UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH

Position analysis based on multi-affine formulations

Arya Shabani

UNIVERSITAT POLITÈCNICA DE CATALUNYA

Doctoral Programme

AUTOMATIC CONTROL, ROBOTICS AND COMPUTER VISION

Ph.D. Thesis

POSITION ANALYSIS BASED ON MULTI-AFFINE FORMULATIONS

Arya Shabani

Advisors:
Federico Thomas and Josep M Porta

April 2021

Position Analysis based on Multi-Affine Formulations

by Arya Shabani

A thesis submitted to the Universitat Politècnica de Catalunya
for the degree of Doctor of Philosophy

Doctoral Programme:
Automatic Control, Robotics and Computer Vision

This thesis has been completed at:
Institut de Robòtica i Informàtica Industrial, CSIC-UPC

Advisors:
Federico Thomas and Josep M Porta

Dissertation Committee:
Prof. Vicente Mata Amela
Prof. Alba Pérez Gracia
Prof. Cecilia García Cena

Abstract

The position analysis problem is a fundamental issue that underlies many problems in Robotics such as the inverse kinematics of serial robots, the forward kinematics of parallel robots, the coordinated manipulation of objects, the generation of valid grasps, the constraint-based object positioning, the simultaneous localization and map building, and the analysis of complex deployable structures. It also arises in other fields, such as in computer aided design, when the location of objects in a design is given in terms of geometric constrains, or in the conformational analysis of biomolecules. The ubiquity of this problem, has motivated an intense quest for methods able of tackling it. Up to now, efficient algorithms for the general problem have remained elusive and they are only available for particular cases. Moreover, the complexity of the problem has typically led to methods difficult to be implemented.

Position analysis can be decomposed into two equally important steps: obtaining a set of closure equations, and solving them. This thesis deals with both of them to obtain a general, simple, and yet efficient solution method that we call the trapezoid method. The first step is addressed relying on dual quaternions. Although it has not been properly highlighted in the past, the use of dual quaternions permits expressing the closure condition of a kinematic loop involving only lower pairs as a system of multi-affine equations. In this thesis, this property is leveraged to introduce an interval-based method specially tailored for solving multi-affine systems. The proposed method is objectively simpler (in the sense that it is easier to understand and to implement) than previous methods based on general techniques such as interval Newton methods, conversions to Bernstein basis, or linear relaxations. Moreover, it relies on two simple operations, namely, linear interpolations and projections on coordinate planes, which can be executed with a high performance. The result is a method that accurately and efficiently bounds the valid solutions of the problem at hand. To further improve the accuracy, we propose the use of redundant, multi-affine equations that are derived from the minimal set of equations describing the problem. To improve the efficiency, we introduce a variable elimination methodology that preserves the multi-affinity of the system of equations. The generality and the performance of the proposed trapezoid method are extensively evaluated on different kind of mechanisms, including spherical mechanisms, generic 6R and 7R loops, over-constrained systems, and multi-loop mechanisms. The proposed method is, in all cases, significantly faster than state of the art alternatives.

Resum

El problema d'anàlisi de posició és una qüestió fonamental que subjau en molts problemes de la robòtica, com ara la cinemàtica inversa dels robots en sèrie, la cinemàtica directa dels robots paral·lels, la manipulació coordinada d'objectes, la generació de prensions vàlides amb mans robòtiques, el posicionament d'objectes basat en restriccions, la localització i la creació de mapes de forma simultània, i l'anàlisi d'estructures desplegables complexes. També sorgeix en altres camps, com en el disseny assistit per ordinador, quan la ubicació dels objectes en un disseny es dóna en termes de restriccions geomètriques o en l'anàlisi conformacional de biomolècules. La omnipresència d'aquest problema ha motivat una intensa recerca de mètodes capaços d'afrontar-lo. Fins al moment, els algoritmes eficients per al problema general han estat esquius i només estan disponibles per a casos particulars. A més, la complexitat del problema sol conduir a mètodes difícils d'implementar.

L'anàlisi de posició es pot descompondre en dos passos igualment importants: l'obtenció d'un sistema d'equacions de tancament i la resolució d'aquest sistema. Aquesta tesi afronta tots dos passos per obtenir un mètode de solució general, senzill i, tot i així, eficient, que anomenem el mètode del trapezoide. El primer pas s'aborda utilitzant quaternions duals. Tot i que no ha estat suficientment destacat en el passat, l'ús de quaternions duals permet expressar la condició de tancament d'un bucle cinemàtic que inclogui només parells inferiors com a un sistema d'equacions multi-afins. En aquesta tesi, s'aprofita aquesta propietat per introduir un mètode especialment dissenyat per resoldre sistemes multi-afins. El mètode proposat és objectivament més senzill (en el sentit que és més fàcil d'entendre i d'implementar) que els mètodes anteriors que utilitzen tècniques generals com ara els mètodes de Newton basats en intervals, les conversions a la base de Bernstein o les relaxacions lineals. A més, el mètode es basa en dues operacions simples, a saber, les interpolacions lineals i les projeccions en plans de coordenades, que es poden executar de forma molt eficient. El resultat és un mètode que acota amb precisió i eficiència les solucions vàlides del problema. Per millorar encara més la precisió, proposem l'ús d'equacions multi-afins redundants derivades del conjunt mínim d'equacions que descriuen el problema. Per altra banda, per millorar l'eficiència, introduïm una metodologia d'eliminació de variables que preserva la multi-afinitat del sistema d'equacions. La generalitat i el rendiment del mètode del trapezoide s'avaluen extensivament en diferents tipus de mecanismes, inclosos mecanismes esfèrics, bucles 6R i 7R genèrics, sistemes sobre-restringits i mecanismes de múltiples bucles. El mètode proposat és, en tots els casos, significativament més ràpid que els mètodes alternatius descrits en la literatura fins al moment.

