INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

On solving the direct kinematics problem for parallel robots

Jean-Charles Faugère — Jean-Pierre Merlet — Fabrice Rouillier

N° 5923

May 2006

Thème SYM

Rapport de recherche

ISSN 0249-6399 ISRN INRIA/RR--5923--FR+ENG

On solving the direct kinematics problem for parallel robots

Jean-Charles Faugère* , Jean-Pierre Merlet† , Fabrice Rouillier‡

Thème SYM — Systèmes symboliques
Projets COPRIN & SALSA

Rapport de recherche n° 5923 — May 2006 — 21 pages

Abstract: In this paper, we propose efficient methods for solving the direct kinematics problem (DKP) for parallel manipulators. By solving, we mean computing all the real solutions in a *certified* way, without any assumption on the manipulator. For example, the precision and the real character of the solutions are guaranteed. The proposed algorithms use as black boxes several recent algorithms from computer algebra such as F4/F5 for computing Gröbner bases, recent methods for isolating the real roots by mean of intervals with rational bounds which are mixed with strategies coming from the interval arithmetics field. The resulting solutions are efficient for such kind of output since the running time never exceeds few seconds (in fact about 1 second for non extreme but general examples) thanks to the algorithms but also to a well adapted translation of DKP problem into algebraic equations. As an example we give a new computational proof of the existence of a parallel robot with 40 real roots.

Key-words: Direct Kinematics Problem, Parallel Robot, Polynomial System Solving, Computer Algebra, Real Roots.

* Laboratoire d'Informatique de Paris VI, SALSA project
† INRIA Sophia-Antipolis, COPRIN project
‡ INRIA Rocquencourt & LIP6, SALSA project

Unité de recherche INRIA Rocquencourt
Domaine de Voluceau, Rocquencourt, BP 105, 78153 Le Chesnay Cedex (France)
Téléphone : +33 1 39 63 55 11 —Télécopie : +33 1 39 63 53 30

Sur la résolution du problème géométrique direct pour les robots parallèles

Résumé : Nous proposons une méthode efficace pour résoudre le problème géométrique direct (DKP) pour les robots parallèles. Le terme résoudre veut dire ici calculer toutes les solutions réelles de façon certifiée sans aucune supposition sur le robot. Par exemple, la précision ou le caractère réel des solutions sont garantis. Les méthodes proposées utilisent différents algorithmes de calcul formel récents tels que F4/F5 pour le calcul de bases de Gröbner ou pour l'isolation des racines réelles de polynômes par des intervalles à bornes rationnelles qui sont associés avec d'autres stratégies utilisant de l'arithmétique par intervalles. L'algorithme final est efficace puisque les temps de calcul sont voisins de la seconde sur des exemples généraux, grâce, en particulier, à une formulation adaptée du DKP. Comme exemple, nous donnons une preuve calculatoire de l'existence d'un robot parallèle avec 40 positions.

Mots-clés : Problème géométrique direct, Robot parallèle, Résolution de systèmes polynomiaux, Calcul Formel, Racines réelles

Contents

1 Introduction and notations

Most industrial robots have a serial mechanical architecture i.e. starting from the fixed base each link is connected through an actuated joint to the next link in the chain. Such structure provides large workspace but is not appropriate in terms of accuracy and payload. For applications requiring a good positioning accuracy and/or large payload it has been suggested the use of closed-loop mechanical structure in which more than one kinematic chains connect the ground and the end-effector. A typical example of such structure is the Gough platform [12], figure 1.

In this mechanism, the end-effector is connected to the ground by 6 articulated legs. The extremities A_i, B_i of the legs are connected to the ground and to the end-effector by spherical joints, allowing free rotations around the joint centers. In each leg is a linear actuator that allows to change the leg lengths and a sensor that allow to measure the leg length. By changing the 6 leg lengths it is possible to control the position/orientation (called a *pose*) of the end-effector with respect to the ground. For controlling such robot it is essential to solve 2 *kinematic* problems: the *inverse kinematic problem* (IKP) and the *direct kinematic problem* (DKP). For the IKP the position/orientation of the platform, defined by a set of parameters $\mathbf{X}$ is known and we must determine the corresponding leg lengths ρ: a simple analysis allows to obtain an analytical relation $\rho = F(\mathbf{X})$. More generally the IKP is to determine the actuated joint variables as a function of the end-effector pose parameters. The DKP is the opposite problem: being given the leg lengths it is necessary to determine

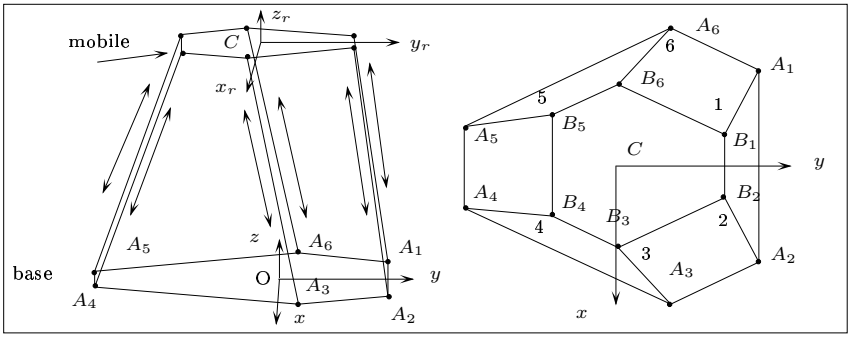

Figure 1: The Gough platform

the current position/orientation of the platform. Determining the position/orientation is essential for the use of the robot as any control law will use this information. Two types of DKP can be considered:

1. *off-line*: when starting the robot it is necessary to determine the start pose of the robot with as only available information the leg lengths as measured by the sensors. Computation time is not critical

2. *real-time*: it will be used for control purpose while the robot is moving and as soon as measurements are available. But here we may assume that the current pose is very close to the one obtained at the previous calculation

In this paper we address mainly the off-line DKP (real-time DKP may be solved in a safe way by other methods [18]) but our purpose is to design a solving algorithm that is numerically robust, provide *certified* solutions and is reasonably fast for off-line studies. By certified solutions we mean that no spurious one will be found (in particular we compute only the real solutions) and that we will be able to refine them up to any arbitrary accuracy. Certification is a key issue for the DKP as any error will leads to a failure in the control of the robot that may have catastrophic consequences. This rules out some solving methods such as elimination whose last numerical step cannot be certified. Indeed another use of the DKP is to simulate the real robot which differs from its theoretical model (e.g. manufacturing tolerances induce uncertainties in the location of the A_i, B_i points and the control law will exhibit positioning errors when the robot follows a nominal trajectory).

