Effective Polynomial system solving

Philippe Trébuchet, INRIA, Spaces/Lip6 Calfor

Équipe CALFOR/INRIA Projet SPACES
Université Pierre et Marie Curie
4 place Jussieu
75252 Paris Cedex 05

21 Mai 2003

Challenge : Dealing with semi-algebraic sets.

Solving Polynomial Systems

Complex solving for Real Geometry
Facts:

- Algorithms in real geometry (BPR, SaSc, Rouillier Roy Safey. . .) reduce the problems to a 0-dimensional system solving.

Solving Polynomial Systems

Complex solving for Real Geometry
Facts:

- Algorithms in real geometry (BPR, SaSc, Rouillier Roy Safey. . .) reduce the problems to a 0-dimensional system solving.

- Need of resolution methods working over Non-archimedian Fields

Solving Polynomial Systems

Complex solving for Real Geometry
Facts:

- Algorithms in real geometry (BPR, SaSc, Rouillier Roy Safey. . .) reduce the problems to a 0-dimensional system solving.

- Need of resolution methods working over Non-archimedian Fields

- Intrinsic height of the variety may be VERY high $\Rightarrow$ resolution algorithms must not introduce artificial instabilities.

The setting

The setting

- $\mathbb{K}$ a field.
- $\mathbb{K}[x_1, \ldots, x_n]$ the ring of n variate polynomials.
- $I = (f_1, \ldots, f_s)$ an ideal defining a variety $\mathcal{V}$.

The setting

- $\mathbb{K}$ a field.
- $\mathbb{K}[x_1, \ldots, x_n]$ the ring of n variate polynomials.
- $I = (f_1, \ldots, f_s)$ an ideal defining a variety $\mathcal{V}$.

What does solving mean?

We will exclude the case where $\mathcal{V}$ contains curves, etc. . .
$\mathcal{V}$ is composed of finitely many points $(\zeta_1, \ldots, \zeta_k)$.
What solving is

The setting

- $\mathbb{K}$ a field.
- $\mathbb{K}[x_1, \ldots, x_n]$ the ring of n variate polynomials.
- $I = (f_1, \ldots, f_s)$ an ideal defining a variety $\mathcal{V}$.

What does solving mean?

We will exclude the case where $\mathcal{V}$ contains curves, etc. . .
$\mathcal{V}$ is composed of finitely many points $(\zeta_1, \ldots, \zeta_k)$.
What solving is

- finding an approximation of the coordinates of the points of $\mathcal{V}$.

The setting

- $\mathbb{K}$ a field.
- $\mathbb{K}[x_1, \ldots, x_n]$ the ring of n variate polynomials.
- $I = (f_1, \ldots, f_s)$ an ideal defining a variety $\mathcal{V}$.

What does solving mean?

We will exclude the case where $\mathcal{V}$ contains curves, etc. . .
$\mathcal{V}$ is composed of finitely many points $(\zeta_1, \ldots, \zeta_k)$.
What solving is

- finding an approximation of the coordinates of the points of $\mathcal{V}$.
- giving a univariate representation of $\mathcal{V}$.

The setting

- $\mathbb{K}$ a field.
- $\mathbb{K}[x_1, \ldots, x_n]$ the ring of n variate polynomials.
- $I = (f_1, \ldots, f_s)$ an ideal defining a variety $\mathcal{V}$.

What does solving mean?

We will exclude the case where $\mathcal{V}$ contains curves, etc. . .
$\mathcal{V}$ is composed of finitely many points $(\zeta_1, \ldots, \zeta_k)$.
What solving is

- finding an approximation of the coordinates of the points of $\mathcal{V}$.
- giving a univariate representation of $\mathcal{V}$.
- decomposing $\mathcal{V}$ into simple parts.

Different approaches

- Numerical methods
  - ⋆ homotopy (Somese, Vershelde,etc . . .).
  - ⋆ interval analysis (J.P. Merlet,).
  - ⋆ Weierstrass (Newton-like) (A. Bellido, O. Ruatta, J.C. Yakoubsohn,etc . . .).
- Algebraic methods
  - ⋆ Matrix methods(J. Canny, I. Emiris, B. Mourrain, . . .).
  - ⋆ Geometric methods (P. Aubry, M. Kalkbrener, D. Lazard, M. Moreno and M. Giusti, G. Lecerf, J. Heintz . . .).
  - ⋆ Rewriting methods (B. Buchberger, J.C. Faugère, . . .).

The quotient algebra

Key structure!

$$A = \mathbb{K}[x_1, \ldots, x_n]/I$$

Problems :

- Where to read the information about the points?
- How to compute with it? (representation)
- Is there a best representation? (numerical conditionning, memory size, **stability**. . .)

Where to read the information about the points?

From A to the ζ_i

Theorem : *[Stickelberger]If* $\mathcal{V} = \zeta_1, \ldots, \zeta_k$, $p \in \mathbb{K}[x_1, \ldots, x_n]$.
We will call the multiplication by p, $\mathcal{M}_p$ *the operator* $\begin{array}{c} A \to A \\ f \to fp \end{array}$. *The operator* $\mathcal{M}_p$ *has the following properties :*

- *The eigenvalues of* $\mathcal{M}_p$ *are the numbers* $p(\zeta_1), \ldots, p(\zeta_k)$ *counted with multiplicities.*
- *The common eigenvectors to all the* $\mathcal{M}_p^t$ *are the evaluations to the* ζ_i.

From here it is easy to compute the Chow form of I i.e. compute

$$Det(\mathcal{M}_{u_0+u_1x_1+\cdots+u_nx_n})$$

From A to the ζ_i (continued)

Theorem : *[Hermite]* $\mathbb{K} = \mathbb{R}$, $h \in \mathbb{R}[\mathbf{x}_1, \ldots, x_n]$ *and* Q_h *be the quadratic form*

$$Q_h : \begin{array}{l} A \rightarrow A \\ p \rightarrow Tr(hp^2) \end{array}.$$

Then we have the following two properties :

- *The number of complex root* ζ_i *such as* $h(\zeta_i) \neq 0$ *is the rank of the quadratic form* Q_h.