Acknowledgments

I would like to thank deeply my supervisors Prof. Dr. Federico Thomas and Dr. Josep M. Porta for their passionate support, motivation, and dedication to my work. From the moment that I set foot in their group, I have learned from their knowledge, their personality, and their patience. If it was not for them I could not pass this stage. I will be forever indebted to them. I am grateful to my colleague and my friend Dr. Soheil Sarabandi, whose friendship and support during these years means a lot to me.

All my family deserves a special mention. You are my true incentive to surmount difficulties and barriers. I can hardly express my feeling for you in words. My mom, Ana, when I was child I always told you that I would thank you when I get an academic prize. Unfortunately I have not achieved anything yet, but here is a good place to say thank you from the bottom of my heart for begin such a compassionate, supportive, wise, and lovely mom. You are the treasure of my life. My little sister, Yasna, whose esprit brings hope and vitality to my life. Your pure love is always in my heart. Eventually, my dad, my true mentor, the realization of wise father archetype, Dr. Alireza Shabani, whose life has always inspired me. Your wisdom and ingenuity have guided me to my current place. Your words and your thoughts are always in my mind. You are the one who has taught me to be free inside. My deepest gratitude, respect, and love to your great personality.

One does not become enlightened by imagining figures of light, but by making the darkness conscious.

Carl Jung

Contents

1 Introduction

This initial chapter poses the problem addressed in this work and reviews the related literature to highlight the novelty and the advantages of the proposed approach. At the end of the chapter, we remark the contributions of the work and the outline of the rest of the document.

1.1 Motivation and problem statement

Mechanisms are sets of links connected to each other via joints. From a pure kinematics point of view, they can be translated into geometric elements pairwise constrained in their motion. Characterizing all the possible configurations of the resulting constrained system is an important branch of kinematics usually called *position analysis* [35, 57, 169]. The position analysis problem is a fundamental issue that underlies many problems in Robotics such as the inverse kinematics of serial robots [184], the forward kinematics of parallel robots [121], the coordinated manipulation of objects [197], the generation of valid grasps [144, 145], the constraint-based object positioning [140], the simultaneous localization and map building [129], and the analysis of complex deployable structures [89]. It also arises in other fields, such as in computer aided design (CAD), when the location of objects in a design is given in terms of geometric constrains [120], or in the conformational analysis of biomolecules [83, 130, 190].

This thesis proposes a new method to solve position analysis problems. Despite the proposed method could in principle be useful in all the aforementioned fields, the focus on this thesis will be on problems in Robotics.

Position analysis can be decomposed into two equally important steps: obtaining a set of closure equations, and solving them. This thesis deals with both of them to obtain a general, simple and yet efficient solution method. In the dominant approach in kinematics, the first step is usually addressed by fixing suitable reference frames to each link, then obtaining the transformation between neighboring links as a function of the corresponding joint variables and, finally, composing the transformations along a set of independent closed loops. This procedure is rather straightforward but, as the complexity of the mechanism increases, the need for the most compact possible 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 whose complexity is not directly related to the number of loops of the mechanism, but to what is known as its coupling number [142]. Distance-based formulations provide important simplifications when the analyzed mechanisms only contain revolute joints in planar mechanism or spherical joints in spatial ones [134, 174]. Unfortunately, deciding beforehand when these formulations are superior to the standard loop-based formulations is still an open problem. Moreover, while formulations based on distance equations are advantageous to obtain closure polynomials [133, 143], their superiority in front of the formulations based on loop equations is not clear when the problem consists in obtaining numerical approximations of the attainable configurations. Therefore, in this thesis the loop-based formalization is adopted. Yet, if one adheres to this approach, a given loop equation can be expressed in many different ways [35]. The different alternatives are proposed with the aim of reducing the complexity of the resulting equations. In this context, the use of dual quaternions emerged as an elegant alternative because its use permits encapsulating both translations and rotations in a very compact way [12, 199]. Moreover, contrary to what happens with other formulations, the use of dual quaternions allows the derivation of a minimal set of multi-affine equations in the joint variables. This property has far reaching consequences in the way these systems of equations can be solved and the main goal of this thesis is to exploit such consequences to its fully extend.

1.2 State of the art

Since finding solutions of systems of nonlinear equations is an ubiquitous problem arising in many fields, ranging from engineering to computer-aided design, and to molecular biology [109, 182], many approaches have been proposed and they can be classified as global or local. While global approaches guarantee to find all possible solutions, local methods, such as those based on the Newton-Raphson algorithm, only find one particular solution, which depends on the initial guess. Here we focus on global methods which can be classified in methods based on elimination theory, methods using continuation techniques, and interval-based methods [75, 133, 148].

1.2.1 Elimination theory

Most algebraic methods to solve systems of polynomial equations rely on elimination theory [34, 45, 94, 95, 101, 138]. The idea behind these methods is to reduce the number of equation and variables transforming the initial set of equations into a minimum degree polynomial in a single variable. The degree of this polynomial is a bound in the number of solutions of the initial system of equations. The most relevant algebraic methods are:

1. **Gröbner bases**. They were introduced in 1966 by Bruno Buchberger in his doctoral thesis [18]. The basic idea of the method is to eliminate the highest-ordered terms in a given set of polynomial equations by adding multiples of the other equations in the set, this process is known as reduction [118]. As a result, the system of polynomial equations $f_1 = 0, f_2 = 0, \dots, f_n = 0$ in the variables $x_1, x_2, \dots, x_n$ is written as a triangular form $g_n(x_n) = 0$, $g_{n-1}(x_n, x_{n-1}) = 0, \dots, g_1(x_1, x_2, \dots, x_n)$, called a Gröbner basis [107]. The process can be applied to any set of polynomial equations and generalizes three familiar techniques: Gaussian elimination for solving linear systems of equations, the Euclidean algorithm for computing the greatest common divisor of two univariate polynomials, and the simplex algorithm for linear programming [148, 167, 168].

