Technical Report

IRI-TR-01-08

A Linear Relaxation Technique for the Position Analysis of Multi-Loop Linkages

Josep M. Porta
Lluís Ros
Federico Thomas

Institut de Robòtica i Informàtica Industrial

Abstract

This report presents a new method able to isolate all configurations that a multi-loop linkage can adopt. We tackle the problem by providing formulation and resolution techniques that fit particularly well together. The adopted formulation yields a system of simple equations (only containing linear and bilinear terms, and trivial trigonometric functions for the helical pair exclusively) whose special structure is later exploited by a branch-and-prune method based on linear relaxations. The method is *general*, as it can be applied to linkages with single or multiple loops with arbitrary topology, involving lower pairs of any kind, and *complete*, as all possible solutions get accurately bounded, irrespectively of whether the linkage is rigid or mobile.

Institut de Robòtica i Informàtica Industrial (IRI)
Consejo Superior de Investigaciones Científicas (CSIC)
Universitat Politècnica de Catalunya (UPC)
Llorens i Artigas 4-6, 08028, Barcelona, Spain
Tel (fax): +34 93 401 5750 (5751)
http://www.iri.upc.edu

Corresponding author:
JM Porta
tel: +34 93 401 0775
porta@iri.upc.edu
http://www.iri.upc.edu/people/porta

1 Introduction

A linkage is an articulated mechanism of rigid *links* connected through lower-pair *joints*. We are interested in linkages forming one or more *kinematic loops*, i.e., closed sequences of pairwise articulated links. This report presents a new method for the position analysis of such linkages, that is, for the computation of the configurations they can adopt, within specified ranges for their degrees of freedom. A *configuration* is here understood in a kinematic sense: as an assignment of positions and orientations to all links that respects the kinematic constraints imposed by all joints, with no regard to possible link-link interferences. Several problems in Robotics translate into the above one, or require an efficient module able to solve it. The problem arises, for instance, when solving the inverse/forward kinematics of serial/parallel manipulators [32, 20], when planning the coordinated manipulation of an object or the locomotion of a reconfigurable robot [56], or in simultaneous localization and map-building [40]. The problem also appears in other domains, such as in the simulation and control of complex deployable structures [27], the theoretical study of rigidity [4], or the conformational analysis of biomolecules [55]. The common denominator in all cases is the existence of one or more kinematic loops in the system at hand, defining a linkage whose feasible configurations must be determined.

For decades, kinematicians have devoted considerable efforts to solve such problem, but eminently on an ad-hoc basis. Closed-form or efficient iterative solutions have been derived for specific linkages, notable ones including those for the inverse kinematics of general 6R manipulators [45], or for the forward kinematics of parallel platforms [50]. However, scarce efforts have been devoted to obtain a *general* solver, able to tackle linkages of any kind. While it is true that such solver could be implemented by algebraizing the problem and using any general technique for systems of polynomial equations (see Section 2), our experience shows that, unfortunately, a solution to both algebraization and resolution, treated as independent problems, does not necessarily yield an efficient procedure. In this work, in contrast, we will propose solution techniques to both problems that fit particularly well together.

A few previous works in the literature specifically provide general linkage solvers, but limited to planar linkages only. These include the work by Nielsen and Roth, who gave an algorithm to derive the Dixon resultant of any planar linkage [39], the work by Wampler, which improves on Nielsen and Roth's by applying a complex plane formulation [54], the work by Celaya *et al.* [8], which provides an interval propagation algorithm, and, finally, the work by Porta *et al.* [41], which attacks the problem via linear relaxations of the equations. A careful examination of these methods shows that [41] is specially amenable for generalization to the spatial case, a task we preliminary addressed in [42] and which we complete in the present work. As a result, we contribute with a method able to solve the position analysis of any spatial linkage, irrespective of the joint types it involves, of the number of loops it presents, and of the loops' interconnection pattern. In every case the method returns a discrete map of the linkage configuration space given as a *box approximation*—a collection of boxes enclosing all of its points at the desired resolution level. Such map allows visualizing the linkage rigid and flexible assembly modes, and bifurcations.

The rest of the report is organized as follows. Section 2 briefly reviews related work concerning the resolution of the problem using general techniques for systems of algebraic equations. Section 3 shows how the position analysis of a multi-loop linkage can be formulated as a canonical system of quadratic, bilinear, and trivial trigonometric equations. A linear relaxation method for solving this system is next presented in Section 4. Section 5 includes several experiments illustrating the performance of a C implementation of the technique. Finally, the report concludes in Section 6 summarizing the main contributions and points deserving further attention.

2 Related Work

Although the position analysis problem can be approached by geometric constructive techniques [13], only the algebraic methods have proved general enough to tackle all problem instances. These methods consist in characterizing the linkage configurations by an algebraic system of equations that is then solved using standard techniques. Reviews of such techniques in the context of Robotics, CAD/CAM and Molecular Modelling can be found for example in [38], [19], and [43], respectively. Broadly speaking, the proposed methods fall into three categories, depending on whether they use algebraic geometry, continuation, or branch-and-prune techniques.

Algebraic-geometric methods, including those based on resultants and Gröbner bases, use variable elimination to reduce the initial system to a univariate polynomial. The roots of this polynomial, once backsubstituted into other equations, yield all solutions of the original system. These methods have proved quite efficient in fairly non-trivial problems such as the inverse kinematics of general 6R manipulators [32, 45], or the forward analysis of general Stewart-Gough platforms [50]. Recent progress on the theory of sparse resultants, moreover, qualifies them as a very promising set of techniques [10].

Continuation methods, in contrast, begin with an initial system whose solutions are known, and then transform it gradually to the system whose solutions are sought, while tracking all solution paths along the way. In its original form, this technique was known as the *Bootstrap Method*, as developed by Roth and Freudenstein [47], and subsequent work by Garcia and Li [14], Garcia and Zangwill [15], Morgan [35], and Li *et al.* [30], among others, led the procedure into its current highly-developed state [49, 17]. These methods have been responsible for the first solutions to many long-standing problems in Kinematics. For example, using them, Tsai and Morgan first showed that the inverse kinematics of the general 6R manipulator has up to sixteen solutions [28], Raghavan showed that the direct kinematics of the general Stewart-Gough platform can have at most forty solutions [44], and Wampler *et al.* solved nine-point path synthesis problems for four-bar linkages [52].

In a different approach, branch-and-prune methods use approximate bounds of the solution set in order to rule out portions of the search space that contain no solution. The initial domain is iteratively reduced as much as possible, and then bisected in two halves. The whole reduction-bisection process is then applied on each half recursively, until a fine-enough approximation of the solution set is derived. The convergence of this scheme is guaranteed by the fact that the bounds get tighter as the intermediate domains get smaller.

While algebraic-geometric and continuation methods are general, they have a number of limitations in practice. On the one hand, algebraic-geometric methods usually explode in complexity, may introduce extraneous roots, and can only be applied to relatively simple systems of equations. Beyond this, they may require the solution of high-degree polynomials, which may be a numerically ill-conditioned step in some cases. On the other hand, continuation methods must be implemented in exact rational arithmetic to avoid numerical instabilities, leading to important memory requirements, and, like elimination methods, they must compute all possible roots, even the complex ones, which slows the process substantially on systems with a small fraction of real roots. Branch-and-prune methods are also general, but present a number of advantages that make them preferable in our case: (1) Contrarily to many elimination methods, they do not require intuition-guided symbolic reductions, (2) they directly isolate the real roots, (3) they can be made numerically robust without resorting to extra-precision arithmetic, and (4) some of them can tackle under- and over-constrained problems without any modification. These are the main reasons that motivate the approach we present, which belongs to this latter category.

Two families of branch-and-prune methods can be distinguished, depending on whether they

bound the solution set via Taylor expansions, or via polytopes.

Within the first family, the interval Newton method [18, 46, 11, 7, 21, 22] is perhaps the most-studied one. This method is based on the mean value theorem or, what is the same, on a zeroth-order Taylor expansion of the equations with a remainder of order one. The method requires the interval evaluation of the inverse of the Jacobian involved in the Taylor remainder and, therefore, it is only applicable to systems where such Jacobian is non-singular at all points of the input domain. Daney [9] and Gavriliu [16] present methods relying on first-order Taylor approximations of the equations, with bounded second-order reminders. While [9] uses Linear Programming to determine the area of feasible solutions, [16] solves a linear system instead, inverting a Jacobian matrix. The main difference with respect to interval Newton methods is that, here, the matrix to be inverted is real-valued and, therefore, the Jacobian is only required to be non-singular at the linearization point. Results show that first-order methods outperform zeroth-order ones [16]. Higher-order Taylor approximations are used in the so-called Taylor forms [3, 36], but they are limited to singe variable, single equation problems.

Methods in the second family bound the solutions by deriving, at each iteration, a convex polytope enclosing the solution set. In fact, such polytope is never determined explicitly, as its exact form can be rather complex. Instead, its rectangular hull is readily derived via Linear Programming. Polytope methods have similar convergence properties to Taylor methods [48, 25], but present a number of advantages: (1) they avoid the computation of derivatives and Jacobian inverses, (2) they naturally account for inequalities in the problem, and (3) they can directly deal with under- or over-constrained problems. The first methods of this kind appeared in the early nineties, by hand of Sherbrooke and Patrikalakis, who derived the polytope from properties of the equations' Bernstein form [48]. Later on, Yamamura [57] and Kolev [26] presented algorithms where the equations are bounded by polytopes made out of bands. In the first case, the bands are aligned according to the linear terms of the equations and the interval evaluation of the non-linear terms defines their width. In the second case, the slopes of trivial functions define the bands' orientation, and the roots of a univariate polynomial yield their width. More recently, linear relaxation techniques have been proposed [29], following a research line that can be traced back to the seventies [1, 33]. A linear relaxation is a set of linear inequalities that tightly bound a particular type of function. The simplest possible relaxation is the one obtained from a first-order Taylor expansion plus a second-order remainder that, when bounded, defines a polytope similar to those of Yamamura's and Kolev's methods. However, rather than resorting to general Taylor expansions, linear relaxations are defined on an ad-hoc basis for each function, including as many linear constraints as necessary to produce better bounds for the function at hand.

Among polytope methods, those based on linear relaxations are usually faster, as they define tighter polytopes with smaller linear programs. Their drawnback is, however, that although linear relaxations are easy to define and yield efficient pruning, they do so only on polynomial equations with simple terms (usually quadratic, or bilinear terms). While in principle any polynomial system could be transformed into such form, the transformation would introduce too extra variables, rendering the solution method inefficient. This is precisely why, being superior in principle, linear relaxation techniques have not been tried in the past on well-established algebraizations of the position analysis problem. This work overcomes this limitation by proposing a new formulation that comes in the adequate form, together with accurate relaxations for its defining equations.

3 Kinematic Equations

This section formulates the kinematic equations of a linkage. Contrarily to the standard approach, all links will be oriented with respect to an absolute frame, rather than by giving their relative angles to vicinal links.

We introduce some definitions for the purpose of discussion. A linkage is a pair $\mathcal{L} = (L, J)$, where $L = \{L_1, \ldots, L_n\}$ is a set of rigid *links*, and $J = \{J_1, \ldots, J_m\}$ is a set of lower-pair *joints*, each connecting a couple of links. Every linkage has an associated graph $G(\mathcal{L}) = (V, E)$ encoding its topology. We place a node in V for every link $L_i \in L$, and an edge (u, v) in E if the links associated with u and v are connected by a joint. We also furnish every link L_i with a local reference frame, denoted $\mathcal{F}_i$, and anchor any one of the links to the ground, letting its reference frame be the absolute frame, $\mathcal{F}_a$. We will use the notation $\mathbf{p}^{\mathcal{F}_i}$ to indicate that the components of vector $\mathbf{p} \in \mathbb{R}^3$, representing the coordinates of point P, are given in the basis of $\mathcal{F}_i$. Vectors with no superscript are assumed to be given in the basis of the absolute frame.

A linkage *configuration* is an assignment of a *pose* $(\mathbf{r}_i, \mathbf{R}_i)$ to each link, where $\mathbf{r}_i \in \mathbb{R}^3$ is the absolute position of $\mathcal{F}_i$'s origin, and $\mathbf{R}_i$ is a 3×3 rotation matrix giving the orientation of $\mathcal{F}_i$ relative to $\mathcal{F}_a$. Note that we cannot assign arbitrary poses to the links, since the joints impose certain constraints that must be fulfilled. We next formulate them explicitly.

3.1 Joint equations

We start assuming that the linkage consists of only one loop, with links and joints numbered from 1 to n, consecutively, and that all joints are of revolute type. Later on, the formulation will be easily extended to arbitrary linkage topologies and joint types.

