

# Some certified methods for Real Solving - applications in robotics

*Fabrice Rouillier*

Fabrice.Rouillier@inria.fr - <http://fgbrs.lip6.fr/~rouillie>

SALSA (INRIA) project and SPIRAL (LIP6) team

*Paris, France*

# General Objectives

$$\mathcal{E} = \{p_1, \dots, p_r\}, \mathcal{F} = \{f_1, \dots, f_l\}, \text{ with } p_i, f_i \in \mathbb{Q}[U, X]$$

$$U = U_1, \dots, U_d \Rightarrow \text{parameters}$$

$$X = X_{d+1}, \dots, X_n \Rightarrow \text{indeterminates}$$

$$\mathcal{C} = \{x \in \mathbb{C}^n, p_1 = 0, \dots, p_r = 0, f_1 \neq 0, \dots, f_s \neq 0\}$$

$$\mathcal{S} = \{x \in \mathbb{R}^n, p_1 = 0, \dots, p_r = 0, f_1 > 0, \dots, f_s > 0\}$$

In part 1 (parallel robot) : we suppose that  $d = 0$  (no parameter) and that the ideal  $\langle p_1, \dots, p_r \rangle$  is zero-dimensional (finite number of complex zeroes).

In part 2 (serial robot/optimization problem) : we suppose that  $d \neq 0$  and that the ideal  $\langle p_1, \dots, p_r \rangle_{U=u}$  is zero-dimensional for almost all  $u \in C^d$  (general results will be described but only this case will be detailed).

## General Objectives (2)

The goal is to propose mathematical objects and algorithms to **SOLVE (answer to the end-user question !)** such systems. The end-user queries we are interested in are :

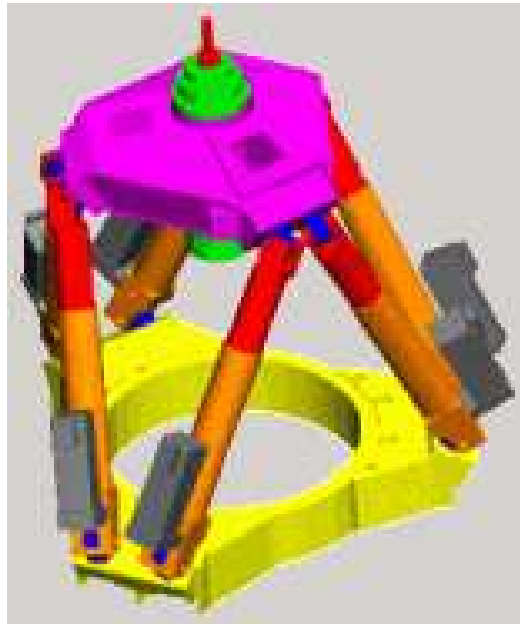
- Zero-dimensional systems :
  - count the real / complex roots ;
  - detect multiple points and compute multiplicities ;
  - provide an accurate and certified approximation of the roots ;
  - signs of polynomials at the real roots of a system ;
- Parametric systems :
  - count the real / complex roots wrt parameter's values ;
  - describe geometrically the solutions set ;
  - provide formal expressions of the roots ;
  - check/compute numerically stable solutions.

## General Objectives (3)

Constraints (complementing numerical methods):

- Exact/certified results : a real root is not a complex root with a small imaginary part ... A numerical approximation must be certified with respect to a precision given by the end-user, etc.
- Universal algorithms : being able to check the assumptions (for example zero-dimensional) and the mathematical arbitrary choices (so called “generic” choices).
- Efficiency : computation time (bit operations) but also memory consuming.

# Part 1 : Zero-dimensional Systems



**The challenge** : given the geometry of the robot and the length of the legs, find the possible positions of the robot.

**Applications** : mostly off-line computations like helping to the design of the robot, checking a path planning process, etc.

# Univariate case

We assume that we know how to solve the univariate case (at least in the real case).

Counting/isolating real roots : Descarte's based methods, Sturm sequences etc . by means of "isolating intervals" with rational bounds.

Evaluating polynomials (or their signs) at the roots of a univariate polynomial (direct application of previous item).

**A reference book :**

Algorithms in Real Algebraic Geometry - Basu Pollack and Roy (Springer)

# Notations

- $\mathcal{E} = \{p_1, \dots, p_s\} \subset \mathbb{Q}[X_1, \dots, X_n]$ ,  $\langle \mathcal{E} \rangle$  is the ideal generated by  $\mathcal{E}$ ;
- $V(\mathcal{E}) \subset \mathbb{C}^n$  is the zero set of  $\langle \mathcal{E} \rangle$  or equivalently the set of complex solutions of  $\{x \in \mathbb{C}^n, p_1(x) = 0, \dots, p_n(x) = 0\}$ ;
- $x \in \mathbb{C}^n$ ,  $x_i$  denotes its  $i$ -th coordinate.

# About Gröbner bases

A Gröbner basis of an ideal  $\langle \mathcal{E} \rangle$  is a set of generators of  $\mathcal{E}$  equipped with a monomial ordering  $<$  and a normal form function (generalizes the Euclidean division to reduce modulo  $\langle \mathcal{E} \rangle$ )

**Theorem 1.** *Let  $G = \{g_1, \dots, g_l\}$  be a Gröbner basis for any (monomial) ordering  $<$  of  $\mathcal{E} = \{p_1, \dots, p_s\} \in \mathbb{Q}[X_1, \dots, X_n]^s$ . The following properties are equivalent:*

- *For all index  $i$ ,  $i = 1..n$ , there exists a polynomial  $g_j \in G$  and a positive integer  $n_j$  such that  $X_i^{n_j} = LM(g_j, <)$  ( $LM$ =leading monomial wrt  $<$ );*
- *The system  $\{p_1 = 0, \dots, p_s = 0\}$  has a finite number of solutions in  $\mathbb{C}^n$ .*
- *$\frac{\mathbb{Q}[X_1, \dots, X_n]}{\langle G \rangle}$  is a finite dimensional  $\mathbb{Q}$ -vector space*