- *The number of real roots* ζ_i *such as* $h(\zeta_i) > 0$ *and the number of real roots such as* $h(\zeta_i) < 0$ *is the signature of* Q_h.

From A to the ζ_i (continued)

Theorem : *[Rouillier] Let* $u \in \mathbb{K}[x_1, \ldots, x_n]$ *we define :*

- $f_u(T) = \Pi_{i=1..k}(T - u(\zeta_i))^{\mu_i}$
- $g_0(T) = \Sigma_{i=1..k}\mu_i \Pi_{j \neq i}(T - u(\zeta_j))$
- $g_l(T) = \Sigma_{i=1..k}\mu_i \zeta_l.i \Pi_{j \neq i}(T - u(\zeta_j))$

From A to the ζ_i (continued)

Theorem : *[Rouillier] Let $u \in \mathbb{K}[x_1, \ldots, x_n]$ we define :*

- $f_u(T) = \Pi_{i=1..k}(T - u(\zeta_i))^{\mu_i}$
- $g_0(T) = \Sigma_{i=1..k}\mu_i \Pi_{j \neq i}(T - u(\zeta_j))$
- $g_l(T) = \Sigma_{i=1..k}\mu_i \zeta_l.i \Pi_{j \neq i}(T - u(\zeta_j))$

$$f_u(\zeta) = 0 \rightarrow \zeta_i = \begin{pmatrix} g_1(\zeta)/g_0(\zeta) \\ \vdots \\ g_n(\zeta)/g_0(\zeta) \end{pmatrix}$$

If u separates the ζ_i then the preceeding polynomials define a Rational Univariate Representation.

From A to the ζ_i (continued)

Theorem : *[Rouillier] Let $u \in \mathbb{K}[x_1, \dots, x_n]$ we define :*

- $f_u(T) = \Pi_{i=1..k}(T - u(\zeta_i))^{\mu_i}$
- $g_0(T) = \Sigma_{i=1..k}\mu_i \Pi_{j \neq i}(T - u(\zeta_j))$
- $g_l(T) = \Sigma_{i=1..k}\mu_i \zeta_l . i \Pi_{j \neq i}(T - u(\zeta_j))$

$$f_u(\zeta) = 0 \rightarrow \zeta_i = \begin{pmatrix} g_1(\zeta)/g_0(\zeta) \\ \vdots \\ g_n(\zeta)/g_0(\zeta) \end{pmatrix}$$

If u separates the ζ_i then the preceeding polynomials define a Rational Univariate Representation.

Proposition : *A separating element can be chosen in the family :*

$$\mathcal{U} = \{x_1 + jx_2 + \cdots + j^{n-1}x_n, j = 0 \dots nk(k-1)/2\}$$

we also have :

$$u \text{ is separating the } \zeta_i \Leftrightarrow deg(minpol((M_u))) == dim(A)$$

Representation of A

Working in A

Finding Canonical representation of the elements of $\mathbb{K}[x_1, \ldots, x_n]$

(Normal Forms)

- $\mathcal{V}$ contains only points $\Rightarrow$ A is a finite dimensionnal $\mathbb{K}$-vector space.
- As a $\mathbb{K}$-vector space $\mathbb{K}[x_1, \ldots, x_n]$ is spanned by the monomials.
- Finding a basis of A can be reduced to finding a **monomial** basis of A (noted B).

Suppose that we know B, a monomial basis of A. Then we have the equality :

$$\mathbb{K}[x_1, \ldots, x_n] = \langle B \rangle \oplus \langle I \rangle$$

($\langle I \rangle$ denotes the $\mathbb{K}$-vector space generated by I)

Here Comes Macaulay!

Computing the canonical expression of a polynomial $p \in \mathbb{K}[x_1, \ldots, x_n]$ in B is simple:

Algorithm 1. INPUT: $p \in \mathbb{K}[x_1, \ldots, x_n]$
$I = (f_1, \ldots, f_s)$
B *a monomial basis of* A
OUTPUT *the representation of* p *in* A.

- *k=0, reduced=*`false`
- *while !reduced*
  - *construct the matrix* $Mat_k(p)$
  - *echelonize* $Mat_k(p)$ *without permuting lines*
  - *if the last line has no nonzero coefficients outside the columns corresponding to* B *then*
    reduced=`true`
- *return the last line of the echelonized matrix.*

Here Comes Macaulay!

Computing the canonical expression of a polynomial $p \in \mathbb{K}[x_1, \ldots, x_n]$ in B is simple:

$$\begin{array}{c} \quad B^c \quad B \\ \left(\begin{array}{c} \mathtt{M}f_1 \\ \vdots \\ \mathtt{M}f_s \\ p \end{array} \right) \end{array}$$

Algorithm 1. Input: $p \in \mathbb{K}[x_1, \ldots, x_n]$
$I = (f_1, \ldots, f_s)$
B a monomial basis of A

Output *the representation of p in A.*

- *k=0, reduced=*`false`
- *while !reduced*
  - ⋆ *construct the matrix $Mat_k(p)$*
  - ⋆ *echelonize $Mat_k(p)$ without permuting lines*
  - ⋆ *if the last line has no nonzero coefficients outside the columns corresponding to B then*
    reduced=`true`
- *return the last line of the echelonized matrix.*

Macaulay Construction

If the f_i are $generic$ of degree d_i then we have :

- the set $E_0 = \{x_1^{\alpha_1} \dots x_n^{\alpha_n}, \alpha_i < d_i\}$ is a basis of A.