Methods from computer algebra have already been used for several purpose in robotics, and especially for solving/studying the DKP. For example, in [14] or [29], a univariate polynomial with as roots as the DKP is explicitly computed. But one can notice that those computer algebra based methods are mainly designed for getting qualitative results (mainly the number of solutions). One goal of this article is to promote such methods by showing

that recent progress make now possible their use for solving the DKP efficiently (few seconds in the worth case) and for any robot (no assumption on the geometry).

In the first section of the present contribution, we propose some general strategies for computing efficiently the real roots of algebraic systems of equations through Gröbner bases.

In the second section, we show how to model the DKP in order to speed up significantly the computations using the algorithms of the first section.

The last section is devoted to experiments on various class of robots. We illustrate the power of the method by computing and comparing the positions of a theoretical planar robot (left-hand) with those of an effective one (the same after small perturbations), studying the behavior of the method with respect to the precision of the input data. As a second example, we prove algorithmically that Dietmaier's robot [6] has 40 real roots: by using Computer Algebra methods we can certify that all the solutions are real numbers (and not only complex numbers with small imaginary parts).

2 Gröbner based Real Solving

This section briefly summarizes some basic results about Gröbner bases and their applications to solving zero-dimensional systems (systems with a finite number of complex roots). The reader interested in a complete introduction of Gröbner bases and the required background is referred to [5],[2].

2.1 Notations and basic definitions

Let denote by $\mathbb{Q}[X_1, \ldots, X_n]$ the ring of polynomials with rational coefficients and unknowns $X_1, \ldots, X_n$ and $S = \{P_1, \ldots, P_s\}$ any subset of $\mathbb{Q}[X_1, \ldots, X_n]$. A point $x \in \mathbb{C}^n$ is a zero of S if $P_i(x) = 0 \quad \forall i = 1 \ldots s$. 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 $U_k \in \mathbb{Q}[X_1, \ldots, X_n]$. Since every element of $\mathcal{I}$ vanishes at each zero of S, we denote by $V_{\mathbb{C}}(S) = V_{\mathbb{C}}(I) = \{x \in \mathbb{C}^n \mid p(x) = 0 \ \ \forall p \in I\}$ (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.

2.2 Gröbner bases

A Gröbner basis of $\mathcal{I}$ is a computable generator set of $\mathcal{I}$ with good algorithmical properties (as described below) and defined with respect to a monomial ordering. In this paper, one will use the following orderings:

- *lexicographic order* : (Lex)

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

- *degree reverse lexicographic order (DRL)* :

$$X_1^{\alpha_1}\cdot\ldots\cdot X_n^{\alpha_n} <_{DRL} X_1^{\beta_1}\cdot\ldots\cdot X_n^{\beta_N} \Leftrightarrow X_0^{\sum_k \alpha_k} X_1^{-\alpha_n}\cdot\ldots\cdot X_n^{-\alpha_1} <_{Lex} X_0^{\sum_k \beta_k} X_1^{-\beta_n}\cdot\ldots\cdot X_n^{-\beta_1} \quad (2)$$

Gröbner bases are mathematical objects with good algorithmical properties : the main ones are summarized in the following.

Definition 1 *For any n-uple $\mu = (\mu_1, \ldots, \mu_n) \in \mathbb{N}^n$, let denote by X^μ the monomial $X_1^{\mu_1} \cdot \ldots \cdot X_n^{\mu_n}$. If $<$ is an admissible monomial ordering and $P = \sum_{i=0}^{r} a_i X^{\mu^{(i)}}$ any polynomial in $\mathbb{Q}[X_1, \ldots, X_n]$, we define:* $\mathrm{LM}(P, <) = \max_{i=0\ldots r} \ , \ < X^{\mu^{(i)}}$ *(leading monomial of P w.r.t. $<$)*

Definition 2 *A set of polynomials G is a Gröbner basis of an ideal $\mathcal{I}$ wrt to a monomial ordering $<$ if for all $f \in I$ there exists $g \in G$ such that LM(g) divides LM(f).*

Given any admissible monomial ordering $<$ one can easily extend the classical Euclidean division to *reduce* a polynomial p by another one or, more generally, by a set of polynomials F, performing the reduction wrt to each polynomial of F until getting an expression which can not be reduced anymore. We denote this function by `Reduce`$(p, F, <)$ (reduction of the polynomial p wrt$F$). Unlike in the univariate case, the result of such a process is not canonical except if $F = G$ is a Gröbner basis:

Theorem 1 *Let G be a Gröbner basis of an ideal $\mathcal{I} \subset \mathbb{Q}[X_1, \ldots, X_n]$ for a fixed ordering $<$.*

(i) a polynomial $p \in \mathbb{Q}[X_1, \ldots, X_n]$ belongs to $\mathcal{I}$ if and only if Reduce$(p, G, <) = 0$,

(ii) Reduce(p,G,<) does not depend on the order of the polynomials in the list G, thus, this is a canonical reduced expression modulus $\mathcal{I}$.

Gröbner bases are computable objects. The historical method for computing them is Buchberger's algorithm ([4, 3]). It has several variants and it is implemented in most general computer algebra systems like Maple or Mathematica. Recently, more efficient algorithms have been proposed to compute Gröbner bases:

- the F_4 algorithm [8] is based on the intensive use of linear algebra methods: in short, the arbitrary choices (S-polynomials) are left to computational strategies related to classical linear algebra problems (mainly the computation of row echelon form), the polynomials being represented as vectors in the monomial basis.

- In [11] a new criterion (the F_5 criterion) for detecting useless computations (most of the S-polynomials to be computed using the classical Buchberger algorithm reduce to 0) has been given; under some regularity conditions on the system, it is proved that the algorithm do never perform useless computations. A new algorithm named F_5 has been built using these two ideas: the F_5 algorithm constructs incrementally the following matrices in degree d:

$$A_d = \begin{array}{c} \\ t_1 f_1 \\ t_2 f_2 \\ t_3 f_3 \\ \dots \end{array} \begin{array}{c} \begin{array}{cccc} m_1 > & m_2 > & m_3 & \dots \end{array} \\ \begin{bmatrix} \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots \end{bmatrix} \end{array}$$

where the indices of the columns are monomials sorted for the admissible ordering $<$and the rows are product of some polynomials f_i by some monomials t_j such that $\deg(t_j f_i) \leq d$. For a regular system the matrices, $A_d$are full rank. In a second step, row echelon forms of the matrices are computed:

$$A'_d = \begin{array}{c} \\ t_1 f_1 \\ t_2 f_2 \\ t_3 f_3 \\ \dots \end{array} \begin{array}{c} \begin{array}{cccc} m_1 & m_2 & m_3 & \dots \end{array} \\ \begin{bmatrix} 1 & 0 & 0 & \dots \\ 0 & 1 & 0 & \dots \\ 0 & 0 & 1 & \dots \\ 0 & 0 & 0 & \dots \end{bmatrix} \end{array}$$

Even if F_5 still computes the same mathematical object (a Gröbner basis), the gap with existing other algorithms is consequent. In particular, due to the range of examples which become computable, Gröbner basis can be considered as a reasonable computable object in large applications. Important parameters to evaluate the complexity of Gröbner bases with the F_5 are the D the maximal degree d occuring in the computation and the size of the matrix A_d. The overall cost is thus dominated by $(\#A_d)^3$.