**Corollary 2.**  *$\mathcal{B} = \{t = X_1^{e_1} \cdot X_n^{e_n}, (e_1, \dots, e_n) \in \mathbb{N}^n \mid \text{normalForm}(t, G, <) = t\} = \{w_1, \dots, w_D\}$  is a basis of  $\mathbb{Q}[X_1, \dots, X_n]/\langle \mathcal{E} \rangle$  as a  $\mathbb{Q}$ -vector space;*

# Stickelberger's theorem

**Definition 3.** Let  $h \in \mathbb{C}[X_1, \dots, X_n]$ ;

$$m_h: \frac{\mathbb{C}[X_1, \dots, X_n]}{\langle \mathcal{E} \rangle} \longrightarrow \frac{\mathbb{C}[X_1, \dots, X_n]}{\langle \mathcal{E} \rangle}$$
$$u \quad \mapsto \quad \overline{h u}$$

**Theorem 4.** The eigenvalues of  $m_h$  are the  $h(\alpha)$ ,  $\alpha \in \mathbf{V}(\langle \mathcal{E} \rangle)$  with multiplicity  $\mu(\alpha)$ .

# Computing in the quotient algebra

For computing a matrix of  $m_h$ , we need :

- A (monomial) basis  $\mathcal{B} = \{w_1, \dots, w_D\}$  of  $\frac{\mathbb{Q}[X_1, \dots, X_n]}{\langle \mathcal{E} \rangle}$  (or equivalently of  $\frac{\mathbb{C}[X_1, \dots, X_n]}{\langle \mathcal{E} \rangle}$ ) : can be deduced from a Gröbner basis for any ordering, a triangular set, Generalized normal forms.
- A way for computing the class  $\bar{h}$  of  $h \in \mathbb{Q}[X_1, \dots, X_n]$  in  $\frac{\mathbb{Q}[X_1, \dots, X_n]}{\langle \mathcal{E} \rangle}$  as a vector wrt  $\mathcal{B}$  (denoted by  $\vec{h}$  from now) : for example the “normalForm” associated to any Gröbner basis.

For example, the columns of the matrix of  $m_h$  wrt  $\mathcal{B}$  are the coordinates of  $\overrightarrow{h w_i}^{\mathcal{B}}$ .

# Hermite's quadratic form

**Double goal** : count the distinct complex/real roots - decrease the number of operations for searching a separating element.

**Definition 5.** (*Hermite's quadratic form*). For  $f \in \mathbb{Q}[X_1, \dots, X_n]$  :

$$q_f: \frac{\mathbb{Q}[X_1, \dots, X_n]}{\langle \mathcal{E} \rangle} \longrightarrow \mathbb{Q}$$
$$\vec{h} \longmapsto \text{Trace}(m_{fh^2})$$

**Theorem 6.** For  $f \in \mathbb{Q}[X_1, \dots, X_n]$ ,

- $\text{rank}(q_f) = \#\{x \in \mathbf{V}(\langle \mathcal{E} \rangle), f(x) \neq 0\}$
- $\text{signature}(q_f) = \#\{x \in \mathbf{V}(\langle \mathcal{E} \rangle) \cap \mathbb{R}^n, f(x) > 0\} - \#\{x \in \mathbf{V}(\langle \mathcal{E} \rangle) \cap \mathbb{R}^n, f(x) < 0\}$

**Corollary 7.** Taking  $f = 1$  :

- $\text{rank}(q_1) = \#\mathbf{V}(\langle \mathcal{E} \rangle)$
- $\text{signature}(q_1) = \#(\mathbf{V}(\langle \mathcal{E} \rangle) \cap \mathbb{R}^n)$

# Separating elements

**Definition 8.** (*Separating element*) Let  $t \in \mathbb{Q}[X_1, \dots, X_n]$ .  $t$  separates  $V(\langle \mathcal{E} \rangle)$  if  $\forall (\alpha, \beta) \in V(\langle \mathcal{E} \rangle)^2, \alpha \neq \beta \Rightarrow t(\alpha) \neq t(\beta)$ .

**Lemma 9.**  $t$  separates  $V(\langle \mathcal{E} \rangle) \Rightarrow \#\mathbf{V}(\text{charpol}(M_t))(\cap \mathbb{R}^n) = \#\mathbf{V}(\langle \mathcal{E} \rangle)(\cap \mathbb{R}^n)$

**Lemma 10.** Almost all polynomials separate  $V(\langle \mathcal{E} \rangle) \subsetneq \mathbb{C}^n$ .

$\Rightarrow$  Probabilistic algorithm for computing  $\#\mathbf{V}(\langle \mathcal{E} \rangle)$  or  $\#\mathbf{V}(\langle \mathcal{E} \rangle) \cap \mathbb{R}^n$

**Lemma 11.**  $\text{charpol}(M_t)$  squarefree  $\Rightarrow t$  separates  $V(\langle \mathcal{E} \rangle)$ .

$\Rightarrow$  Deterministic filter

**Lemma 12.**  $\text{degree}(\text{charpol}(M_t)) = \text{rank}(q_1) \Rightarrow t$  separates  $V(\langle \mathcal{E} \rangle)$ .