- the sets $E_i = \{x_1^{\alpha_1} \dots x_n^{\alpha_n}, \Sigma_{j=1..n}\alpha_j \leq \Sigma_{j=1..n}(d_j - 1) + 1,\ \forall n \geq j \geq i,\ \alpha_j < d_j\}$

Algorithm 2. Input: $f_0 \in \mathbb{K}[x_1, \dots, x_n]$ *whom we want the multiplication operator*
$f_1, \dots, f_s$ *generic polynomials*
Output: *The multiplication matrix of* f_0 *in* A

- *construct the matrix :*

$$\begin{array}{c} \begin{array}{cccc} E_0 f_0 & E_1 f_1 & \dots & E_n f_n \end{array} \\ \left(\begin{array}{c|c} \mathtt{A} & \mathtt{C} \\ \hline \mathtt{B} & \mathtt{D} \end{array}\right) \end{array}$$

- *return the Schur complement*$= A - CD^{-1}B$

Matrix methods

Study of the sylvester endomorphism :

$$\begin{array}{ccc} < E_0 > \times < E_1 > \times \cdots \times < E_s > & \longrightarrow & \mathbb{K}[x_1, \ldots, x_n] \\ (p_0, p_1, \ldots, p_s) & \longrightarrow & \Sigma_{i=0..s} p_i f_i \end{array}$$

- E_0 must be a basis of A.
- E_i must be wide enought for the Schur complement of the bloc `A` gives the multiplication operator.

Ex: Sparse resultant (J. Canny, I. Emiris. . .), etc...

Gröbner bases

Algorithm 3. INPUT : $f_1, \ldots, f_s$ *generating* I*.(any dimensional)*
An admissible monomial order γ
OUTPUT : *A representation of* A

- $G = \{f_1, \ldots, f_s\}$
- **repeat**
  - ⋆ $G' = G$
  - ⋆ *for all pair* $(p, q), \ p, q \in G'$ *do*
    - ∗ $S = \overline{S(p,q)}^{G'}$
    - ∗ *if* $S \neq 0$ *then* $G = G \cup S$
- **until** $G' == G$
- *return* G

Gröbner bases (Faugère,Lazard,Lombardi)

At a second thought. . .

- Can substitute polynomial algebra by linear algebra

Gröbner bases (Faugère,Lazard,Lombardi)

At a second thought. . .

- Can substitute polynomial algebra by linear algebra

Then we can do the following changes :

- see the polynomials of I as linear dependence relations

- substitute S-polynomial reductions with echelonisations of matrices

Reduce S-pol

Gröbner bases (Faugère,Lazard,Lombardi)

At a second thought. . .

- Can substitute polynomial algebra by linear algebra

Then we can do the following changes :

- see the polynomials of I as linear dependence relations

- substitute S-polynomial reductions with echelonisations of matrices

$$
\begin{array}{c}
\text{monomials} \\
\left(\begin{array}{c}
\frac{lcm(\gamma(p),\gamma(q))}{\gamma(p)}p \\
\frac{lcm(\gamma(p),\gamma(q))}{\gamma(q)}q \\
\vdots \\
\hline \\
\text{Reductor polynomials}
\end{array}\right)
\end{array}
$$

Gröbner bases (Faugère,Lazard,Lombardi)

At a second thought. . .

- Can substitute polynomial algebra by linear algebra

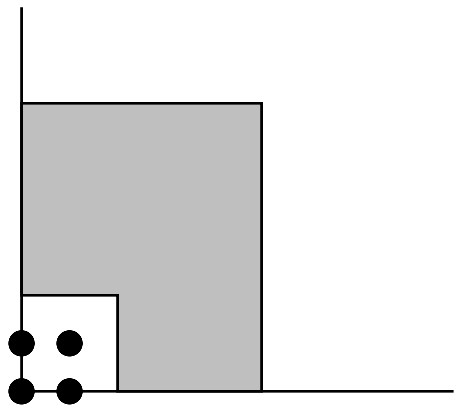
Then we can do the following changes :

- see the polynomials of I as linear dependence relations
- substitute S-polynomial reductions with echelonisations of matrices
- proceed incrementaly

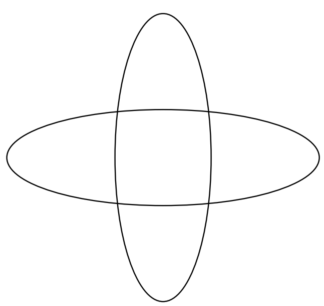
$$\begin{array}{c} \text{monomials} \\ \left(\begin{array}{c} \frac{lcm(\gamma(p),\gamma(q))}{\gamma(p)} p \\ \frac{lcm(\gamma(p),\gamma(q))}{\gamma(q)} q \\ \vdots \\ \hline \\ \text{Reductor polynomials} \end{array} \right) \end{array}$$

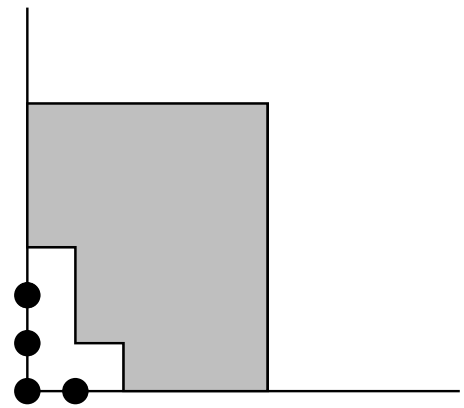
Unhappy with Gröbner though

ex: $p_1 = ax_1^2 + bx_2^2 + n_1(x_1, x_2)$
$p_2 = cx_1^2 + dx_2^2 + n_2(x_1, x_2)$

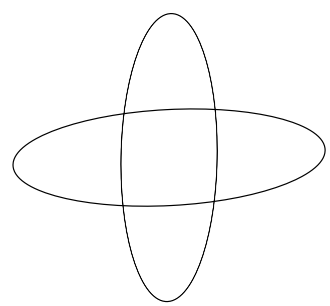


$$n_1 = n_2 = 0$$





$$n_1 = \varepsilon_1 x_1 x_2$$
$$n_2 = \varepsilon_2 x_1 x_2$$



Computing better representations

Notations

- A Normal form is :• A monomial basis B of A
  • An algorithm to project $\mathbb{K}[x_1, \ldots, x_n]$ onto B

- A choice function refining the degree, γ, is a function that takes a polynomial p and returns one monomial $\gamma(p)$ of the support of p such that $deg(\gamma(p)) = deg(p)$.

