A Branch-and-Prune Method to Solve Closure Equations in Dual Quaternions

Arya Shabani, Josep M. Porta, Federico Thomas*

Institut de Robòtica i Informàtica Industrial (CSIC-UPC)
Llorens Artigas 4-6, 08028 Barcelona, Spain

Abstract

Using dual quaternions, the closure equations of a kinematic loop can be expressed as a system of multiaffine equations. In this paper, this property is leveraged to introduce a branch-and-prune method specially tailored for solving such systems of equations. The new method is objectively simpler (in the sense that it is easier to understand and to implement) than previous approaches relying on general techniques such as interval Newton methods or methods based on Bernstein polynomials or linear relaxations. Moreover, it relies on two basic operations — linear interpolation and projection onto coordinate planes— that can be efficiently computed. The generality of the proposed method is evaluated on position analysis problems with 0-, 1-, and 2-dimensional solution sets, including the inverse kinematics of serial robots and the forward kinematics of parallel ones. The results obtained on these problems show that the efficiency of the method compares favorably to state-of-the-art alternatives.

Keywords: Position analysis, dual quaternions, multiaffine polynomials, branch-and-prune methods.

1. Introduction

The position analysis problem entails determining the valid poses of a set of rigid links given joint constraints between them. This is a fundamental issue that underlies many problems such as the inverse kinematics of serial robots [1], the forward kinematics of parallel robots [2], the coordinated manipulation of objects [3], the generation of valid grasps [4], the constrained object positioning [5], the simultaneous localization and map building [6], the analysis of complex deployable structures [7], and also problems in other fields such as the conformational analysis of biomolecules [8].

The position analysis of mechanisms can be decomposed into two equally important subproblems: obtaining closure equations and solving them. The first problem is usually solved by fixing suitable reference frames to each link, then obtaining the transformation between the

*Corresponding author at: Tel: +34 934015757. Fax: +34 934015750.
Email addresses: ashabani@iri.upc.edu (Arya Shabani), porta@iri.upc.edu (Josep M. Porta), fthomas@iri.upc.edu (Federico Thomas)

frames at neighboring links as a function of the corresponding joint variables, and, finally, composing the transformations along each independent loop. This procedure is straightforward, but, as the complexity of the mechanism increases, the simplicity of the formulation becomes an issue of paramount importance. An alternative is to abandon the standard loop-based paradigm and use, for instance, distance-based closure conditions, which provide important simplifications when the analyzed mechanisms contain spherical joints [9, 10]. This is so because their complexity is not related to the number of loops in the mechanism, but to its coupling number [11]. Unfortunately, deciding beforehand when these formulations are superior to the standard loop-based formulations is an open problem. If one still adheres to the loop-based approach, the complexity of the formulation can only be reduced by simplifying the equations. In this context, dual quaternions emerged as an elegant alternative to vector-, matrix- and quaternion-based formulations, because they compactly encapsulate both translations and rotations [12, 13]. Moreover, contrary to what happens with other formulations, the use of dual quaternions leads to a minimal set of multiaffine equations. This property was first identified in [14] where it was exploited to reduce the system of equations to a generalized eigenproblem. Unfortunately and despite its relevance, this work received little attention. As we will show in this paper, the multiaffinity has far-reaching consequences when using interval-based methods.

Different variants of dual quaternions can be found in the literature [15, 16, 17, 18, 19, 20]. Some authors use unit dual quaternions where the sine and cosine of each joint angle appear in the formulation [16]. The natural exponential function substitution is then used to avoid treating such expressions as independent terms, which would duplicate the number of variables in the problem [21]. However, this substitution introduces extraneous roots and the new variables must be considered in the complex domain. As an alternative, the normalization of the real term of the dual quaternions to one significantly simplifies the formulation [22], but it suffers from the drawback associated with the tangent half-angle substitution (*i.e.*, it is singular for joint angles equal to π). Nevertheless, as we will show later, when using a branch-and-prune method this drawback can be elegantly overcome.

Concerning the second sub-problem —that of solving the system of closure equations— three main global methods (*i.e.*, those able to obtain all possible solutions) have been proposed in the literature: methods based on elimination theory, continuation methods, and interval methods. As a general rule, we can say that elimination methods [23, 24] often require intuition-guided steps and are unsuitable for large problems. Continuation methods [25, 26] determine all complex solutions, which substantially slows down the process on problems with a small fraction of real solutions. In contrast, interval methods are not affected by these limitations. Moreover, when the system has a continuous set of solutions, interval methods can provide an approximation of this set, independently of its dimension.

Interval methods [27, 28] apply a branch-and-prune approach. Therefore, they reduce an initial box, defined as the Cartesian product of the ranges of the input variables, ruling out regions that are proven to contain no solution. If the resulting box is not small enough to be considered as a solution, it is bisected into two sub-boxes and the reduce and bisect procedures are recursively applied to the two of them. The result is a set of boxes that necessarily contains all the solutions of the system within the initial box. The key element of these methods is how to reduce the boxes. Different techniques have been proposed to this end.

Interval Newton methods, described in detail in [29], have been applied to solve the position analysis of serial [27] and parallel [28] robots. These methods require the inversion of an interval Jacobian matrix. This is a complex operation, only feasible for systems where the Jacobian is full-rank all over the considered domain. Expressing the polynomial system in the Bernstein

basis permits avoiding interval Jacobian inversions [30]. Under this approach, the problem is fully represented in terms of the so-called *control points* whose convex hull necessarily contains the sought solutions. As a result, the box reduction process has an elegant and simple geometric interpretation. Unfortunately, the size of the linear programs used to implicitly define the convex hull rapidly grows with the number of variables in the problem. To address this issue one can resort to the use of linear relaxations [31] which, instead of relying on control points, explicitly represent the half-planes bounding the considered polynomials. This defines smaller linear programs that can be solved faster, but requires the input polynomials to include only linear, bilinear, or quadratic monomials. The transformations to reduce any polynomial system to this simplified form introduce many extra variables, reducing the efficiency of the method.

In this paper, we show that solving a multiaffine system of closure equations in dual quaternions via a branch-and-prune method becomes remarkably simple. This is because the conversion to the Bernstein basis becomes unnecessary as the evaluation of each function at the corners of the considered box directly gives the control points. The result is a simple and yet efficient method that can be easily parallelized. A preliminary version of these ideas were applied to the position analysis of spherical mechanism in [32]. Here, we extend the approach to general spatial mechanisms including cases with continuous sets of solutions. This allows us to obtain the input-output curves of complicated spatial mechanisms, or even an approximation of the self-motion manifold of kinematotropic mechanisms, in a remarkably simple way.

This paper is organized as follows. Section 2 describes how to use non-unit dual quaternions to formalize position analysis problems as the solution set of multiaffine closure equations. Then, Section 3 presents a rigorous proof of two important properties of multiaffine equations, namely the interpolation and convex-hull properties. These properties are exploited in Section 4 to derive a general and efficient branch-and-prune resolution method. Section 5 illustrates the performance of this method on problems with 0-, 1-, and 2-dimensional solution sets and, finally, Section 6 summarizes the contributions of this paper and identifies points deserving further attention.

2. Deriving closure equations using dual quaternions

2.1. Basics on dual quaternions

We next include the basic facts on dual quaternions needed in our analysis. A complete treatment of the subject can be found, for example, in [17].

Using quaternions, a rotation through an angle θ about the axis defined by the unit vector $\mathbf{p} = (p_x, p_y, p_z)^T$ can be expressed as

$$\mathbf{r_p}(\theta) = \cos\left(\frac{\theta}{2}\right) + \sin\left(\frac{\theta}{2}\right)\hat{p}. \tag{1}$$

where $\hat{p} = p_x\mathrm{i} + p_y\mathrm{j} + p_z\mathrm{k}$, i, j, and k being imaginary units that satisfy the relationships $\mathrm{i}^2 = \mathrm{j}^2 = \mathrm{k}^2 = \mathrm{ijk} = -1$.

Since $\mathbf{r_p}(\theta)$ and $\mathbf{r_{-p}}(-\theta)$ represent the same rotation, we can projectivize this representation so that a rotation is identified with a point in the three dimensional projective space. Thus, the rotation represented by the quaternion in Eq. (1) is also represented by the following non-unit quaternion

$$\mathbf{r_p}(t) = 1 + t\,\hat{p}, \tag{2}$$

where $t = \tan(\theta/2)$. This representation is singular at $\theta = \pi$.

A translation d in the direction also given by $\mathbf{p}$ can be seen as a rotation in the dual magnitude $\varepsilon\, d$, where $\varepsilon^2 = 0$ [33]. That is,

$$\mathbf{t}_\mathbf{p}(d) = \cos\left(\varepsilon\,\frac{d}{2}\right) + \sin\left(\varepsilon\,\frac{d}{2}\right)\,\hat{p}. \tag{3}$$

Then, expanding the trigonometric functions using Maclaurin series, we have that

$$\mathbf{t}_\mathbf{p}(d) = 1 + \varepsilon\,\frac{d}{2}\,\hat{p}. \tag{4}$$

With this representation for rotations and translations, we have the essential building blocks to derive loop equations. Nevertheless, it is interesting to consider the screw displacement consisting of a rotation through an angle θ about the axis defined by $\mathbf{p}$ and a translation d in the direction also given by $\mathbf{p}$. This displacement can be expressed as:

$$\mathbf{s}_\mathbf{p}(t, d) = \mathbf{r}_\mathbf{p}(t)\,\mathbf{t}_\mathbf{p}(d) = 1 + t\,\hat{p} + \varepsilon\frac{d}{2}\,(-t + \hat{p})\,. \tag{5}$$

Notice that this does not represent a general screw displacement as the axis of rotation passes through the origin. The general form can be found, for example, in [18, 34].