Let us derive the assembly constraints imposed by joint J_i on its two links, L_i and L_{i+1}. If P_i and Q_i are two different points on the axis of J_i (Fig. 1), the valid poses for such links are those that fulfill[1]

$$\mathbf{r}_i + \mathbf{R}_i\, \mathbf{q}_i^{\mathcal{F}_i} = \mathbf{r}_{i+1} + \mathbf{R}_{i+1}\, \mathbf{q}_i^{\mathcal{F}_{i+1}}, \tag{1}$$

$$\mathbf{R}_i\, \mathbf{d}_i^{\mathcal{F}_i} = \mathbf{R}_{i+1}\, \mathbf{d}_i^{\mathcal{F}_{i+1}}, \tag{2}$$

where $\mathbf{d}_i = \mathbf{q}_i - \mathbf{p}_i$. The first condition forces L_i to be placed so that the point Q_i, seen as attached to $\mathcal{F}_i$, coincides with the same point, seen as attached to $\mathcal{F}_{i+1}$. The second condition does a similar identification for the $\mathbf{d}_i$ vector. Note that, if $\mathbf{R}_i = (\hat{\mathbf{u}}_i, \hat{\mathbf{v}}_i, \hat{\mathbf{w}}_i)$, then the following orthonormality conditions must also hold for $i = 1, \ldots, n$, so that $\mathbf{R}_i$ represents a valid rotation:

$$\|\hat{\mathbf{u}}_i\| = 1, \tag{3}$$

$$\|\hat{\mathbf{v}}_i\| = 1, \tag{4}$$

$$\hat{\mathbf{u}}_i \cdot \hat{\mathbf{v}}_i = 0, \tag{5}$$

$$\hat{\mathbf{u}}_i \times \hat{\mathbf{v}}_i = \hat{\mathbf{w}}_i. \tag{6}$$

Definition 1. The *basic* system of equations of a linkage is the one formed by Eqs. (1) and (2), gathered for all joints, and Eqs. (3)-(6), gathered for all links.

Definition 2. A *feasible* configuration is a tuple $(\mathbf{r}_1, \mathbf{R}_1, \ldots, \mathbf{r}_n, \mathbf{R}_n)$ that satisfies all equations of the basic system.

3.2 Loop equations

Our next goal is to show that the basic system can actually be simplified, reducing the number of equations and variables involved. Note for this that we can write Eq. (1) as

$$\mathbf{r}_i + \mathbf{R}_i\, \mathbf{p}_i^{\mathcal{F}_i} + \mathbf{R}_i\, \mathbf{d}_i^{\mathcal{F}_i} - \mathbf{R}_{i+1}\, \mathbf{q}_i^{\mathcal{F}_{i+1}} - \mathbf{r}_{i+1} = 0 \tag{7}$$

[1] In what follows, indices will be cyclic, so that indices $n+1$ and 0 will refer to indices 1 and n, respectively.

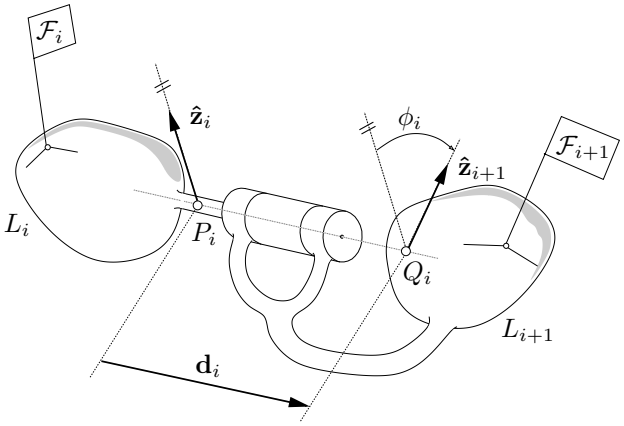


Figure 1: Elements intervening in the assembly constraints of a revolute joint.

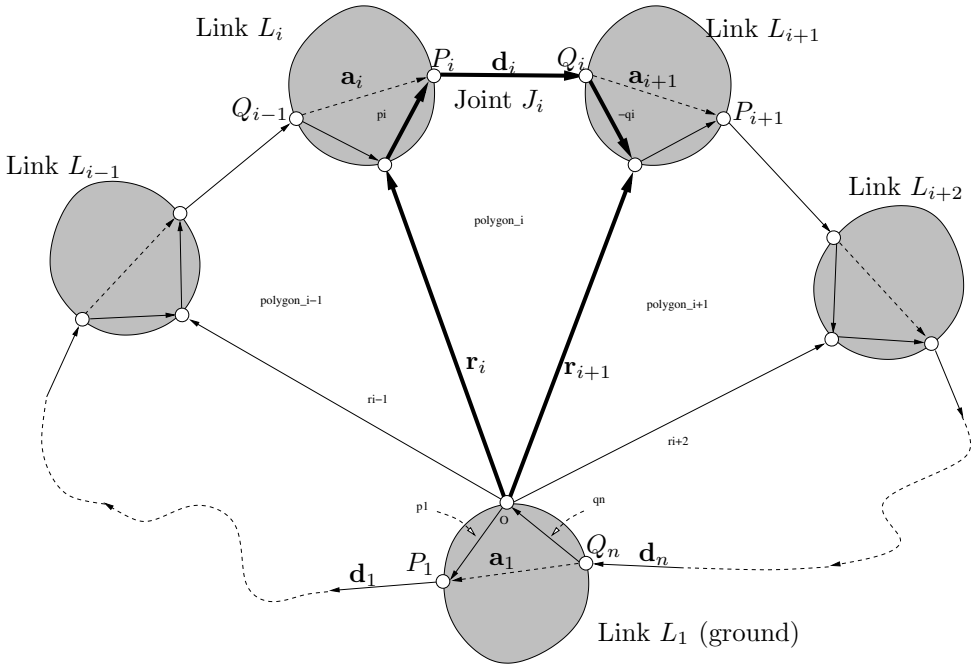


Figure 2: Schematic representation of the joint and loop equations. The joint equation for J_i expresses the closure of a pentagon through J_i (thick arrows). The loop equation expresses the closure of the polygon defined by the $\mathbf{a}_i$ and $\mathbf{d}_i$ vectors (the outer perimeter of all pentagons).

which can be interpreted as the sum of five vectors forming a closed polygon (Fig. 2). This implies that if we add up equation (7), for $i = 1, \ldots, n$, we obtain

$$\sum_{i=1}^{n} \mathbf{R}_i \, \mathbf{a}_i^{\mathcal{F}_i} + \mathbf{R}_i \, \mathbf{d}_i^{\mathcal{F}_i} = 0, \tag{8}$$

where $\mathbf{a}_i = \mathbf{p}_i - \mathbf{q}_{i-1}$. We call Eq. (8) a loop equation, as it expresses the fact that the $\mathbf{a}_i$ and $\mathbf{d}_i$ vectors form a closed polygon.

Definition 3. The *reduced system* of a single-loop linkage $\mathcal{L}$ is the one formed by Eqs. (2), gathered for all joints, Eqs. (3)-(6), gathered for all links, and Eq. (8).

Definition 4. A *feasible orientation* of a linkage is an assignment of a 3×3 matrix $\mathbf{R}_i$ to each of its links such that all equations in the reduced system get satisfied.

Observe that the reduced system only contains the $\mathbf{R}_i$ matrices as variables (the $\mathbf{r}_i$ vectors have been eliminated) and much less equations (n equations of the basic system have been replaced by just one equation). We next prove that solving any of the two systems is equivalent.

Proposition 5. Assuming that link L_1 is the ground, with $\mathbf{r}_1 = \mathbf{0}$ and $\mathbf{R}_1 = \mathbf{I}$, the solutions of the reduced system are in one-to-one correspondence with the solutions of the basic system.

Proof 1 *Clearly, if the tuple* $(\mathbf{r}_1, \mathbf{R}_1, \ldots, \mathbf{r}_n, \mathbf{R}_n)$ *satisfies the basic system, the tuple* $(\mathbf{R}_1, \ldots, \mathbf{R}_n)$ *also satisfies the reduced system, and we only need to prove the converse.*

Given a feasible orientation $(\mathbf{R}_1, \ldots, \mathbf{R}_n)$ *we only need to give the associated* $\mathbf{r}_i$ *vectors,* $i = 1, \ldots, n$*, to complete it to a feasible configuration. For a link* L_i*, we can compute its* $\mathbf{r}_i$ *using the recurrence*

$$\mathbf{r}_k = \mathbf{r}_{k-1} + \mathbf{R}_{k-1}\, \mathbf{p}_{k-1}^{\mathcal{F}_{k-1}} + \mathbf{R}_{k-1}\, \mathbf{d}_{k-1}^{\mathcal{F}_{k-1}} - \mathbf{R}_k\, \mathbf{q}_{k-1}^{\mathcal{F}_k}, \tag{9}$$

for $k = 2, \ldots, i$*. Observe that the use of this recurrence makes Eq. (7), and thus Eq. (1), be satisfied on joints* $J_1, \ldots, J_{i-1}$*. However, we can also compute* $\mathbf{r}_i$ *using the recurrence*

$$\mathbf{r}_k = \mathbf{r}_{k+1} - \mathbf{R}_k\, \mathbf{p}_k^{\mathcal{F}_k} - \mathbf{R}_k\, \mathbf{d}_k^{\mathcal{F}_k} + \mathbf{R}_{k+1}\, \mathbf{q}_k^{\mathcal{F}_{k+1}} \tag{10}$$

for $k = i, \ldots, n$*, which renders Eq. (7) satisfied on joints* $J_i, J_{i+1}, \ldots, J_n$*. To conclude the proof, we only need to check whether the* $\mathbf{r}_i$ *values, computed either via Eq. (9) or Eq. (10), do coincide. Note for this that, taking into account that* $\mathbf{p}_1 = \mathbf{q}_n + \mathbf{R}_1\, \mathbf{a}_1^{\mathcal{F}_1}$*, and using (9), then* $\mathbf{r}_i$ *can be written in solved form as*

$$\mathbf{r}_i = \mathbf{q}_n + \left(\sum_{j=1}^{i-1} \mathbf{R}_j\, \mathbf{a}_j^{\mathcal{F}_j} + \mathbf{R}_j\, \mathbf{d}_j^{\mathcal{F}_j} \right) - \mathbf{R}_i\, \mathbf{q}_{i-1}^{\mathcal{F}_i}, \tag{11}$$

and using (10) as

$$\mathbf{r}_i = \mathbf{q}_n - \left(\sum_{j=i}^{n} \mathbf{R}_j\, \mathbf{a}_j^{\mathcal{F}_j} + \mathbf{R}_j\, \mathbf{d}_j^{\mathcal{F}_j} \right) - \mathbf{R}_i\, \mathbf{q}_{i-1}^{\mathcal{F}_i}. \tag{12}$$

The two values will coincide if the difference between the right-hand sides of Eqs. (11) and (12) vanishes. But this is the case, since such difference is

$$\sum_{j=1}^{n} \mathbf{R}_j\, \mathbf{a}_j^{\mathcal{F}_j} + \mathbf{R}_j\, \mathbf{d}_j^{\mathcal{F}_j},$$

and according to Eq. (8) this summation vanishes.