2. **Sylvester resultant**. For systems of two polynomials, it is defined as the determinant of a $(\alpha_1 + \alpha_2) \times (\alpha_1 + \alpha_2)$ matrix, with α_1 and α_2 the degree of the variable to eliminate in the polynomials. In the matrix, the block of coefficients of the equations are arranged row-wise and they are shifted one column to the right at each block [192].

3. **Macaulay resultant**. It is also called the multivariate resultant or the multipolynomial resultant. It is a generalization of the homogeneous resultant to n homogeneous polynomials in n variables. Let T denote the set of all terms of degree d_m in the variables $x_1, x_2, \dots x_n$, that is, $T = x_1^{\alpha_1} x_2^{\alpha_2} \dots x_n^{\alpha_n}$ such that $\alpha_1 + \alpha_2 + \dots + \alpha_n = d_m$. Now, let $T^{(i)}$

the terms of degree $d_m - d_i$ that are not divisible by $x_1^{d_1}, x_2^{d_2}, \ldots, x_{i-1}^{d_{i-1}}$. Then, the multiplication of terms in $T^{(i)}$ by $f_i(X)$ forms the Macaulay matrix, whose determinant give the sought resultant [200]. For $n = 2$ the Macaulay resultant reduces to the Sylvester one.

4. **Dixon resultant**. It is generated from the following expression:

$$\Delta(x_i, \bar{x}_j) = \frac{\begin{vmatrix} f_1(x_1, x_2, \cdots, x_m) & \ldots & f_m(x_1, x_2, \cdots, x_m) \\ f_1(\bar{x}_1, x_2, \cdots, x_m) & \ldots & f_m(\bar{x}_1, x_2, \cdots, x_m) \\ f_1(\bar{x}_1, \bar{x}_2, \cdots, x_m) & \ldots & f_m(\bar{x}_1, \bar{x}_2, \cdots, x_m) \\ & \vdots & \\ f_1(\bar{x}_1, \bar{x}_2, \cdots, \bar{x}_{m-1}, x_m) & \ldots & f_m(\bar{x}_1, \bar{x}_2, \cdots, \bar{x}_{m-1}, x_m) \end{vmatrix}}{(x_1 - \bar{x}_1) \ \ldots \ (x_{m-1} - \bar{x}_{m-1})}, \tag{1.1}$$

which is a polynomial in x_m, where $\bar{x}_i$ is a new variable used to replace x_i, whenever used. This resultant exploits the sparsity of the equations and, thus, it is in general more efficient than alternative methods [75]. The Dixon resultant has been used to produce outstanding results in the position analysis of manipulators [202]. However, as in all methods generating the resultant from the determinant of a matrix, in the Dixon method non-generic equations generate singularities. Therefore, techniques to avoid them by extracting non-trivial projection operators are necessary.

Despite the results obtained using elimination-based methods, in general, such methods often require intuition-guided steps and are unsuitable for large problems.

1.2.2 Continuation

Polynomial continuation methods [165] start from a system of equations with known solutions and gradually transform it into the system whose solutions are sought. During this transformation, the solutions are tracked using prediction-correction methods so that at the end of the process the sought after solutions of the input system are obtained.

This technique was known as early as 1930 and first used in kinematics by Roth and Freudenstein [146]. Subsequent work by Garcia and Li [53], Garcia and Zangwill [54], Morgan [111], and Li et al. [96], among others, led the procedure into its current highly-developed state [189]. This method has been responsible for the first solutions of 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 sixteen solutions [180] and Raghavan showed that the direct kinematics of the general Stewart-Gough platform can have forty solutions [180]. It was also used in some recent works for the geometric design of serial manipulators [93], to solve the forward kinematic of a 3-UPU parallel robot [182], the inverse kinematic of a 4P3R spatial robot manipulator [196], and to find singular configurations of manipulators [148].

Although solutions with small imaginary parts may give information about configuration which will be almost but not quite valid [188], in general computing the imaginary solutions substantially slows down the approaches based on continuation, particularly on problems with a small fraction of real solutions.

1.2.3 Interval methods

Interval-based methods [117] apply a branch-and-prune approach. Therefore, they reduce a given initial box defined from the valid ranges of the joints by 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 split into two sub-boxes and the reduce and split procedures are recursively applied to the two of them. The result is a set of boxes that necessarily contains all the solutions to the system within the initial box. The key element of these methods is how to reduce the boxes, for which different techniques have been proposed.

Interval Newton methods, described in detail in [80, 117], have been applied to solve the position analysis of serial [23, 24] and parallel [106] 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 Bernstein basis permits avoiding interval Jacobian inversions [13, 159]. 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 resolution 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 [82, 92, 105, 131, 198] which, instead of relying on control points, directly bound the the considered polynomials in the space of their defining variables. This defines smaller linear programs which can be solved significantly faster, but require 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 thesis, we introduce a remarkably simple branch-and-prune method to solve a multi-affine system of closure equations obtained using dual quaternions. The simplicity arises from the fact that the conversion to Bernstein basis becomes unnecessary as the evaluation of each function at the corners of an orthotope in the space defined by the variables directly leads to control points. The result is a simple-to-implement and yet efficient algorithm than can be easily parallelized at different levels. In this thesis, we apply the approach both to the position analysis of spherical and to general spatial mechanisms, including cases with positive-dimensional solution sets or multi-loop mechanisms. This allows us to obtain the coupler curves of complicated spatial mechanisms, or even a discretization of the self-motion manifold of a reconfigurable mechanism, in a remarkably simple way.

The multi-affine property in the closure equations of a spatial closed-loop mechanism was identified for the first time in [113] where this property was exploited to reduce the resolution of the equations to a generalized eigenproblem. Unfortunately, many conjectures were introduced, and, therefore, this work received little attention despite its relevance. From the available literature, it can actually be said that most kinematicians are not well aware of the multi-affine property of the closure equations of multi-loop spatial mechanisms when formulated using dual quaternions. Thus, it is the objective of this work to highlight this property and show the advantages derived from it.