A general dual quaternion can be written as $\mathbf{g} = \mathbf{q}_1 + \varepsilon\,\mathbf{q}_2$, where $\mathbf{q}_1$ and $\mathbf{q}_2$ are ordinary quaternions. When representing spatial displacements using unit dual quaternions, it is not difficult to prove that

$$\|\mathbf{q}_1\| = 1 \tag{6}$$

and

$$\mathbf{q}_1 \cdot \mathbf{q}_2 = 0. \tag{7}$$

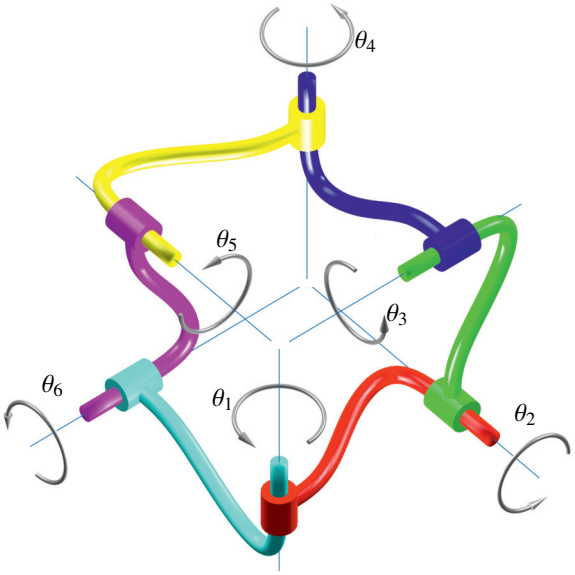
In our case, while Eq. (7) is satisfied, Eq. (6) is not because $\mathbf{q}_1$ is not necessarily a unit quaternion. As a consequence, in our case, $\mathbf{g}\mathbf{g}^*$ with $\mathbf{g}^* = \bar{\mathbf{q}}_1 + \varepsilon\,\bar{\mathbf{q}}_2$ and $\bar{\mathbf{q}}_i$ the quaternion conjugate, is a pure real magnitude not necessarily equal to 1. In this way, rigid-body displacements are simply identified with points in the six-dimensional quadric defined by Eq. (7), called the Study quadric [35].

2.2. Loop equations

Using the Denavit-Hartenberg (DH) convention, the reference frame of link i can be obtained from that of link $i-1$ by first rotating about its axis Z with angle θ_i and translating d_i along the same axis, and then rotating the new reference frame about its axis X with angle α_i and translating it a_i along the same axis. Thus, the 4-tuple $(\theta_i, d_i, \alpha_i, a_i)$ includes the DH parameters of link i. Using Eq. (5), and denoting $t_i = \tan(\theta_i/2)$ and $u_i = \tan(\alpha_i/2)$, this displacement can be compactly expressed as

$$\mathbf{s}_\mathbf{z}(t_i, d_i)\,\mathbf{s}_\mathbf{x}(u_i, a_i) = \left(1 + t_i\,\mathrm{k} + \varepsilon\,\frac{d_i}{2}\,(-t_i + \mathrm{k})\right)\left(1 + u_i\,\mathrm{i} + \varepsilon\,\frac{a_i}{2}\,(-u_i + \mathrm{i})\right), \tag{8}$$

where $\mathbf{z}$ and $\mathbf{x}$ are unit vectors along the Z and X axes, respectively. Since we do not adhere to the unity condition, the above expression is simpler than those appearing, for example, in [16] and [36].



θ_i	d_i	α_i	a_i
θ_1	0	$\pi/2$	1
θ_2	0	$-\pi/2$	1
θ_3	0	$\pi/2$	1
θ_4	0	$-\pi/2$	1
θ_5	0	$\pi/2$	1
θ_6	0	$-\pi/2$	1

Figure 1: The overconstrained 6R closed-loop mechanism described by Bricard and its DH parameters. All joint angles in the shown configuration are $\pi/2$.

We can derive the closure equations for any closed-loop mechanism expressed in terms of its DH parameters simply composing terms as the one in Eq. (8). This composition leads to expressions of the form

$$\mathbf{s_z}(t_1, d_1)\, \mathbf{s_x}(u_1, a_1) \ldots \mathbf{s_z}(t_n, d_n)\, \mathbf{s_x}(u_n, a_n) = c, \tag{9}$$

where c is a real scalar. This means that the components in i, j, k, ε, ε i, ε j, and ε k resulting from expanding the left hand side of Eq. (9) necessarily vanish. As a result, we obtain seven scalar equations from which only six are independent due to the constraint in Eq. (7). As it is done, for example, in [20], we simply discard the one corresponding to the term in ε and keep the rest of equations, which have a direct interpretation in terms of translations and rotations. The relevance of these equations is that they are multiaffine in terms of t_i and d_i, for $i = 1, \ldots, n$. This is better understood through an example.

2.3. Formulation example

Let us consider the 6R mechanism depicted in Fig. 1 described by Bricard in [37]. According to Grübler-Kutzbach formula, it should be rigid, but it is mobile because of the special choice of its design parameters. This kind of exceptional mechanisms are called *overconstrained* (see, for example, [38] and the references therein for details on this kind of 6R mechanisms). In general, 6R closed-loop mechanisms can have up to 16 possible rigid configurations [39], but this overconstrained mechanism has a 1-dimensional configuration space, which is technically referred to as a 1-dimensional self-motion manifold.

The closure equation of this mechanism can be expressed as

$$\begin{aligned}&(1+t_1\,\mathrm{k})(1+\mathrm{i}+\varepsilon\,(-1+\mathrm{i})/2)\,(1+t_2\,\mathrm{k})(1-\mathrm{i}+\varepsilon\,(-1+\mathrm{i})/2)\\&(1+t_3\,\mathrm{k})(1+\mathrm{i}+\varepsilon\,(-1+\mathrm{i})/2)\,(1+t_4\,\mathrm{k})(1-\mathrm{i}+\varepsilon\,(-1+\mathrm{i})/2)\\&(1+t_5\,\mathrm{k})(1+\mathrm{i}+\varepsilon\,(-1+\mathrm{i})/2)\,(1+t_6\,\mathrm{k})(1-\mathrm{i}+\varepsilon\,(-1+\mathrm{i})/2)=c.\end{aligned}\tag{10}$$

After expanding the left hand side in Eq. (10) and grouping the terms in i, j, and k, we obtain the following three equations:

$$\begin{aligned}&t_2t_3-t_1t_4-t_1t_2-t_1t_6+t_2t_5-t_3t_4-t_3t_6+t_4t_5-t_5t_6+t_1t_2t_3t_5+t_1t_2t_4t_6\\&\qquad-t_1t_3t_4t_5+t_1t_3t_5t_6-t_2t_3t_4t_6+t_2t_4t_5t_6+t_1t_2t_3t_4t_5t_6=0,\end{aligned}\tag{11}$$

$$\begin{aligned}&-t_4-t_6-t_2+t_1t_3t_4-t_1t_2t_3-t_1t_2t_5+t_1t_3t_6-t_1t_4t_5+t_2t_3t_5+t_1t_5t_6\\&\qquad+t_2t_4t_6-t_3t_4t_5+t_3t_5t_6+t_1t_2t_3t_4t_6-t_1t_2t_4t_5t_6+t_2t_3t_4t_5t_6=0,\end{aligned}\tag{12}$$

$$\begin{aligned}&-t_3-t_5-t_1+t_1t_2t_4+t_1t_2t_6+t_1t_3t_5-t_2t_3t_4+t_1t_4t_6-t_2t_3t_6+t_2t_4t_5-t_2t_5t_6\\&\qquad+t_3t_4t_6-t_4t_5t_6+t_1t_2t_3t_4t_5-t_1t_2t_3t_5t_6+t_1t_3t_4t_5t_6=0.\end{aligned}\tag{13}$$

Likewise, repeating the same process for the terms multiplying ε i, ε j, and ε k, we obtain the following equations:

$$\begin{aligned}&t_1t_3-t_1t_5+t_2t_4-t_2t_6+t_3t_5+t_4t_6+t_1t_2t_3t_4-t_1t_2t_3t_6-t_1t_2t_4t_5+t_1t_2t_5t_6\\&\qquad+t_1t_3t_4t_6+t_2t_3t_4t_5-t_1t_4t_5t_6-t_2t_3t_5t_6+t_3t_4t_5t_6-3=0,\end{aligned}\tag{14}$$

$$\begin{aligned}&t_5-t_3-3t_1+t_1t_2t_4-t_1t_2t_6+t_1t_3t_5-t_2t_3t_4+t_1t_4t_6+t_2t_3t_6+t_2t_4t_5-t_2t_5t_6\\&\qquad-t_3t_4t_6+t_4t_5t_6+t_1t_2t_3t_4t_5-t_1t_2t_3t_5t_6+t_1t_3t_4t_5t_6=0,\end{aligned}\tag{15}$$

$$t_2-t_6+t_1t_2t_3-t_1t_3t_4+t_1t_4t_5-t_1t_5t_6=0.\tag{16}$$

Equations (11)-(13) actually correspond to the closure conditions for the *spherical indicatrix* of the mechanism, and Eqs. (14)-(16) can be seen as a condition in the tangent space of the configuration space of this spherical indicatrix [40]. It can be verified that all these equations are multiaffine. That is, all of them can be expressed as $h_i\,t_i+k_i=0$, for $i=1,\ldots,6$, where h_i and k_i depend on all the variables except t_i. We next investigate the properties of this type of equations.