3.3 Multiple loops

We next extend the formulation to cope with linkages $\mathcal{L} = (L, J)$ with an arbitrary graph $G(\mathcal{L})$. Note first that if $G(\mathcal{L})$ is a tree (Fig. 3a), then the position analysis of $\mathcal{L}$ becomes trivial: we can parameterize the configurations of $\mathcal{L}$ by the relative joint angles of one link with respect to its predecessor in the tree, and the set of valid angle tuples is simply T^m, the m-dimensional torus. Similarly, the graph of a general linkage can be seen as a tree of "ordinary" nodes and

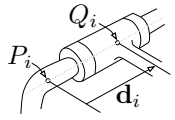
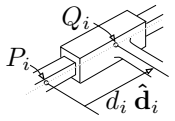
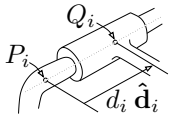
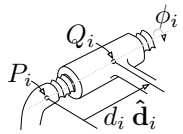
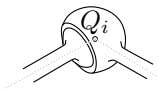
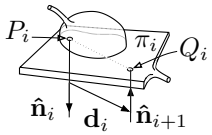
Pair	Shape	Joint equations	δ_i term
Revolute		$\mathbf{r}_i + \mathbf{R}_i\,\mathbf{q}_i^{\mathcal{F}_i} = \mathbf{r}_{i+1} + \mathbf{R}_{i+1}\,\mathbf{q}_i^{\mathcal{F}_{i+1}}$ $\mathbf{R}_i\,\mathbf{d}_i^{\mathcal{F}_i} = \mathbf{R}_{i+1}\,\mathbf{d}_i^{\mathcal{F}_{i+1}}$	$\mathbf{R}_i\,\mathbf{d}_i^{\mathcal{F}_i}$
Prismatic		$\mathbf{r}_i + \mathbf{R}_i\,\mathbf{p}_i^{\mathcal{F}_i} + d_i\,\mathbf{R}_i\,\hat{\mathbf{d}}_i^{\mathcal{F}_i} = \mathbf{r}_{i+1} + \mathbf{R}_{i+1}\,\mathbf{q}_i^{\mathcal{F}_{i+1}}.$ $\mathbf{R}_i = \mathbf{R}_{i+1}$	$d_i\,\mathbf{R}_i\,\hat{\mathbf{d}}_i^{\mathcal{F}_i}$
Cylindrical		$\mathbf{r}_i + \mathbf{R}_i\,\mathbf{p}_i^{\mathcal{F}_i} + d_i\mathbf{R}_i\,\hat{\mathbf{d}}_i^{\mathcal{F}_i} = \mathbf{r}_{i+1} + \mathbf{R}_{i+1}\,\mathbf{q}_i^{\mathcal{F}_{i+1}}$ $\mathbf{R}_i\,\hat{\mathbf{d}}_i^{\mathcal{F}_i} = \mathbf{R}_{i+1}\,\hat{\mathbf{d}}_i^{\mathcal{F}_{i+1}},$	
Helical		$\mathbf{r}_i + \mathbf{R}_i\,\mathbf{p}_i^{\mathcal{F}_i} + d_i\,\mathbf{R}_i\,\hat{\mathbf{d}}_i^{\mathcal{F}_i} = \mathbf{r}_{i+1} + \mathbf{R}_{i+1}\,\mathbf{q}_i^{\mathcal{F}_{i+1}}$ $\mathbf{R}_i\,\hat{\mathbf{d}}_i^{\mathcal{F}_i} = \mathbf{R}_{i+1}\,\hat{\mathbf{d}}_i^{\mathcal{F}_{i+1}},$ $\phi_i = k_i\,d_i$	
Spherical		$\mathbf{r}_i + \mathbf{R}_i\,\mathbf{q}_i^{\mathcal{F}_i} = \mathbf{r}_{i+1} + \mathbf{R}_{i+1}\,\mathbf{q}_i^{\mathcal{F}_{i+1}}$	0
Planar		$\mathbf{r}_i + \mathbf{R}_i\,\mathbf{p}_i^{\mathcal{F}_i} + \mathbf{d}_i = \mathbf{r}_{i+1} + \mathbf{R}_{i+1}\,\mathbf{q}_i^{\mathcal{F}_{i+1}}$ $\mathbf{d}_i \cdot \mathbf{R}_i\,\hat{\mathbf{n}}_i^{\mathcal{F}_i} = 0$ $\mathbf{R}_i\,\hat{\mathbf{n}}_i^{\mathcal{F}_i} = -\mathbf{R}_{i+1}\,\hat{\mathbf{n}}_{i+1}^{\mathcal{F}_{i+1}}.$	$\mathbf{d}_i$

Table 1: Joint equations and δ_i term for all lower pairs.

"block" nodes[2] (Fig. 3b) and the position analysis of $\mathcal{L}$ reduces to the position analysis of its constituent blocks. We assume hereafter, thus, that $G(\mathcal{L})$ is a block graph.

We now realize that the loop equation (8) must hold for every cycle of $G(\mathcal{L})$ since, along the cycle, the $\mathbf{a}_i$ and $\mathbf{d}_i$ vectors must necessarily form a closed polygon. However, not all loop equations are independent.

Proposition 6: For a linkage $\mathcal{L}$, every loop equation can be expressed as a linear combination of the loop equations associated with a cycle basis of $G(\mathcal{L})$.

Proof 2 *Let C be a cycle of $G(\mathcal{L}) = (V, E)$. Assuming $E = \{e_1, \ldots, e_m\}$, we may encode C with the binary vector $\mathbf{c} = (c_1, \ldots, c_m)$, where $c_i = 1$ if edge e_i belongs to the cycle, and $c_i = 0$ otherwise. It is well known that, using the symmetric difference $\oplus$ as vector addition[3], the set of cycle vectors defined in this way forms a vector space, C_G. Thus, if A and B are cycles with vectors $\mathbf{a}$ and $\mathbf{b}$, then $\mathbf{c} = \mathbf{a} \oplus \mathbf{b}$ also defines a cycle. It is also known that any cycle $\mathbf{c}$ can be expressed as*

$$\mathbf{c} = \mathbf{c}_1 \oplus \ldots \oplus \mathbf{c}_k, \tag{13}$$

[2] A block is a maximal subgraph where every two vertices are traversed by one cycle

[3] The symmetric difference of two binary vectors, $\mathbf{c} = \mathbf{a} \oplus \mathbf{b}$, is their element-wise XOR. I.e., $c_i = 1$ if a_i and b_i have different values, and $c_i = 0$ otherwise.

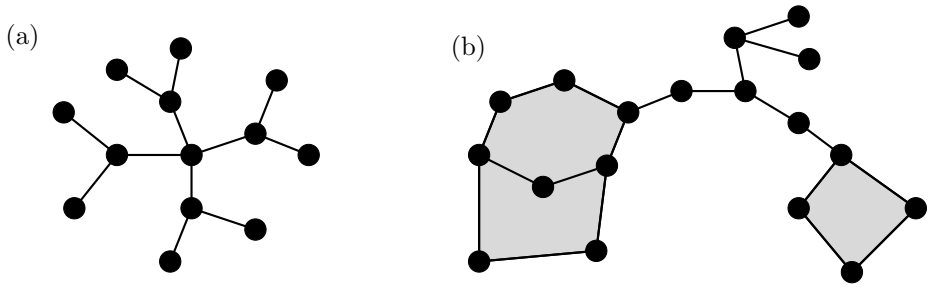


Figure 3: (a) A tree (b) A graph with two blocks, shown as shaded regions.

where $\mathbf{c}_1, \ldots, \mathbf{c}_k$ *are vectors of a basis of* C_G*. The proposition can be proved by showing that the loop equation corresponding to* $\mathbf{c}$ *is the sum of the loop equations corresponding to* $\mathbf{c}_1, \ldots, \mathbf{c}_k$*. Actually, we only need to prove this claim for the* $k = 2$ *case, since the general case follows by induction on the number of sumands in Eq. (13). Suppose thus that a cycle* C *is the symmetric difference of two other cycles,* C_1 *and* C_2*, i.e.,* $\mathbf{c} = \mathbf{c}_1 \oplus \mathbf{c}_2$*. Then, using Fig. (4) we easily see that the loop equation of* C *is the sum of the loop equations of* C_1 *and* C_2 *since (1) such sum will cancel out the* $\mathbf{a}_i$ *and* $\mathbf{d}_i$ *terms belonging to edges both in* C_1 *and* C_2*, and (2) all* $\mathbf{a}_i$ *terms corresponding to* C *are either* $\mathbf{a}_i$ *terms corresponding to* C_1 *or* C_2*, or a sum of them.*

Definitions 1 and 3 also apply to multi-loop linkages, with the exception that the reduced system contains one loop equation for each cycle of a selected cycle basis of $G(\mathcal{L})$. The analogue of Proposition 5 also holds:

Proposition 7. Let $\mathcal{L}$ be a linkage for which $G(\mathcal{L})$ is a block graph. Assuming that link L_1 is the ground, with $\mathbf{r}_1 = \mathbf{0}$ and $\mathbf{R}_1 = \mathbf{I}$, the solutions of the reduced system are in one-to-one correspondence with the solutions of the basic system.

Proof 3 *As for Proposition 5, we only need to prove that every feasible orientation can be completed to a feasible configuration. This only requires assigning a position vector* $\mathbf{r}_i$ *to every link* $L_i \in L$ *so that all joint equations get satisfied. Let* u_1 *be the node of* $G(\mathcal{L})$ *corresponding to the ground link* L_1*, and let* u_i *be the node corresponding to any other link* L_i*. Let* P *be any path of* $G(\mathcal{L})$ *connecting* u_1 *with* u_i*. We can let* $\mathbf{r}_i$ *be the position vector defined by recurrence (9), applied on the chain of links found along* P*. If we apply such recurrence on a different path* P' *connecting* u_1 *with* u_i*, we will obtain the same* $\mathbf{r}_i$ *value because, as in Proposition 5,* P *and* P' *form a closed loop. Thus, the* $\mathbf{r}_i$ *values computed in this way are unique and consistent, and satisfy all joint equations of the linkage.*

As a corollary, we have that the reduced system provides a set of necessary and sufficient conditions characterizing the feasible configurations of the linkage. Although the basic system could also be used to solve the position analysis problem, we will prefer using the reduced system in general, because it involves much less variables (the $\mathbf{r}_i$ vectors do not intervene) and equations (m joint equations of the basic system have been replaced by $m - n + 1$ loop equations).

3.4 Accounting for joint limits

Let L_i and L_{i+1} be two links connected through joint J_i. The *relative angle* between L_i and L_{i+1}, denoted ϕ_i, is defined as the angle between two unit vectors, $\hat{\mathbf{z}}_i$ and $\hat{\mathbf{z}}_{i+1}$, chosen orthogonal to J_i's axis, solidary to L_i and L_{i+1} respectively (Fig. 1). Suppose that we want to limit ϕ_i to lie

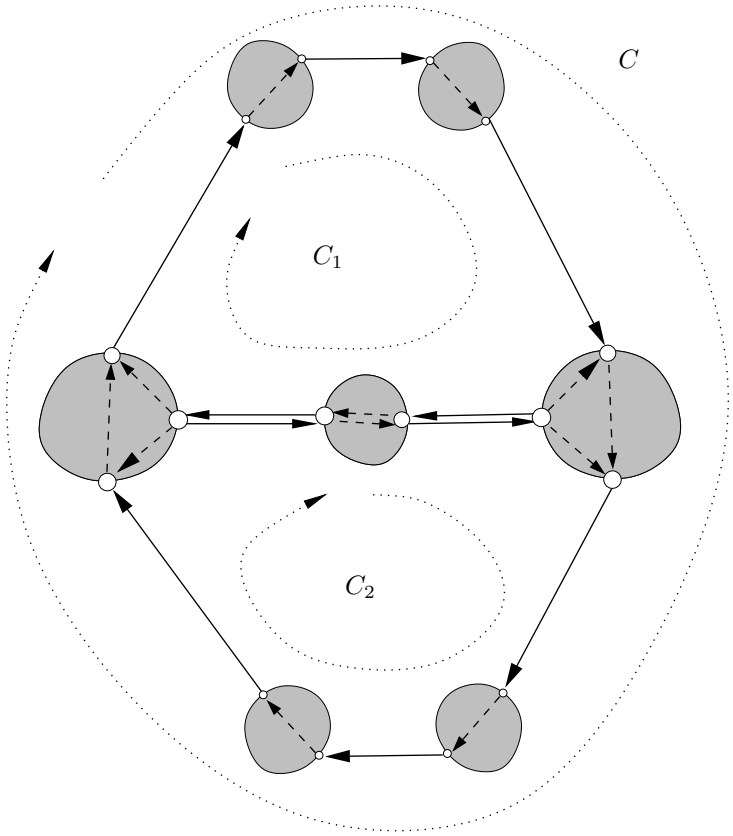


Figure 4: The loop equation of cycle C is the sum of the loop equations corresponding to cycles C_1 and C_2. The figure follows the same conventions used in Fig. 2: shaded regions are links, intralink arrows correspond to $\mathbf{a}_i$ vectors, and interlink arrows correspond to $\mathbf{d}_i$ vectors.

within the interval $[\phi_i^l, \phi_i^u] \subset [0, 2\pi]$. We will take these bounds into account by limiting the range of the sine and cosine of ϕ_i. Note for this that, if

$$c_i = \cos(\phi_i), \tag{14}$$
$$s_i = \sin(\phi_i), \tag{15}$$

then c_i and s_i can be obtained from $\mathbf{R}_i$ and $\mathbf{R}_{i+1}$ by realizing that

$$c_i = \hat{\mathbf{z}}_i \cdot \hat{\mathbf{z}}_{i+1}, \tag{16}$$
$$s_i\, \hat{\mathbf{d}}_i = \hat{\mathbf{z}}_i \times \hat{\mathbf{z}}_{i+1}, \tag{17}$$

where

$$\hat{\mathbf{z}}_i = \mathbf{R}_i\, \hat{\mathbf{z}}_i^{\mathcal{F}_i}, \tag{18}$$
$$\hat{\mathbf{z}}_{i+1} = \mathbf{R}_{i+1}\, \hat{\mathbf{z}}_{i+1}^{\mathcal{F}_{i+1}}, \tag{19}$$
$$\hat{\mathbf{d}}_i = \mathbf{R}_i\, \hat{\mathbf{d}}_i^{\mathcal{F}_i}. \tag{20}$$

Thus, to limit ϕ_i we can simply add Eqs. (16)-(20) to the system to be solved, and fix the ranges of the s_i and c_i variables to the interval evaluation of $\sin(\phi_i)$ and $\cos(\phi_i)$ on $[\phi_i^l, \phi_i^u]$.