- Let S be a subset of $\mathbb{K}[x_1, \ldots, x_n]$ S^+ is the set :

$$S^+ = x_1 S \cup \cdots \cup x_n S$$

- $P \subset \mathbb{K}[x_1, \ldots, x_n], M \subset \{x^\alpha,\ \alpha \in \mathbb{Z}^n\}$,$(P|M)$, is the matrix whose columns are index by M and lines by P.

- Let p_1, p_2 $\in$ $\mathbb{K}[x_1, \ldots, x_n]$, the C-polynomial of p_1 and p_2 is $C(p_1, p_2) = \frac{lcm(\gamma(p_1),\gamma(p_2)),\gamma(p_1)}{p}_1 - \frac{lcm(\gamma(p_1),\gamma(p_2)),\gamma(p_2)}{p}_2$, and $deg_C(C(p_1, p_2)) = deg(lcm(\gamma(p_1), \gamma(p_2)))$.

Macaulay revisited

Provide a way to write any polynomial p under the form
$p = b + i,\ b \in \langle B \rangle,\ i \in \langle I \rangle$

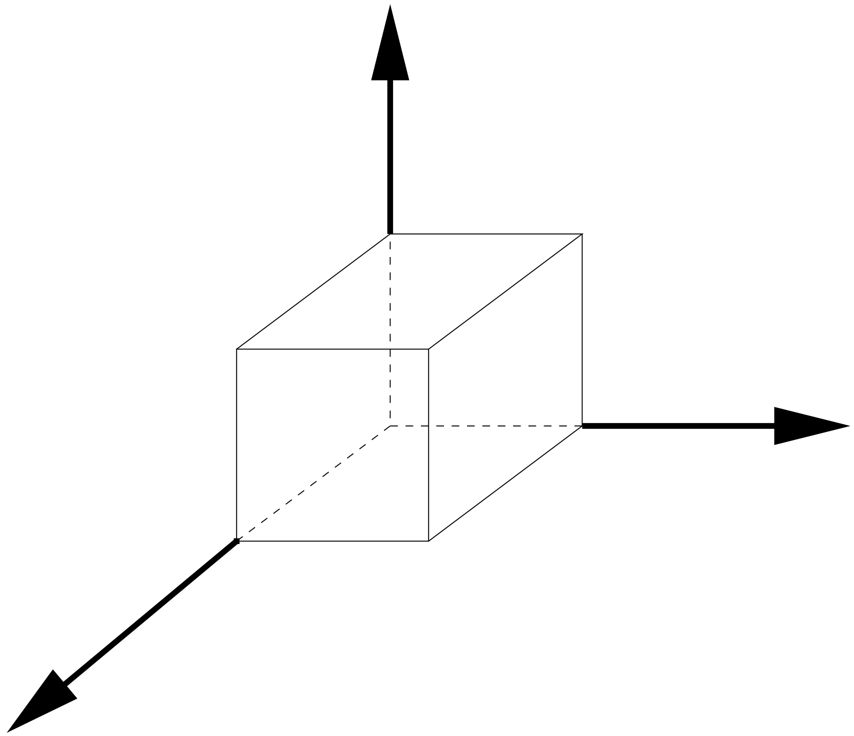
Macaulay revisited

Provide a way to write any polynomial p under the form
$p = b + i,\ b \in \langle B \rangle,\ i \in \langle I \rangle$

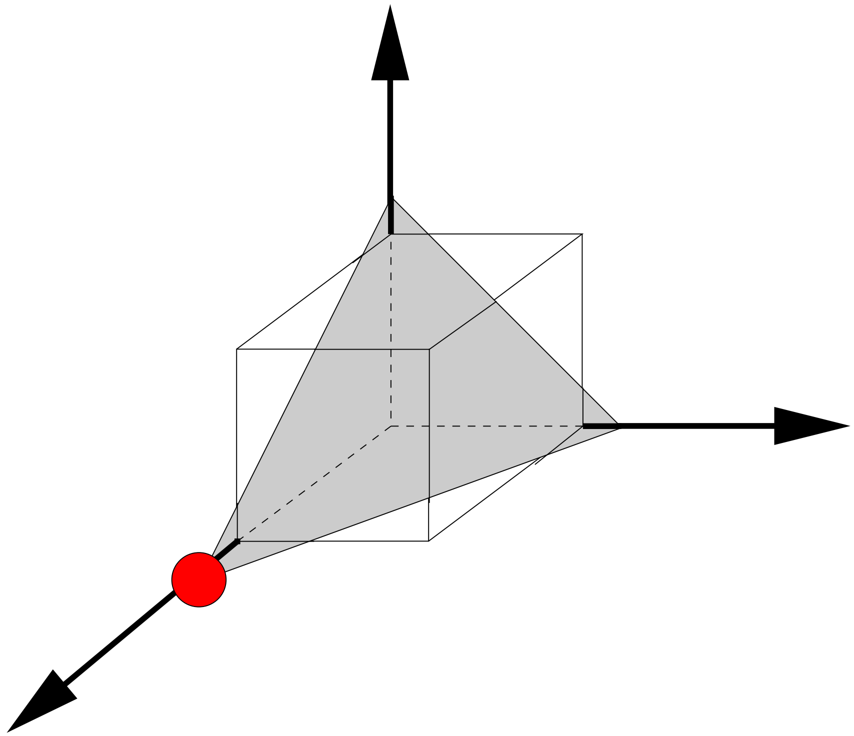
Algorithm 4. [Mourrain, T.] INPUT : $F = f_1, \ldots, f_s$ `generic` *polynomials*
OUTPUT : *The representation of A given by Macaulay.*

- $k = min(deg(f_i),\ i = 1..s)$.
- $P_k = F[k]$, $M_k = \{x_i^k,\ f_i \in P_k\}$
- **while** $k \leq \Sigma_{i=1..s}(d_i - 1) + 1$ **do**
  - ⋆ $P_{k+1} = P_k^+ \cap B^+ \cup proj(P[k+1])$, $M_{k+1} = M_k^+ \cap B^+ \cup \{x_i^{k+1}, f_i \in P[k+1]\}$
  - ⋆ *Solve the linear system* $(P_{k+1}|M_{k+1})X = P_{k+1}$.
  - ⋆ $k = k + 1$
- *return* $P_i, i = min(deg(f_i),\ i = 1..s)..\Sigma_{i=1..s}(d_i - 1) + 1$