1.3 Overview of contributions and outline of the text

The main objective of this thesis is to develop the trapezoid method, a simple, general, and yet efficient method for the position analysis of mechanisms which relies on formulating the problem as a multi-affine algebraic system of equations. The interest of this kind of formulations is due to the fact that they can be solved using a simple branch-and-prune numerical method. This method can isolate all the solutions to the desired resolution level, irrespective of the dimension of the solution set. The combination of the proposed formulation and resolution methods will have the following interesting properties:

1. **General**. The use of a set of kinematic loop equations is general and can be easily automatized using standard algorithms developed in graph theory.
2. **Algebraic**. The formulation is fully algebraic, it does not involve any mathematical operations other than additions, subtractions, multiplications, and divisions [191].
3. **Simple**. Although the theory that leads to it cannot be classified as simple, the method is easily implementable.

The introduction of the new proposed methodology for the resolution of position analysis problems is based on the following contributions:

- The formulation of position analysis problems using dual quaternions to obtain a multi-affine formulation.

- A fast branch-and-prune solver for multi-affine systems of equations which we call the *trapezoid* solver. Multi-affine systems exhibit a set of properties similar to those of systems expressed in Berstein basis. The most important of them being the interpolation and the convex hull properties. Exploiting them would lead to a very fast and efficient branch-and-prune method for solving systems of multi-affine systems.

- A methodology to generate redundant multi-affine equations that can be leveraged to significantly improve the quality in the output of the solver.

- A variable elimination procedure that reduces the complexity of the equations and significantly speeds up the proposed solver.

- Parallelization of the resolution of position analysis problems. Previous schemes for the resolution of position analysis problems cannot easily parallelized. In the proposed branch-and-prune method, different levels of parallelization naturally arise.

The method presented in this work is applicable to the position analysis of challenging mechanisms. As testbeds, we use:

1. **Spherical mechanisms**. In a spherical mechanism, the axes of revolution intersect at a single point, which constraints the motion of the end-effector to a sphere centered at this point. Spherical mechanisms are used, for instance, in automotive differential or in robotic wrist and, thus, characterizing their motions has relevant practical applications.

2. **General 6R serial robots**. The inverse kinematics problem of a serial robot consists in finding the values of its joint variables given the position and orientation of its end-effector, relative to the base, and the values of all of the geometric link parameters [162]. This problem can be seen to be equivalent to determine the feasible configurations of a single-loop kinematic chain.

3. **Overconstrained mechanisms**. An overconstrained mechanism is a linkage that has more degrees of freedom than is predicted by the Grübler-Kutzbach mobility formula [90]. Many overconstrained mechanisms have been described and several of them have relevant practical applications [11, 152]. One of the fundamental problems in this context is to derive the input-output relations for arbitrary overconstrained mechanism. The analytical derivation of such relations is done case by case and it is a complex process in general. Thus, the availability of an efficient solver to numerically derive them is a tool of capital relevance in this field.

4. **Mobile mechanisms**. We consider kinematic loops with 7 joints, which in general have a one-degree of freedom solution set although kinematotropic linkages can have a variable number degrees of freedom, depending on the configuration. Both cases are considered herein to evaluate the performance of the proposed method.

5. **Multi-loop mechanisms**. These problems appear, for instance, in parallel robots and are used to illustrate the generality of the proposed solver.

The rest of this thesis is organized as follows. Chapter 2 describes in detail the multi-affine formulation used in this work. The formulation is first introduced for spherical mechanisms, where only quaternions are necessary and then extended to spatial mechanism using dual quaternions. Chapter 3 characterizes the properties of multi-affine maps and presents the trapezoid method, an interval-based solver relying on them. Chapter 4 analyses the performance of this basic solver on different testbeds, including spherical mechanisms, generic 6R loops, overcontrained systems, mobile kinematic loops, and multi-loop mechanisms. Chapters 5 and 6 present extensions on the basic solver relying, respectively, on the generation of redundant equations and on the elimination of variables. We will show that these extensions significantly improve both the accuracy and the performance of the solver. Finally, Chapter 7 summarizes the contributions of this work and points to directions deserving further attention.

2 A multi-affine formulation

In this chapter we describe basic tools on quaternions and dual quaternions and then we use them to define multi-affine formulations both for spherical and spatial mechanisms.

2.1 Basics on quaternions and dual quaternions

We next include some basic facts on quaternions needed in our analysis. A complete treatment of the subject can be found, for example, in [156].

Quaternions are numbers, first introduced by Hamilton in 1843 [60], that can written as $a + \mathrm{i}b + \mathrm{j}c + \mathrm{k}d$ where $\mathrm{i}^2 = \mathrm{j}^2 = \mathrm{k}^2 = \mathrm{ijk} = -1$. They can be seen as a generalization of complex numbers. It was rapidly realized that quaternion algebra yields more efficient algorithms than matrix algebra for applications involving rigid-body transformations. Nowadays, quaternions play a fundamental role in the representation of spatial rotations [150]. The idea after the invention of quaternions was to represent a rotation in $\mathbb{R}^3$ as it was done in $\mathbb{R}^2$ using complex numbers. Cayley solved the problem in $\mathbb{R}^3$ and he also showed that a rotation in $\mathbb{R}^4$ can also be described by means of quaternions [71, 173]. Afterward, Clifford introduced the concept of double quaternion, $\mathbf{q}_1 + e\,\mathbf{q}_2$, and dual quaternion, $\mathbf{q}_1 + \varepsilon\,\mathbf{q}_2$, where $\mathbf{q}_1$ and $\mathbf{q}_2$ are ordinary quaternions and $e^2 = 1$ and $\varepsilon^2 = 0$. While double quaternions found direct application to represent rotations in $\mathbb{R}^4$, dual quaternions found application to encapsulate both translations and rotations in $\mathbb{R}^3$ into a unified representation. It has been found that spatial displacements can be approximated by rotations in $\mathbb{R}^4$ [173]. In other words, a dual quaternion can be approximated by a double quaternion.