3.5 Other Lower Pairs

This section extends the formulation given in Sections 3.1 and 3.2 to deal with lower pairs of any kind. We start reviewing the joint equations that each lower pair introduces into the basic system, and then examine the loop equations they yield in the reduced system.

Regarding the joint equations, note that a lower pair introduces motions about points (as in spherical joints), lines (as in revolute, prismatic, cylindrical, or helical joints), or planes (as in planar joints). In general, we define the P_i and Q_i points of a joint as in Section 3.1, but selected on the point, line, or plane characterizing the motion and with P_i rigidly attached to $\mathcal{F}_i$ and Q_i rigidly attached to $\mathcal{F}_{i+1}$. The equations of each lower pair are as follows (see Table 1):

- When J_i is revolute, we have the equations of Table 1, first row, already discussed above.
- When J_i is prismatic, L_{i+1} can only translate with respect to L_i. This can be enforced by choosing parallel reference frames for L_i and L_{i+1}, and adding $\mathbf{R}_i = \mathbf{R}_{i+1}$ to the system. Then, the valid poses for L_i and L_{i+1} are the pairs $(\mathbf{r}_i, \mathbf{R}_i)$ and $(\mathbf{r}_{i+1}, \mathbf{R}_{i+1})$ that verify the equations in Table 1, second row, where $\hat{\mathbf{d}}_i$ is a unit vector along the axis of the joint, pointing from P_i to Q_i, and d_i is a variable parameter taking values within some range.
- When J_i is cylindrical, L_{i+1} can freely rotate and translate along J_i's axis and the valid poses must satisfy the equations in Table 1, third row, where d_i and $\hat{\mathbf{d}}_i$ are defined as for prismatic joints.
- When J_i is helical, it can be seen as a cylindrical joint where the rotated angle ϕ_i and the displacement d_i are related by
$$\phi_i = k_i d_i, \tag{21}$$
where k_i is the pitch of the helix. Recall from Section 3.4, that ϕ_i must verify Eqs. (14)-(20) and thus, in addition to these equations, each helical joint contributes with the equations in Table 1, fourth row, to the basic system.
- When J_i is spherical, P_i coincides with Q_i and L_{i+1} can freely rotate with respect to L_i, meaning that the valid poses for the two links are those verifying the equation in Table 1, fifth row.
- When J_i is planar, the contact of links L_i and L_{i+1} is constrained to a plane π_i. The situation is depicted in the last row of Table 1 The points P_i and Q_i defining $\mathbf{d}_i$ are selected on π_i, the former attached to L_i and the latter to L_{i+1}. The vectors $\hat{\mathbf{n}}_i$ and $\hat{\mathbf{n}}_{i+1}$ are the normals to such links in P_i and Q_i, respectively. Three conditions for a proper assembly of L_i and L_{i+1} are also given in Table 1, last row.

Regarding the loop equations, note that on any lower pair, the first joint equation given in each row of the table plays a role similar to Eq. (1) for revolute joints. These equations merely force L_i and L_{i+1} to be placed with their Q_i points coinciding. It is not difficult to see again that, as done for revolute joints, we can eliminate the $\mathbf{r}_i$ vectors by adding the joint equations around a loop of the linkage. This will yield a loop equation of the form

$$\sum_{i=1}^{n} \mathbf{R}_i \, \mathbf{a}_i^{\mathcal{F}_i} + \delta_i = 0, \tag{22}$$

where the δ_i term depends on the joint type, as given in Table 1. Proposition 5 also holds for such generalized loop equations, with trivial modifications in its proof.

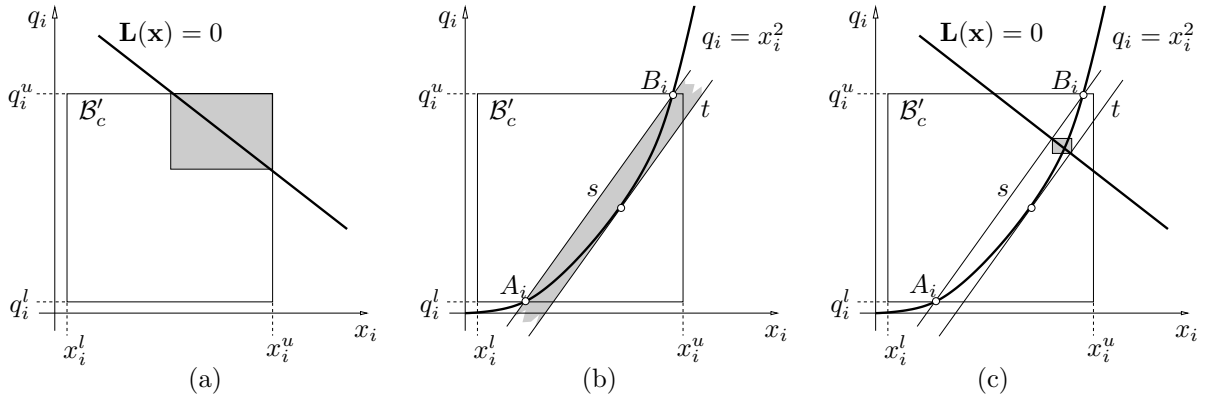


Figure 5: (a) Shrinking $\mathcal{B}'_c$ to fit the linear variety $L(\mathbf{x}) = 0$. (b) Half-planes approximating the part of the parabola inside $\mathcal{B}'_c$. (c) Smallest box enclosing the intersection of $L(\mathbf{x}) = 0$ with the half-planes in (b).

4 Solution Strategy

This section provides a method to solve the reduced system of equations of a multi-loop linkage. The method involves a preprocessing step to leave the equations in a canonical form (Section 4.1) and a numerical method that exploits this form to isolate all solutions (Section 4.2). Detailed pseudocode of the algorithm is given in Section 4.3, and its performace is discussed in Section 4.4.

4.1 Equation expansion

Observe that most of the equations in the reduced system are polynomial and, if x_i and x_j refer to any two of their variables, the involved monomials can only be of the form x_i^2, $x_i x_j$, or x_i. In other words, there can only be quadratic, bilinear, or linear terms. Only in the case of a linkage with helical joints, trigonometric equations of the form $x_i = \cos(x_j)$ or $x_i = \sin(x_j)$ appear in the system. For the sake of conciseness, we skip this particular case in the following, but it can be easily accommodated if necessary.

Let us define the changes of variables $q_i = x_i^2$ for each quadratic monomial, and $b_k = x_i x_j$ for each bilinear monomial. Clearly, by substituting the q_i's and b_k's into the equations of the reduced system, we obtain a new system of the form

$$F(\mathbf{x}) = (L(\mathbf{x}), P(\mathbf{x}), H(\mathbf{x})) = 0, \tag{23}$$

where $\mathbf{x} = (x_1, \ldots, x_{n_l}, q_1, \ldots, q_{n_q}, b_1, \ldots, b_{n_b})$ is a tuple including the original and newly defined variables, and:

- $L(\mathbf{x}) = (l_1(\mathbf{x}), \ldots, l_{m_l}(\mathbf{x}))$ is a block of linear functions.
- $P(\mathbf{x}) = (p_1(\mathbf{x}), \ldots, p_{m_p}(\mathbf{x}))$ is a block of parabolic functions of the form $q_i - x_i^2$.
- $H(\mathbf{x}) = (h_1(\mathbf{x}), \ldots, h_{m_h}(\mathbf{x}))$ is a block of hyperbolic functions of the form $b_k - x_i x_j$.

Hereafter, the x_i's will be refereed to as *primary* variables, and the q_i's and b_i's as *dummy* ones. Also, we will let $n = n_l + n_p + n_h$, and $m = m_l + m_p + m_h$.

As it turns out, all component functions of $F(\mathbf{x})$ are relatively simple. They are either linear functions, or simple quadratic or bilinear functions involving two or three variables each. Moreover, since the $\hat{\mathbf{u}}_i$, $\hat{\mathbf{v}}_i$, and $\hat{\mathbf{w}}_i$ vectors are unit vectors, the maximum ranges for the corresponding

x_i variables are $[-1, 1]$. For the d_i variables appearing in prismatic, cylindrical, or helical joints their range can be easily scaled to $[0, 1]$. With this, the ranges for the q_i's are $[0, 1]$, and for the b_i's are $[-1, 1]$. As a result, the *search space* $\mathcal{B}$ where the solutions of System (23) must be sought for is the Cartesian product of such ranges. In the text below, any subset of this space defined by the Cartesian product of a number of intervals will be referred to as a *box*, and we will write $[x_i^l, x_i^u]$ to denote the interval of a box along dimension i.

4.2 Equation solving

The algorithm starts with the initial box $\mathcal{B}$, and isolates the valid configurations it contains by iterating over two operations, *box shrinking* and *box splitting*. Using box shrinking, portions of $\mathcal{B}$ containing no solution are eliminated by narrowing some of its defining intervals. This process is repeated until either (1) the box is reduced to an empty set, in which case it contains no solution, or (2) the box is "sufficiently" small, in which case it is considered a solution box, or (3) the box cannot be "significantly" reduced, in which case it is bisected into two sub-boxes via box splitting (which simply bisects its largest interval). To converge to all solutions, the whole process is then repeated for the newly created sub-boxes, and for the sub-boxes recursively created thereafter, until one ends up with a collection of (solution) boxes whose side lengths are below a given threshold, σ.

Before further precising this process, we will first see how to eliminate portions of a box that cannot contain any solution. Detailed pseudo-code of the whole strategy will be given later, in Section 4.3.

When reducing any box $\mathcal{B}_c \subseteq \mathcal{B}$ note first that, since any solution inside $\mathcal{B}_c$ must be in the linear variety $L(\mathbf{x}) = 0$, we may shrink $\mathcal{B}_c$ to the smallest possible box bounding the portion of this variety falling inside $\mathcal{B}_c$. The limits of this new box along, say, dimension x_i can be easily found by solving the two linear programs

$$\textbf{LP1: Minimize } x_i, \text{ subject to: } L(\mathbf{x}) = 0, \mathbf{x} \in \mathcal{B}_c,$$
$$\textbf{LP2: Maximize } x_i, \text{ subject to: } L(\mathbf{x}) = 0, \mathbf{x} \in \mathcal{B}_c,$$

giving, respectively, the new lower and upper bounds for x_i. Fig. 5-(a) illustrates the process on the x_i-q_i plane, in the case that $L(\mathbf{x}) = 0$ is a straight line. However, note that $\mathcal{B}_c$ can be further reduced, as the parabolic and hyperbolic equations must also be satisfied.

Regarding the parabolic equations, $q_i = x_i^2$, we incorporate them into the previous linear programs as follows. The section of the parabola lying inside the box $\mathcal{B}'_c = [x_i^l, x_i^u] \times [q_i^l, q_i^u]$ is bounded to lie in the shaded area between lines s and t in Fig. 5-(b). Line s is defined by the intersection points, A_i and B_i, of the parabola with the box. Line, t is the tangent to the parabola parallel to s. The two inequalities defining the area between these lines can be added to **LP1** and **LP2**, which, in conjunction with $L(\mathbf{x}) = 0$, usually produces a much larger reduction of $\mathcal{B}'_c$, as illustrated in Fig. 5-(c).

Regarding the hyperbolic equations, we linearize them as follows. If we consider one of these equations, say $b_k = x_i x_j$, and we know that its variables can take values inside the ranges $x_i \in [a, b]$, $x_j \in [c, d]$, and $b_k \in [e, f]$, all we need is a collection of half-planes tightly delimiting the set of points that satisfy $b_k = x_i x_j$ inside the box $\mathcal{B}'_c = [a, b] \times [c, d] \times [e, f]$. For this purpose, consider the vertexes of the rectangle $[a, b] \times [c, d]$ in the $x_i - x_j$ plane, and lift them vertically to the points D_1, D_2, D_3 and D_4 on the hyperbolic paraboloid $b_k = x_i x_j$, as shown in Fig. 6. Using the fact that this is a doubly-ruled surface, it is easy to see that the tetrahedron defined by D_1, D_2, D_3 and D_4 completely contains the portion of the surface inside $\mathcal{B}'_c$. Hence, to prune portions of a box that do not satisfy the hyperbolic equations, one can simply introduce the half-planes defining this tetrahedron into **LP1** and **LP2** above.