$\Rightarrow$  Deterministic algorithm

# Variable's elimination

An easy example :

- compute  $G$  a Gröbner basis of  $\langle \mathcal{E} \rangle$ ;
- compute  $D = \dim_{\mathbb{Q}} \frac{\mathbb{Q}[X_1, \dots, X_n]}{\langle \mathcal{E} \rangle}$  and  $\{w_1, \dots, w_D\}$  from  $G$ ;
- compute  $\vec{1}, \vec{X}_1, \dots, \vec{X}_1^{D-1}$  (normalForm) and check they are linearly independent;

- if so, solve the linear system  $[\vec{1}, \vec{X}_1, \dots, \vec{X}_1^{D-1}] \beta = [\vec{X}_1^D, \vec{X}_2, \dots, \vec{X}_n]$

and get a system equivalent to  $\mathcal{E}$  :

$$\begin{cases} X_1^D - \sum_{i=0}^{D-1} \beta_{1,i+1} X_1^i = 0 \\ X_2 - \sum_{i=0}^{D-1} \beta_{2,i+1} X_1^i = 0 \\ \vdots \\ X_n - \sum_{i=0}^{D-1} \beta_{n,i+1} X_1^i = 0 \end{cases}$$

- **ELSE ??**

This is how works the (famous) FGLM algorithm on such examples

# Remarks on lexicographic Gröbner bases

General shape in the zero-dimensional case for  $<_{\text{lex}}$  :

$$\left\{ \begin{array}{l} f_1(X_1) \\ f_2(X_1, X_2) \\ \vdots \\ f_{k_3-1}(X_1, X_2) \\ f_{k_3}(X_1, X_2, X_3) \\ \vdots \\ f_{k_n}(X_1, \dots, X_n) \\ \vdots \\ f_{k_n+1}(X_1, \dots, X_n) \end{array} \right.$$

Case of ideals in “Shape position ideals for  $<_{\text{lex}}$ ” (degree( $f_1$ ) =  $D$ ) :

$$\left\{ \begin{array}{l} f_1(X_1) \\ X_2 - f_2(X_1, X_2) \\ \vdots \\ X_n - f_n(X_1, \dots, X_n) \end{array} \right.$$

This occurs for example when  $X_1$  is separating  $V(\langle \mathcal{E} \rangle)$  and  $\langle \mathcal{E} \rangle = \sqrt{\langle \mathcal{E} \rangle}$ .

Triangular sets (lexicographic Gröbner bases in the zero-dimensional case) :

$$\left\{ \begin{array}{l} f_1(X_1) = 0 \\ X_2^{n_2} + f_2(X_1) = 0 \\ \vdots \\ X_n^{n_n} + f_n(X_1, \dots, X_{n-1}) = 0 \end{array} \right.$$

# The Rational Univariate Representation (RUR)

**Definition 13.** For  $t \in \mathbb{Q}[X_1, \dots, X_n]$ :

- $f_t = \sum_{i=0}^D a_i T^{D-i} = \text{charpol}(m_t)$ ,  $\tilde{f}_t$  square-free part.
- $\forall v \in \mathbb{Q}[X_1, \dots, X_n]$ ,  $g_{t,v}(T) = \sum_{i=0}^{d-1} \text{Trace}(m_{vt^i}) H_{d-i-1}(\tilde{f}_t, T)$ , where  
 $d = \text{degree}(\tilde{f}_t)$  and  $H_j(\tilde{f}_t, T) = \sum_{i=0}^j a_i T^{j-i}$

**Theorem 14.** If  $t$  separates  $V(\langle \mathcal{E} \rangle)$ , then

$$\begin{array}{ccc} V(\langle \mathcal{E} \rangle)(\cap \mathbb{R}^n) & \approx & V(f_t)(\cap \mathbb{R}) \\ \alpha = (\alpha_1, \dots, \alpha_n) & \rightarrow & t(\alpha) \\ \mathcal{R}_t(\beta) = \left( \frac{g_{t,X_1}(\beta)}{g_{t,1}(\beta)}, \dots, \frac{g_{t,X_n}(\beta)}{g_{t,1}(\beta)} \right) & \leftarrow & \beta \end{array}$$

and :

- $f_t \in \mathbb{Q}[T]$  and  $\forall v \in \mathbb{Q}[X_1, \dots, X_n]$ ,  $g_{t,v} \in \mathbb{Q}[T]$ ;
- $\mu(\alpha) = \mu(t(\alpha)) = \mu(\mathcal{R}_t(t(\alpha))) = \frac{(\widetilde{f_t}') (t(\alpha))}{(\widetilde{f_t})' (t(\alpha))}$

This generalizes the “elimination process” to the case of ideals which are not “shape position”

## The RUR : remarks

For computing RUR, we only need

- a monomial basis  $\mathcal{B} = \{w_1, \dots, w_D\}$  of  $\frac{\mathbb{Q}[X_1, \dots, X_n]}{I}$ ; Suppose that  $w_1 = 1$  and  $\forall i > 1, \exists k, w_i = X_k w_j'$
- the matrices  $M_{X_i}$  of  $m_{X_i}$  wrt  $\mathcal{B}$ ;

This input can be provided by Gröbner bases but also by some new alternatives (B. Mourrain and P. Trebuchet's work).

**From now we suppose we only have these data as input and that the rational numbers in the  $M_{X_i}$  are of binary size  $t$ .**

# Computing a RUR

RUR = separating element  $t$  AND a RUR-Candidate  $\{f_t(T), g_{t,1}(T), g_{t,X_1}(T), \dots, g_{t,X_n}(T)\}$  : if  $t$  separates  $V(I)$ ,

$$\begin{array}{ccc}
 V(\langle \mathcal{E} \rangle)(\cap \mathbb{R}) & \approx & V(f_t)(\cap \mathbb{R}) \\
 \alpha = (\alpha_1, \dots, \alpha_n) & \rightarrow & t(\alpha) \\
 (X_1(\alpha) = \frac{g_{t,X_1}(t(\alpha))}{g_{t,1}(t(\alpha))}, \dots, X_n(\alpha) = \frac{g_{t,X_n}(t(\alpha))}{g_{t,1}(t(\alpha))}) & \leftarrow & t(\alpha)
 \end{array}$$

preserving **multiplicities and real roots**.

Many variants for computing a RUR-Candidate - few solutions for computing a RUR. This difference is critical for “decision” algorithms.

Can be computed as a lexicographic Gröbner basis when  $I$  is radical and  $X_1$  separates  $V(\langle \mathcal{E} \rangle)$  (Faugère, Yokoyama/Noro, ... multi-mod. or p-adic methods).