Macaulay revisited an example

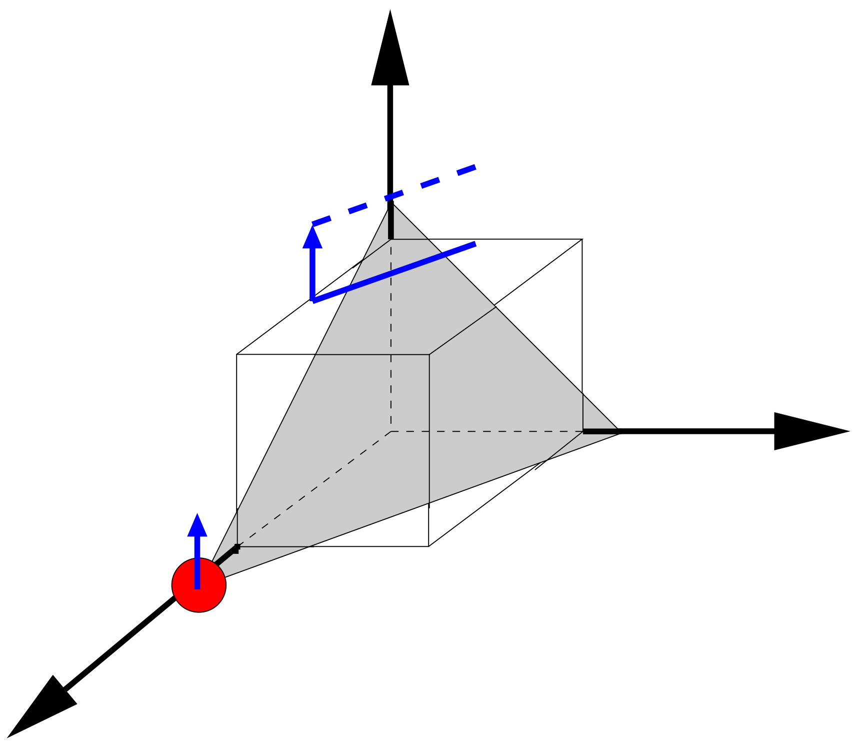


Macaulay revisited an example



$$f_i = x_i^{d_i} + \dots$$

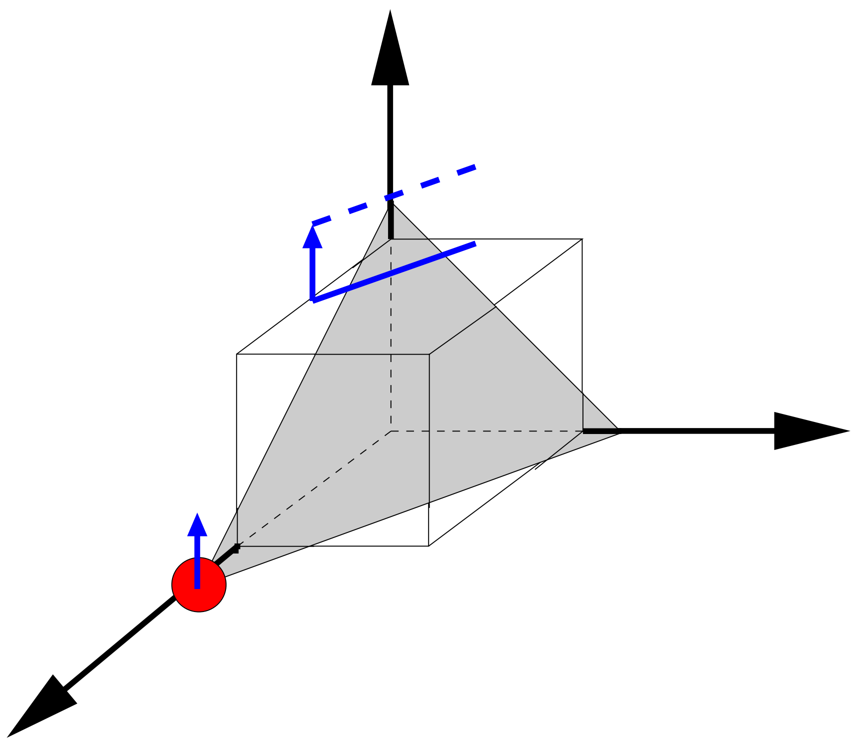
Macaulay revisited an example



$$f_i = x_i^{d_i} + \dots$$

$$x_1 f_i = x_1 x_i^{d_i} + \alpha_2 x_1^{d_1} x_2 + \cdots + \alpha_n x_1^{d_1} x_n + \dots$$