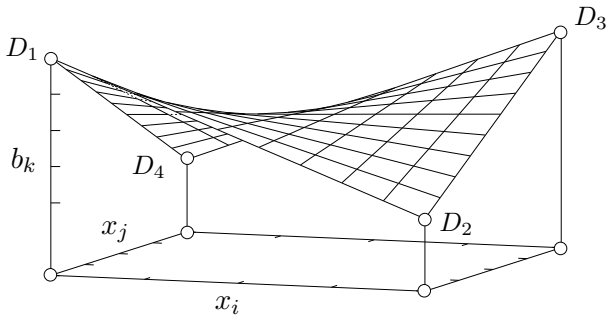


Figure 6: The tetrahedron defined by the D_i's bounds this surface inside $\mathcal{B}'_c$.

Observe that we can define linear relaxations directly for more complex equations such as circle or sphere equations. This reduces the number of dummy variables introduced during equation expansion (Section 4.1), and thus speeds up the execution of each iteration. But such relaxations are in general more conservative, increasing the number of iterations required to solve the problem.

4.3 Pseudocode

Algorithm 1 gives the main loop of the process. As input, it receives the box $\mathcal{B}$, the list F containing the equations $L(\mathbf{x}) = 0$, $P(\mathbf{x}) = 0$, and $H(\mathbf{x}) = 0$, and two threshold parameters σ and ρ. As output, it returns a list S of "solution boxes" that enclose all points of the solution set. The functions VOLUME($\mathcal{B}$) and SIZE($\mathcal{B}$) compute the volume and the length of the longest side of $\mathcal{B}$, respectively. These and other low-level procedures of straightforward implementation will be left unspecified in the algorithms below.

Initially, two lists are set up in lines 1 and 2: an empty list S of solution boxes, and a list P of boxes to be processed, containing $\mathcal{B}$. A **while** loop is then executed until P gets empty (lines 3-18), by iterating the following steps. Line 4 extracts one box from P. Lines 5-9 repeatedly reduce this box as much as possible, via the SHRINK-BOX function, until either the box is an empty set (IS-VOID($\mathcal{B}_c$) is true), or it cannot be significantly reduced ($V_c/V_p > \rho$), or it becomes small enough (SIZE($\mathcal{B}$) $\leq \sigma$). In the latter case, the box is considered a solution for the problem. If a box is neither a solution nor it is empty, lines 14 and 15 split it into two sub-boxes and add them to P for further processing (line 15).

Notice that this algorithm implicitly explores a binary tree of boxes, whose internal nodes are boxes that have been split at some time, and whose leaves are either solution or empty boxes. Solution boxes are collected in list S and returned as output in line 19. Clearly, the tree may be explored in either depth-first or breadth-first order, depending on whether line 15 inserts the boxes at the head or tail of P, getting identical output in any case.

The SHRINK-BOX procedure is sketched in Algorithm 2. It takes as input the box $\mathcal{B}_c$ to shrink, and the list of equations F. The procedure starts by collecting in C all linear equations (line 1), all half planes approximating the parabolic equations (lines 2-4), and, finally, all half planes approximating the hyperbolic equations (lines 5-7). Then, the procedure uses these constraints to reduce every dimension of the box, solving the linear programs in lines 8 to 11, which possibly give tighter bounds for the corresponding intervals. Observe that the linear programs need only be solved for the primary variables $(x_1, \ldots, x_{v_l})$ and not for the dummy ones.

If System (23) has a finite number of isolated solutions, the previous algorithm returns a

```
SOLVE-LINKAGE(B, F, σ, ρ)
 1: S ← ∅
 2: P ← {B}
 3: while P ≠ ∅ do
 4:    B_c ← EXTRACT(P)
 5:    repeat
 6:       V_p ← VOLUME(B_c)
 7:       SHRINK-BOX(B_c, F)
 8:       V_c ← VOLUME(B_c)
 9:    until IS-VOID(B_c) or SIZE(B_c) ≤ σ or V_c/V_p > ρ
10:    if not IS-VOID(B_c) then
11:       if SIZE(B_c) ≤ σ then
12:          S ← S ∪ {B_c}
13:       else
14:          (B_1, B_2)← SPLIT-BOX(B_c)
15:          P ← P ∪ {B_1, B_2}
16:       end if
17:    end if
18: end while
19: return S
```

Algorithm 1: The top-level search scheme.

collection of small boxes containing them all, with each solution lying in one, and only one box. If, on the contrary, the solution space is an algebraic variety of dimension one or higher, the returned boxes will form a discrete envelope of the variety. The precision of the output can be adjusted at will by using the σ parameter, which fixes an upper limit for the width of the widest interval on all returned boxes.

4.4 Performance analysis

The performance of a root finding algorithm is normally assessed in terms of its completeness, correctness, and convergence order.

An algorithm is *complete* if its output includes all solutions of the problem at hand. As for the proposed method, we note that it iterates over two basic operations: the linear relaxation of non-linear functions and the solution of linear programs. Both operations are designed in a conservative way: as defined, a linear relaxation fully includes the graph of the function it approximates (within the box where solutions are sought), and the output of the linear programs always defines an axis-aligned orthotope enclosing the solution set. The proposed method is thus complete, because solution points are never ruled out anywhere in the algorithm. While it is true that numerical issues could arise due to the use of floating-point arithmetic, both in the computation of the linear relaxations and in the solution of the linear programs, these problems can be easily overcome. Linear relaxations can be made conservative by carefully considering the rounding when computing them [29] and, with a cheap post-process, the output of the Simplex method can be correctly interpreted so that it is also numerically safe [37, 23].

An algorithm is *correct* if its output only includes solution points. In the context of branch-and-prune methods, the algorithm is correct if all of the returned boxes contain, at least, one solution each. We next provide a sufficient condition that allows checking the existence of solutions of $F(\mathbf{x}) = 0$ in a given box $\mathcal{B}_c$. To this end, consider the system formed by $L(\mathbf{x}) = 0$ together with the linear relaxations of $P(\mathbf{x}) = 0$ and $H(\mathbf{x}) = 0$, derived for $\mathcal{B}_c$ as explained

SHRINK-BOX($\mathcal{B}_c, F$)
1: $C \leftarrow$ { Linear equations in F }
2: **for all** equations $q_i = x_i^2$ in F **do**
3: $C \leftarrow C \cup$ { Two half planes bounding the feasible area of the equation for the ranges of q_i, x_i }
4: **end for**
5: **for all** equations $b_k = x_i x_j$ in F **do**
6: $C \leftarrow C \cup$ { Four half planes bounding the feasible area of the equation for the ranges of b_k, x_i, x_j }
7: **end for**
8: **for all** $i \in \{1, \ldots, v_l\}$ **do**
9: $x_i^l \leftarrow$ min. x_i subject to all eqs. in C and $\mathbf{x} \in \mathcal{B}_c$
10: $x_i^u \leftarrow$ max. x_i subject to all eqs. in C and $\mathbf{x} \in \mathcal{B}_c$
11: **end for**

Algorithm 2: The SHRINK-BOX procedure.

in Section 4.2. Note that the solution set of this system is a convex polytope $\mathcal{P}(\mathcal{B}_c) \subset \mathbb{R}^n$. If $F(\mathbf{x}) = 0$ has as many variables as equations (i.e., $n = m$), the following existence condition can be used:

If $\mathcal{P}(\mathcal{B}_c) \subset \mathcal{B}_c$, then $\mathcal{B}_c$ contains at least one solution point of $F(\mathbf{x}) = 0$.

Note that, in practice, the condition can be easily checked by deriving the smallest orthotope enclosing $\mathcal{P}(\mathcal{B}_c)$ via Linear Programming, and checking whether it is contained in $\mathcal{B}_c$.

To prove the condition, we first realize that, by linearizing $F(\mathbf{x})$ about a point $\mathbf{x}_c \in \mathcal{B}_c$, $F(\mathbf{x}) = 0$ can be written as

$$F(\mathbf{x}_c) + J_F(\mathbf{x}_c)\,(\mathbf{x} - \mathbf{x}_c) + \varepsilon(\mathbf{x}, \mathbf{x}_c) = 0, \tag{24}$$

where $J_F(\mathbf{x}_c)$ is the Jacobian of $F(\mathbf{x})$ at $\mathbf{x}_c$, and $\varepsilon(\mathbf{x}, \mathbf{x}_c)$ is a second-order error term. But, if $m = n$ and $\mathbf{C}$ is full rank, Eq. (24) is equivalent to

$$\mathbf{x} = J_F(\mathbf{x}_c)^{-1}(J_F(\mathbf{x}_c)\,\mathbf{x}_c - F(\mathbf{x}_c) - \varepsilon(\mathbf{x}, \mathbf{x}_c)). \tag{25}$$

Thus, finding the solutions of $F(\mathbf{x}) = 0$ is equivalent to finding the fixed points of the right hand side of Eq. (25), $M(\mathbf{x}) = J_F(\mathbf{x}_c)^{-1}(J_F(\mathbf{x}_c)\,\mathbf{x}_c - F(\mathbf{x}_c) - \varepsilon(\mathbf{x}, \mathbf{x}_c))$. Since $\mathbf{y} = M(\mathbf{x})$ maps points $\mathbf{x} \in \mathcal{B}_c$ to points $\mathbf{y} \in \mathcal{P}(\mathcal{B}_c)$, by Brouwer's fixed point theorem [6] we can assert that, if $\mathcal{P}(\mathcal{B}_c) \subset \mathcal{B}_c$, then there exists an $\mathbf{x}^* \in \mathcal{B}_c$ for which $\mathbf{x}^* = M(\mathbf{x}^*)$, which implies that $\mathbf{x}^*$ is a solution of $F(\mathbf{x}) = 0$.

It is worth mentioning that the existence condition just described is less restrictive than other sufficient criteria [24, 34, 5], yet easier to integrate in our framework. Also, while the test proposed by Miranda [34] can be extended to non-squared systems, the resulting sufficient condition is too weak to be useful in general. To our knowledge, no results are available to derive necessary and sufficient conditions for the existence of solutions in a given box. Therefore, as it happens on all algorithms of this kind, our algorithm can in principle return boxes for which it is not possible to elucidate whether they include a solution. In any case, the error in all function approximations, $\varepsilon(\mathbf{x}, \mathbf{x}_c)$, is quadratic with respect to the size of the box. Since the algorithm returns boxes whose largest side is below σ, the error in the equations is always $O(\sigma^2)$. Thus, only boxes with small errors can be misleadingly taken as solutions. In practice this occurs for linkage configurations that are close to a singularity.

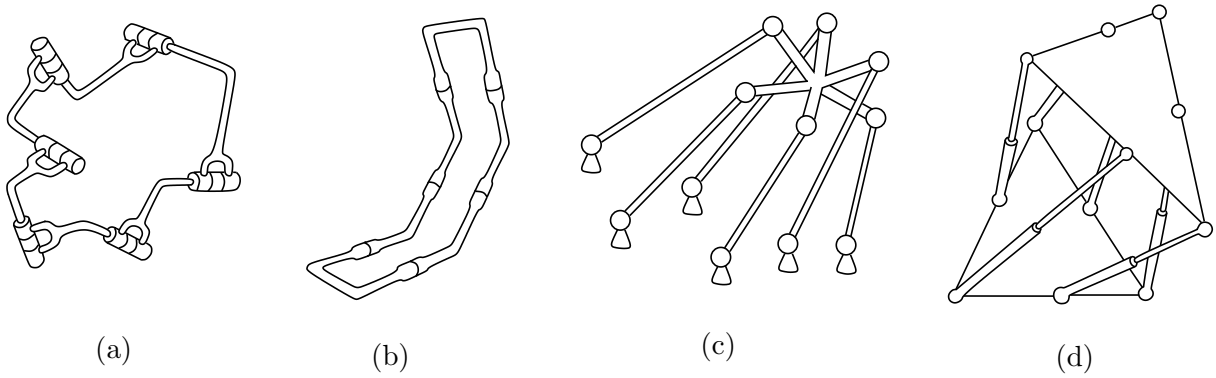


Figure 7: Test cases analyzed: (a) A generic 6R loop, (b) a special 6R loop, (c) a generic 6-6 Stewart platform, and (d) a special 6-6 Stewart platform.

General 6R				Parameters interpretation	Special 6R			
i	a_i	d_i	α_i		i	a_i	d_i	α_i
1	0.3	0.0106	$\pi/2$		1	0.5	1	$\pi/3$
2	1	0	0.0175		2	0	1	$\pi/3$
3	0	0.2	$\pi/2$		3	0	1	$\pi/3$
4	1.5	0	0.0175		4	0.5	1	$\pi/3$
5	0	0	$\pi/2$		5	0	1	$\pi/3$
6	1.1353	0.1049	1.4716		6	0	1	$\pi/3$

Table 2: Denavit-Hartenberg parameters of the solved 6R loops.