3. Two important properties of multiaffine maps

A multivariate polynomial $f(\mathbf{x})$ is usually expressed in terms of monomials as

$$f(\mathbf{x})=\sum_{\mathbf{p}=\mathbf{0}}^{\mathbf{m}} a_{\mathbf{p}}\,\mathbf{x}^{\mathbf{p}},\tag{17}$$

where the sum expands over the multi-index combination up to $\mathbf{m} = (m_1, \ldots, m_n)$, *i.e.*, all $\mathbf{p} = (p_1, \ldots, p_n)$ such that $0 \leq p_i \leq m_i$ for $i = 1, \ldots, n$, where $a_{\mathbf{p}}$ are scalar coefficients, $\mathbf{x} = (x_1, \ldots, x_n)$, and $\mathbf{x}^{\mathbf{p}}$ denotes the product $x_1^{p_1} \ldots x_n^{p_n}$. For our purposes, however, it is more convenient to express the polynomial using the multivariate Bernstein basis [41], which includes the polynomials of the form

$$b_{\mathbf{p},\mathbf{m}}(\mathbf{x}) = b_{p_1,m_1}(x_1) \ldots b_{p_n,m_n}(x_n), \tag{18}$$

where

$$b_{m_i,p_i}(x_i) = \binom{m_i}{p_i} x_i^{p_i} \, (1 - x_i)^{m_i - p_i}. \tag{19}$$

Using this basis

$$f(\mathbf{x}) = \sum_{\mathbf{p}=\mathbf{0}}^{\mathbf{m}} c_{\mathbf{p}} \, b_{\mathbf{p},\mathbf{m}}(\mathbf{x}), \tag{20}$$

where

$$c_{\mathbf{p}} = \sum_{\mathbf{j}=\mathbf{0}}^{\mathbf{p}} \frac{\binom{\mathbf{p}}{\mathbf{j}}}{\binom{\mathbf{m}}{\mathbf{j}}} \, a_{\mathbf{j}} \tag{21}$$

are the so-called control points and where

$$\binom{\mathbf{p}}{\mathbf{j}} = \binom{p_1}{j_1} \ldots \binom{p_n}{j_n}$$

and

$$\binom{\mathbf{m}}{\mathbf{j}} = \binom{m_1}{j_1} \ldots \binom{m_n}{j_n}.$$

Since the formulation for the closure conditions derived in the previous section involves only multiaffine equations, we are interested in the particular case where $\mathbf{m} = (1, \ldots, 1)$ and where p_i is either 0 or 1. In this case,

$$b_{p_i,1}(x_i) = \begin{cases} (1 - x_i) & p_i = 0 \\ x_i & p_i = 1 \end{cases} \tag{22}$$

and it can be seen that the control points simplify to

$$c_{\mathbf{p}} = f(\mathbf{p}), \tag{23}$$

which correspond to the evaluation of f on the corners of the unitary box

$$\mathcal{B}_1 = [0, 1]_1 \times [0, 1]_2 \times \cdots \times [0, 1]_n. \tag{24}$$

Using Eqs. (22) and (23), f can be expressed as

$$f(\mathbf{x}) = \sum_{\mathbf{p}=\mathbf{0}}^{(1,\ldots,1)} f(\mathbf{p}) \prod_{i=1}^{n} b_{p_i,1}(x_i). \tag{25}$$

This result can be generalized to any box $\mathcal{B}$ in $\mathbb{R}^n$, i.e., any axis-aligned $n-$rectangle (also known as an *orthotope*) defined as

$$\mathcal{B} = [x_1^l, x_1^u] \times [x_2^l, x_2^u] \times \cdots \times [x_n^l, x_n^u], \tag{26}$$

where $x_i^l, x_i^u \in \mathbb{R}$, $x_i^l \leq x_i^u$ and where the set of 2^n vertices of $\mathcal{B}$ is

$$\mathcal{V}(\mathcal{B}) = \left\{(v_1, \ldots, v_n) \in \mathbb{R}^n \mid v_i \in \left\{x_i^l, x_i^u\right\}\right\}. \tag{27}$$

Then, it can be seen that

$$f(\mathbf{x}) = \sum_{\mathbf{p}=\mathbf{0}}^{(1,\ldots,1)} f(\mathbf{v_p}) \prod_{i=1}^{n} b_{p_i,1}(s(x_i)), \tag{28}$$

where $\mathbf{v_p}$ is the element of $\mathcal{V}(\mathcal{B})$ with $v_i = x_i^l$ if $p_i = 0$, and $v_i = x_i^u$ otherwise, and with

$$s(x_i) = \frac{x_i - x_i^l}{x_i^u - x_i^l}$$

a variable substitution so that Bernstein polynomials are still evaluated in the $[0, 1]$ interval.

Two important properties of multiaffine maps directly follow from the previous considerations (alternative presentations can be found, for instance, in [42]). The first one is the interpolation property. The expression in Eq. (28) can be rewritten taking into account Eq. (22) as the following linear interpolation between two functions of $n - 1$ variables

$$f(\mathbf{x}) = (1 - s(x_1))\, f_l(x_2, \ldots, x_n) + s(x_1)\, f_u(x_2, \ldots, x_n) \tag{29}$$

where

$$f_l(x_2, \ldots, x_n) = \sum_{\mathbf{p}=(0,0,\ldots,0)}^{(0,1,\ldots,1)} f(\mathbf{v_p}) \prod_{i=2}^{n} b_{p_i,1}(s(x_i)),$$

and

$$f_u(x_2, \ldots, x_n) = \sum_{\mathbf{p}=(1,0,\ldots,0)}^{(1,1,\ldots,1)} f(\mathbf{v_p}) \prod_{i=2}^{n} b_{p_i,1}(s(x_i)).$$

The same decomposition can be applied recursively to f_l and f_u until the resulting functions only involve one variable. Therefore, the evaluation of f can be merely computed as a linear interpolation of its control points. Actually this can be seen as a specialization for multiaffine polynomials of the De Casteljau's algorithm [43], which is a robust and efficient way to evaluate polynomials in Bernstein form.

The second property of multiaffine maps is the convex hull property. Although it also applies to f, this property is more useful when applied to

$$g(\mathbf{x}) = (\mathbf{x}, f(\mathbf{x})), \tag{30}$$

which defines a function in $\mathbb{R}^{n+1}$. Finding the roots of f is equivalent to determining the points in the form $(\mathbf{x}, 0)^T$ in the graph of g. This function is represented in Bernstein form as

$$g(\mathbf{x}) = \sum_{\mathbf{p}=\mathbf{0}}^{(1,\ldots,1)} \mathbf{d_p} \prod_{i=1}^{n} b_{p_i,1}(s(x_i)), \tag{31}$$

where $\mathbf{d_p}$ is the element corresponding to $\mathbf{v_p}$ in the set

$$\mathcal{D}(\mathcal{B}) = \{(\mathbf{v}, f(\mathbf{v})) \in \mathbb{R}^{n+1} | \mathbf{v} \in \mathcal{V}(\mathcal{B})\}. \tag{32}$$

Since the values of the Bernstein polynomials in the range $[0, 1]$ are non-negative and form a partition of the unity [43], Eq. (31) is a particular convex combination of the points in $\mathcal{D}(\mathcal{B})$ for each $\mathbf{x}$. Thus, the value of g in $\mathbf{x} \in \mathcal{B}$ is fully included in the convex hull of the points in $\mathcal{D}(\mathcal{B})$

$$\mathcal{H} = \left\{ \sum_{\mathbf{p}=\mathbf{0}}^{(1,\ldots,1)} \mathbf{d_p}\, \lambda_\mathbf{p} \mid \sum \lambda_\mathbf{p} = 1,\ 0 \leq \lambda_\mathbf{p} \leq 1 \right\}. \tag{33}$$

Observe that, since the $\mathbf{d_p}$ points are fixed, $\mathcal{H}$ is fully-defined by linear constraints and, thus, it can be readily included in a linear program.

4. The proposed branch-and-prune method

Let us define the system of equations

$$\mathbf{f}(\mathbf{x}) = \mathbf{0}, \tag{34}$$

where $\mathbf{f} = (f_1(\mathbf{x}), \ldots, f_m(\mathbf{x}))$ and each f_j for $j = 1, \ldots, m$, is a multiaffine polynomial in $\mathbf{x} = (x_1, \ldots, x_n)$. The proposed algorithm identifies arbitrarily small boxes containing solution candidates of System (34) within a given box $\mathcal{B} \subset \mathbb{R}^n$ using a branch-and-prune scheme which iterates two operations, *box reduction* and *box bisection*. Using box reduction, portions of $\mathcal{B}$ containing no solution are ruled out by narrowing some of its defining intervals. This process is iterated until either (a) the box is reduced to an empty set, in which case it contains no solution, or (b) the longest side of the box is under a specified threshold σ, in which case it is considered a solution box, or (c) the decrease of the box volume between two consecutive iterations is below a given ratio ρ, in which case the box is bisected along its largest side. If the later occurs, the whole process is repeated for the newly created sub-boxes, and for the sub-boxes recursively created thereafter, until the process ends up with a collection of solution boxes whose size is under σ. Thus, this algorithm returns a collection of boxes, with all their side lengths below σ, containing all the solutions. If the solution space is an algebraic variety of dimension greater than zero, the algorithm returns a collection of boxes bounding the portions of this variety contained in $\mathcal{B}$. The algorithm is *complete*, in the sense that no solution is missed, and *conservative*, in the sense that some boxes might contain no solution.

Since the box bisection operation is trivial, we next focus on the description of a box reduction procedure, which can be seen as a simplification of a method presented in [30]. The simplification is possible thanks to the multiaffinity of the closure equations derived using dual quaternions and the associated properties described in the previous section.

For the sake of clarity, we describe the procedure when System (34) consists of just one equation in two variables, and we latter describe how it applies to the general case.

Assume that we want to find all solutions of a multiaffine equation $f(\mathbf{x}) = 0$, with $\mathbf{x} = (x_1, x_2)$, in $\mathcal{B} = [x_1^l, x_1^u] \times [x_2^l, x_2^u] \in \mathbb{R}^2$ (Fig. 2-top). Since $(\mathbf{x}, f(\mathbf{x}))$ must lie within the convex hull $\mathcal{H}$ of the 2^2 points of $\mathbb{R}^3$ $\{(\mathbf{x}, f(\mathbf{x}))|\ \mathbf{x} \in \{x_1^l, x_1^u\} \times \{x_2^l, x_2^u\}\}$, we simply project $\mathcal{H}$ onto each x_i-$f(\mathbf{x})$ coordinate plane, as depicted in Fig. 2-bottom. This projection defines a trapezoid whose vertices