A major problem is to check that a RUR-Candidate is a RUR or equivalently that  $t$  separates  $V(\langle \mathcal{E} \rangle)$  for non radical ideals.

## From “few” traces

**Theorem 15.** *Given  $q_1[1] = [\text{Trace}(m_{w_1}), \dots, \text{Trace}(m_{w_D})]$  (first line of Hermite’s quadratic form), a **RUR-Candidate** can be computed in  $O(D^3 + nD^2)$  arithmetic operations.*

**Proof.** See algorithms below

□

# The first polynomial

According to Stickelberger's theorem,  $\text{Trace}(m_{ti}) = \sum_{i=1}^D \alpha^i$  is the  $i$ -th Newton's sum of  $f_t = \text{charpol}(m_t) = \sum_{i=0}^D a_i T^{D-i}$ . One can thus deduce the coefficients of  $f_t$  from the scalars  $\text{Trace}(m_{ti}), i = 0 \dots D$  by solving the triangular linear system :  $\left\{ (D - i) a_i = \sum_{j=0}^{i-1} a_{i-j} \text{Trace}(m_{tj}) \right\}_{i=0 \dots D}$

**Algorithm** charpol :

**Input** =  $q_1[1] = [\text{Trace}(m_{w_i})]_{i=1 \dots D}, + \mathcal{B} + M_{X_i}, i = 1 \dots n$

$\vec{y} = [1, 0, \dots, 0]$ ;  $M_t = \sum_{i=1}^D t_i M_{X_i}$ ;

For  $i = 0 \dots D$  do

- $\text{Trace}(m_{ti}) = \vec{y} \cdot q_1[1]$
- $\vec{y} = M_t \cdot \vec{y} /^*$  at the  $i$ -th step  $\vec{y} = t^i$

Solve  $(D - i) a_i = \sum_{j=0}^{i-1} a_{i-j} \text{Trace}(m_{tj})$

$O(D^3)$  operations (note this is a general algorithm for computing the characteristic polynomial of any element in  $\frac{\mathbb{Q}[X_1, \dots, X_n]}{I}$ ).

# The coordinates

Algorithm coordinates :

Input = data from algorithm charpol +  $f_t$

$\tilde{f}_t = \sum_{i=0}^j a_i T^{d-i}$  the squarefree part of  $f_t$

For  $i = 0 \dots d$   $\overrightarrow{H_i(\tilde{f}_t, t)} = \sum_{j=0}^i a_j \underbrace{t^{i-j}}_{\text{already computed}}$

For  $v \in \{1, X_1, \dots, X_n\}$

- $[\text{Trace}(m_{vw_1}), \dots, \text{Trace}(m_{vw_D})] = M_v \cdot q_1[1]^t$
- For  $i = \dots d - 1$

$$\text{Trace}(m_{vH_i(t)}) = \overrightarrow{H_i(t)} \cdot [\text{Trace}(m_{vw_1}), \dots, \text{Trace}(m_{vw_D})]^t$$

$$g_{t,v}(T) = \sum_{i=0}^{d-1} \text{Trace}(m_{vH_i(\tilde{f}_t, t)}) T^{d-i-1} = \sum_{i=0}^{d-1} \text{Trace}(m_{X_j t^i}) H_{d-i-1}(\tilde{f}_t, T)$$

$O(n D^2)$  arithmetic operations

# Computing Hermite's quadratic form

$$q_1 = [\text{Trace}(m_{w_i w_j})]_{i=1 \dots D}^{j=1 \dots D}$$

**Remark :**

- $\text{Trace}(w_i w_j) = \overline{w_i} w_j \cdot q_1[1];$

**Proposition 16.** *Hermite's quadratic form can be computed from  $q_1[1]$  performing  $O(\#\mathcal{T}D)$  arithmetic operations where  $\mathcal{T} = \{w_i w_j, i = 1 \dots D, j = 1 \dots D\}$ .*

# RUR from “few” traces

Algorithm RUR-Classic :

Input =  $q_1[1] = [\text{Trace}(m_{w_i})]_{i=1\dots D}$ , +  $\mathcal{B}$  +  $M_{X_i}, i = 1\dots n$

- $d = \#\mathbf{V}(\langle \mathcal{E} \rangle)$  (Hermite’s quadratic form);  $O(D^3)$
- compute  $f_t = \text{charpol}(m_t)$  for  $t \in \{ \sum_{i=1}^n j^{i-1} X_i, j = 0\dots n \frac{D(D-1)}{2} \}$   
until  $\text{degree}(\tilde{f}_t) = d$ ;  $O(D^3)$  – practical /  $O(n D^5)$  – theoretical
- compute  $g_{t,v}, v = 1, X_1, \dots, X_n$   $O(n D^2)$
- return  $f_t, g_{t,v}, v = 1, X_1, \dots, X_n$

The theoretical complexity is a worst case : all the possible separating elements excepted the last one are not separating.

In practice, I haven’t any example with more than two tries ...

# The “few traces” from a multiplication table

**Definition 17.**  $\mathcal{T} = \{\overrightarrow{w_j w_j}, i = 1 \dots D, j = 1 \dots D\}$

**Remark :**  $\#\mathcal{T} < D^2$

**Examples :**

- from a lexicographic G. basis in "Shape position" :  $\#\mathcal{T} = O(D^2)$
- If  $D = \delta^n$  (Bezout),  $\#\mathcal{T} = O(2^n D^2)$  (ex. when  $\#G = n$ )

**Computing  $\mathcal{T}$**

Computing  $\overrightarrow{w_i w_j}$  :  $\exists i', k, w_i w_j = X_k w_{i'} w_j \Rightarrow \overrightarrow{w_i w_j} = M_{X_k} \overrightarrow{w_{i'} w_j}$

Requires  $O(\#\mathcal{T} D^2)$  arithmetic operations

**Computing  $q_1$  from  $\mathcal{T}$**  : for  $i = 1 \dots D$ ,  $\text{Trace}(m_{w_i}) = \sum_{j=1}^D \overrightarrow{w_i w_j}[j]$   
( $O(D^2)$ )