Macaulay revisited an example

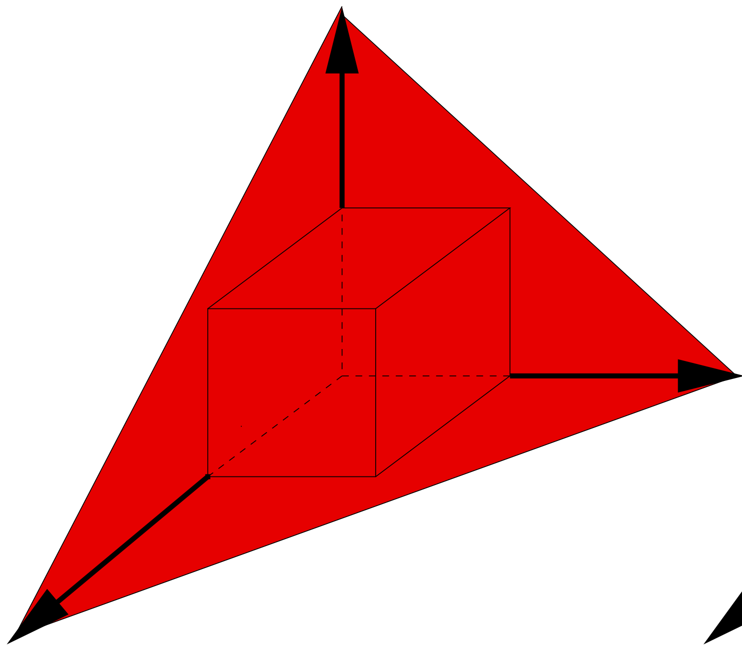


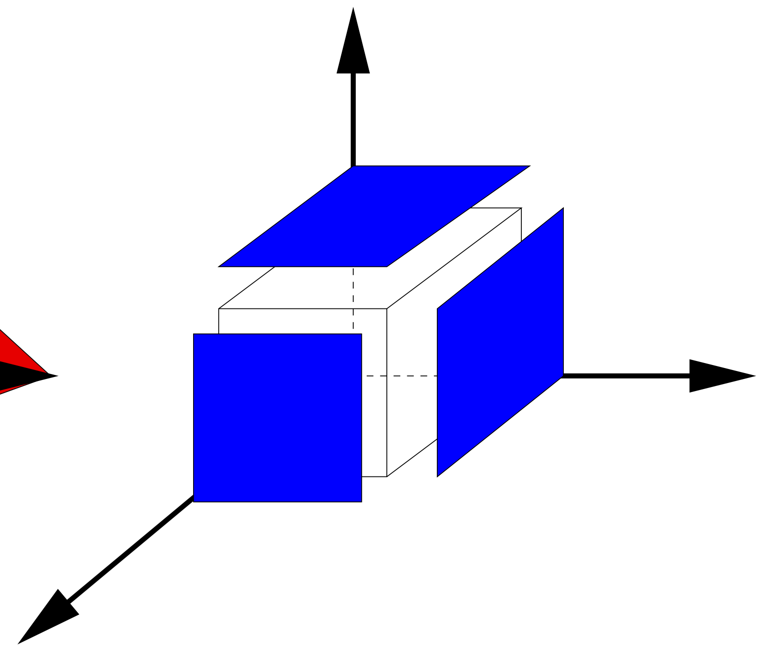
$$f_i = x_i^{d_i} + \dots$$

$$x_1 f_i = x_1 x_i^{d_i} + \alpha_2 x_1^{d_1} x_2 + \dots + \alpha_n x_1^{d_1} x_n + \dots$$

$$\left(\begin{array}{cccccccc} & & \dots\dots\dots\dots\dots\dots & & & & & \\ 1 & \dots & \alpha_2 & \alpha_3 & \dots & \alpha_n & \dots \\ & & \dots\dots\dots\dots\dots\dots & & & & & \end{array} \right)$$

comparing to the original





Comparing to the original

n	5	6	7	8	9	10	11
size	5	6	7	8	9	10	11
of the	20	30	42	56	72	90	110
matrices	30	60	105	168	252	360	495
	20	60	140	280	504	840	1320
	5	30	105	280	630	1260	2310
		6	42	168	504	1260	2772
			7	56	252	840	2310
				8	72	360	1320
					9	90	495
						10	110
							11
Sum	80	192	448	1024	2304	5120	11264
Macaulay	462	1 716	6 435	24 310	92 378	352 716	1 352 078
Nb points	32	64	128	256	512	1024	2048

Table 1: Size of the systems to invert

What to do if B is not known?

Algorithm 5. [Mourrain] INPUT : $F = f_1, \ldots, f_n$
L *a* $\mathbb{K}$*-vector space connex to* 1
OUTPUT : *The multiplicative structure of* A

1) $K_0 = \langle f_1, \ldots, f_n \rangle$, $n = 0$

2) **repeat**

- $\star$ $K_{n+1} = K_n^+ \cap L$
- $\star$ $n = n + 1$

3) **until** $K_n == K_{n-1}$

4) Compute B *a supplementary of* K_n *in* L

5) If $B^+ \not\subset L$, $L = L^+$ *and go back to 1)*

What to do if not generic

a new criterion :

- $\mathbb{K}[x_1, \ldots, x_n] =< B > \oplus < I >$.

$\Longleftrightarrow$

- The multiplication operators by the variables commute.

What to do if not generic

a new criterion :

- $\mathbb{K}[x_1, \ldots, x_n] = < B > \oplus < I >$.

$\Longleftrightarrow$

- The multiplication operators by the variables commute.

Properties

$\oplus$ Very good numerical stability.

$\oplus$ Possibility to take into account the geometry of the problem.

$\ominus$ **VERY** expensive computation.

Computing better for computing less

What went wrong :

- Postpone the effective computation of B to the last step.

Computing better for computing less

What went wrong :

- Postpone the effective computation of B to the last step.

What we should do to mimic Gröbner bases :

- try since the first step to guess what will be B
- check our guess is correct
- do not compute polynomials that go $too\ far$ from B

Computing better for computing less

Algorithm 6. [T.] NORMAL FORMS

INPUT : $F = f_1, \ldots, f_s$ *defining* I *(*0*-dimensionnal)*

γ *a choice* **function** *refining the degree.*

Initilisation : *Choose the* f_i *of minimal degree*

$b = (\gamma(f_i))$, $k = deg(f_i)$, $P_k = \{f_i\}$, $M_k = \{\gamma(f_i)\}$

Core Loop : `While` $(newmon||k \leq Maxdeg(F))$ *do*

Computing better for computing less

Algorithm 6. [T.] NORMAL FORMS

INPUT : $F = f_1, \ldots, f_s$ *defining* I *(*0*-dimensionnal)*

γ *a choice* **function** *refining the degree.*

Initilisation : *Choose the* f_i *of minimal degree*

$b = (\gamma(f_i))$, $k = deg(f_i)$, $P_k = \{f_i\}$, $M_k = \{\gamma(f_i)\}$

Core Loop : `While` $(newmon || k \leq Maxdeg(F))$ *do*

- *Compute the* C*-pol of degree* $k + 1$

Computing better for computing less

Algorithm 6. [T.] NORMAL FORMS

INPUT : $F = f_1, \ldots, f_s$ *defining* I *(*0*-dimensionnal)*

γ *a choice* **function** *refining the degree.*

Initilisation : *Choose the* f_i *of minimal degree*

$b = (\gamma(f_i))$, $k = deg(f_i)$, $P_k = \{f_i\}$, $M_k = \{\gamma(f_i)\}$

Core Loop : `While` $(newmon || k \leq Maxdeg(F))$ *do*

- *Compute the* C*-pol of degree* $k+1$
- *Compute* $P_{k+1} = P_k^+ \cap b^+$ *and take into account the* f_i *of degree* $k+1$