The use of quaternions in kinematics has attracted the attention of many researchers due to its compactness and robustness. It actually can be considered as the primary singularity-free representation of rotations in $\mathbb{R}^3$ [203]. Its use for formulating loop equations in spherical mechanism was already treated in detail in [164]. Yang and Freudenstein introduced the use of dual quaternions for representing spatial displacements [199] and, recently, they have been used to obtain the closure polynomial of the general 6R serial robot [202]. The use of dual quaternions has been proven advantageous with respect to alternative representations in several comparative studies [7, 49, 151, 158].

Different variations and notations of dual quaternions can be found in the literature [51, 55, 85, 97, 156, 173]. Some authors use unit dual quaternions, in whose case terms in the sine and cosine in each joint angle appear in the formulation [51]. 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 [87]. Moreover, 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 greatly simplifies the formulation [65], but it suffers from the drawback associated with the tangent half-angle substitution (it is singular for joint angles equal to π). Nevertheless, as we will show later, when using a branch-and-prune resolution method this drawback can be elegantly overcome.

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

$$\mathbf{r}_{\mathbf{p}}(\theta) = \cos\left(\frac{\theta}{2}\right) + \sin\left(\frac{\theta}{2}\right)\hat{p}. \tag{2.1}$$

where $\hat{p} = p_x\mathrm{i} + p_y\mathrm{j} + p_z\mathrm{k}$. Since $\mathbf{r}_{\mathbf{p}}(\theta)$ and $\mathbf{r}_{-\mathbf{p}}(-\theta)$ 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. (2.1) can also be represented by the following non-unit quaternion

$$\mathbf{r}_{\mathbf{p}}(t) = 1 + t\,\hat{p}, \tag{2.2}$$

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 $\varepsilon^2 = 0$ [48]. That is,

$$\mathbf{t}_{\mathbf{p}}(d) = \cos\left(\varepsilon\,\frac{d}{2}\right) + \sin\left(\varepsilon\,\frac{d}{2}\right)\,\hat{p}. \tag{2.3}$$

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

$$\mathbf{t}_{\mathbf{p}}(d) = 1 + \varepsilon\,\frac{d}{2}\,\hat{p}. \tag{2.4}$$

With the above 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:

$$\mathbf{s}_{\mathbf{p}}(t,d) = \mathbf{r}_{\mathbf{p}}(t)\,\mathbf{t}_{\mathbf{p}}(d) = 1 + t\,\hat{p} + \varepsilon\frac{d}{2}\left(-t+\hat{p}\right). \tag{2.5}$$

Notice that this does not represent a general screw displacement as the rotation axis passes through the origin. The general form can be found, for example, in [157, 173].

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 using unit dual quaternions to represent spatial displacements, it is not difficult to prove that

$$\|\mathbf{q}_1\| = 1 \tag{2.6}$$

and

$$\mathbf{q}_1 \cdot \mathbf{q}_2 = 0. \tag{2.7}$$

Nevertheless, in our case, while Eq. (2.7) is satisfied, Eq. (2.6) 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. (2.7), called the Study quadric after his discoverer [166].

2.2 Closure equations for spherical mechanisms using quaternions

An spherical kinematic loop equation can always be formulated in terms of products of unit quaternions

$$\mathbf{r}_{\mathbf{p}_1}(t_1)\cdot\mathbf{r}_{\mathbf{p}_2}(t_1)\ldots\mathbf{r}_{\mathbf{p}_r}(t_r) = c, \tag{2.8}$$

where c is a real scalar and where each quaternion has the form

$$\mathbf{r}_{\mathbf{p}_i}(t_i) = 1 + t_i\,(p_{ix}\,\mathrm{i} + p_{iy}\,\mathrm{j} + p_{iz}\,\mathrm{k}), \tag{2.9}$$

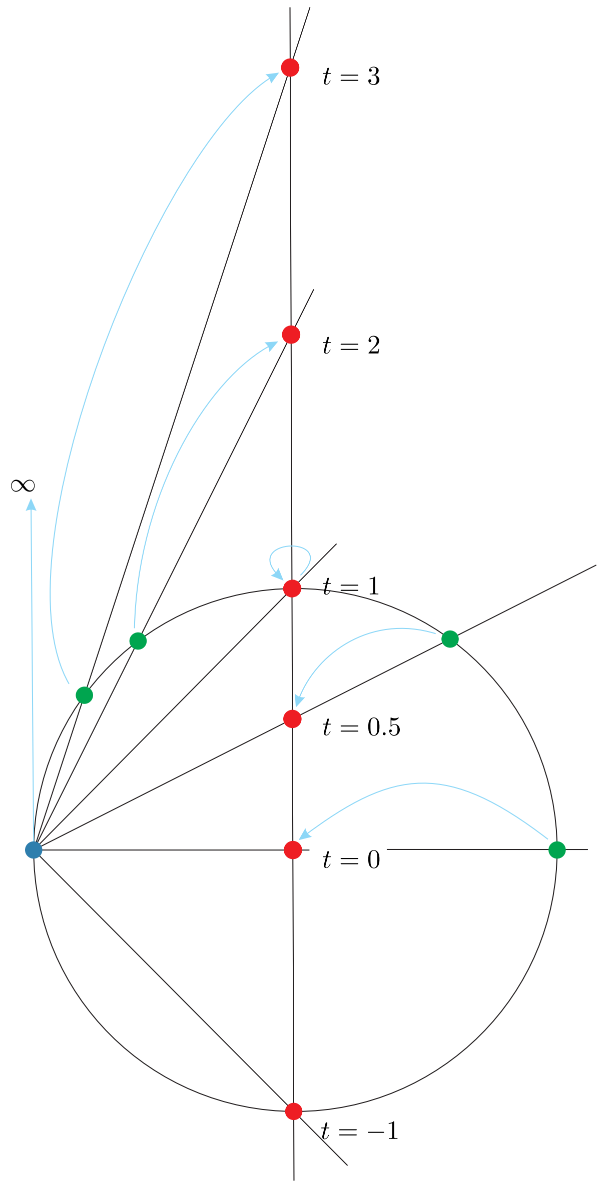


Figure 2.1: The Weierstrass substitution maps the interval $[-\pi, \pi]$ onto the interval $(-\infty, \infty)$. Numerical problems arise as we approach $\pm\pi$.