	Generic 6-6 Stewart platform			Special 6-6 Stewart platform		
i	$\mathbf{q}_i^{\mathcal{B}}$	$\mathbf{q}_i^{\mathcal{P}}$	l_i	$\mathbf{q}_i^{\mathcal{B}}$	$\mathbf{q}_i^{\mathcal{P}}$	l_i
1	$(0, 0, 0)$	$(0, 0, 0)$	$l_1 = 1$	$(0, 0, 0)$	$(0, 0, 0)$	1.519640
2	$(1.107915, 0, 0)$	$(0.542805, 0, 0)$	$l_2 = 0.645275$	$(c, s, 0)$	$(-c, s, 0)$	1.922131
3	$(0.549094, 0.756063, 0)$	$(0.956919, -0.528915, 0)$	$l_3 = 1.086284$	$(2c, 2s, 0)$	$(c, s, 0)$	1.812880
4	$(0.735077, -0.223935, 0.525991)$	$(0.665885, -0.353482, 1.402538)$	$l_4 = 1.503439$	$(1 + c, s, 0)$	$(3c, s, 0)$	1.380117
5	$(0.514188, -0.526063, -0.368418)$	$(0.478359, 1.158742, 0.107672)$	$l_5 = 1.281933$	$(2, 0, 0)$	$(2c, 0, 0)$	1.715536
6	$(0.590473, 0.094733, -0.205018)$	$(-0.137087, -0.235121, 0.353913)$	$l_6 = 0.771071$	$(1, 0, 0)$	$(c, -s, 0)$	1.714524

Table 3: Geometric parameters of the solved 6-6 platforms. The letters s and c abbreviate the sine and cosine of $\pi/3$, respectively.

The *convergence order* of a root finding algorithm gives information about its asymptotic performance. An algorithm is said to exhibit a convergence of order r if there exists a constant $k \in (0, 1)$, such that

$$\epsilon(\mathbf{x}_{i+1}, \mathbf{x}^*) \leq k \cdot \epsilon(\mathbf{x}_i, \mathbf{x}^*)^r,$$

where $\mathbf{x}_i$ is an estimation of the exact root $\mathbf{x}^*$ at iteration i, and $\epsilon(\mathbf{x}_i, \mathbf{x}^*)$ indicates the distance from $\mathbf{x}_i^*$ to $\mathbf{x}^*$. As mentioned in Section 2, branch-and-prune methods rely on conservative bounds to discard subsets of the input domain that do not containt solutions. The tighter the bounds, the faster the convergence of the method. Therefore, we can compare the convergence order of different families of branch-and-prune methods by comparing the quality of the bounds used in each method.

The recursion used by the Newton method is derived from applying the mean value theorem to the individual functions f_i of $F(\mathbf{x})$, at some point $\mathbf{x}_c \in \mathcal{B}_c$,

$$f_i(\mathbf{x}) = f_i(\mathbf{x}_c) + \nabla \mathbf{f}_i(\zeta)\ (\mathbf{x} - \mathbf{x}_c).$$

Here, ζ is also a point of $\mathcal{B}_c$, but it is in general unknown. The interval extension of the Newton recursion overcomes this problem using an interval evaluation of $\nabla \mathbf{f}_i(\zeta)$ for all possible $\zeta \in \mathcal{B}_c$. However, if all functions f_i are quadratic (as it occurs in our formulation), any $f_i(\mathbf{x})$ can be expressed in the form

$$f_i(\mathbf{x}) = \mathbf{x}^\top\ \mathbf{A}_i\ \mathbf{x} + \mathbf{b}_i^\top\ \mathbf{x} + c_i,$$

where $\mathbf{A}_i$ is an $n \times n$ symmetric matrix, and $\mathbf{b}_i$ is an n-dimensional vector. Thus, we can write

$$\begin{aligned}
\nabla \mathbf{f}_i(\mathbf{x}) &= (2\ \mathbf{A}_i\ \mathbf{x} + \mathbf{b}_i)^\top \\
&= (2\ \mathbf{A}_i\ (\mathbf{x}_c + \mathbf{x} - \mathbf{x}_c) + \mathbf{b}_i)^\top \\
&= (2\ \mathbf{A}_i\ \mathbf{x}_c + \mathbf{b}_i)^\top + (2\ \mathbf{A}_i\ (\mathbf{x} - \mathbf{x}_c))^\top \\
&= (2\ \mathbf{A}_i\ \mathbf{x}_c + \mathbf{b}_i)^\top + (\mathbf{x} - \mathbf{x}_c)^\top 2\ \mathbf{A}_i \\
&= \nabla \mathbf{f}_i(\mathbf{x}_c) + (\mathbf{x} - \mathbf{x}_c)^\top \mathbf{H}_{f_i},
\end{aligned}$$

where $\mathbf{H}_{f_i}$ is the Hessian of f_i, which is constant. Therefore, each f_i can be written as

$$f_i(\mathbf{x}) = f(\mathbf{x}_c) + \nabla \mathbf{f}_i(\mathbf{x}_c)\ (\mathbf{x} - \mathbf{x}_c) + (\zeta - \mathbf{x}_c)^\top \mathbf{H}_{f_i}\ (\mathbf{x} - \mathbf{x}_c).$$

On the other hand, a quadratic function can be exactly represented by its first-order Taylor expansion about $\mathbf{x}_c$ as

$$f_i(\mathbf{x}) = f(\mathbf{x}_c) + \nabla \mathbf{f}_i(\mathbf{x}_c)\ (\mathbf{x} - \mathbf{x}_c) + \frac{1}{2}(\mathbf{x} - \mathbf{x}_c)^\top \mathbf{H}_{f_i}\ (\mathbf{x} - \mathbf{x}_c).$$

Comparing the last two equations, we see that the mean value approximation is exact when $\zeta = (\mathbf{x} + \mathbf{x}_c)/2$. Therefore, if $\mathbf{x}$ is subject to lie in $\mathcal{B}_c$, ζ can only be in

$$\mathcal{B}'_c = \{\mathbf{x}' \mid \mathbf{x}' = \frac{\mathbf{x} + \mathbf{x}_c}{2},\ \mathbf{x} \in \mathcal{B}_c\}.$$

Clearly, $\mathcal{B}'_c$ is fully included in $\mathcal{B}_c$ and, thus, by evaluating $\nabla \mathbf{f}_i(\zeta)$ for all ζ in $\mathcal{B}_c$, and not only in $\mathcal{B}'_c$, the interval Newton method overestimates the error. For this reason, at least for quadratic functions, branch-and-prune methods based on first-order Taylor approximations, like those used in [16], converge faster than the interval Newton method, which is known to be quadratically convergent [2]. Since the approximations derived from linear relaxations are equal or tighter than those derived from first-order Taylor approximations, we can conclude that the convergence order of methods based on the former is equal or greater than methods based on the latter.

Note finally that, contrarily to interval Newton methods, our method can be applied to under- or over-constrained systems. On over-constrained systems, our method exhibits the same convergence order than when applied to well-constrained ones, since the addition of extra equations does not hinder the convergence in any way. On the contrary, redundancy produces larger box reductions in Shrink-Box and thus reduces the number of iterations. The drawback is that the higher the number of equations, the slower the execution of each iteration. On under-constrained systems, the convergence order of the algorithm is difficult to derive precisely. A worst-case analysis, though, sheds some light on it. Note that, for an n-dimensional box $\mathcal{B}_c$ all of whose sides are of length σ, the error at step i is

$$\epsilon_i = \sqrt{\sigma^2\, n} = \sigma\sqrt{n}.$$

The worst possible case occurs when the Shrink-Box procedure is completely ineffective, which makes the method rely on bisection only to isolate the solutions. Should this be the case, after splitting $\mathcal{B}_c$ the error on each one of the child boxes would be

$$\epsilon_{i+1} = \sqrt{\sigma^2\,(n-1) + \frac{\sigma^2}{4}} = k\,\epsilon_i.$$

where $k = \sqrt{(4\,n-3)/(4\,n)}$. Thus, in this situation the method would exhibit linear convergence, with k approaching one (i.e., to the non-convergence case) as the number of variables grows. We point out, however, that the worst-case just depicted is rather improbable and, in fact, experiments show that when isolating positive-dimensional solutions, the convergence order is linear, but k is always substantially smaller than $\sqrt{(4\,n-3)/(4\,n)}$ because the Shrink-Box procedure always performs some reduction.

5 Experiments

The algorithm has been implemented in C, using the *glpk* library to solve the linear programs involved [31]. We next illustrate its performance on a Pentium Core 2 at 2.4 GHz by way of solving the four test cases shown in Fig. 7: a generic 6R loop, a generic 6-6 Stewart platform, and special versions of these two linkages. Efficient specific solutions for them were obtained in 1993 [45] and 2001 [50], respectively, via elimination techniques. We note, however, that although the methods in [45] and [50] can solve the generic versions of such linkages, none of them can isolate the one-dimensional configuration space of mobility-one instances like those in Fig. 7 (b) and (d).

The algorithm has also been tested successfully on numerous other examples, ranging from planar linkages to spatial robots and molecules. Details on such experiments (including their formulation, output solutions, and linkage animations) are provided in [51]. In all cases, the presented method is more than one order of magnitude faster than general polytope methods like [48], whose implementation is notably more intricate.

5.1 Solving generic and special 6R loops

Loops with six revolute joints typically arise when solving the inverse kinematics of serial 6R robot arms. Since in this context the ground link and the end effector have fixed relative positions, the problem boils down to finding all possible configurations of a 6R loop. The geometry of 6R loops is easily described in terms of Denavit-Hartenberg parameters, provided in Table 2 for the two linkages herein analysed. The parameters in Table 2, left, correspond to problem number 6 introduced in [53]. It has a generic geometry, with no special dimensions or line coincidences, and exhibits 16 isolated solutions. The parameters in Table 2, right, correspond

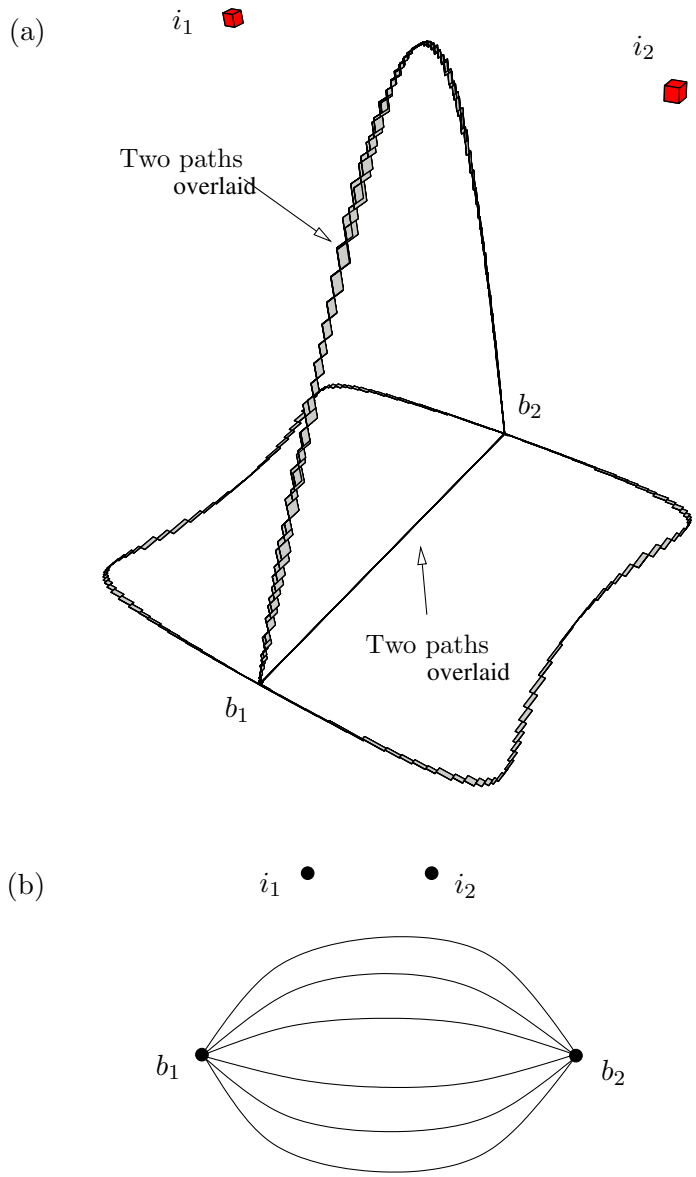


Figure 8: (a) One-dimensional configuration space of the Bricard loop, as computed by the presented technique. Actually, the boxes are 22-dimensional, but are here shown projected onto three of the problem's variables. Boxes corresponding to the two isolated solutions have been enlarged to make them visible. (b) The actual topology of the configuration space.

to a special Bricard linkage. Because all of its pairs of adjacent axes are intersecting (two of them at infinity), this linkage exhibits a one-dimensional solution set, with bifurcation points and additional isolated solutions. We next show that the parameters required by our formulation can be easily written in terms of Denavit-Hartenberg parameters.