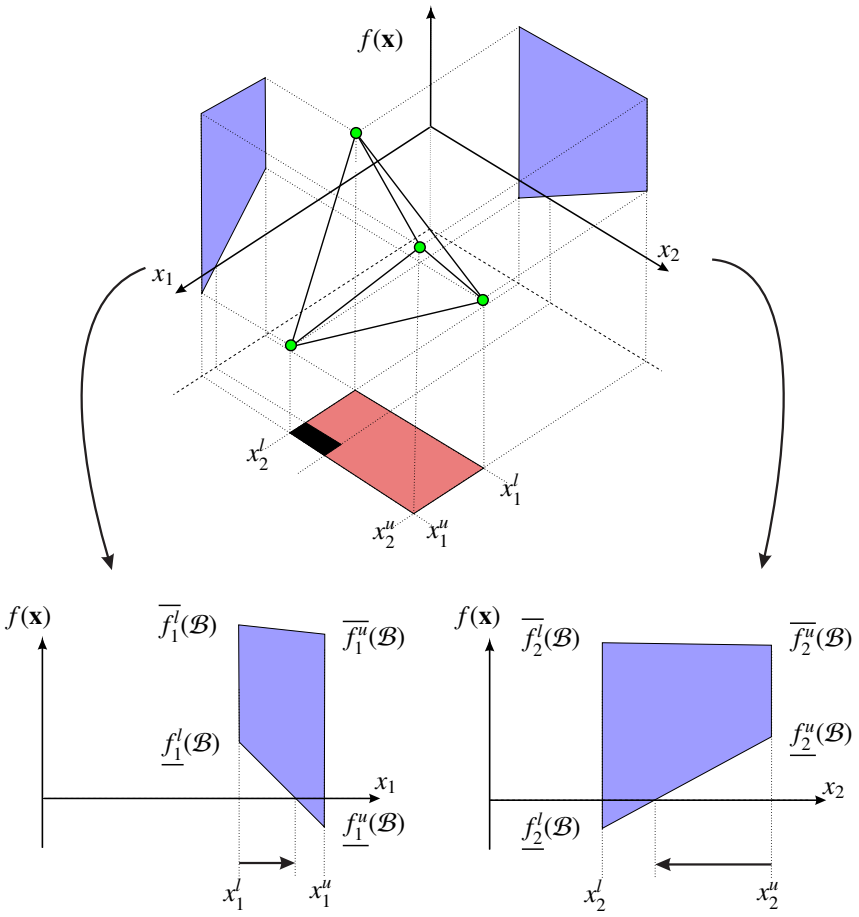


Figure 2: The image of the points in the box $[x_1^l, x_1^u] \times [x_2^l, x_2^u]$ (shown in red) for any multiaffine function $f(x_1, x_2)$ necessarily lies inside a tetrahedron. The vertices of the tetrahedron (shown in green) are obtained by evaluating f in the corners of the box. Then, from the projections of the tetrahedron onto the coordinate planes (shown in blue), the initial ranges for the variables can be reduced to the regions where these projections intersect the line $f = 0$. The reduction in both x_1 and x_2 defines a new box (shown in black) better bounding the sought-after solution set.

have x_i^l and x_i^u as horizontal coordinates and where the vertical coordinates are defined as

$$\begin{aligned}
\underline{f_i^l}(\mathcal{B}) &= \min\{f(\mathbf{v}) | \mathbf{v} \in \mathcal{V}(\mathcal{B}), x_i = x_i^l\},\\
\overline{f_i^l}(\mathcal{B}) &= \max\{f(\mathbf{v}) | \mathbf{v} \in \mathcal{V}(\mathcal{B}), x_i = x_i^l\},\\
\underline{f_i^u}(\mathcal{B}) &= \min\{f(\mathbf{v}) | \mathbf{v} \in \mathcal{V}(\mathcal{B}), x_i = x_i^u\},\\
\overline{f_i^u}(\mathcal{B}) &= \max\{f(\mathbf{v}) | \mathbf{v} \in \mathcal{V}(\mathcal{B}), x_i = x_i^u\},
\end{aligned}$$

with $i \in \{1, 2\}$. Clearly, we can prune the ranges of x_i for which the trapezoid does not intersect the line defined by $f(\mathbf{x}) = 0$. These trapezoid-line clippings usually reduce the ranges of some variables giving a smaller box that still bounds the sought solutions. If the trapezoid and the line do not intersect, the considered box includes no solution and is discarded. This strategy typically

produces less pruning than alternative methods, but the results reported in Section 5 show that it is advantageous due to its low computational cost.

Note that the same pruning strategy can be applied to a multiaffine equation in $n > 2$ variables because the convex hull of the then involved 2^n points will also yield a trapezoid when projected to a plane defined by the x_i and $f(\mathbf{x})$ axes, for $i = 1, \ldots, n$. Finally, if we have several equations in the system, the process can be iterated for all of them to obtain the intersection of the reduction obtained with each equation in each variable range.

One feature of this approach is that it leads to the so-called *clustering effect*. This is a well-known effect of branch-and-prune methods that apply one necessary condition at a time instead of a set of necessary and sufficient conditions at once. In the former case, a solution is returned as a cluster of boxes instead of a single box containing it [44]. Increasing the resolution, *i.e.*, reducing the value of the parameter σ bounding the size of the returned boxes, does not eliminates the clustering because the same effect is reproduced just with smaller boxes. Clustering is exacerbated when some of the solution varieties of the individual equations in the system are close. In practice, the size of the cluster gives an idea of the shakiness of the mechanism in the corresponding set of configurations. Indeed, if the effect of manufacturing inaccuracies and/or small elasticities in the kinematic chain were incorporated as ranges in the coefficients of the closure conditions, clustering-free branch-and-prune methods would also produce clusters. Thus, although from the pure mathematical point of view clusters of boxes might be seen as a problem, from a mechanical point of view their presence provides qualitative information on which configurations of the mechanism might lead to a loss of rigidity. Having said that, we next explain ways to mitigate the clustering effect.

4.1. Verification of Solution Boxes

The clustering effect would be avoided if necessary and sufficient conditions were available to determine whether a given box contains a solution. Unfortunately, such conditions are not available in general. However, a variety of sufficient conditions exist. The theorems by Kantorovich [45, 46], Moore [47] or Smale [48] are particularly tailored for interval Newton methods. In some cases they can not only prove the existence of a solution in a given box, but also guarantee the uniqueness of such solution. However, in practice such theorems only hold on small boxes with low probability of including more than one solution. Moreover, all these theorems rely on the Jacobian of the system of equations, which may be problematic, specially when it is necessary to invert it on a given range for the input variables.

In contrast, Miranda's theorem [49] relies only on the values of the function on the different faces of the analyzed box. It can be stated as follows:

Let $\mathbf{f} : \mathcal{S} \subset \mathbb{R}^n \to \mathbb{R}^n$ *and assume that* $\mathcal{B} = \{[x_1^l, x_1^u], \ldots, [x_n^l, x_n^u]\}$ *is a box included in* $\mathcal{S}$*. Let* $\mathcal{L}_i = \{\mathbf{x} \in \mathcal{B} | x_i = x_i^l\}$ *and* $\mathcal{U}_i = \{\mathbf{x} \in \mathcal{B} | x_i = x_i^u\}$ *be the n pairs of opposite faces of* $\mathcal{B}$ *along dimension i. If* $\mathbf{f}_j(\mathbf{x}) \cdot \mathbf{f}_j(\mathbf{y}) \leq 0$ *for all* $\mathbf{x} \in \mathcal{L}_i$, $\mathbf{y} \in \mathcal{U}_i$*, for* $i, j = 1, \ldots, n$*, then* $\mathbf{f}$ *has at least one root in* $\mathcal{B}$*.*

It has been proven that there is a hierarchy in terms of generality between the different existence theorems [50]. In this hierarchy, Moore's theorem is more general than Kantorovich's theorem which in turn is more general than Smale's theorem [51]. However, Miranda's theorem is more general than all of them meaning that if any of them holds Miranda's theorem also holds. Only Borsuk's theorem is more general than Miranda's theorem for arbitrary norms, but when

using the infinity norm where inclusion sets are boxes, as it is our case, they are equivalent. Thus, it makes sense to consider Miranda's test as the best option. This test can be incorporated in the box reduction strategy based on the trapezoid-clipping with a negligible computational cost: if for a given box all the trapezoids resulting from the projection of the control points on the different x_i-f_j coordinate planes have their extremes on opposites sides of the $f_j = 0$ line, then the theorem holds and, consequently, the box is guaranteed to include at least one solution.

None of the existence tests available in the literature, however, applies to problems with solution sets of dimension greater than zero. For 1-dimensional solution sets, one can still use Miranda's test on the faces of the boxes returned by the algorithm. If any of them can be certified to include an isolated solution, the box can be guaranteed to include part of the sought-after solution variety. For solution sets of dimension two or higher, one should check lower-dimensional components defining the boundary of the analyzed box.

A simpler alternative strategy that that will be used here, and that works for any dimension, is to initiate a Newton-Raphson process in a point in the box potentially including part of the solution and check whether it converges to a solution inside the box. In this regard, note that the control points can also be used to efficiently evaluate the derivatives of a multiaffine function. For instance, using Eq. (29) the derivative of f with respect to x_1 is

$$\frac{\partial f}{\partial x_1} = \frac{f_u(x_2, \ldots, x_n) - f_l(x_2, \ldots, x_n)}{x_1^u - x_1^l}, \tag{35}$$

where, as described in Section 3, f_l and f_u can be computed by linear interpolation of the control points.

4.2. Implementation Details

From the geometric point of view, the change of variable $t_i = \tan\left(\frac{\theta_i}{2}\right)$ can be seen as the stereographic projection of the unit circle, from $x = -1$, to the line $y = 0$. Thus, if θ_i=π, t_i goes to infinity. Thus, numerical problems will occur if any of the joint angles of a solution configuration is π. Moreover, if we want to apply a branch-and-prune method to obtain the solutions of our system of equations, it is not a good decision to start with a domain ranging from $-\infty$ to $+\infty$ in all variables. One workaround to this issue is to split the problem in two, one for $\theta_i \in [0, \pi]$, and another one for $\theta_i \in [-\pi, 0]$, and shift the origin of θ_i by $\pi/2$ and $-\pi/2$ respectively so that the range for t_i is $[-1, 1]$ in the two subproblems. Note that using small ranges for the variables increases the numerical stability of the algorithm. This shift can be applied to all the rotational variables, while the domains for non-rotational variables remain the same for all subproblems. In other words, if our problem has r rotation variables, we will decompose it into 2^r subproblems. Such sub-problems are independent between them and, consequently, they can be solved in parallel.

5. Examples

The performance of the proposed method has been evaluated on the following four test-cases:

1. Two 6R loops with zero-dimensional solution sets;
2. A C5R loop with a one-dimensional solution set;
3. A 7R kinematotropic loop with a solution set with dimension up to two; and
4. A 3-PRRR parallel robot.