**Theorem 18.** *Computing a RUR from  $\mathcal{T}$  requires  $O(\#\mathcal{T} D^2 + D^3 + n D^2)$  arithmetic operations.*

# Complexity : choosing the right model

Suppose that  $t$  is the binary size of the integers in  $M_{X_i} = \frac{1}{d_{X_i}} M'_{X_i}$

**The multiplication table** : at most  $\delta = \max(\deg(w_i w_j))$  iterations  $v = M \cdot v$ .

Growth of coefficients :  $t \rightarrow \delta(t + D)$

Bound on the binary cost :  $O(\#TD^2\delta(t + D))$

**Hermite's quadratic form** :

- $\text{Trace}(w_i w_j) = \overline{w_i} w_j \cdot q_1[1]$ ; size  $O(\delta(t + D))$  bin. cost :  $O(\#TD\delta(t + D))$

**Reduction of Hermite's quadratic form** : growth of coefficients  $t' \rightarrow D t'$   
(Rouillier's fraction-free algorithm) binary cost :  $O(D^4\delta(t + D))$

**First polynomial of the RUR** : it is the characteristic polynomial of  $M_t$  thus its coefficients are of size  $O(Dt)$  (see G. Villard's survey).

The intermediate iteration :  $t^{i+1} = M_t t^i$  induces a growth  $t \rightarrow D(t + D)$

(thus the resolution of the triangular system do not induce any growth)

The binary cost is bounded by  $O(D^4(t + D))$ .

**Coordinates** : no significant growth (compared with the computation of the first polynomial)  $\Rightarrow O(n D^3(t + D))$

# Complexity : First remarks

The computation of the multiplication table is not so costly compared to the rest since the data are not growing too much.

The reduction of Hermite's quadratic form is the main operation.

**Algorithm RUR-Classic:**

**Input** =  $q_1[1] = [\text{Trace}(m_{w_i})]_{i=1\dots D}, + \mathcal{B} + M_{X_i}, i = 1\dots n$

- $d = \#\mathbf{V}(\langle \mathcal{E} \rangle)$  (Hermite's quadratic form);  $O(D^3)$
- compute  $f_t = \text{charpol}(m_t)$  for  $t \in \{ \sum_{i=1}^n j^{i-1} X_i, j = 0\dots n \frac{D(D-1)}{2} \}$   
until  $\text{degree}(\tilde{f}_t) = d$ ;  $O(D^3)$  – practical /  $O(n D^5)$  – theoretical
- compute  $g_{t,v}, v = 1, X_1, \dots, X_n$   $O(n D^2)$
- return  $f_t, g_{t,v}, v = 1, X_1, \dots, X_n$

We need to change the strategy !

# Computing faster

## The strategy :

- compute  $d$  and  $t$  using modular arithmetic (modulo a prime number). The result will be correct excepted for a finite number of primes.
- check that the RUR-Candidate is a RUR after the computation to get a deterministic algorithm.

## Advantages :

- fast computations (fixed precision) for “predicting“  $t$
- if the prime numbers are big enough : very few are ”bad”;
- take  $t = X_i$  when possible (smaller results due to the sparsity of  $M_{X_i}$ )
- $t$  may be given by the user.

# Checking a RUR-Candidate

There exists a filter : checking that  $f_t$  is squarefree

For the general case :

**Proposition 19.** *A RUR-Candidate  $\mathcal{R}_t(\langle \mathcal{E} \rangle) = \{f_t, g_{t,1}, g_{t,X_1}, \dots, g_{t,X_n}\}$  is a RUR iff  $\text{Trace}(m_{p_i w_j}) = 0, \forall i = 1 \dots n, j = 1 \dots D$  with  $p_i = X_i g_{t,1}(t) - g_{t,X_i}(t)$ ;*

**Proof.**  $\mathcal{R}_t(\langle \mathcal{E} \rangle)$  is a RUR iff  $p_i(\alpha) = 0, \forall i = 1 \dots n, \forall \alpha \in \mathbf{V}(\langle \mathcal{E} \rangle)$  since  $\#\mathbf{V}(f_t) \leq \mathbf{V}(\langle \mathcal{E} \rangle)$ .

Hermite's quadratic form must be of rank 0:

$$\text{rank}(q_{p_i}) = \text{rank}\left(\left[\text{Trace}(m_{p_i w_j w_k})\right]_{j=1 \dots D}^{k=1 \dots D}\right) = \#\{\alpha \in \mathbf{V}(\langle \mathcal{E} \rangle), p_i(\alpha) \neq 0\}$$

$$\Leftrightarrow \text{its first line is null} \Leftrightarrow \text{Trace}(m_{p_i w_j}) = 0, j = 1 \dots D$$

□

# Checking a RUR-Candidate

**Corollary 20.** A RUR-Candidate  $\mathcal{R}_t(\langle \mathcal{E} \rangle) = \{f_t, g_{t,1}, g_{t,X_1}, \dots, g_{t,X_n}\}$  is a RUR iff  $q_1 \vec{p}_i = 0, \forall i = 1 \dots n, j = 1 \dots D$  with  $p_i = X_i g_{t,1}(t) - g_{t,X_i}(t)$ ;

**Proposition 21.** One can check that a RUR-Candidate is a RUR in  $O(D^3 + n D^2)$  arithmetic operations (if the RUR-Candidate has been computed using RUR-Classic).