where $\mathbf{p}_i = (p_{ix}, p_{iy}, p_{iz})^T$ is a unit vector representing the rotation axis and $t_i = \tan\left(\frac{\theta_i}{2}\right)$, θ_i being the rotated angle about the vector. The angle θ_i can be constant or variable.

The expansion of Eq. (2.8) leads to the following four scalar equations

$$
\begin{aligned}
f_0(t_1, \ldots, t_n) &= \sqrt{(t_1^2 + 1) \cdots (t_n^2 + 1)}, \\
f_j(t_1, \ldots, t_n) &= 0, \quad j = 1, 2, 3.
\end{aligned}
$$

These four equations are not independent because they correspond to the components of a unit quaternion. While f_j, $j = 1, 2, 3$, are affine polynomials in each variable, f_0, after clearing the radical, leads to a quadratic polynomial. For obvious simplicity reasons, in what follows, we will take f_j, $j = 1, 2, 3$, as the set of three independent equations. Therefore, if our problem can be described in terms of n independent loop equations, we will have a set of $3n$ multi-affine polynomial equations.

A word of caution should be added here. Sometimes, in an abuse of language, a system of equations like the one given by f_j, $j = 1, 2, 3$, is called multi-linear. Strictly speaking, it is multi-affine. Indeed, f_j can be expressed as a function of t_i as $f_j = a_j t_i + b_j$ which is not linear in t_i, but affine.

2.2.1 Reducing numerical issues by shifting rotations

The procedure just described solves the position analysis of the studied mechanism provided that we exclude those configurations in which at least one of the revolute joint angles is π. 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 $y = 0$ (see Fig. 2.1). Thus, if $\theta_i = \pi$, t_i goes to infinity. Then, numerical problems will occur when identifying valid configurations of the mechanisms in the case in which at least one of its revolute joint angles 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 possible solution to this situation is to split the problem in two: one for $\theta_i^+ \in [0, \pi]$, and another one for $\theta_i^- \in [-\pi, 0]$, where $\theta_i^+ = \theta_i + \pi/2$ and $\theta_i^- = \theta_i - \pi/2$ which can be accomplished by shifting the origin $\pm\pi/2$. This operation can be done for all the rotational variables whose valid range includes π. In other words, if our problem has n of such rotation variables, we will decompose it into 2^n problems where, for each problem, the domain for all rotational variables is $[-1, 1]$. Note that using small ranges for the variables increases the numerical stability of the algorithm and that the 2^n sub-problems generated are independent between them and, consequently, they can be solved in parallel.

2.3 Closure equations for spatial mechanisms using dual quaternions

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. The 4-tuples $(\theta_i, d_i, \alpha_i, a_i)$ are defined as the Denavit-Hartenberg parameters (DH parameters, for short) of link i. Using Eq. (2.5), and denoting $u_i = \tan\frac{\alpha_i}{2}$, this displacement can be compactly expressed as

$$\mathbf{s_z}(t_i, d_i)\,\mathbf{s_x}(u_i, a_i) = \left(1 + t_i\,\mathrm{k} + \varepsilon\,\frac{d_i}{2}\,(-t_i + \mathrm{k})\right)\left(1 + u_i\,\mathrm{i} + \varepsilon\,\frac{a_i}{2}\,(-u_i + \mathrm{i})\right). \tag{2.10}$$

Since we do not adhere to the unity condition, the above expression is simpler than those used, for example, in [51, 202].

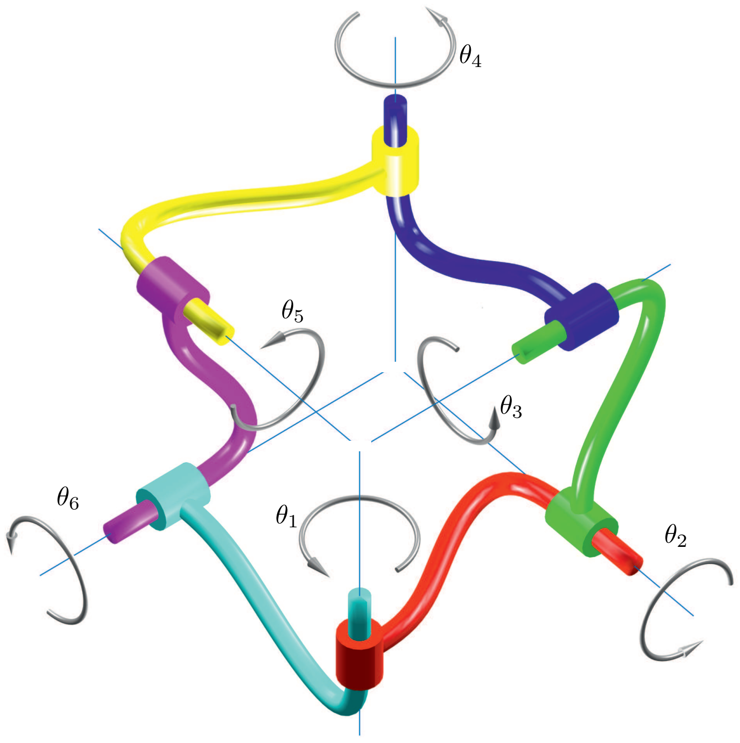
By composing terms of the form given in Eq. (2.10), we can derive the closure equations for any closed-loop mechanism expressed in terms of DH parameters. This composition leads to expressions of the form