The two 6R loops are classical benchmarks in the field of inverse kinematics of serial robots and are used to compare the performance of the proposed method with standard approaches. The rest of test cases are used to show the generality and robustness of the method in problems with solution sets of dimension greater than zero, which can not be characterized using standard approaches based, for instance, on elimination theory or on continuation techniques. In particular, the second test-case is a mobile linkage that includes different types of joints and shows the ability of the method to isolate one-dimensional solution sets. The third is a kinematotropic linkage with solution subsets of different dimension and illustrates the versatility of the method even in cases where some of the equations become dependent at particular configurations. Finally, the last test-case is a parallel robot used to show the performance of the method in multiple-loop mechanisms.

In all these test-cases, we will examine the performance of the method described in Section 4 based on the trapezoid-line clipping, which we will refer to as the *trapezoid* method. This method will be compared with the two alternative methods described respectively in [30] and [31]. The first one is based on linear programming directly using the convex hull defined in Eq. (33) (it will be denoted as the LP method), and the second one also uses linear programming, but on smaller linear program derived using the linear relaxations of the equations (it will be denoted as the LR method). Observe that while the trapezoid and the LP methods are defined in the same space, the LR method uses a different formalization in terms of variables and equations and, thus, the outputs of the latter cannot be directly compared with the results of the two other methods. The LP approach does not include any box verification procedure, but the LR one incorporates a procedure based on the Brouwer's fixed point theorem. In both the LP and LR methods, the linear programs are solved using the CoinOR linear programming library [52]. In contrast, and due to the simplicity of the underlying operations, the trapezoid method does not rely on any external library and it can be implemented in few lines of code.

All the reported results have been obtained with a computer equipped with 32Gb of RAM and an i7 processor with 4 cores running at 4.2 GHz, where two concurrent threads can be executed at each core. These capacities are used to execute each one of the subproblems defined in Section 4.2 in a separate thread. In all cases we use $\rho = 0.5$ and either $\sigma = 10^{-4}$, for the problems with zero-dimensional solution sets, or $\sigma = 10^{-2}$, for the problems with solution sets with dimension greater than zero. The implementation of the proposed method in C and the input files for the test-cases are available at [53].

5.1. 6R loops with zero-dimensional solution sets

The general inverse kinematics of 6R robots is a decades-old problem which still stands as a challenging test-case [54]. The early work on this subject concentrated on particular architectures and identified instances with up to 16 different solutions [55]. Later, this proved to be the maximum number of solutions for rigid 6R loops [39]. In the 90's, Manocha and Canny derived the first efficient numerical technique for this problem [56]. This approach works seamlessly for 6R loops with generic parameters, but it may fail for loops where the parameters fulfill particular conditions.

We will use as a first test-case the example appearing in [57] and also in [56][1], which has 16 real solutions. The DH parameters for this test-case appear in Table 1. Table 2 shows the performance statistics of the compared methods.

[1] The version of this example given in [56] has a wrong sign in one of the entries of $\mathbf{A}_{\text{hand}}$.

θ_i	d_i	α_i	a_i
θ_1	0	$\pi/2$	0.3
θ_2	0	0.017	1
θ_3	0.2	$\pi/2$	0
θ_4	0	0.017	1.5
θ_5	0	$\pi/2$	0
θ_6	0	0.017	0

$$\mathbf{A}_{\text{hand}} = \begin{pmatrix} -0.7601 & -0.6416 & 0.1022 & -1.1401 \\ 0.1333 & 0 & 0.9910 & 0 \\ -0.6359 & 0.7669 & 0.0855 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Table 1: DH parameters and closure condition for a 6R loop with 16 real solutions.

	LP	**LR**	**Trapezoid**
Processed boxes	8644	10025	20270
Box reductions	13064	11149	27698
Bisected boxes	4290	5012	10103
Empty boxes	4338	4997	10149
Solution boxes	16	16	18
Verified solutions	-	16	6
Execution time [s]	15	22	0.033

Table 2: Performance of the three compared methods for a 6R loop with 16 solutions.

Using the LP method, the simplex tableau has 48 rows and 320 columns (*i.e.*, 15230 entries) and the 16 solution boxes of this problem are identified in 15 seconds. None of them are verified because, as said, this approach does not include any verification procedure. As a comparison, the LR method takes 22 seconds to solve the same problem and correctly verifies all the solution boxes. In this case, the tableau has 223 rows and 59 columns (*i.e.*, 13157 entries), only slightly smaller than the one used in the LR method. The proposed trapezoid method only takes 33 ms to isolate the 16 solutions of the problem. Two of the solutions are returned as a cluster of two boxes. However, the maximum error in the center of any of the returned boxes is below 10^{-3}, which means that all of them would be considered solutions if manufacturing or assembly errors were taken into account.

Using the trapezoid method, six of the returned boxes are verified using Miranda's theorem. Since this theorem is the most general existence theorem none of the alternative approaches would be able to verify more solution boxes. In any case, the sixteen boxes actually including a solution can be readily identified with negligible computational cost using a Newton-Raphson process initialized at the center of each of the returned boxes.

The same problem can be solved with a redundant formulation, taking into account the equation corresponding to the ϵ term in dual quaternion giving the loop-closure condition. This reduces the number of explored boxes, but slightly increases the execution time (due to the need to process one more equation) and it does not alleviate the clustering effect. Similar results are

θ_i	d_i	α_i	a_i
θ_1	0.1875	$-\pi/2$	0.5
θ_2	0.375	0	1
θ_3	0.25	$\pi/2$	0.125
θ_4	0.875	$-\pi/2$	0
θ_5	0	$\pi/2$	0
θ_6	0.125	0	0.25

$$\mathbf{A}_{\text{hand}} = \begin{pmatrix} -1 & 0 & 0 & 0.41150993 \\ 0 & -1 & 0 & 0.14908956 \\ 0 & 0 & 1 & 0.4889994 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Table 3: DH parameters and closure condition of the *bad_eg0* example from [58].

	LP	**LR**	**Trapezoid**
Processed boxes	10140	7761	45802
Box reductions	16087	8842	74481
Bisected boxes	5038	3880	22869
Empty boxes	5094	3873	22277
Solution boxes	8	8	656
Verified solutions	-	2	0
Execution time [s]	20	19	0.095

Table 4: Performance of the three compared methods for the *bad_eg0* example from [58].

obtained in the other test-cases included in this paper.

Note that the algorithm described in [56], whose implementation is available at [58], can solve this problem in 4 ms. Unfortunately, due to an unreported bug, it fails to provide the correct value of θ_6 for one of the solutions. This method is based on an eigenproblem which can be efficiently solved relying on highly optimized linear algebra libraries [59]. However, its implementation is not trivial due to the numerous reasons that lead to an ill-conditioning of the involved matrices. For example, this approach fails when applied to the problem in Table 3. In contrast, the method presented in this paper has no problem and solves the problem in just 95 ms. The only relevant issue is that, due to the particularities of the mechanism, each solution is returned as a cluster of boxes. However, the error in the center of the returned boxes is, in all cases, below 10^{-2}. The LP and LR methods identify the eight solution boxes, but they are about 200 times slower than the trapezoid method. If the result provided by the trapezoid method is combined with a Newton-Raphson process, the eight boxes containing the exact solutions are readily identified. This post-process adds less than 2 ms to the overall execution time. An alternative strategy to limit the clustering effect is to use the LP method only for the boxes considered as solutions by the trapezoid method. In this case, the eight solutions are identified in just 0.69 s, still significantly faster than the LP or the LR methods alone.