**Algorithm** CHECKRUR-CLASSIC

- $\vec{H}_0 = a_0[1, 0, \dots, 0]$ ; for  $i = 1 \dots D$   $\vec{H}_i(t) = M_t \vec{H}_{i-1}(t) + a_i \vec{H}_{i-1}(t)$ ;
- if  $q_1 \vec{H}_D(t) \neq [0, \dots, 0]$  then return(FALSE)
- for  $i = 1 \dots n$   $\vec{g}_{t,X_i}(t) = \sum_{j=1}^D \underbrace{\text{Trace}(X_i t^j)}_{\text{already computed}} \vec{H}_{D-i-1}(t)$
- $\vec{g}_{t,1}(t) = \sum_{j=1}^D \text{Trace}(t^j) \vec{H}_{D-i-1}(t)$
- for  $i = 1 \dots n$  if  $q_1(\vec{g}_{t,X_i}(t) + M_{X_i} \vec{g}_{t,1}(t)) \neq [0, \dots, 0]$  then return(FALSE)
- return(TRUE)

$O(D^3 + n D^2)$  arithmetic operations - no significant growth of coefficients (compared to the other sub-algorithms)

## RUR vs Lexicographic Gröbner bases in the shape position case

When  $X_1$  separates  $V(\langle \mathcal{E} \rangle) + \langle \mathcal{E} \rangle$  is radical

**RUR**

**Lex. G.B.**

$$\left\{ \begin{array}{l} f(X_1) = 0 \\ X_2 = \frac{h_2(X_1)}{f'(X_1)} \\ \vdots \\ X_n = \frac{h_n(X_1)}{f'(X_1)} \end{array} \right. \quad \left\{ \begin{array}{l} f(X_1) = 0 \\ X_2 = f_2(X_1) \\ \vdots \\ X_n = f_n(X_1) \end{array} \right.$$

(same number of arithmetic operations in the “shape position” case)

**Proposition :**  $f'(X_1) X_i - h_i(X_1) = X_i - f_i(X_1) \pmod I$

**Equivalently :**  $f_i = f'^{-1}(X_1) h_n(X_1) \pmod I$

**Remarks :**

all the coefficients of the RUR have the “same” size ( $O(D t)$ )

In a Lex. G. B. the coefficients of  $f$  appear to be “small” ( $O(D t)$ ) compared to those of  $f_i$ .

# Adding inequations/inequalities

$$\mathcal{E} = \{p_1, \dots, p_r\}, \mathcal{F} = \{f_1, \dots, f_l\}, \text{ with } p_i, f_i \in \mathbb{Q}[X_1, \dots, X_n]$$

$$\mathcal{C} = \{x \in \mathbb{C}^n, p_1 = 0, \dots, p_r = 0, f_1 \neq 0, \dots, f_s \neq 0\}$$

$$\mathcal{S} = \{x \in \mathbb{R}^n, p_1 = 0, \dots, p_r = 0, f_1 > 0, \dots, f_s > 0\}$$

## A straightforward method :

Compute  $f_i(\frac{g_{t,X_1}}{g_{t,1}}, \dots, \frac{g_{t,X_n}}{g_{t,1}})$ ,  $i = 1 \dots s$  and study the signs of these univariate polynomials at the roots of  $f_t$ .

**Drawback** : terrible computations (substitutions+reduction modulo  $f_t$ )

## A less straightforward method :

$$\mathcal{E}' = \{p_1, \dots, p_r, T_1 - f_1, \dots, T_s - f_s\} \subset \mathbb{Q}[X_1, \dots, X_n]$$

Compute the RUR of  $\langle \mathcal{E}' \rangle$  :  $\{f_t, g_{t,1}, g_{t,X_1}, \dots, g_{t,X_n}, g_{t,T_1}, \dots, g_{t,T_s}\}$  and study the signs of  $g_{t,1}, g_{t,T_1}, \dots, g_{t,T_s}$  at the roots of  $f_t$  (univariate problem).

**Drawback** : can not study  $f_i, i = 1 \dots s$  without re-computing an ideal

# Adding inequations/inequalities

**A solution** : “simulating” Gröbner bases

**Definition 22.** Given two monomial orderings  $<_U$  (w.r.t. the variables  $U_1, \dots, U_d$ ) and  $<_X$  (w.r.t. the variables  $X_{d+1}, \dots, X_n$ ) one can define a “block“ ordering  $<_{U,X} : m <_{U,X} m'$  if and only if

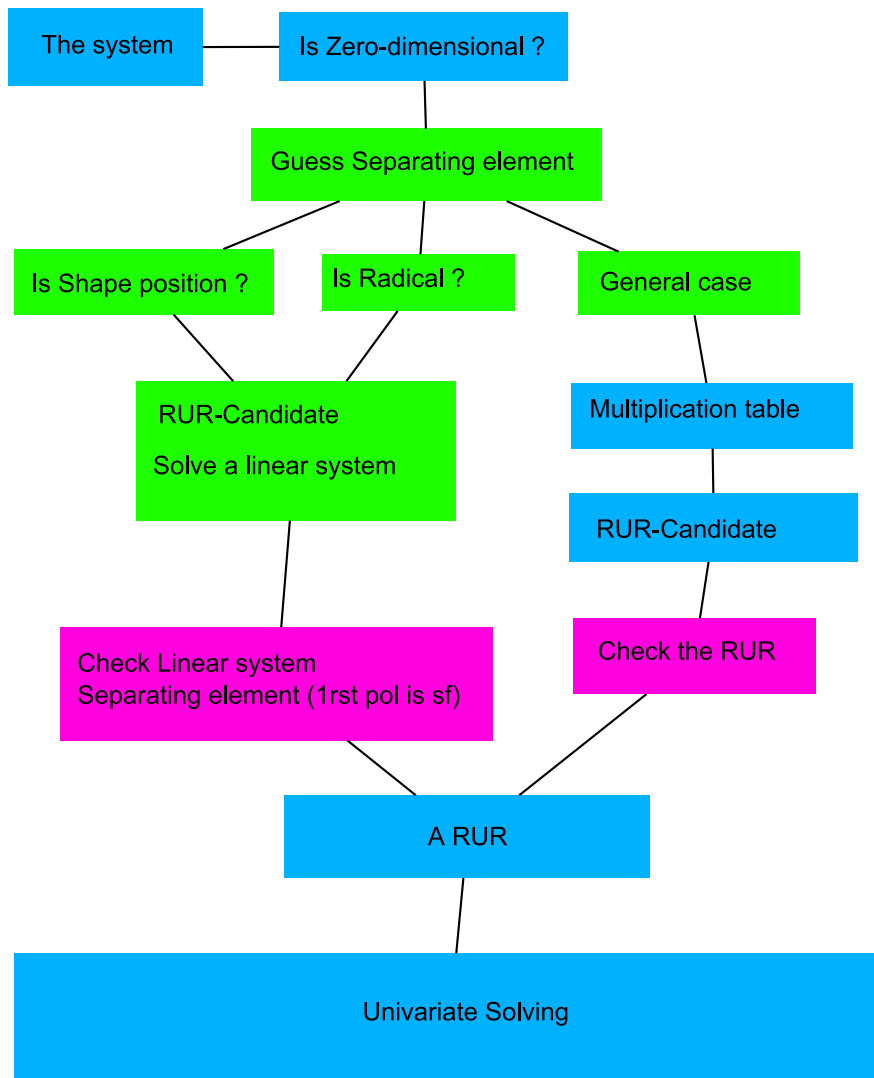
$$m|_{U_1=1, \dots, U_d=1} <_X m'|_{U_1=1, \dots, U_d=1} \text{ or}$$