We pay a particular attention to Gröbner bases computed for elimination orderings since they provide a way to *simplify* the system (an equivalent system with a structured shape). For example, a lexicographic Gröbner basis of zero dimensional system (when the number of complex solutions is finite) has always the following shape (if we suppose that $X_1 < X_2 \dots < X_n$):

$$\begin{cases} f(X_1) = 0 \\ f_2(X_1, X_2) = 0 \\ \vdots \\ f_{k_2}(X_1, X_2) = 0 \\ f_{k_2+1}(X_1, X_2, X_3) = 0 \\ \vdots \\ f_{k_{n-1}+1}(X_1, \dots, X_n) = 0 \\ \vdots \\ f_{k_n}(X_1, \dots, X_n) = 0 \end{cases}.$$

(depending on some geometrical properties of the solutions, some of the polynomials may be identically null). A well known property is that the zeros of the smallest (w.r.t. $<$) non null polynomial define the Zariski closure (classical closure in the case of complex coefficients) of the projection on the coordinate's space associated with the smallest variables.

More generally, an admissible ordering $<$ on the monomials depending on variables $[X_1, \ldots, X_n]$ is an ordering which eliminates $X_{d+1}, \ldots, X_n$ if $X_i < X_j \ \forall i = 1 \ldots d, j = d + 1 \ldots n$. The lexicographic ordering is a particular elimination ordering. Given two monomial orderings $<_1$(w.r.t. the variables $X_1, \ldots, X_d$) and $<_2$(w.r.t. the variables $X_{d+1}, \ldots, X_n$) one can define an ordering which eliminates $X_{d+1}, \ldots, X_n$ by setting the so called block ordering $<_{1,2}$as follows : given two monomials m and m', $m <_{1,2} m'$ if and only if $m_{|_{X_1=1,\ldots,X_n=1}} <_2 m'_{|_{X_1=1,\ldots,X_n=1}}$ or $(m_{|_{X_1=1,\ldots,X_n=1}} = m'_{|_{X_1=1,\ldots,X_n=1}}$ and $m <_1 m')$.

2.3 Zero-dimensional systems

Zero-dimensional systems are polynomial systems with a finite number of complex solutions. This specific case is fundamental for many engineering applications including parallel manipulator DKPs which are known to have, at most, 40 complex solutions in non degenerated cases ([16, 19, 21]). The following known theorem shows that we can detect easily that a system is zero dimensional or not by computing a Gröbner base for any monomial ordering :

Theorem 2 *Let $G = \{g_1, \ldots, g_l\}$ be a Gröbner basis for any ordering $<$ of any system $S = \{P_1, \ldots, P_s\} \in \mathbb{Q}[X_1, \ldots, X_n]^s$. The two following properties are equivalent :*

- *For all index i, $i = 1 \ldots n$, there exists a polynomial $g_j \in G$ and a positive integer n_j such that $X_i^{n_j} = LM(g_j, <)$;*
- *The system $\{P_1 = 0, \ldots, P_s = 0\}$ has a finite number of solutions in $\mathbb{C}^n$.*

If S is zero-dimensional, then, according to theorem 2, only a finite number of monomials $m \in \mathbb{Q}[X_1, \ldots, X_n]$ are not reducible modulo G, meaning that Reduce(m,G,$<$)$= m$. Mathematically, a system is zero-dimensional if and only if $\mathbb{Q}[X_1, \ldots, X_n]/I$ is a $\mathbb{Q}$-vector space of finite dimension, I being the ideal generated by the system. This vector space can fully be characterized when knowing a Gröbner basis:

Theorem 3 *Let $S = \{p_1, \ldots, p_s\}$ be a set of polynomials with $p_i \in \mathbb{Q}[X_1, \ldots, X_n] \quad ; \forall i = 1 \ldots s$, and suppose that G is a Gröbner basis of $I = \langle S \rangle$ with respect to any monomial ordering $<$. Then :*

- $\mathbb{Q}[X_1, \ldots, X_n]/I = \{\text{Reduce}(p, G, <) \quad p \in I\}$ *is a vector space of finite dimension;*
- $\mathcal{B} = \{t = X_1^{e_1} \cdot X_n^{e_n} \quad , \quad (e_1, \ldots, e_n) \in \mathbb{N}^n | \quad \text{Reduce}(t, G, <) = t\} = \{w_1, \ldots, w_D\}$ *is a basis of $\mathbb{Q}[X_1, \ldots, X_n]/I$ as a $\mathbb{Q}$-vector space;*
- $D = \sharp\mathcal{B}$ *is exactly the number of elements of complex zeroes of the system $\{P = 0, \quad \forall P \in S\}$ counted with multiplicities.*

Thus, in the zero-dimensional case, once a Gröbner basis is known, a basis of $\mathbb{Q}[X_1, \ldots, X_n]/I$ can easily be computed (Theorem 3) so that linear algebra methods can be applied for doing several computations and getting informations about the roots.

For any polynomial $q \in \mathbb{Q}[X_1, \ldots, X_n]$ the decomposition $\overline{q}$ =Reduce(q,G,$<$)=$\sum_{i=1}^{D} a_i w_i$ is unique (Theorem 1) and we denote by $\vec{q} = [a_1, \ldots, a_D]$ the representation of $\overline{q}$ in the basis $\mathcal{B}$. For example, the matrix w.r.t. $\mathcal{B}$ of the linear map

$$m_{q:} \left(\begin{array}{ccc} \mathbb{Q}[X_1, \ldots, X_n]/I & \longrightarrow & \mathbb{Q}[X_1, \ldots, X_n]/I \\ \vec{p} & \longmapsto & \overrightarrow{pq} \end{array} \right)$$

can explicitly be computed (its columns are the vectors $\overrightarrow{qw_i}$) and one can then apply the following well-known theorem:

Theorem 4 *(Stickelberger) The eigenvalues of m_q are exactly the $q(\alpha)$ where $\alpha \in V_{\mathbb{C}}(S)$.*

According to Theorem 4, the i-th coordinate of all $\alpha \in V_{\mathbb{C}}(S)$ can be obtained from M_{X_i} eigenvalues but the issue of finding all the coordinates of all the $\alpha \in V_{\mathbb{C}}(S)$ from $M_{X_1}, \ldots, M_{X_n}$ eigenvalues is not explicit nor straightforward (see [1] for example). Note also that some authors propose algorithms to compute numerically the matrices $M_{X_1}, \ldots, M_{X_n}$ without computing Gröbner bases (see [20]). Up to our experiments, such computations are not numerically stable for general manipulators and it may be preferable to compute, for example, the characteristic polynomial of the matrix M_{X_i} and then isolate its real roots to obtain all the possible i-th coordinates of the solutions : we prefer to follow with exact computations a little bit more, providing exact formulas as explained in the next section.