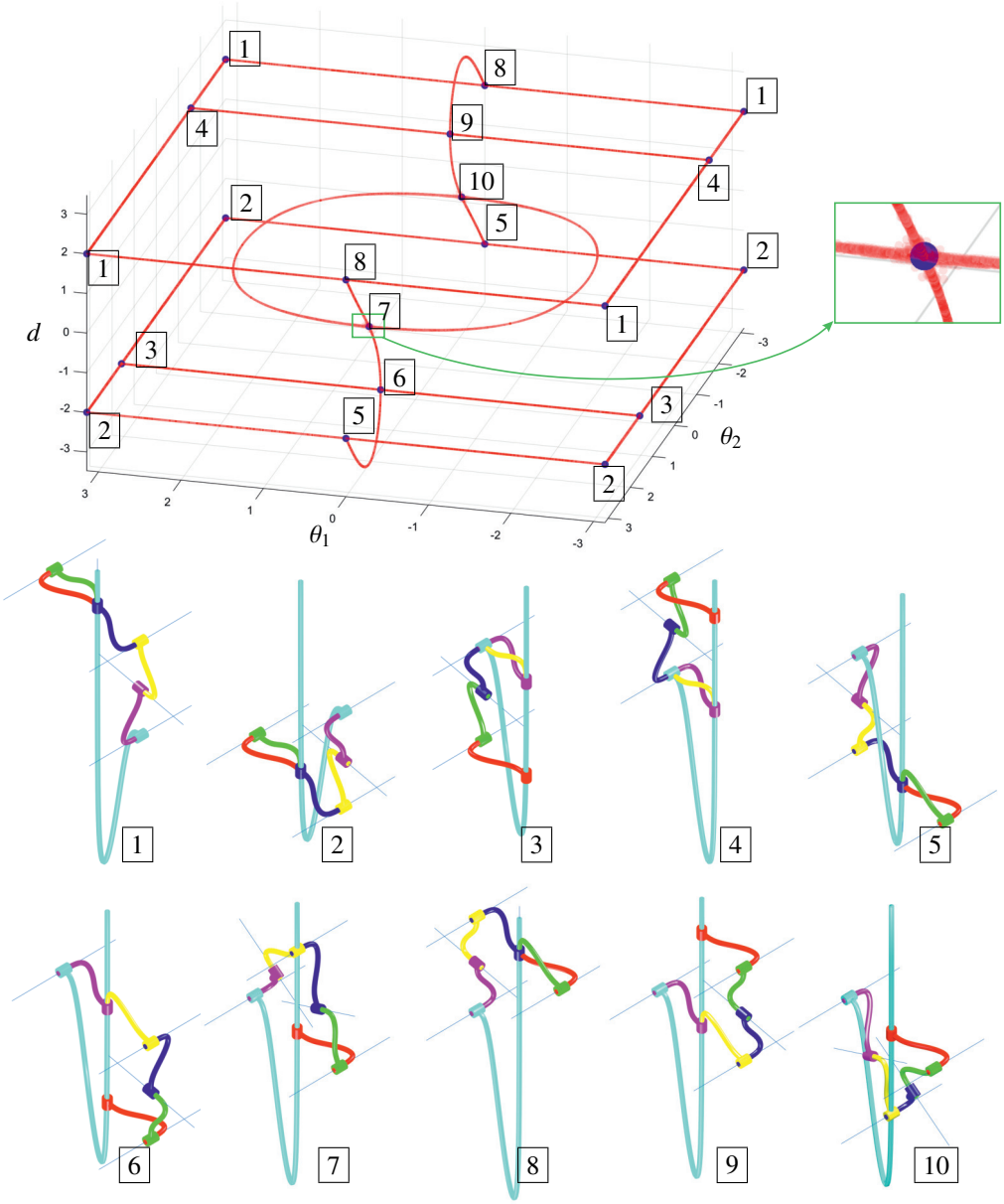


Figure 3: Top: The self-motion manifold of the analyzed C5R closed-loop mechanism obtained with the trapezoid method projected on the subspace defined by θ_1, θ_2, and d. The solution boxes are represented as semitransparent spots to better appreciate the accumulation of boxes. The only noticeable difference with respect to the solutions obtained using the LP method is the cluster effect around the nodes, as shown on the right inset. Bottom: mechanism configurations at the nodes of the self-motion. Some nodes are repeated at the boundaries of the represented region because this self-motion is periodic in θ_1 and θ_2. The elliptic shape at $d = 0$ is the solution set of the Bricard's overconstrained mechanism shown in Fig. 1.

	LP	LR	Trapezoid
Processed boxes	321866	227137	926804
Box reductions	531473	268578	1408882
Bisected boxes	160901	113568	463370
Empty boxes	22081	3416	191606
Solution boxes	138884	110153	271828
Execution time [s]	1237	101	2.83

Table 5: Performance of the three compared methods for the self-motion manifold computation of the mechanism in the C5R closed-loop mechanism.

	LP	LR	Trapezoid
Processed boxes	706658	679935	12699530
Box reductions	1090058	733929	18397748
Bisected boxes	353265	339967	6349701
Empty boxes	115952	61423	3279112
Solution boxes	237441	278545	3070717
Execution time [s]	7000	339	57

Table 6: Performance of the three compared methods for the computation of the self-motion manifold of the 7R closed-loop kinematotropic mechanism in Fig. 4.

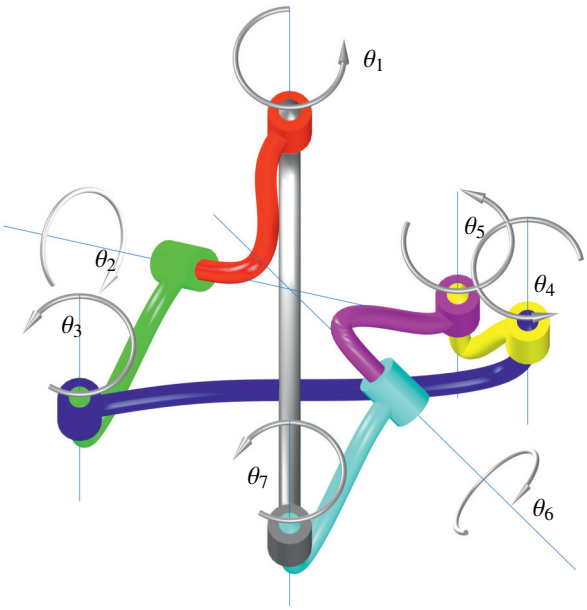
5.2. A C5R loop with a singular 1-dimensional solution set

Let us consider the C5R closed-loop mechanism resulting from substituting the first revolute joint of the Bricard's 6R mechanism shown in Fig. 1 with a cylindrical joint. The self-motion manifold of this mechanism is one-dimensional and includes singularities (i.e., it contains nodes). When there is no translation in the cylindrical joint, we obviously have the standard Bricard's mechanism. Therefore, the solution set of the Bricard's overconstrained mechanism would appear as a subset of the solution set of this mechanism.

Figure 3 shows the characterization of the self-motion manifold of this mechanism obtained using the trapezoid method which includes 271828 small boxes. The characterization obtained with the LP method avoids the cluster effect near the nodes as it only includes 138884 solution boxes. However, as detailed in Table 5, obtaining it is about 500 times more expensive in terms of computational time. As in the previous example, one possibility is to use the trapezoid method first and then filter out the possible solutions using the LP method. This procedure isolates the solution set in about 466 s. This is still about three times faster than the LP method alone. Using the Newton-based procedure 5747 of these boxes can be verified to include a solution.

5.3. A 7R closed-loop kinematotropic mechanism

Kinematotropic mechanisms are a class of reconfigurable mechanisms whose motion modes have a variable number of degrees of freedom [60]. The position analysis of these mechanisms is hard since different dependencies between the equations appear depending on the configuration. Among all possible 7R kinematotropic closed-loop mechanisms, the one studied in [20] (see Fig. 4), which is a particular case of the one proposed in [21], has five motion modes, the largest number of modes known to date for such a kind of mechanism. According to the results



θ_i	d_i	α_i	a_i
θ_1	0	$2\arctan 2$	5
θ_2	0	$-2\arctan 2$	5
θ_3	0	0	12
θ_4	0	0	2
θ_5	0	$-2\arctan 2$	3
θ_6	0	$2\arctan 2$	5
θ_7	0	0	12

Figure 4: 7R closed-loop kinematotropic mechanism studied in [20], and its DH-parameters.

presented in [21], one of these modes is 2-dimensional and the other four, 1-dimensional. All these modes are connected through ten transition configurations. Thus, this mechanism is a challenging example in which we can check the effectiveness of our method.

Table 6 compares the performance of the LP, LR, and trapezoid methods when approximating the self-motion manifold of this mechanism. The representation of the obtained solutions using the LP and the trapezoid methods appear in Fig. 5-top. In Fig. 5-top, we have identified the ten transition configurations and the corresponding mechanism configurations are represented in Fig. 5-bottom. They perfectly match those reported in [20], validating the obtained result. In this example, while both methods provide an excellent approximation of the 2-dimensional component, the approximation of the one-dimensional ones are much rougher for the trapezoid method. This enforces the idea that this method is very good at characterizing the components of higher

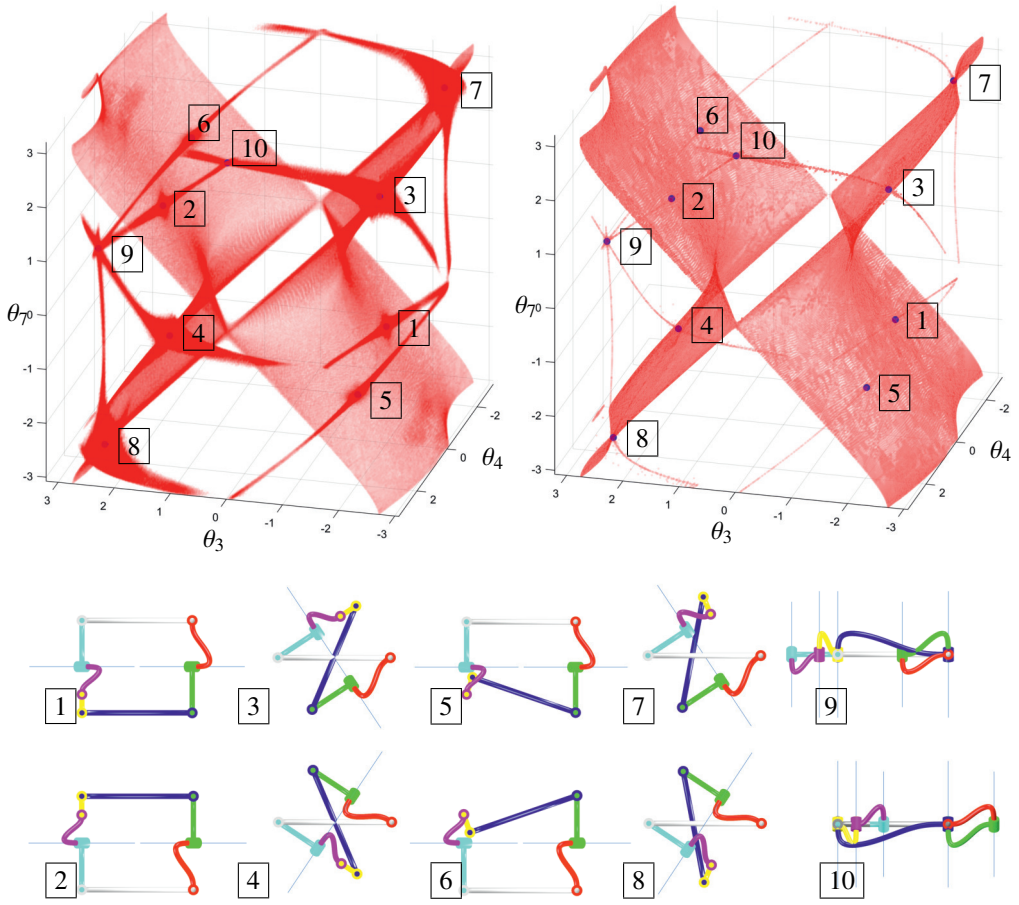


Figure 5: Top: Approximation of the self-motion manifold of the mechanism in Fig. 4 using the trapezoid (left) and the LP (right) methods. The results have been projected onto the subspace defined by θ_3, θ_4, and θ_7. Bottom: The configurations for the transition points between the different motion modes.