Computing better for computing less

Algorithm 6. [T.] NORMAL FORMS

INPUT : $F = f_1, \ldots, f_s$ *defining* I *(*0*-dimensionnal)*
γ *a choice* **function** *refining the degree.*

Initilisation : *Choose the* f_i *of minimal degree*
$b = (\gamma(f_i))$, $k = deg(f_i)$, $P_k = \{f_i\}$, $M_k = \{\gamma(f_i)\}$

Core Loop : `While` $(newmon || k \leq Maxdeg(F))$ *do*

- *Compute the* C*-pol of degree* $k+1$
- *Compute* $P_{k+1} = P_k^+ \cap b^+$ *and take into account the* f_i *of degree* $k+1$
- $M_{k+1} = \{M_k^+ \cap b^+\}$

Computing better for computing less

Algorithm 6. [T.] NORMAL FORMS

INPUT : $F = f_1, \ldots, f_s$ *defining* I *(*0*-dimensionnal)*

γ *a choice* **function** *refining the degree.*

Initilisation : *Choose the* f_i *of minimal degree*

$b = (\gamma(f_i))$, $k = deg(f_i)$, $P_k = \{f_i\}$, $M_k = \{\gamma(f_i)\}$

Core Loop : `While` $(newmon || k \leq Maxdeg(F))$ *do*

- *Compute the* C*-pol of degree* $k+1$
- *Compute* $P_{k+1} = P_k^+ \cap b^+$ *and take into account the* f_i *of degree* $k+1$
- $M_{k+1} = \{M_k^+ \cap b^+\}$
- $PseudoSolve\,(P_{k+1}|M_{k+1})X = P_{k+1}$

Computing better for computing less

Algorithm 6. [T.] NORMAL FORMS

INPUT : $F = f_1, \ldots, f_s$ *defining* I *(*0*-dimensionnal)*
γ *a choice* **function** *refining the degree.*

Initilisation : *Choose the* f_i *of minimal degree*
$b = (\gamma(f_i))$, $k = deg(f_i)$, $P_k = \{f_i\}$, $M_k = \{\gamma(f_i)\}$

Core Loop : `While` $(newmon || k \leq Maxdeg(F))$ *do*

- *Compute the* C*-pol of degree* $k+1$
- *Compute* $P_{k+1} = P_k^+ \cap b^+$ *and take into account the* f_i *of degree* $k+1$
- $M_{k+1} = \{M_k^+ \cap b^+\}$
- $PseudoSolve\ (P_{k+1}|M_{k+1})X = P_{k+1}$
- *Reduce the* C*-pol with respect to* P_j

Computing better for computing less

Algorithm 6. [T.] $\textsc{Normal Forms}$

INPUT : $F = f_1, \ldots, f_s$ *defining* I *(*0*-dimensionnal)*

γ *a choice* **function** *refining the degree.*

Initilisation : *Choose the* f_i *of minimal degree*

$b = (\gamma(f_i))$, $k = deg(f_i)$, $P_k = \{f_i\}$, $M_k = \{\gamma(f_i)\}$

Core Loop : `While` $(newmon || k \leq Maxdeg(F))$ *do*

- *Compute the* C*-*pol *of degree* $k+1$
- *Compute* $P_{k+1} = P_k^+ \cap b^+$ *and take into account the* f_i *of degree* $k+1$
- $M_{k+1} = \{M_k^+ \cap b^+\}$
- $PseudoSolve$ $(P_{k+1}|M_{k+1})X = P_{k+1}$
- *Reduce the* C*-*pol *with respect to* P_j
- *Whether the* C*-*pol *reduce to* 0 *or not and whether* $(P_{k+1}|M_{k+1})$ *is of maximal rank update* b, P_{k+1}, k, M_{k+1}

End `While`

Computing better for computing less

Algorithm 6. [T.] NORMAL FORMS

INPUT : $F = f_1, \ldots, f_s$ *defining* I *(*0*-dimensionnal)*

γ *a choice* **function** *refining the degree.*

Initilisation : *Choose the* f_i *of minimal degree*

$b = (\gamma(f_i))$, $k = deg(f_i)$, $P_k = \{f_i\}$, $M_k = \{\gamma(f_i)\}$

Core Loop : `While` $(newmon || k \leq Maxdeg(F))$ *do*

- *Compute the* C*-pol of degree* $k+1$
- *Compute* $P_{k+1} = P_k^+ \cap b^+$ *and take into account the* f_i *of degree* $k+1$
- $M_{k+1} = \{M_k^+ \cap b^+\}$
- $PseudoSolve\ (P_{k+1}|M_{k+1})X = P_{k+1}$
- *Reduce the* C*-pol with respect to* P_j
- *Whether the* C*-pol reduce to* 0 *or not and whether* $(P_{k+1}|M_{k+1})$ *is of maximal rank update* b, P_{k+1}, k, M_{k+1}

End `While`

OUTPUT : $\{P_j, j = 0..k\}$ *that allow to construct a system of normal form* $\forall k \in \mathbb{N}$

$$r = \#M_{k+1} - Rank((M_{k+1}|P_{k+1}))$$
$$c = \#\{\text{non reduced to } 0\}$$

$\mathbf{r = 0}$ && $\mathbf{c = 0}$	$\mathbf{r = 0}$ && $\mathbf{c \neq 0}$	$\mathbf{r \neq 0}$ && $\mathbf{c \neq 0}$	$\mathbf{r \neq 0}$ && $\mathbf{c = 0}$
Go to the next step	Treat the non reduced and go back at their degree	Add to B and treat the nonzero	Add to b Go to the next step

An example

$$F = \begin{cases} x^2 + xy \\ y^2 + xy \\ xy^3 \end{cases} \qquad \gamma = \text{ask user}$$