2.4 The Rational Univariate Representation

The Rational Univariate Representation [26] is, with the end-user point of view, a simple way for representing symbolically the roots of a zero-dimensional system without loosing information (multiplicities or real roots) since one can get all the information on the roots of the system by solving exclusively univariate polynomials.

Given a zero-dimensional system $I =< p_1, \ldots, p_s >$ where the $p_i \in \mathbb{Q}[X_1, \ldots, X_n]$, a Rational Univariate Representation of $\boldsymbol{V}(I)$ has the following shape : $f_t(T) = 0, X_1 = \frac{g_{t,X_1}(T)}{g_{t,1}(T)}, \ldots, X_n = \frac{g_{t,X_n}(T)}{g_{t,1}(T)}$, where $f_t, g_{t,1}, g_{t,X_1}, \ldots, g_{t,X_n} \in \mathbb{Q}[T]$ (T is a new variable). It is uniquely defined w.r.t. a given polynomial t which separates $V(I)$ (injective on $V(I)$), the polynomial f_t being necessarily the characteristic polynomial of m_t (see above section) in $\mathbb{Q}[X_1, \ldots, X_n]/I$ [26]. The RUR defines a bijection between the roots of $\mathcal{F}$ and those of f_t preserving the multiplicities and the real roots :

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

For computing a RUR one has to solve two problems :

- finding a separating element t;

- given any polynomial t, compute a RUR-Candidate $f_t, g_{t,1}, g_{t,X_1}, \ldots, g_{t,X_n}$ such that if t is a separating polynomial, then the RUR-Candidate is a RUR.

According to [26], a RUR-Candidate can explicitly be computed when knowing a suitable representation of $\mathbb{Q}[X_1, \ldots, X_n]/I$:

- $f_t = \sum_{i=0}^{D} a_i T^i$ is the characteristic polynomial of m_t. Lets denotes by $\overline{f_t}$ its square-free part.
- for any $v \in \mathbb{Q}[X_1, \ldots, X_n]$, $g_{t,v}(T) = \sum_{i=0}^{d-1} Trace(m_{vt^i}) H_{d-i-1}(T)$, $d = \deg(\overline{f_t})$ and $H_j(T) = \sum_{i=0}^{j} a_i T^{i-j}$.

In [26], a strategy is proposed for computing a RUR for any system (a RUR-Candidate and a separating element), but there are special cases where it can be computed differently. When X_1 is separating $\boldsymbol{V}(I)$ and when I is a radical ideal, the system is said to be in *shape position*. In such cases, the shape of the lexicographic Gröbner basis is always the following :

$$\begin{cases} f(X_1) = 0 \\ X_2 = f_2(X_1) \\ \vdots \\ X_n = f_n(X_1) \end{cases} . \tag{3}$$

As shown in [26], if the system is in shape position, $g_{X_1,1} = f'_{X_1}$ and we have $f_{X_1} = f$ and $f_i(X_1) = g_{X_1,X_i}(X_1) g_{X_1,1}(X_1) \bmod f$. Thus the RUR associated with X_1 and the lexicographic Gröbner basis are equivalent up to the inversion of $g_{X_1,1} = f'_{X_1}$ modulo f_{X_1}. In the rest of the paper we call this object a RR-Form of the corresponding lexicographic Gröbner basis. The RUR is well known to be much smaller than the lexicographic Gröbner basis in general (this may be explained by the inversion of the denominator) and thus will be our priviligied symbolic output. Note that it is easy to check that a system is in shape position once knowing a RUR-Candidate (and so to check that X_1 separates $\boldsymbol{V}(I)$): it is necessary and sufficient that f_{X_1} is square-free.

These results have many practical drawbacks since, for most robots, the systems which model the DKP are in shape position. We thus can multiply the strategies for computing a symbolic solution : one can compute the RR-Form Gröbner directly using [8] or [11] for example or by change of ordering like in [9] or a RUR using the algorithm from [26]. Choosing the right strategy will be part of our experimental section.

2.5 From formal to numerical solutions

Computing a RUR reduces the resolution of a zero-dimensional system to solving one polynomial in one variable (f_t) and to evaluating n rational fractions ($\frac{g_{t,X_i}(T)}{g_{t,1}(T)}, i = 1 \ldots n$) at its roots (note that if one simply wants to compute the number of real roots of the system, there is no need to consider the rational coordinates). The next task is thus is to compute all

the real roots of the system (and only the real roots), providing a numerical approximation with an arbitrary precision (set by the user) of the coordinates.

The isolation of the real roots of f_t can be done using the algorithm proposed in [27] : the output will be a list l_{f_t} of intervals with rational bounds such that for each real root α of f_t, there exists a unique interval in l_{f_t} which contains α. The second step consists in refining each interval in order to ensure that it does not contain any real root of $g_{t,1}$. Since f_t and $g_{t,1}$ are co-prime this computation is easy and we then can ensure that the rational functions can be evaluated using interval arithmetics without any cancellation of the denominator. This last evaluation is performed using multi-precision arithmetics (MPFI package - [22]). As we will see in the experiments, the precision needed for the computations is poor and, moreover, the rational functions defined by the RUR are stable under numerical evaluation, even if their coefficients are huge (rational numbers), and thus this part of the computation is still efficient. For increasing the precision of the result, it is only necessary to decrease the length of the intervals in l_{f_t} which can easily be done by bisection or using a certified Newton's algorithm. Note that is is quite simple to certify the sign of the coordinates : one simply has to compute some gcds (gcds of the numerators and of f_t) and split, when necessary the RUR.

3 Algebraic modelisation

The computation of a Gröbner basis is very sensitive to the algebraic equations used to model the DKP. In this section, we show how to reduce significantly the computing time by modifying the standard algebraic equations describing the DKP.

3.1 Standard algebraic equations.

Lets introduce the nomenclature used in this part :

- 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.