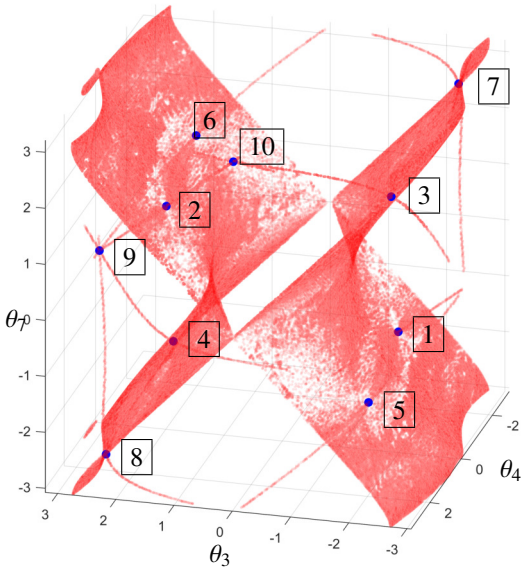


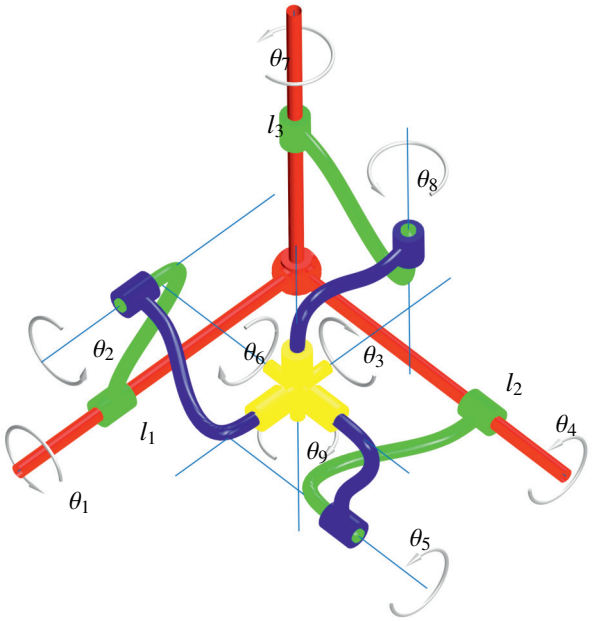
Figure 6: The set of verified solution boxes on the self-motion manifold of the mechanism in Fig. 4 obtained using the trapezoid method and the Newton-Raphson verification procedure. The results have been projected onto the subspace defined by θ_3, θ_4, and θ_7.

dimension, but suffers from the clustering effect in the characterization of those of lower dimension. We can say that the lower the dimension of the component, the more relevant the clustering effect. However, if we use the Newthon-Raphson procedure, 238261 of the solution boxes are verified to include a solution point. This is about the same number of solution boxes returned by the LP method and, thus, the verified boxes already offers a dense, cluster-free approximation of the sought after configuration space. This can be appreciated in Fig. 6 that shows only the subset of verified solution boxes obtained with the trapezoid method.

5.4. A 3-PRRR parallel robot

In principle, multiple-loop mechanisms do not offer any extra conceptual complexity for the trapezoid method if formalized as follows. First, a graph is defined where the vertices represent the links of the mechanism and the edges are given by the joints connecting them. Then, a set of independent loops is obtained defining a spanning tree of the graph: each edge of the graph not included in the tree determines an independent loop [61]. Finally, the equations for each one of these independent loops are generated using the procedure defined in Section 2.

We use the Tripteron mechanism shown in Fig. 7 and described in [62] and [63] to illustrate this procedure. This is a 3-PRRR parallel robot with a moving platform linked to a fixed based through three legs where each leg has an active prismatic joint and three passive revolute joints. The platform has linear displacements in all directions, as a Cartesian robot, but with the characteristic reduced inertia and increased rigidity of parallel robots. In our particular analysis, we fix the prismatic joints of legs two and three, but we let free the variable l_1 representing the



θ_i	d_i	α_i	a_i
0	l_k	0	0
θ_{3k-2}	0	0	1
θ_{3k-1}	0	0	1
θ_{3k}	-0.2	0	0

Figure 7: A 3-PRRR parallel robot with a moving platform (in yellow) linked to a fixed based (in red) through three legs and the DH-parameters of these legs for $k \in \{1, 2, 3\}$. Each leg has an active prismatic joint and three passive revolute joints.

displacement of the slider joint of the first leg. This robot defines three loops, but only two of them are independent.

Table 7 compares the performance of the trapezoid method with the LP and LR methods. In contrast to what happens with the previous test-cases, the LR method is the most efficient one in this particular case. This can be attributed to the regularity of the parameters for this robot which results in a particularly simple set of equations using the LR formulation. Despite its simplicity, the trapezoid method is also quite efficient. For instance, it is more than two orders of magnitude faster than the LP method, which is based on the same formulation as the trapezoid method. The trapezoid method bounds the solution set with 14906 small boxes. Using the Newton-based verification procedure, 8813 of them are guaranteed to include a point of the sought solution set. These results are obtained taking into account the independent loops formed by legs one and two and legs one and three. As shown in Table 8, different results are obtained using the equations derived from all the pairs of independent loops in this problem or even when considering the three loops simultaneously, i.e., using redundant equations. Notable differences can be observed in the execution time and, specially, in the total number of processed boxes. Clearly if redundant equations are considered, the number of processed boxes reduces, but this

	LP	**LR**	**Trapezoid**
Processed boxes	1084304	14611	7875392
Box reductions	1735675	18310	8868336
Bisected boxes	541896	7305	3937440
Empty boxes	535483	20	3923046
Solution boxes	6925	7305	14906
Execution time [s]	10302	14	26

Table 7: Performance of the three compared methods for the self-motion manifold computation of the 3-PRRR parallel robot in Fig. 7.

Loops	$(1,2),(1,3)$	$(1,2),(2,3)$	$(1,3),(2,3)$	$(1,2),(1,3),(2,3)$
Processed boxes	7875392	9361756	10762026	6332110
Box reductions	8868336	10322872	11861654	6874236
Bisected boxes	3937440	4680622	5380757	3165799
Empty boxes	3923046	4656327	5356841	3153721
Solution boxes	14906	24807	24428	12590
Execution time [s]	26	27	30	26

Table 8: Performance of the trapezoid method considering different loops. The notation (a,b) is used to denote the loop formed by leg a and b.

does not directly translate to a reduction in the execution time due to the burden of processing more equations. However, significant differences in the number of processed boxes also appear even when considering different sets of non-redundant equations. Therefore, to efficiently solve more complex multi-loop problems, a careful analysis of the propagation of the ranges between the variables in the loops would be necessary [64]. We leave this as a point for further research.

Figure 8 shows the self-motion manifold of this mechanism together with some representative configurations. The configurations labeled with 1 to 4 are at extremes of the range for l_1 and configurations 5-6 and 7-8 are very close in this particular projection but, as it can be seen in the displayed configurations, they differ in the rest of variables.

6. Conclusions

The multiaffinity of the closure equations of the kinematic loops has been exploited in this paper to its latest consequences. As a result, we have derived an interval method for solving systems of this kind of equations —which we have named *trapezoid method*— which is significantly simpler than all other interval methods used in the past for position analysis in kinematics. The reason for this comparative simplicity is that all previous approaches apply to general systems of algebraic equations and the one derived here is specific for multiaffine systems, *i.e.*, to closure equations in its simplest form.

We have presented a variety of examples including single- and multiple-loop mechanisms and, in all of them, the trapezoid method is at least two orders of magnitude faster that the alternative branch-and-prune method relying on the same formulation. The obtained results for 6R closed loops with 0-dimensional solution sets are even comparable, in terms of performance,

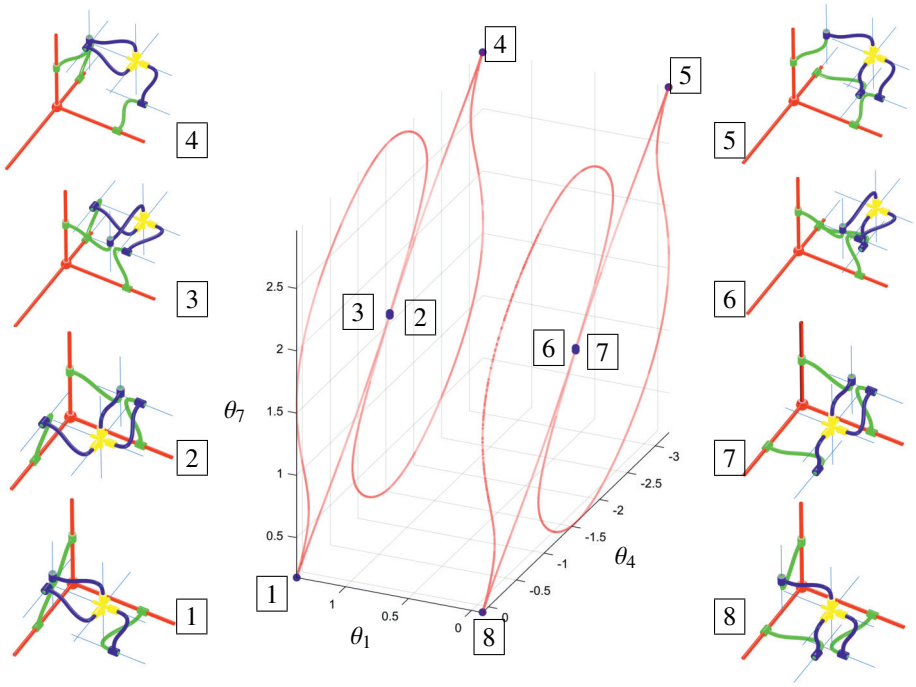


Figure 8: The self-motion manifold of the 3-PRRR parallel robot and some representative configurations.

with a specialized numerical approach. To obtain even better performance, it would be interesting to increment the level of parallelization via GPUs or networks of low-cost single-board computers. In any case, the position analysis problem is NP-hard in general [65] and, thus, despite the efficiency of the trapezoid method in problems of practical interest, and despite their potential improvements, an exponential increment in computational cost has to be expected as the number of variables in the problem grows. Regarding the quality in the approximation of the solution, in the case of continuous solution sets, commonly known as self-motion manifolds, the trapezoid method provides good approximations for the components of dimension larger than one, but those of lower dimension are degraded by the clustering effect. In these cases, the trapezoid method has to be complemented with a final refinement step, which can be based on the use of necessary and sufficient conditions or on solution verification procedures. Both approaches have been evaluated in this paper.

Finally, it is worth exploring other numerical alternatives to solve multiaffine systems. For example, as we mentioned in the introduction of this article, the method presented in [14] reduces the resolution of such kind of systems to a generalized eigenproblem. It would be worth reexamining this approach in the light of results presented here.

Acknowledgments

This work was partially supported by the Spanish Ministry of Economy and Competitiveness through the project DPI2017-88282-P.