$$\mathbf{s_z}(t_1, d_1)\,\mathbf{s_x}(u_1, a_1)\cdots\mathbf{s_z}(t_n, d_n)\,\mathbf{s_x}(u_n, a_n) = c, \tag{2.11}$$

where c is a real scalar. This means that the components in i, j, k, ε, εi, εj, and εk resulting from expanding the left hand side of Eq. (2.11) necessarily vanish. As a result, we obtain seven scalar equations from which only six are independent due to the constraint in Eq. (2.7). In what follows, we simply discard the one corresponding to the term in ε, as it is done, for example, in [85]. The relevance of these equations is that they are multi-affine in terms of t_i and d_i, $i = 1, \ldots, n$. This is better understood through an example.

2.3.1 Formulation example

Let us consider the Bricard's 6R mechanism depicted in Fig. 2.2 [193]. According to Grübler-Kutzbach formula, it should be rigid, but it is mobile because of the special choice of their design parameters. This kind of exceptional mechanisms are called *overconstrained* (see, for example, [124] and the references therein for details on these kind of 6R mechanisms). In general, 6R closed-loop mechanisms can have up to 16 possible rigid configurations [136], but this exceptional mechanism has a 1-dimensional configuration space, which is technically referred to as a 1-dimensional self-motion manifold.



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 2.2: Bricard's overconstrained 6R closed-loop mechanism and its DH parameters. All joint angles, θ_i, in the shown mechanism configuration are $\pi/2$.

The closure equation of this mechanism can be expressed as:

$$\begin{aligned}
&(1+t_1\,\mathrm{k})(1+\mathrm{i}+\varepsilon\,(-1+\mathrm{i})/2)\\
&(1+t_2\,\mathrm{k})(1-\mathrm{i}+\varepsilon\,(-1+\mathrm{i})/2)\\
&(1+t_3\,\mathrm{k})(1+\mathrm{i}+\varepsilon\,(-1+\mathrm{i})/2)\\
&(1+t_4\,\mathrm{k})(1-\mathrm{i}+\varepsilon\,(-1+\mathrm{i})/2)\\
&(1+t_5\,\mathrm{k})(1+\mathrm{i}+\varepsilon\,(-1+\mathrm{i})/2)\\
&(1+t_6\,\mathrm{k})(1-\mathrm{i}+\varepsilon\,(-1+\mathrm{i})/2)=c.
\end{aligned} \tag{2.12}$$

Therefore, after expanding the left hand side in Eq. (2.12) and grouping the terms in i, j, and k, we obtain the following three equations:

$$\begin{aligned}
t_2t_3-t_1t_4-t_1t_2-t_1t_6+t_2t_5-t_3t_4-\\
t_3t_6+t_4t_5-t_5t_6+t_1t_2t_3t_5+t_1t_2t_4t_6-\\
t_1t_3t_4t_5+t_1t_3t_5t_6-t_2t_3t_4t_6+t_2t_4t_5t_6+\\
t_1t_2t_3t_4t_5t_6=0,
\end{aligned} \tag{2.13}$$

$$\begin{aligned}
-t_4-t_6-t_2+t_1t_3t_4-t_1t_2t_3-t_1t_2t_5+\\
t_1t_3t_6-t_1t_4t_5+t_2t_3t_5+t_1t_5t_6+\\
t_2t_4t_6-t_3t_4t_5+t_3t_5t_6+t_1t_2t_3t_4t_6-\\
t_1t_2t_4t_5t_6+t_2t_3t_4t_5t_6=0,
\end{aligned} \tag{2.14}$$

$$\begin{aligned}
-t_3-t_5-t_1+t_1t_2t_4+t_1t_2t_6+t_1t_3t_5-\\
t_2t_3t_4+t_1t_4t_6-t_2t_3t_6+t_2t_4t_5-t_2t_5t_6+\\
t_3t_4t_6-t_4t_5t_6+t_1t_2t_3t_4t_5-t_1t_2t_3t_5t_6+\\
t_1t_3t_4t_5t_6=0.
\end{aligned} \tag{2.15}$$

Likewise, repeating the same process for the terms multiplying $\varepsilon\mathrm{i}$, $\varepsilon\mathrm{j}$, and $\varepsilon\mathrm{k}$, we obtain the following equations:

$$\begin{aligned}
t_1t_3-t_1t_5+t_2t_4-t_2t_6+t_3t_5+t_4t_6+\\
t_1t_2t_3t_4-t_1t_2t_3t_6-t_1t_2t_4t_5+t_1t_2t_5t_6+\\
t_1t_3t_4t_6+t_2t_3t_4t_5-t_1t_4t_5t_6-t_2t_3t_5t_6+\\
t_3t_4t_5t_6-3=0,
\end{aligned} \tag{2.16}$$

$$\begin{aligned}
&t_5 - t_3 - 3t_1 + t_1t_2t_4 - t_1t_2t_6 + t_1t_3t_5 - \\
&t_2t_3t_4 + t_1t_4t_6 + t_2t_3t_6 + t_2t_4t_5 - t_2t_5t_6 - \\
&t_3t_4t_6 + t_4t_5t_6 + t_1t_2t_3t_4t_5 - t_1t_2t_3t_5t_6 + \\
&t_1t_3t_4t_5t_6 = 0, \qquad (2.17)\\
&t_2 - t_6 + t_1t_2t_3 - t_1t_3t_4 + t_1t_4t_5 - t_1t_5t_6 = 0. \qquad (2.18)
\end{aligned}$$

Equations (2.13)-(2.15) actually correspond to the closure conditions for the *spherical indicatrix* of the mechanism, and equations Eqs. (2.16)-(2.18) can be seen as a condition in the tangent space of the configuration space of this spherical indicatrix [172]. It can be easily checked that all these equations are multi-affine. In next Chapter, we investigate the properties of such type of equations, which will be later exploited to come up with an efficient branch-and-prune algorithm to solve them.