Lets denote by $\vec{v}_{|R}$ (resp. $M_{|R}$) the coordinates of the vector $\vec{v}$ (resp. the point M) with respect to the Cartesian reference frame R, and by $\mathbf{NM}_{|R}$ the matrix which columns are $\overrightarrow{M_iN_i}_{|R}, \quad i = 1 \ldots 6$.

An hexapod shall be described by a geometric model where:

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

According to these notations, solving the *DKP* problem consists in computing $\overrightarrow{\mathrm{OB}}_{i|_{R_f}}$ with respect to the constraints : $L_i^2 = ||A_iB_i||^2 \quad , \quad i = 1 \ldots 6$. Among the numerous existing *DKP* formulations, there exists two main families which are commonly used in 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 $\mathbf{OB}_{|_{R_f}}$ which meets the constrains $L_i^2 = ||A_iB_i||^2 \quad , \quad i = 1 \ldots 6$, then there exists a rotation $\mathcal{R}$ such that :
$$\overrightarrow{\mathrm{OB}}_{i|_{R_f}} = \overrightarrow{\mathrm{OC}}_{|_{R_f}} + \mathcal{R} \cdot \overrightarrow{\mathrm{CB}}_{i|_{R_m}} \quad , \quad i = 1 \ldots 6 \tag{4}$$

Position based equations have already been used by several authors and we describe first the ones from [15], which are already optimized for Gröbner bases computations. The systems consists in 9 equations depending on 9 variables and will be denoted by $\mathcal{S}_{3\,\mathrm{pt}}$.

As used by several authors, the natural way to set an algebraic equation system from (4) is to straightforwardly use the rotation matrix parameters and the vector $\overrightarrow{\mathrm{OC}}_{|_{R_f}}$ coordinates as unknowns. Since $\mathcal{R}$ is a rotation (a 3×3 matrix), the following relations hold :

$$\mathcal{R}^t\mathcal{R} = I_3, \det(\mathcal{R}) = 1 \tag{5}$$

This first set of equations depend neither on the geometry of the platform, nor on the length of the legs and will be denoted by $\mathcal{S}_{\mathrm{dis,rot}}$.

The constraints $L_i^2 = ||A_iB_i||^2 \quad , \quad i = 1 \ldots 6$ can then be expressed accordingly to equations (4):

$$L_i^2 = ||\overrightarrow{\mathrm{OB}}_{i|_{R_f}} - \overrightarrow{\mathrm{OA}}_{i|_{R_f}}||^2 = ||\overrightarrow{\mathrm{OC}}_{|_{R_f}} + \mathcal{R} \cdot \overrightarrow{\mathrm{CB}}_{i|_{R_m}} - \overrightarrow{\mathrm{OA}}_{i|_{R_f}}||^2. \tag{6}$$

Let denote by $\mathcal{S}_{\mathrm{dis,geom}}$ the resulting system. The full modelisation of the displacement leads to an algebraic system $\mathcal{S}_{\mathrm{rot}=}\mathcal{S}_{\mathrm{dis,rot}} \cup \mathcal{S}_{\mathrm{dis,geom}}$ made of 13 polynomial equations depending on 12 unknowns ($[X, Y, Z, r_{ij}, j = 1 \ldots 3, i = 1 \ldots 3]$), where $[X, Y, Z]$ are the coordinates of $\overrightarrow{\mathrm{OC}}_{|_{R_f}}$. Note that a useful trick to speed up the final computation is to precompute a Gröbner basis of $\mathcal{S}_{\mathrm{dis,rot}}$ and to add to the previous computation all the equations coming from (3.1).

3.2 Quaternions

A standard way for describing the orientation of a parallel robot is to use quaternions ([13],[23],[7], etc.). We propose to exploit a little bit more some blind properties of such a modelization together with specific behavior of modern algorithms for computing Gröbner bases. In short, we extract variables which are "intrinsincly" linear early in the algorithm to decrease the number of variables with a significant influence on the computation times.

3.2.1 The generic case

To parameterize the group of rotations we can use the Cayley transform; if H is any anti-symmetric matrix:

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

then, provided that 1 is not an eigenvalue of H, we have that

$$\mathcal{R} = \frac{I+H}{I-H} = \begin{bmatrix} -\frac{-1-c^2+a^2+b^2}{1+c^2+a^2+b^2} & 2\,\frac{a-\mathrm{bc}}{1+c^2+a^2+b^2} & 2\,\frac{\mathrm{ac}+b}{1+c^2+a^2+b^2} \\ -2\,\frac{a+\mathrm{bc}}{1+c^2+a^2+b^2} & -\frac{a^2-1-b^2+c^2}{1+c^2+a^2+b^2} & -2\,\frac{-c+\mathrm{ab}}{1+c^2+a^2+b^2} \\ 2\,\frac{\mathrm{ac}-b}{1+c^2+a^2+b^2} & -2\,\frac{c+\mathrm{ab}}{1+c^2+a^2+b^2} & \frac{-b^2-c^2+1+a^2}{1+c^2+a^2+b^2} \end{bmatrix} \tag{7}$$

is a rotation. Conversely, if $\mathcal{R}$ is a rotation then $H = \frac{\mathcal{R}-I}{\mathcal{R}+I}$ is anti-symmetric, assuming that -1 is not an eigenvalue of $\mathcal{R}$.

Expressing relation (4) and removing the denominators, one obtain a system depending on 6 variables $[X, Y, Z, a, b, c]$. We can suppose that $\overrightarrow{\mathrm{CB}_1}_{|R_m} = 0$ and $\overrightarrow{\mathrm{OA}_1} = 0$. Then equations (6) can be rewritten:

$$\begin{aligned} L_1^2 &= ||\overrightarrow{\mathrm{OC}}_{|R_f}||^2 \\ L_i^2 - L_1^2 &= ||\mathcal{R}\cdot\overrightarrow{\mathrm{CB}}_{i|R_m} - \overrightarrow{\mathrm{OA}}_{i|R_f}||^2 \quad +2 \quad \left\langle \mathcal{R}\cdot\overrightarrow{\mathrm{CB}}_{i|R_m} - \overrightarrow{\mathrm{OA}}_{i|R_f}, \overrightarrow{\mathrm{OC}}_{|R_f} \right\rangle \text{ for } i = 2, \ldots \end{aligned} \tag{8}$$

Hence if we assume for a while that $\mathcal{R}$ is known, then $[X, Y, Z] = \overrightarrow{\mathrm{OC}}_{|R_f}$ is solution of a linear system (5 equations and 3 unknowns). Therefore it is enough to compute directly the Gröbner basis of (8) for an elimination order such that $[X, Y, Z] > [a, b, c]$ and to eliminate the first block of variables (in fact the elimination is done very quickly in the first steps of the algorithm). Hence the computed Gröbner basis depends only on a, b, c and the shape of the result is *usually* in shape position (and thus in the form (2.4) with $n = 3$). Let denote by $\mathcal{S}_{\mathrm{quat}}$ the resulting system. Based on the F_5 algorithm, a specific Gröbner engine has been implemented which computes a lexicographic Gröbner basis and eliminates as soon as possible a block of k variables. As we will see in the next section the computation of such a basis is not only faster but also the size of the result (mainly the size of the coefficients) is much smaller.