References

[1] J. J. Vicker, J. Denavit, R. S. Hartenberg, An iterative method for the displacement analysis of spatial mechanisms, ASME Journal of Applied Mechanics 31 (1964) 309–314.

[2] H. Pang, M. Shahinpoor, Inverse dynamics of a parallel manipulator, Journal of Robotic Systems 11 (8) (1994) 693–702.

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

[4] C. Rosales, L. Ros, J. M. Porta, R. Suárez, Synthesizing grasp configurations with specified contact regions, The International Journal of Robotics Research 30 (4) (2011) 431–443.

[5] A. Rodríguez, L. Basañez, E. Celaya, A relational positioning methodology for robot task specification and execution, IEEE Transactions on Robotics 24 (3) (2008) 600–611.

[6] J. M. Porta, CuikSlam: A kinematics-based approach to SLAM, in: IEEE International Conference on Robotics and Automation, 2005, pp. 2436–2442.

[7] P. Kumar, S. Pellegrino, Computation of kinematic paths and bifurcation points, International Journal of Solids and Structures 37 (46-47) (2000) 7003–70027.

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

[9] F. Thomas, A distance geometry approach to the singularity analysis of 3R robots, ASME Journal of Mechanisms and Robotics 8 (1) (2016) 011001 (11 pages).

[10] J. M. Porta, F. Thomas, Yet another approach to the Gough-Stewart platform forward kinematics, in: Proceedings of the IEEE International Conference in Robotics, 2018, pp. 974–980.

[11] N. Rojas, F. Thomas, On closed-form solutions to the position analysis of Baranov trusses, Mechanism and Machine Theory 50 (2012) 179–196.

[12] W. Blaschke, Kinematik und Quaternionen, VEB Deutscher Verlag der Wissenschaften, Berlin, 1960.

[13] A. T. Yang, F. Freudenstein, Application of dual-number quaternion algebra to the analysis of spatial mechanisms, Journal of Applied Mechanics 31 (2) (1964) 300–308.

[14] B. Morton, M. Elgersma, A new computational algorithm for 7R spatial mechanisms, Mechanism and Machine Theory 31 (1) (1996) 23–43.

[15] P. Gervasi, V. Karakusevic, P. Zsombor-Murray, An algorithm for solving the inverse kinematics of a 6R serial manipulator using dual quaternions and grassmannians, in: J. Lenarcic, M. Husty (Eds.), Advances in Robot Kinematics: Analysis and Control, Springer, 1998, pp. 383–392.

[16] D. Gan, Q. Liao, S. Wei, J. Dai, S. Qiao, Dual quaternion-based inverse kinematics of the general spatial 7R mechanism, Journal of Mechanical Engineering Science 222 (8) (2008) 1593–1598.

[17] J. M. Selig, Geometric fundamentals of robotics, Springer, New York, 2005.

[18] F. Thomas, Approaching dual quaternions from matrix algebra, IEEE Trans. on Robotics 30 (5) (2014) 1037–1048.

[19] Z. Li, J. Schicho, A technique for deriving equational conditions on the Denavit-Hartenberg parameters of 6R linkages that are necessary for movability, Mechanism and Machine Theory 94 (2015) 1–8.

[20] X. Kong, A variable-DOF single-loop 7R spatial mechanism with five motion modes, Mechanism and Machine Theory 120 (2018) 239–249.

[21] X. Kong, M. Pfurner, Type synthesis and reconfiguration analysis of a class of variable-dof single-loop mechanisms, Mechanism and Machine Theory 85 (2015) 116–128.

[22] G. Hegedüs, J. Schichob, H.-P. Schröcker, The theory of bonds: A new method for the analysis of linkages, Mechanism and Machine Theory 70 (2013) 407–424.

[23] M. Raghavan, B. Roth, Inverse kinematics of the general 6R manipulator and related linkages, Transactions of the ASME Journal of Mechanical Design 115 (3) (1993) 502–508.

[24] T.-Y. Lee, J.-K. Shim, Forward kinematics of the general 6-6 Stewart platform using algebraic elimination, Mechanism and Machine Theory 36 (9) (2001) 1073–1085.

[25] B. Roth, F. Freudenstein, Synthesis of path-generating mechanisms by numerical methods, ASME Journal of Engineering for Industry 85 (1963) 298–307.

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

[27] A. Castellet, F. Thomas, Towards an efficient interval method for solving inverse kinematic problems, in: IEEE International Conference on Robotics and Automation, Vol. IV, 1997, pp. 3615–3620.

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

[29] A. Neumaier, Interval Methods for Systems of Equations, Cambridge University Press, Cambridge, 1990.

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

[31] J. M. Porta, L. Ros, F. Thomas, A linear relaxation technique for the position analysis of multi-loop linkages, IEEE Transactions on Robotics 25 (2) (2009) 225–140.
[32] A. Shabani, S. Sarabandi, J. M. Porta, F. Thomas, A fast branch-and-prune algorithm for the position analysis of spherical mechanisms, in: Mechanisms and Machine Science, Vol. 73, Springer, 2019, pp. 549–558.
[33] I. Fischer, Dual-Number Methods in Kinematics, Statics and Dynamics, CRC Press, 1999.
[34] J. M. Selig, Exponential and Cayley maps for dual quaternions, Advances in Applied Clifford Algebras 20 (3) (2010) 923–936.
[35] E. Study, Von den bewegungen und umlegungen, Mathematische Annalen 39 (1891) 441–565.
[36] Z. Zhao, T. Wang, D. Wang, Inverse kinematic analysis of the general 6R serial manipulators based on unit dual quaternion and Dixon resultant, in: Chinese Automation Congress (CAC), 2017, pp. 2646–2650.
[37] R. Bricard, Mémoire sur la théorie de l'octaèdre articulé, Journal de Mathématiques Pures et Appliquées 3 (1897) 113–148.
[38] M. Pfurner, A new family of overconstrained 6R-mechanisms, in: EUCOMES 08, 2009, pp. 117–124.
[39] E. J. Primrose, On the input-output equation of the general 7R-mechanism, Mechanism and Machine Theory 21 (6) (1986) 509–510.
[40] F. Thomas, On the n-bar mechanism, or how to find global solutions to redundant single loop spatial kinematic chains, in: IEEE International Conference on Robotics and Automation, Vol. I, 1992, pp. 403–408.
[41] R. T. Farouki, The Bernstein polynomial basis: A centennial retrospective, Computer Aided Geometric Design 29 (6) (2012) 379–419.
[42] A. D. Rikun, A convex envelope formula for multilinear functions, Journal of Global Optimization 10 (1997) 425–437.
[43] G. Farin, Curves and Surfaces for Computer Aided Geometric Design - A Practical Guide, Academic Press, 1990.
[44] A. Morgan, V. Shapiro, Box-bisection for solving second-degree systems and the problem of clustering, ACM Transactions on Mathematical Software 13 (2) (1987) 152–167.
[45] L. V. Kantorovich, On Newton's method for functional equations, Proceedings of the USSR Academy of Sciences 59 (1948) 1237–1240.
[46] J. Ortega, W. Rheinboldt, Iterative Solution of Nonlinear Equations in Several Variables, Academic Press, 1970.
[47] R. E. Moore, A test for existence of solutions to nonlinear systems, ACM Journal of Numerical Analysis 14 (1977) 611–615.
[48] S. Smale, Algorithms for solving equations, in: International Congress of Mathematicians, 1986, pp. 172–195.
[49] C. Miranda, Un'osservazione su un teorema di Brouwer, Bollettino dell'Unione Matematica Italiana 2 (3) (1941) 5–7.
[50] G. Alefeld, A. Frommer, G. Heindl, J. Mayer, On the existence theorems of Kantorovich, Miranda and Borsuk, Electronic Transactions on Numerical Analysis 17 (2004) 102–111.
[51] G. E. Alefeld, F. A. Potra, Z. Shen, On the existence theorems of Kantorovich, Moore and Miranda, in: Topics in numerical analysis, Springer, 2001, pp. 21–28.
[52] The coin linear program code (CLP), https://github.com/coin-or/Clp.
[53] KRD Research Group: A solver for systems of multi-affine equations, http://www.iri.upc.edu/people/porta.
[54] K. Hauser, Learning the problem-optimum map: Analysis and application to global optimization in robotics, IEEE Transactions on Robotics 33 (1) (2017) 141–152.
[55] D. Pieper, The kinematics of manipulators under computer control, Ph.D. thesis, Stanford University (1968).
[56] D. Manocha, J. Canny, Efficient inverse kinematics for general 6R manipulators, IEEE Transactions on Robotics and Automation 10 (1994) 648–657.
[57] C. Wampler, A. Morgan, Solving the 6R inverse position problem using a generic-case solution methodology, Mechanism and Machine Theory 26 (1) (1991) 91–106.
[58] GAMMA Research Group: Inverse kinematics code for serial manipulators, http://gamma.cs.unc.edu/software/downloads/IK/ik-1.0.tar.gz.
[59] E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. D. Croz, A. Greenbaum, S. Hammarling, A. McKenney, D. Sorensen, LAPACK Users' Guide, 3rd Edition, Society for Industrial and Applied Mathematics, 1999.
[60] K. Wohlhart, Kinematotropic linkages, in: J. Lenarcic, V. Parenti-Castelli (Eds.), Recent Advances in Robot Kinematics, Springer, Dordrecht, 1996, pp. 359–368.
[61] G. Chartrand, L. Lesniak, Graphs and Digraphs, Chapman and Hall, 1996.
[62] M. Carricato, V. Parenti-Castelli, Singularity-free fully-isotropic translational parallel mechanisms, The International Journal of Robotics Research 21 (2) (2002) 161–174.
[63] X. Kong, C. M. Gosselin, Kinematics and singularity analysis of a novel type of 3-crr 3-dof translational parallel manipulator, The International Journal of Robotics Research 21 (9) (2002) 791–798.
[64] E. Celaya, T. Creemers, L. Ros, Exact interval propagation for the efficient solution of position analysis problems on planar linkages, Mechanism and Machine Theory 54 (2012) 116–131.

[65] J. B. Saxe, Embeddability of weighted graphs in k-space is strongly np-hard, in: Allerton Conference in Communications, Control and Computing, 1979, pp. 480–489.