In our formulation, the system to be solved is formed by the loop equation (8), and equations (2) to (6) gathered for all joints. Thus, we need to obtain the vectors $\mathbf{a}_i^{\mathcal{F}_i}$, $\mathbf{d}_i^{\mathcal{F}_i}$, and $\mathbf{d}_i^{\mathcal{F}_{i+1}}$. As done in the Denavit-Hartenberg convention, we number the links consecutively from 1 to n, and define a reference frame $\mathcal{F}_i$ for each link L_i, with the z_i axis directed along the axis of the

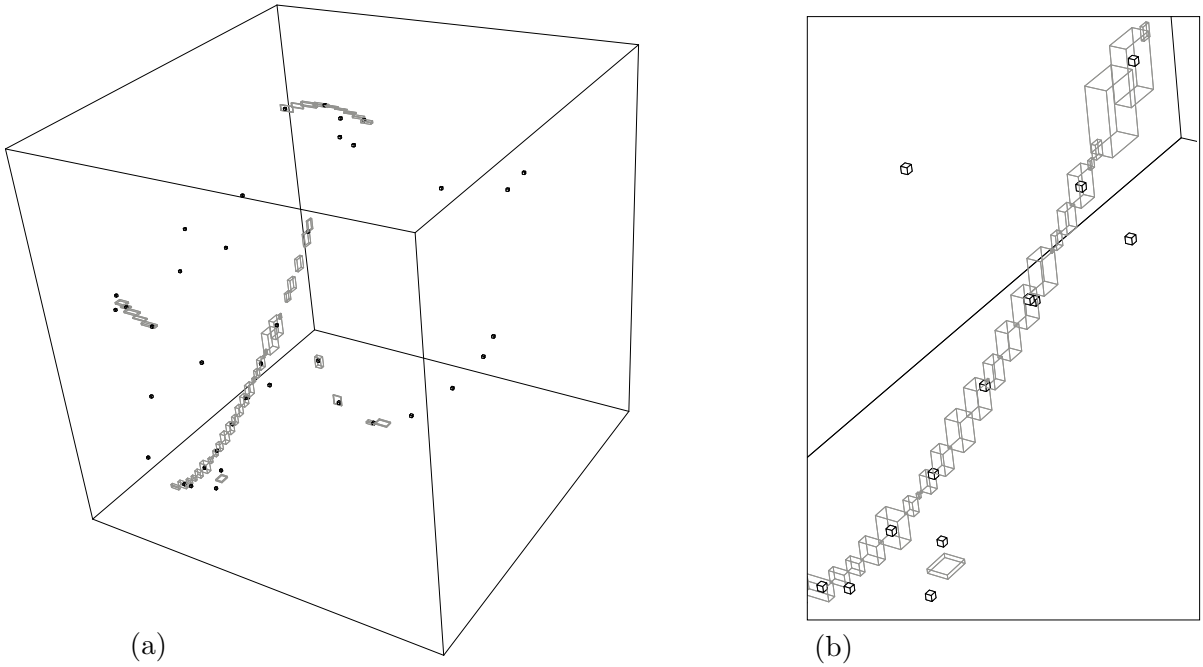


Figure 9: Left: Dietmeier's 6-6 platform, solved at two different precisions. Right: A closer view of the output. In the two plots, transparent and opaque boxes correspond to two different runs, at $\sigma = 0.1$ and $\sigma = 10^{-7}$, respectively. All opaque boxes have been slightly enlarged to make them visible.

$i+1$-th joint, and the x_i axis directed along the normal line through joint axes i and $i+1$, with both axes pointing towards a selected positive sense for the loop.

It is easy to realise that, if we select the P_i and Q_i points of joint i as indicated in Table 2, middle figure, then

$$\mathbf{a}_i^{\mathcal{F}_i} = (a_i, 0, 0), \tag{26}$$

$$\mathbf{d}_i^{\mathcal{F}_i} = (0, 0, d_i), \tag{27}$$

$$\mathbf{d}_i^{\mathcal{F}_{i+1}} = (0, d_i \sin\alpha_{i+1}, d_i \cos\alpha_{i+1}), \tag{28}$$

where:

- a_i is the distance between joints i and $i+1$ along their common normal.
- d_i is the distance between consecutive normals along joint i.
- α_i is the angle from the z_{i-1} axis to z_i axis, turning around the direction of the positive x_i axis.

In sum, as for the problem formulation, we obtain a system of 28 linear, 21 parabolic and 23 hyperbolic equations, involving 23 primary variables and 44 dummy ones.

Choosing the parameters in Table 2, left, and setting $\sigma = 10^{-2}$ and $\rho = 0.95$, we solve the generic 6R loop in about 14 seconds, correctly isolating 16 boxes corresponding to the solutions published in [32]. In this case, the system processed 131 boxes, 16 of which contain a solution, 50 were found to be empty, and 65 were split for recursive processing. All 16 solution boxes are validated to include a solution point according to the existence condition described in Section 4.4.

Choosing the parameters in Table 2, right, the 6R loop becomes an overconstrained mechanism. While existing methods like [45, 32] can not deal with this degenerate case, the proposed procedure is immune to such situations and obtains a complete box approximation of the whole configuration space, as shown in Fig. 8 (a). With $\sigma = 0.1$ and $\rho = 0.95$, we obtain the shown 1863 boxes in 23 seconds, after processing 3749 boxes, 12 of which were found to be empty. The number of empty boxes is pretty small, taking into account the total amount of processed boxes and solutions, indicating that the box-shrinking strategy is efficient. Note that, ideally, by iterating box-shrinking, one should end up with a box with solutions lying on its walls and, therefore, splitting a box at such point should always separate portions of the search space containing solutions. In other words, the ideal algorithm should not generate empty boxes.

From the output of the algorithm we can readily define a graph with one node for each box and one edge joining two nodes if their corresponding boxes are adjacent. Note that the structure of this graph reflects the structure of the configuration space of the analysed linkage. For the special 6R loop, the analysis reveals that such space has two isolated points, corresponding to two rigid assembly modes, and a one-dimensional cyclic curve, corresponding to a one-degree-of-freedom assembly mode. Such curve, moreover, has two bifurcation points connected by six different paths. The actual topology is represented in Fig. 8 (b).

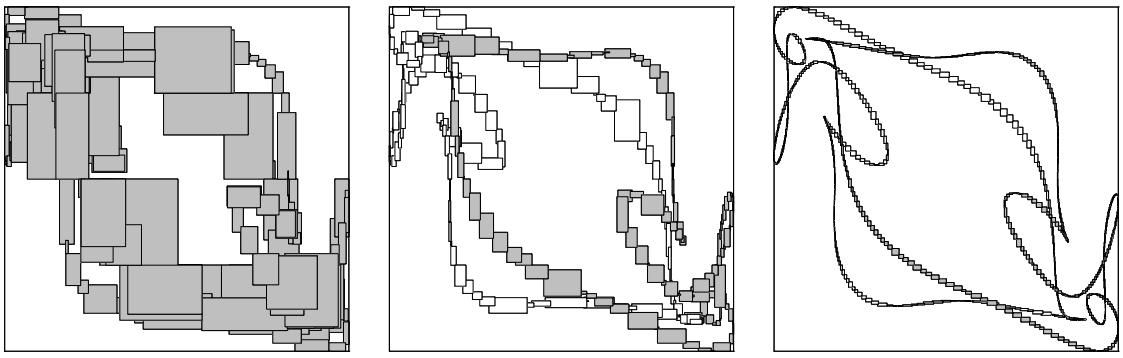
5.2 Solving generic and special 6-6 platforms

A generic 6-6 Stewart platform can have up to 40 solutions. One case giving rise to exactly 40 solutions was found by Dietmeier [12], with the geometric parameters indicated in Table 3, left. For each leg, the table gives the coordinates of the base ($\mathbf{q}_i^{\mathcal{B}}$) and platform ($\mathbf{q}_i^{\mathcal{P}}$) anchor points, relative to base and platform frames respectively, and the leg length (l_i). The problem formulation involves 24 equations in 24 variables, derived from five kinematic loops. When solving the system with $\sigma = 10^{-3}$ and $\rho = 0.95$ we obtain 35 isolated solutions and several one-dimensional "curve segments" of boxes that include the remaining 11 solutions (Fig. 9). All points included in these curve segments are quasi solutions: recall that the error is quadratic with the size of the boxes and, thus for $\sigma = 10^{-3}$ the error in the returned boxes is below $\sigma^2 = 10^{-6}$. Therefore, the presence of curve segments of quasi solutions indicates that the linkage is close to a singular configuration. It is possible to isolate the true solution points within these curve segments by just re-running the method with a smaller σ. For $\sigma = 10^{-7}$, our implementation isolates the correct 40 solutions in 260 seconds, after processing 3395 boxes, 1658 of which were found to be empty. In this case, also all the solutions boxes are verified to include a solution point.

Choosing the geometric parameters of Table 3, the 6-6 platform becomes moveable with one degree of freedom (despite it has all of its leg-lengths fixed). We highlight that the problem formulation for this case is identical to the one used for the general 6-6 platform, only differing on the mentioned geometric parameters. Fig. 10 shows the solution boxes obtained by the algorithm, for decreasing values of the σ parameter, projected onto two of the problem's dimensions. For $\sigma = 0.5$, the obtained box approximation is too crude to reveal the toplogy of the configuration space. However, as we reduce σ, two separated one-dimensional components rapidly emerge (depicted in white and grey in the figure), allowing for a correct motion analysis of the linkage at hand. Finally, Fig. 11 is a 3D version of the last plot in Fig. 10 where the two connected components can be better appreciated.

6 Conclusions

We have presented a complete method able to give box approximations of the configuration space of arbitrary multi-loop linkages. The method is *general*, in the sense that it can manage



$\sigma = 0.5$, $t = 94$, $n_s = 193$ $\sigma = 0.25$, $t = 109$, $n_s = 256$ $\sigma = 0.0625$, $t = 140$, $n_s = 703$

Figure 10: Box approximations obtained for the Griffis-Duffy platform, at three different resolutions, projected onto two of the problem's variables. In each plot we indicate the σ parameter used, the CPU seconds employed (t), and the number of solution boxes found (n_s).

any type of links and lower-pair joints, forming kinematic loops of arbitrary topology. It is also *complete*, meaning that every solution point will be contained in one of the returned boxes. Moreover, in all experiments done so far the algorithm was also *correct*, since, by using a small enough σ value, all output boxes contained at least one solution point each. Although we cannot verify the presence of solutions in all boxes, returning boxes with no solution seems rather improbable due to the fact that the linearizations introduce errors smaller than the size of the considered boxes. Moreover, the fact that all equations are simultaneously taken into account during box reduction (whether directly or in a linearized form) palliates the so-called *cluster effect*, a known problem of bisection-based techniques of this kind, whereby each solution is obtained as a compact cluster of boxes instead of a single box containing it, irrespectively of the precision used [35]. In the experiments performed so far, we encountered spurious output on linkages with close-to-singular configurations, but this can not be attributed to clustering problems since the phenomenon disappeared when running the algorithm at smaller σ values.

A main contribution with respect to previous work is the method's ability to deal with configuration spaces of general structure. This is accomplished by maintaining a collection of boxes that form a tight envelope of such spaces, which can be refined to the desired precision in a multi-resolutive fashion. The method is quadratically convergent to all roots if these are isolated points, and linearly convergent to them if these form positive-dimensional connected components. Although the method's performance is notable for a general technique of this kind, an extensive study should be endeavored to determine how it scales with the complexity of the tackled linkages.

References

[1] C. S. Adjiman, S. Dallwig, C. A. Foudas, and A. Neumaier. A global optimization method, αBB, for general twice-differentiable constrained NLPs - I theoretical advances. *Computer and Chemical Engineering*, 22:1137–1158, 1998.

[2] G. Alefeld and J. Herzberger. *Introduction to Interval Computations*. Academic Press, Orlando, Florida, 1983.