$$(m|_{U_1=1, \dots, U_d=1} = m'|_{U_1=1, \dots, U_d=1} \text{ and } m|_{X_{d+1}=1, \dots, X_n=1} <_U m'|_{X_{d+1}=1, \dots, X_n=1}).$$

**Lemma 23.** If  $G$  is a Gröbner basis of  $\langle \mathcal{E} \rangle = \langle p_1, \dots, p_r, T_1 - f_1, \dots, T_s - f_s \rangle$  for  $<_X$ , then  $G \cup \{T_i - \text{normalform}(f_i), i = 1 \dots s\}$  is a Gröbner basis of  $\langle \mathcal{E}' \rangle = \langle p_1, \dots, p_r, T_1 - f_1, \dots, T_s - f_s \rangle$  for  $<_{T,X}$  where  $T$  is any admissible monomial ordering wrt  $T = [T_1, \dots, T_n]$ ;

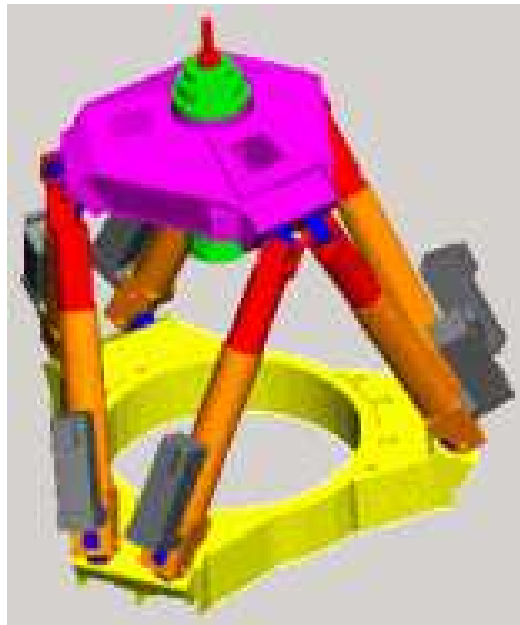
**Remark :**  $\frac{\mathbb{Q}[X_1, \dots, X_n]}{\langle \mathcal{E} \rangle} \cong \frac{\mathbb{Q}[X_1, \dots, X_n, T_1, \dots, T_n]}{\langle \mathcal{E}' \rangle}$  so that the only extraneous computations needed to extend the RUR are the construction of the  $M_{T_i}$  and the computations of the  $\text{Trace}(m_{T_i t^j})$ .

**Complexity** :  $O(\underbrace{D^3 + n D^2}_{\text{RUR}} + s D^2)$



**Figure 1.** A general black box

# Solving the direct kinematics problem for parallel robots



**The challenge** : given the geometry of the robot and the length of the legs, find the possible positions of the robot.

**Applications** : mostly off-line computations like helping to the design of the robot, checking a path planning process, etc.

# Algebraic systems !

- $R_f$ : the base Cartesian reference frame of center  $O$ .
- $R_m$ : the Cartesian reference frame of center  $C$  (end-effector) relative to the mobile platform.
- $A_i$ : location of the center of the joint of kinematics chain  $i$  on the base.
- $B_i$ : location of the center of the joint of kinematics chain  $i$  on the moving platform.
- $L_i$ : overall effective distance between  $B_i$  and  $A_i$ .

An hexapod shall be described by a geometric model where:

- $\overrightarrow{OA}_i|_{R_f}$  describe the base geometry;
- $\overrightarrow{OB}_i|_{R_m}$  describe the mobile platform geometry;
- $L_i$  ,  $i = 1...6$  are the kinematics chains lengths.

## Several classical models

**The goal:** computing  $\overline{OB}_{i|R_f}$  knowing  $L_i^2 = \|A_i B_i\|^2$  ,  $i = 1 \dots 6$ .

Two families of models well suited for computer algebra :

- **Position based equations.** Any rigid body such as the mobile platform can also be spatially located through the position of three of its distinct points such as the first three joints  $B_1, B_2$  and  $B_3$  (supposed to be distinct). Since the body is rigid, the positions of  $B_4, B_5, B_6$  and  $C$  can be deduced from the positions of  $B_1, B_2$  and  $B_3$ .
- **Displacement based equations.** If there exists any mobile platform position  $O B_{|R_f}$  which meets the constraints  $L_i^2 = \|A_i B_i\|^2$  ,  $i = 1 \dots 6$ , then there exists a rotation  $\mathcal{R}$  such that :

$$\overline{OB}_{i|R_f} = \overline{OC}_{|R_f} + \mathcal{R} \cdot \overline{CB}_{i|R_m} \quad , \quad i = 1 \dots 6 \quad (1)$$