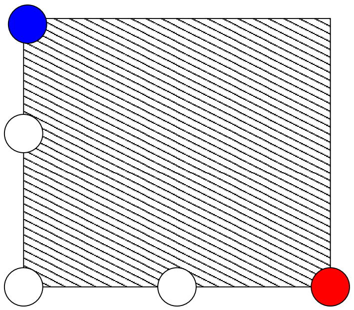
Compute the quotient :

An example

$$F = \begin{cases} x^2 + xy \\ y^2 + xy \\ xy^3 \end{cases}$$

$\gamma =$ ask user

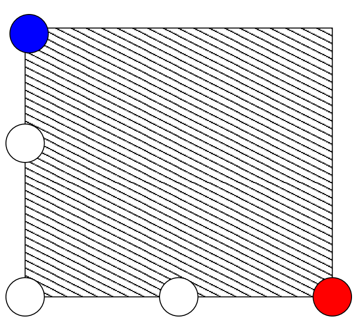
Compute the quotient :

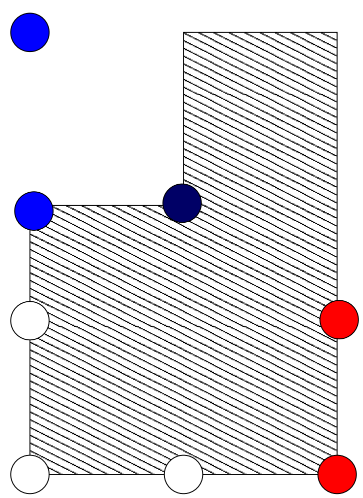


An example

$$F = \begin{cases} x^2 + xy \\ y^2 + xy \\ xy^3 \end{cases} \qquad \gamma = \text{ask user}$$

Compute the quotient :



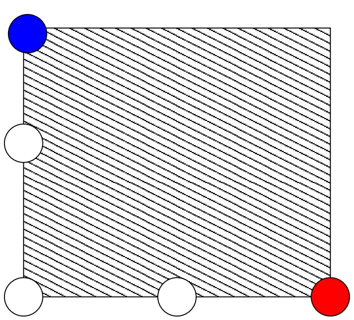


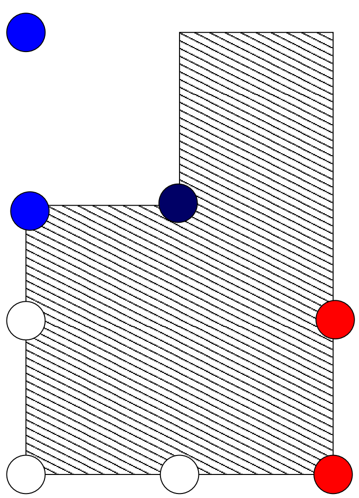
An example

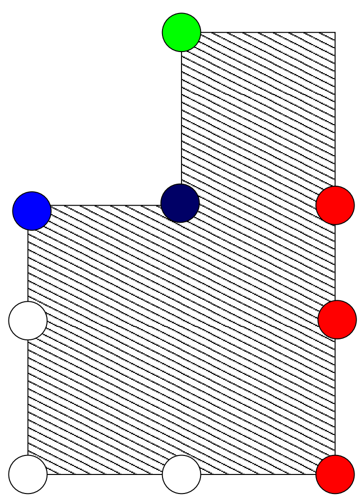
$$F = \begin{cases} x^2 + xy \\ y^2 + xy \\ xy^3 \end{cases}$$

$\gamma =$ ask user

Compute the quotient :







What to do if we do not refine the degree

An example : the `Lex` monomial order
The Problems :

- we do not have always *reduced* polynomials.
- we do not know in advance how to determine what polynomials will be reduced
- we must avoid dead lock (i.e. polynomial p depending on polynomial q and polynomial q depending on polynomial p).

What to do if we do not refine the degree

An example : the `Lex` monomial order

The Problems :

- we do not have always *reduced* polynomials.
- we do not know in advance how to determine what polynomials will be reduced
- we must avoid dead lock (i.e. polynomial p depending on polynomial q and polynomial q depending on polynomial p).

The solutions:

- put in stand by non fully reduced polynomials.
- at the end of each step determine what will be the polynomial set to consider.
- use a linear form from the first quadrant to avoid dead-locks.

Stability in practice

Parallel Robot with quaternion parametrization:
`http://www-sop.inria.fr/saga/POL/BASE/2.multipol/rbpl16.html` :

γ	time in s	Peak mem	average of $cond(M_i)$
Macaulay	632	17M	10^7
Dlex	3325	40M	10^7
Dinvlex	2554	40M	10^7
Size	1240	20M	10^8
Numeric	9889	50M	10^6
Random	636	30M	10^7

Stability in practice

Parallel Robot with quaternion parametrization:
`http://www-sop.inria.fr/saga/POL/BASE/2.multipol/rbpl16.html` :

γ	time in s	Peak mem	average of $cond(M_i)$
Macaulay	632	17M	10^7
Dlex	3325	40M	10^7
Dinvlex	2554	40M	10^7
Size	1240	20M	10^8
Numeric	9889	50M	10^6
Random	636	30M	10^7

ε-Computations!!

Conclusion

The algebraic solving process is essentially composed of two mandatory steps :

- Computation of a representation of the quotient algebra.

Conclusion

The algebraic solving process is essentially composed of two mandatory steps :

- Computation of a representation of the quotient algebra.
- Determination of certain quantities linked with multiplication operators.

Conclusion

The algebraic solving process is essentially composed of two mandatory steps :

- Computation of a representation of the quotient algebra.
- Determination of certain quantities linked with multiplication operators.

It is greatly profitable to take into account the needs of the second step during the first one.

Conclusion

The algebraic solving process is essentially composed of two mandatory steps :

- Computation of a representation of the quotient algebra.
- Determination of certain quantities linked with multiplication operators.

It is greatly profitable to take into account the needs of the second step during the first one.

What remains to do :

- Find a suitable stopping criterion in positive dimension
- Use and optimize the infinitesimals for real life applications.