3.2.2 The non generic case

Since the above formulations are valid only if -1 is not an eigenvalue value of $\mathcal{R}$ we need to detect and solve these exceptional situations to certify the results.

i. we first compute a Gröbner basis of $\mathcal{S}_{\text{quat}}$ for any ordering and apply theorem 3 to count the number of complex solutions counted with multiplicities; If it is 40 or infinite we are sure that no solution is missing or that the robots is singular and we are done. Note that for singular robots it may happens that $\mathcal{S}_{\text{quat}}$ admits a finite number of solutions but that the additional linear system admits an infinite number of solutions.

ii. other else , we take the projective version of (3.2.1) by applying the following transform$[a \longrightarrow \frac{a}{d}, b \longrightarrow \frac{b}{d}, c \longrightarrow \frac{c}{d}]$ and then set $d = 0$ we obtain

$$\mathcal{R}_\infty = \begin{bmatrix} \frac{c^2-a^2-b^2}{c^2+a^2+b^2} & -2\,\frac{\text{bc}}{c^2+a^2+b^2} & 2\,\frac{\text{ac}}{c^2+a^2+b^2} \\ -2\,\frac{\text{bc}}{c^2+a^2+b^2} & -\frac{a^2-b^2+c^2}{c^2+a^2+b^2} & -2\,\frac{\text{ab}}{c^2+a^2+b^2} \\ 2\,\frac{\text{ac}}{c^2+a^2+b^2} & -2\,\frac{\text{ab}}{c^2+a^2+b^2} & -\frac{b^2+c^2-a^2}{c^2+a^2+b^2} \end{bmatrix}$$

We thus generate an algebraic system as in the generic situation (replacing $\mathcal{R}$ by $\mathcal{R}_\infty$) and we test whether there are some solutions such that $a^2 + b^2 + c^2 \neq 0$. Note that this computation is much faster than the general computation (less than 0.1 sec for all the examples).

3.2.3 Recovering the real solutions of the DKP

If we assume that $\mathcal{S}_{\text{quat}}$ has been solved using a RR-form or a RUR and the isolation process exposed in sections 2.4 and 2.3, one needs to compute, for each real solution (rotations), the related translation $[X, Y, Z]$ which remains to solve a linear system of 5 equations in 3 variables. In a first stage, this can be tried using multi-precision interval arithmetic.

- A - check that at least one 3×3 minor never vanishes (by means of intervals not containing 0) at the real roots of $\mathcal{S}_{\text{quat}}$, we are done (usual situation).
- B - If not (exceptional situation), one symbolically computes all the 3×3 minors and, for each; check if it vanishes at some root of the system : one simply needs to add the minor to the system and solve it. Note that if such a system has a solution, the fist polynomial of the RUR is a factor of the first polynomial of the RUR of $\mathcal{S}_{\text{quat}}$.
- C - If at least one of these minors do not vanish at any root of $\mathcal{S}_{\text{quat}}$, one increases the precision of the computed solutions and the precision of the interval arithmetic until finding the unique translations corresponding to the rotations : we are sure that such a process will end since the filter from step A will conclude that at least one of the 3×3 minor never vanishes after a finite number of tries.

- D - Else, one can ignore the roots of $\mathcal{S}_{\text{quat}}$ which drive to an non invertible system : they correspond to singular positions of the robot, and apply the above process on the others. In pracice, this can be done by removing the factor of the first polynomial of the RUR of $\mathcal{S}_{\text{quat}}$ by the one obtained at step B and run this process again from step A.

Note that the solution can be obtained with an arbitrary precision when using multi-precision interval arithmetics such as MPFI since the solutions of $\mathcal{S}_{\text{quat}}$ can be refined on demand accordingly to section 2.5.

4 Experiments

All the computations were done on a PC Xeon (2.8 Ghz + 2 Go RAM). In the following experiments we use the following benchmark:

- Using a numerical global optimization program, Dietmaier [6] gives explicitly an example of a robot with 40 real roots; The coordinates of the base and platform are the following :

```
Base:= [
  [0,0,0],
  [0.542805,0,0],
  [0.956919,-0.528915,0],
  [0.665885,-0.353482,1.402538],
  [0.478359,1.158742,0.107672],
  [-0.137087,-0.235121,0.353913]
]:
Platform:=[
  [0,0,0],
  [1.107915,0,0],
  [0.549094,0.756063,0],
  [0.735077,-0.223935,0.525991],
  [0.514188,-0.526063,-0.368418],
  [0.590473,0.094733,-0.205018]
]:
```

  The 40 real solutions are obtained for the following legs lengths :

```
Legs:=[1,0.645275,1.086284,1.503439,1.281933,0.771071]:
```

  we show that using the tools presented in the paper it is very easy to check that the solutions are really real numbers (and not complex number with a very small

imaginary part). If δ is any positive integer we use the following function to convert original floating point coefficients to rationals:

$$f_\delta(x) = \frac{\lfloor 2^\delta x \rceil}{2^\delta}$$

and thus `dietmaier_` δ `_method` is the algebraic system $\{\mathcal{S}_{\text{quat}}, \mathcal{S}_{\ni \text{pt}}, \mathcal{S}_{\text{rot}}\}$ depending on the value of `method`. We will see in that if δ is too small then we have not 40 real roots: for instance when $\delta = 17$ we have only 38 real roots and two complex numbers whose imaginary parts are $\approx \pm 0.021$.