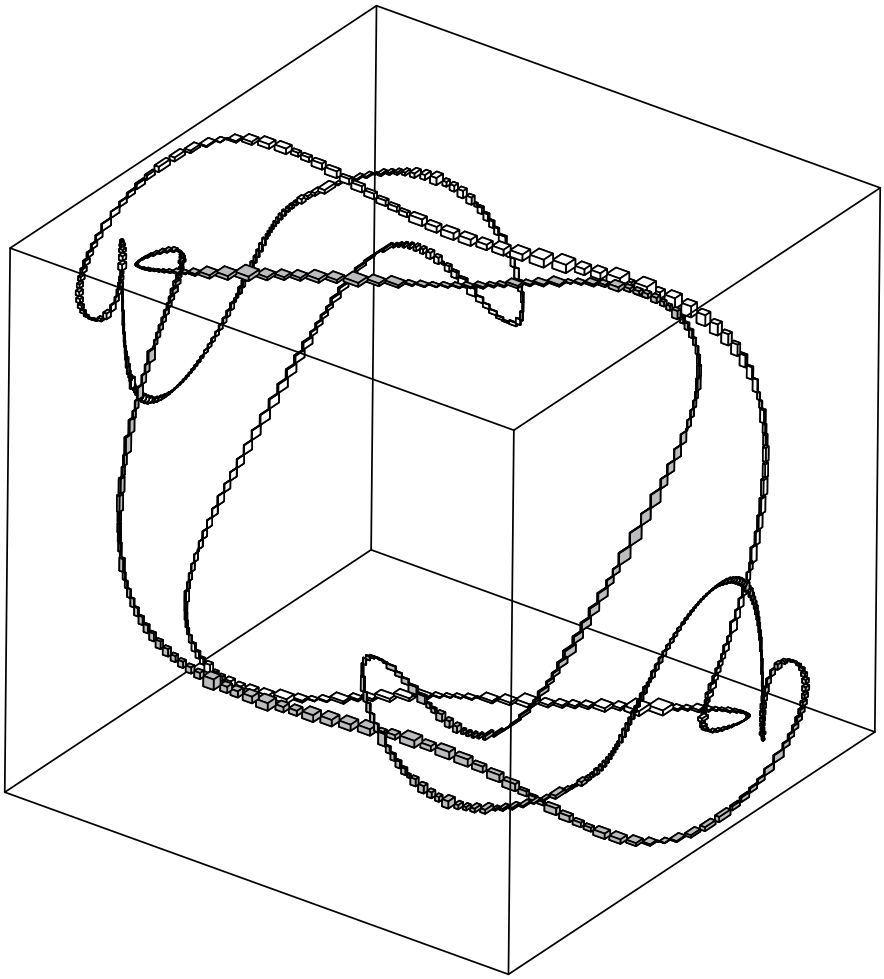


Figure 11: Box approximation obtained for the Griffis-Duffy platform, for $\sigma = 0.0625$, projected onto three of the problem's variables.

[3] M. Berz and G. Hoffstätter. Computation and applications of Taylor polynomials with interval remainder bounds. *Reliable Computing*, 4:83–97, 1998.

[4] C. Borcea and I. Streinu. The number of embeddings of a minimally-rigid graph. *Discrete and Computational Geometry*, 31(2):287–303, 2004.

[5] K. Borsuk. Drei sätze über die n-dimensionale sphäre. *Fund. Math.*, 20:177–190, 1933.

[6] L. E. J. Brouwer. Ueber eineindeutige, stetige transformationen von flächen in sich. *Math. Ann.*, 69:176–180, 1910.

[7] A. Castellet and F. Thomas. An algorithm for the solution of inverse kinematics problems based on an interval method. In M. Husty and J. Lenarcic, editors, *Advances in Robot Kinematics*, pages 393–403. Kluwer Academic Publishers, 1998.

[8] E. Celaya, T. Creemers, and L. Ros. Exact interval propagation for the efficient solution of planar linkages. In *Proceedings of 12th World Conference in Mechanism and Machine Science*, 2007.

[9] D. Daney, Y. Papegay, and A. Neumainer. Interval methods for certification of the kinematic calibration of parallel robots. In *IEEE International Conference on Robotics and Automation*, pages 1913–1918, 2004.

[10] A. Dickenstein and I. Z. Emiris. *Solving polynomial equations: Foundations, algorithms, and applications*. Algorithms and Computation in Mathematics. Springer-Verlag, 2005.

[11] O. Didrit, M. Petitot, and E. Walter. Guaranteed solution of direct kinematic problems for general configurations of parallel manipulators. *IEEE Transactions on Robotics and Automation*, 14(2):259–266, 1998.

[12] P. Dietmeier. The Stewart-Gough platform of general geometry can have 40 real postures. In J. Lenarcic and M. Husty, editors, *Advances in Robot Kinematics: Analysis and Control*, pages 7–16. Springer, 1998.

[13] I. Fudos and C. M. Hoffmann. A graph-constructive approach to solving systems of geometric constraints. *ACM Transactions on Graphics*, 16(2):179–216, 1997.

[14] C. B. Garcia and T. Y. Li. On the number of solutions to polynomial systems of equations. *SIAM Journal of Numerical Analysis*, 17:540–546, 1980.

[15] C. B. Garcia and W. I. Zangwill. *Pathways to solutions, fixed points, and equilibria*. Prentice Hall, Upper Saddle River, NJ, 1981.

[16] M. Gavriliu. *Towards More Efficient Interval Analysis: Corner Forms and a Remainder Interval Newton Method*. PhD thesis, California Institute of Technology, 2005.

[17] H.-J. Su, J. M. McCarthy, M. Sosonkina, and L. T. Watson. Algorithm 857: POLSYS GLP-a parallel linear product homotopy code for solving polynomial systems of equations. *ACM Transactions on Mathematical Software*, 32(4):561–579, 2006.

[18] E. Hansen. *Global Optimization Using Interval Analisis*. Marcel Dekker Inc., New York, 1992.

[19] C. H. Hoffmann and B. Yuan. On spatial constraints solving approaches. In *Third International Workshop on Automated Deduction in Geometry*, volume 2061 of *Lecture Notes in Computer Science*, 2000.

[20] J.-P. Merlet. *Parallel Robots*. Springer, 2000.

[21] J.-P. Merlet. A formal numerical approach to determine the presence of singularity within the workspace of a parallel robot. In *Proceedings of the 2nd Workshop on Computational Kinematics*, pages 167–176, 2001.

[22] J.-P. Merlet. An improved design algorithm based on interval analysis for parallel manipulator with specified workspace. In *IEEE International Conference on Robotics and Automation*, pages 1289–1294, 2001.

[23] C. Jansson. Rigorous lower and upper bounds in linear programming. *SIAM Journal of Optimization*, 14(3):914–935, 2004.

[24] L. Kantorivich. On Newton's method for functional equations. *Dokl. Akad. Nauk SSSR*, 59:1237–1240, 1948.

[25] R. B. Kearfott. Discussion and empirical comparisons of linear relaxations and alternate techniques in validated deterministic global optimization. *Optimization Methods and Software*, 21(5):715–731, 2006.

[26] L. V. Kolev. A new method for global solution of systems of non-linear equations. *Reliable Computing*, 4:125–146, 1998.

[27] P. Kumar and S. Pellegrino. Computation of kinematic paths and bifurcation points. *International Journal of Solids and Structures*, (37):7003–70027, 2000.

[28] L. -W. Tsai and A. Morgan. Solving the kinematics of the most general six- and five-degree-of-freedom manipulators by continuation methods. *ASME Journal of Mechanisms, Transmissions, and Automation in Design*, 107:189–200, 1985.

[29] Y. Lebbah, C. Michel, M. Rueher, D. Daney, and J.-P. Merlet. Efficient and safe global constraints for handling numerical constraint systems. *SIAM Journal of Numerical Analysis*, 42(5):2076–2097, 2005.

[30] T. Y. Li, T. Sauer, and J. A. York. The cheater's homotopy: An efficient procedure for solving systems of polynomial equations. *SIAM Journal of Numerical Analysis*, 18(2):173–177, 1988.

[31] A. Makhorin. GLPK - The GNU linear programming toolkit. `http://www.gnu.org/software/glpk`.

[32] D. Manocha and J. Canny. Efficient inverse kinematics for general 6R manipulators. *IEEE Transactions on Robotics and Automation*, 10:648–657, 1994.

[33] G. P. McCormick. Computability of global solutions to factorable nonconvex programs: Part I - convex underestimating problems. *Mathematical Programming*, 10:147–175, 1976.

[34] C. Miranda. Un'osservatione su un teorema di Brouwer. *Bolletino dell'Unione Matematica Italiana*, 3(2):5–7, 1940.

[35] A. Morgan and V. Shapiro. Box-bisection for solving second-degree systems and the problem of clustering. *ACM Transactions on Mathematical Software*, 13(2):152–167, 1987.

[36] A. Neumaier. Taylor forms–use and limits. *Reliable Computing*, 9:43–79, 2002.

[37] A. Neumaier and O. Shcherbina. Safe bounds in linear and mixed-integer programming. *Mathematical Programming*, 99:283–296, 2004.

[38] J. Nielsen and B. Roth. On the kinematic analysis of robotic mechanisms. *The International Journal of Robotics Research*, 18(12):1147–1160, 1999.

[39] J. Nielsen and B. Roth. Solving the input/output problem for planar mechanisms. *ASME Journal of Mechanical Design*, 121:206–211, 1999.

[40] J. M. Porta. CuikSlam: A kinematics-based approach to SLAM. In *IEEE International Conference on Robotics and Automation*, pages 2436–2442, 2005.

[41] J. M. Porta, L. Ros, T. Creemers, and F. Thomas. Box approximations of planar linkage configuration spaces. *ASME Journal of Mechanical Design*, 129(4):397–405, 2007.

[42] J. M. Porta, L. Ros, and F. Thomas. Multi-loop position analysis via iterated linear programming. In *Robotics: Science and Systems II*, pages 169–178. MIT Press, 2006.

[43] J. M. Porta, L. Ros, F. Thomas, F. Corcho, J. Cantó, and J. J. Pérez. Complete maps of molecular-loop conformational spaces. *Journal of Computational Chemistry*, 28(13):2170–2189, 2007.

[44] M. Raghavan. The Stewart platform of general geometry has 40 configurations. *ASME Journal of Mechanical Design*, 115:277–282, 1993.

[45] M. Raghavan and B. Roth. Inverse kinematics of the general 6R manipulator and related linkages. *Transactions of the ASME Journal of Mechanical Design*, (115):502–508, 1993.

[46] R. S. Rao, A. Asaithambi, and S. K. Agrawal. Inverse kinematic solution of robot manipulators using interval analysis. *ASME Journal of Mechanical Design*, 120:147–150, 1998.

[47] B. Roth and F. Freudenstein. Synthesis of path-generating mechanisms by numerical methods. *ASME Journal of Engineering for Industry*, 85:298–307, 1963.

[48] E. C. Sherbrooke and N. M. Patrikalakis. Computation of the solutions of nonlinear polynomial systems. *Computer Aided Geometric Design*, 10(5):379–405, 1993.

[49] A. J. Sommese and C. W. Wampler. *The Numerical Solution of Systems of Polynomials Arising in Engineering and Science*. World Scientific, 2005.

[50] T.-Y. Lee and J.-K. Shim. Forward kinematics of the general 6-6 Stewart platform using algebraic elimination. *Mechanism and Machine Theory*, (36):1073–1085, 2001.

[51] The CUIK project home page. `http://www-iri.upc.es/groups/gmr/cuikweb`.

[52] C. Wampler, A. Morgan, and A. Sommese. Complete solution of the nine-point path synthesis problem for four-bar linkages. *ASME Journal of Mechanical Design*, 114:153–159, 1992.

[53] C. Wampler and A. P. Morgan. Solving the 6R inverse position problem using a generic-case solution methodology. *Mechanism and Machine Theory*, 26(1):91–106, 1991.

[54] C. W. Wampler. Solving the kinematics of planar mechanisms by Dixon's determinant and a complex plane formulation. *ASME Journal of Mechanical Design*, 123:382–387, 2001.

[55] W. J. Wedemeyer and H. Scheraga. Exact analytical loop closure in proteins using polynomial equations. *Journal of Computational Chemistry*, 20(8):819–844, 1999.

[56] J. H. Yakey, S. M. LaValle, and L. E. Kavraki. Randomized path planning for linkages with closed kinematic chains. *IEEE Transactions on Robotics and Automation*, 17(6):951–958, 2001.

[57] K. Yamamura. Interval solution of nonlinear equations using linear programming. *BIT*, 38(1):186–199, 1998.

Acknowledgements

This work has been partially supported by the Spanish Ministry of Education and Science (DPI2007-60858), by the "Comunitat de Treball dels Pirineus" (2006ITT-10004), and by Ramón y Cajal and I3 program funds.

IRI reports

This report is in the series of IRI technical reports.
All IRI technical reports are available for download at the IRI website
`http://www.iri.upc.edu`.