# Quaternion based model

$$\text{take } H = \begin{bmatrix} 0 & a & b \\ -a & 0 & c \\ -b & -c & 0 \end{bmatrix} \text{ with}$$

If 1 is not an eigenvalue of  $H$  :

$$\mathcal{R} = \frac{I+H}{I-H} = \begin{bmatrix} -\frac{-1-c^2+a^2+b^2}{1+c^2+a^2+b^2} & 2\frac{a-bc}{1+c^2+a^2+b^2} & 2\frac{ac+b}{1+c^2+a^2+b^2} \\ -2\frac{a+bc}{1+c^2+a^2+b^2} & -\frac{a^2-1-b^2+c^2}{1+c^2+a^2+b^2} & -2\frac{-c+ab}{1+c^2+a^2+b^2} \\ 2\frac{ac-b}{1+c^2+a^2+b^2} & -2\frac{c+ab}{1+c^2+a^2+b^2} & \frac{-b^2-c^2+1+a^2}{1+c^2+a^2+b^2} \end{bmatrix}$$

Choosing  $CB_1 = 0$  and  $OA_1 = 0$ , the constraints are :

$$L_1^2 = \|\overrightarrow{OC}_{|R_f}\|^2$$

$$L_i^2 - L_1^2 = \|\mathcal{R} \cdot \overrightarrow{CB}_{i|R_m} - \overrightarrow{OA}_{i|R_f}\|^2 + 2 \langle \mathcal{R} \cdot \overrightarrow{CB}_{i|R_m} - \overrightarrow{OA}_{i|R_f}, \overrightarrow{OC}_{|R_f} \rangle, i = 2, \dots, 6$$

If 1 is not an eigenvalue of  $H \Leftrightarrow$  the system has no “roots at infinity”  
(easy to detect on a Gröbner basis)

# Quaternion based model

If the system has “roots at infinity”,  $\mathcal{R}$  can be replaced by :

$$R_{\infty} = \begin{bmatrix} \frac{c^2 - a^2 - b^2}{c^2 + a^2 + b^2} & -2 \frac{bc}{c^2 + a^2 + b^2} & 2 \frac{ac}{c^2 + a^2 + b^2} \\ -2 \frac{bc}{c^2 + a^2 + b^2} & -\frac{a^2 - b^2 + c^2}{c^2 + a^2 + b^2} & -2 \frac{ab}{c^2 + a^2 + b^2} \\ 2 \frac{ac}{c^2 + a^2 + b^2} & -2 \frac{ab}{c^2 + a^2 + b^2} & -\frac{b^2 + c^2 - a^2}{c^2 + a^2 + b^2} \end{bmatrix}$$

This leads to an easy to solve new system. Let's focus on the generic case :

It depends on 6 variables :  $a, b, c$  and  $t_1, t_2, t_3$  but contains 5 equations which are linear wrt  $t_1, t_2, t_3$ .

**The trick** : compute a Gröbner basis of  $\langle \mathcal{E} \rangle \cap \mathbb{Q}(a, b, c)$

Algorithm F4 (Faugère) can detect early in the algorithm that in a lexicographic Gröbner basis with  $t_i > a, b, c$ , the variables  $t_i$  will, **in general** be linearly expressed wrt  $a, b, c$ .

It can exploit the relations involving these variables without computing the full (huge) Gröbner basis of the system but can return the “small” G.B. of  $\langle \mathcal{E} \rangle \cap \mathbb{Q}(a, b, c)$

# The algorithm

Compute a RUR of  $\langle \mathcal{E} \rangle \cap \mathbb{Q}(a, b, c)$  :

- The systems are in general in shape position : this is automatically detected

Check that the linear systems in  $t_1, t_2, t_3$  are invertible and solve them :

- Compute the sign of some polynomials (minors) at the roots of the system

If not : split the system (adding minors in  $t_1, t_2, t_3$ ) :

- study  $\mathcal{E} \cup (T^{p-1})$  and eliminate  $T$  to get the solutions (if any) where  $p \neq 0$ .

Numerical certified approximations are obtained using multi-precision interval arithmetic (MPFI)

# Software

<http://fgbrs.lip6.fr/Software>

SALSA library containing

- FGb (F4 algorithm implemented by J.C. Faugère) - Dynamic library written in C
- RS (RUR + real root isolation by F. Rouillier) - Dynamic library written in C
- An interface with Maple Software (version >9.5)
- ... (see next lecture).

Will be linked and distributed with Maple 11 (official distribution)

# Experiments

Example	GB+RUR	ISO(32)	ISO(64)	Comment
dietmaier_16	1.0	0.20	0.28	40 complex roots / 38 real roots
dietmaier_18	1.1	0.21	0.32	40 real roots
dietmaier_24	1.4	0.25	0.36	"
dietmaier_30	1.8	0.28	0.40	"
dietmaier_40	2.0	0.31	0.43	"
dietmaier_60	2.2	0.32	0.46	"
dietmaier_120	2.9	0.40	0.56	"
Spat66_quat	0.7	0.00	0.01	40 complex roots / 0 real roots
LeftHand $\mathcal{R}$	0.4	0.12	0.16	30 complex roots / 8 real roots
LeftHand $\mathcal{R}_\infty$	0.04	0.00	0.01	6 complex roots / 0 real roots
LeftHand $+\varepsilon$	0.9	0.10	0.12	40 complex roots / 8 real roots

Spat66 : using classical Gröbner bases (Buchberger) and position based model .... 20 minutes !

# Conclusion

Solving applications with computer algebra claims expertise to find the right strategy and the right model.

When it works, the advantages are many : universal methods, certification of the results. It comes as a complement to numerical methods, mostly for off-line studies.

In particular computer algebra tools are not well suited for solving systems set for numerical methods.

Providing a fast general algorithm is a balance between :

- adapted computable mathematical objects
- facility of implementation
- careful study of arithmetic/binary/practical complexity

I hope that the IMA workshop will help to transfer some knowledge for promoting algebraic computations !