- `Left Hand` This example is a particular case of the "left hand" robot by Jean-Pierre Merlet. We took the following coordinates for defining its base and its platform :

```
Base:=[
  [-9.704000000, 9.107000000, 0],
  [9.708000000, 9.110000000, 0],
  [12.75400000, 3.892000000, 0],
  [599/200, -12993/1000, 3/500],
  [-1503/500, -13, 1/200],
  [-12.77000000, 3.901000000, 1/1000]
]:

Platform:=[
  [-3003/1000, 7.290000000, 0],
  [601/200, 7.296000000, 0],
  [7.814000000, -1.053000000, 0],
  [4.812000000, -6.246000000, 3/1000],
  [-4.823000000, -6.257000000, 1/125],
  [-7.818000000, -1.057000000, 1/100]
]:
```

Moreover, we set the length of the legs to the following values for our experiments :

```
Legs:=[9999/500,4001/200,20007/1000,
       20007/1000,19993/1000,5001/250]:
```

It is interesting to test this configuration because it is a *exceptional* case: the system has only 36 complex solutions (and not 40). In that case, 1 is always an eigenvalue of $\mathcal{R}$ so that

$$\mathcal{R} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & ? & ? \\ 0 & ? & ? \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & \mathcal{R}_\in \end{bmatrix} \text{where } \mathcal{R}_\in \text{ is } a \text{ rotation}.$$

moreover we can split the 36 solutions of the problem into two sets:

i. computing with $\mathcal{R}_\infty$:-1 is eigenvalue of $\mathcal{R}$ (or $\mathcal{R}_\in$) of multiplicity 2 for 6 solutions.

ii. computing with $\mathcal{R}$:-1 is not an eigenvalue of $\mathcal{R}$ for the other 30 complex solutions.

`Left Hand` $+\varepsilon$ is a perturbed version of the previous robot: it has now 40 real roots and the same number of real roots (8).

- `Spat 66` is a generic system with 40 complex roots (the coordinates of the robot as well as the lengths of the legs are chosen randomly in $[-1000, 1000]$). It has been used as a challenging example in [10] and [24]. The solution proposed in [24], based on Gröbner bases for computing the number of real roots, took several hours. By running this old strategy on the machine used for the benchmarks included in this article, we observe a computation time of 819 sec :

  - Gröbner basis for the DRL ordering using an optimized variant of burchberger's algorithm (t-grobner) : 415 sec;
  - Univariate Representation using symetric functions such as in [25] : 243 sec;
  - Sturm sequences for counting the real roots : 161 sec.

  These results must be compared with 0.75 sec we now observe by using modern algorithm (F4/F5 for getting an RR-Form + Descartes' method for getting the certified solutions) and the modelization of Section 3.2. The speed-up is about 1000.

In the following table *size* is twice the size of the output in the basis $b = 2^{32}$; for instance hence 146 means that the biggest integer is not bigger than $2^{\frac{32*146}{2}} = 2^{2336}$. For all but one example we give only the CPU time for computing the lexico Gröbner basis in Rational Form (denoted by `RR` in the table); for the first example (`dietmaier_16_quat`) a DRL Gröbner basis has been computed (`DRL`); surprisingly the computation is more difficult for a DRL Gröbner basis than for a Lexico Gröbner basis. It might seen than the size of the coefficients in the Rational Form of the Gröbner bases is smaller than in any other representation.

Benchmark	rr/drl	size	CPU	RUR	ISO(32)	ISO(64)	Comment
`dietmaier_16_quat`	rr	146	1.0	-	0.20	0.28	40 complex roots / 38 real roots
`dietmaier_16_quat`	drl	690	8.7	6.5	0.46	0.59	"
`dietmaier_16_3pt`	drl	842	13.3	9.4	0.54	0.75	"
`dietmaier_16_3pt`	rr	325	4.0	-	0.56	0.71	"
`dietmaier_16_rot`	rr	288	2.0	-	0.64	0.82	"
`dietmaier_18_quat`	rr	165	1.1	-	0.21	0.32	40 real roots
`dietmaier_18_3pt`	rr	366	4.5	-	0.65	0.84	"
`dietmaier_18_rot`	rr	326	2.2	-	0.75	0.93	"
`dietmaier_24_quat`	rr	219	1.4	-	0.25	0.36	40 real roots
`dietmaier_30_quat`	rr	272	1.8	-	0.28	0.40	"
`dietmaier_40_quat`	rr	312	2.0	-	0.31	0.43	"
`dietmaier_60_quat`	rr	349	2.2	-	0.32	0.46	"
`dietmaier_120_quat`	rr	456	2.9	-	0.40	0.56	"
`Spat66_quat`	rr	93	0.7	-	0.00	0.01	40 complex roots / 0 real roots
`LeftHand_3pt`	rr	225	0.4	-	0.16	0.21	36 complex roots / 8 real roots
`LeftHand_rot`	rr	228	0.6	-	0.37	0.39	"
`LeftHand_quat`	rr	166	0.4	-	0.12	0.16	30 complex roots / 8 real roots
`LeftHand_rot` $\mathcal{R}_\infty$	rr	14	0.04	-	0.00	0.01	6 complex roots / 0 real roots
LeftHand _ quat $+\varepsilon$	rr	126	0.9	-	0.10	0.12	40 complex roots / 8 real roots

Therefore the previous table establish a cause-and-effect relation between the size of the coefficients in the output and the CPU time for computing the corresponding object.

In [6] a floating point implantation of Gröbner bases is given; note that only an approximate Gröbner basis is given by the algorithm; the authors report a CPU time of 1067 sec (DEC Alpha 3000) to compute a lexicographic Gröbner basis. In our implementation (and a faster computer) it takes only 1 sec to compute a lexico Gröbner basis for the dietmaier example and ... sec to compute exactly all the real roots of the system.

The conclusion of the experiments is that the CPU time for computing a RR-Form is very stable (around 1 sec) and is minimally sensitive to configuration of the robot and the precision of the input data (δ parameter). Surprisingly, it is much more efficient to compute a lexicographical Gröbner basis or a RUR than a Gröbner basis for a DRL ordering: the simplest explanation of this behavior is that the size of the coefficients in a DRL Gröbner are bigger than in the corresponding RUR (or RR-Form).

5 Conclusion

We have improved significantly algorithms based on Gröbner bases computations for solving the DKP for parallel robots (see [24] for example) as well as the algebraic formulation of the problem. We did not compare our results with other valuable certified strategies such as homothopy methods ([28], [30]) or strategies based on interval analysis (see [18]) for two reasons :

- they do not solve the same kind of problems : for example, interval analysis based methods could solve robots with uncertain coordinates while we cannot; homothopy methods compute all the complex roots but cannot discriminate in a certified way the real roots from the others;
- as shown in the present paper, the modelization is of high importance, so that comparing objectively the solvers would require to find the "best" modelization for each method.

However, keeping track of the above remarks, our strategy seems to be currently the most powerfull one, with such a level of certification, for studying all the postures of an exactly knwon general robot.

The modelization proposed in section 3.2 give a precise characterization of "generic solutions" : these are those for which the linear part of the system is degenerated. A next step would be to study specifically the non generic solutions relaxing some parameters (lengths of the legs or coordinates of some joints) using recent algorithms such as [17]. A direct application would be, for example, to characterize singular robots.

Still finding **all** solutions is only one of the major part of the DKP. Indeed the roboticians are interested only in the *current* pose, i.e. the pose in which is the robot they have in front of them. Evidently this pose is in the set of the poses that is calculated but there is no known method which allows to determine which pose in the set is the current one. To show the difficulty of this problem we must define the concept of *valid pose*. For that purpose we consider the pose of the robot that is obtained when it is assembled for the first time (the *initial assembly mode*), i.e. when the 6th leg is connected to the platform, that is supposed to be known. A valid pose is defined as a pose that may be reached from the initial assembly mode by a path such that:

- the whole path must lie within the workspace of the robot (this workspace is limited by the possible change in the leg lengths and by mecahnical constraints on the joint)
- the whole path is interference-free: the legs do no intersect on the path
- the whole path is singularity-free: the jacobian of the IKP must never be singular

Hence determining is a pose is valid is equivalent to solve a robotics motion planning problem which is NP hard. There is also no known theoretical result on the problem of determining under which conditions there will be only one valid pose in the set of poses that are obtained when solving the DKP.

References

[1] Auzinger and Stetter. An elimination algorithm for the computation of all zeros of a system of multivariate polynomial equations. *Int. Series of Numerical Math.*, 86:11–30, 1998.

[2] S. Basu, R. Pollack, and M.-F. Roy. *Algorithms in real algebraic geometry*, volume 10 of *Algorithms and Computations in Mathematics*. Springer-Verlag, 2003.

[3] B. Buchberger. *Gröbner bases : an algorithmic method in polynomial ideal theory.* Recent trends in multidimensional systems theory. Reider ed. Bose, 1985.

[4] B. Buchberger, G.-E. Collins, and R. Loos. *Computer Algebra Symbolic and Algebraic Computation.* Springer-Verlag, second edition edition, 1982.

[5] D. Cox, J. Little, and D. O'Shea. *Ideals, varieties, and algorithms an introduction to computational algebraic geometry and commutative algebra.* Undergraduate texts in mathematics. Springer-Verlag New York-Berlin-Paris, 1992.

[6] P. Dietmaier. The stewart-gough platform of general geometry can have 40 real postures. In *Advances in Robot Kinematics: Analysis and Control*, pages 1–10. 1998.

[7] A.G. Erdman. *Modern Kinematics.* Wiley, New-York, 1993.

[8] J.-C. Faugère. A new efficient algorithm for computing gröbner bases (f_4). *Journal of Pure and Applied Algebra*, 139(1-3):61–88, 1999.

[9] J.-C. Faugère, P. Gianni, D. Lazard, and T. Mora. Efficient computation of zero-dimensional gröbner basis by change of ordering. *Journal of Symbolic Computation*, 16(4):329–344, 1993.

[10] J.-C. Faugère and D. Lazard. The combinatorial classes of parallel manipulators. *Mechanism and Machine Theory*, 30:765–776, 1995.

[11] Jean-Charles Faugère. A new efficient algorithm for computing gröbner bases without reduction to zero f_5. In *Proceedings of the ACM SIGSAM International Symposium on Symbolic and Algebraic Computation*, 2002.

[12] V.E. Gough. Contribution to discussion of papers on research in automobile stability, control and tyre performance, 1956-1957. Proc. Auto Div. Inst. Mech. Eng.

[13] K.H. Hunt. *Kinematic geometry of mechanisms.* Clarendon Press, Oxford, 1978.

[14] M.L. Husty. An algorithm for solving the direct kinematic of Stewart-Gough-type platforms. *Mechanism and Machine Theory*, 31(4):365–380, 1996.

[15] D. Lazard. Solving zero - dimensional algebraic systems. *Journal of Symbolic Computation*, 13:117–132, 1992.

[16] D. Lazard. Stewart platforms and gröbner basis. In *Proceedings of Advances in Robotics Kinematics*, pages 136–142, 1992.

[17] D. Lazard and F. Rouillier. Solving parametric polynomial systems. *Journal of Symbolic Computation*, 2006. to appear.

[18] J-P. Merlet. Solving the forward kinematics of a Gough-type parallel manipulator with interval analysis. *Int. J. of Robotics Research*, 23(3):221–236, 2004.

[19] B. Mourrain. The 40 generic positions of a parallel robot. In *ISSAC'93*, pages 173–182, 1993.

[20] B. Mourrain. An introduction to linear algebra methods for solving polynomial equations, 1998.

[21] M. Raghavan. The Stewart platform of general geometry has 40 configurations. In *ASME Design and Automation Conf.*, volume 32-2, pages 397–402, Chicago, 1991.

[22] N. Revol and F. Rouillier. Motivations for an arbitrary precision interval arithmetic and the mpfi library. *Reliable Computing*, 11:1–16, 2005.

[23] B. Roth. Computation in kinematics. In P. Kovacs J. Angeles, G. Hommel, editor, *Computational Kinematics*, pages 3–14. Kluwer, 1993.

[24] F. Rouillier. Real roots counting for some robotics problems. In *Computational Kinematics*, pages 73–82. Kluwer, 1995.

[25] F. Rouillier. *Algorithmes efficaces pour l'étude des zéros réels des systèmes polynomiaux.* PhD thesis, Université de Rennes I, 1996.

[26] F. Rouillier. Solving zero-dimensional systems through the rational univariate representation. *Journal of Applicable Algebra in Engineering, Communication and Computing*, 9(5):433–461, 1999.

[27] F. Rouillier and P. Zimmermann. Efficient isolation of polynomial real roots. *Journal of Computational and Applied Mathematics*, 162(1):33–50, 2003.

[28] A.J. Sommese, J. Verschelde, and C.W. Wampler. Advances in polynomial continuation for solving problems in kinematics. *ASME J. of Mechanical Design*, 126(2):262–268, 2004.

[29] C.W. Wampler. Forward displacement analysis of general six-in-parallel SPS (Stewart) platform manipulators using soma coordinates. *Mechanism and Machine Theory*, 31(3):331–337, 1996.

[30] C.W. Wampler, A.P. Morgan, and A.J. Sommese. Numerical continuation methods for solving polynomial systems arising in kinematics. *ASME J. of Mechanical Design*, 112:59–68, 1990.

Unité de recherche INRIA Rocquencourt
Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex (France)

Unité de recherche INRIA Futurs : Parc Club Orsay Université - ZAC des Vignes
4, rue Jacques Monod - 91893 ORSAY Cedex (France)
Unité de recherche INRIA Lorraine : LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex (France)
Unité de recherche INRIA Rennes : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex (France)
Unité de recherche INRIA Rhône-Alpes : 655, avenue de l'Europe - 38334 Montbonnot Saint-Ismier (France)
Unité de recherche INRIA Sophia Antipolis : 2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex (France)

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399