3

A solver for multi-affine formulations

In this chapter we describe two fundamental properties of multi-affine maps (namely, the interpolation and convex hull properties) and then we exploit these properties to define and efficient and highly parallelizable solver for position analysis problems which we call the *trapezoid method*.

3.1 Properties of affine maps

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

$$f(\mathbf{x}) = \sum_{\mathbf{p}=\mathbf{0}}^{\mathbf{m}} a_{\mathbf{p}}\, \mathbf{x}^{\mathbf{P}}, \tag{3.1}$$

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 $i \in \{1, \ldots, n\}$ and where $\mathbf{x} = (x_1, \ldots, x_n)$ 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 [47],

$$B_{\mathbf{p},\mathbf{m}}(\mathbf{x}) = b_{p_1,m_1}(x_1) \ldots b_{p_n,m_n}(x_n), \tag{3.2}$$

where

$$b_{p_i,m_i}(x_i) = \binom{m_i}{p_i} x_i^{p_i}\, (1 - x_i)^{m_i - p_i}. \tag{3.3}$$

Using this basis

$$f(\mathbf{x}) = \sum_{\mathbf{p}=\mathbf{0}}^{\mathbf{m}} c_{\mathbf{p}}\, B_{\mathbf{p},\mathbf{m}}(\mathbf{x}), \tag{3.4}$$

where

$$c_{\mathbf{p}} = \sum_{\mathbf{j}=\mathbf{0}}^{\mathbf{p}} \frac{\binom{\mathbf{p}}{\mathbf{j}}}{\binom{\mathbf{m}}{\mathbf{j}}} a_{\mathbf{j}} \tag{3.5}$$

are the so-called control points and where

$$\binom{\mathbf{p}}{\mathbf{j}} = \binom{p_1}{j_1} \cdots \binom{p_n}{j_n} \tag{3.6}$$

and

$$\binom{\mathbf{m}}{\mathbf{j}} = \binom{m_1}{j_1} \cdots \binom{m_n}{j_n}. \tag{3.7}$$

Since the formulation for the kinematic constraints derived in the previous section involves only multi-affine equations, we are only interested in the case where $\mathbf{m} = (1, \ldots, 1)$. In this case, the Bernstein-based representation reduces to

$$b_{p_i,1}(x_k) = \begin{cases} (1 - x_i) & p_i = 0 \\ x_i & p_i = 1 \end{cases} \tag{3.8}$$

and it can be seen that the control points simplify to

$$c_{\mathbf{p}} = f(\mathbf{p}), \tag{3.9}$$

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

$$\mathcal{B}_1 = [0,1]_1 \times [0,1]_2 \times \cdots \times [0,1]_n. \tag{3.10}$$

Using Eqs. (3.8) and (3.9), f can be expressed as

$$f(\mathbf{x}) = \sum_{\mathbf{p}=0}^{(1,\ldots,1)} f(\mathbf{p}) \prod_{i=1}^{n} b_{p_i,1}(x_i). \tag{3.11}$$

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

$$\mathcal{B} = [x_1^l, x_1^u] \times [x_2^l, x_2^u] \times \cdots \times [x_n^l, x_n^u], \tag{3.12}$$

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

$$\mathcal{V} = \left\{ (v_1, \ldots, v_n) \in \mathbb{R}^n \mid v_i \in \left\{ x_i^l, x_i^u \right\} \right\}. \tag{3.13}$$

Then, it can be seen that

$$f(\mathbf{x}) = \sum_{\mathbf{p}=\mathbf{0}}^{(1,\dots,1)} f(\mathbf{v_p}) \prod_{i=1}^{n} b_{p_i,1}(s(x_i)), \tag{3.14}$$

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

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

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

Two important properties of multi-affine maps directly follow from the previous considerations (alternative presentations can be found in [10,139,171]). The first one is the interpolation property. Equation (3.14) can be expressed, using Eq. (3.8), as the following linear interpolation between two functions of $n-1$ variables

$$f(\mathbf{x}) = (1 - s(x_1))\, f_l(x_2, \dots, x_n) + s(x_1)\, f_u(x_2, \dots, x_n) \tag{3.15}$$

where

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

and

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

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 by the linear interpolation of its control points. Actually this can be seen as a specialization for multi-affine polynomials of the De Casteljau's algorithm [47], which is a robust and efficient way to evaluate polynomials in Bernstein form.

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

$$g(\mathbf{x}) = (\mathbf{x}, f(\mathbf{x})), \tag{3.16}$$

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

$$g(\mathbf{x}) = \sum_{\mathbf{p}=\mathbf{0}}^{(1,\dots,1)} \mathbf{d}_\mathbf{p} \prod_{i=1}^{n} b_{p_i,1}(s(x_i)), \tag{3.17}$$

where $\mathbf{d}_\mathbf{p}$ is the element of

$$\mathcal{D} = \{(\mathbf{v}, f(\mathbf{v})) \in \mathbb{R}^{n+1} | \mathbf{v} \in \mathcal{V}\}, \tag{3.18}$$

corresponding to $\mathbf{v}_\mathbf{p}$.

Since the values of the Bernstein polynomials in the range $[0, 1]$ are non-negative and form a partition of the unity [47], Eq. (3.17) is a particular convex combination of the points in $\mathcal{D}$ 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 $\mathcal{D}$:

$$\mathcal{H} = \left\{ \sum_{\mathbf{p}=\mathbf{0}}^{(1,\dots,1)} \mathbf{d}_\mathbf{p}\, \lambda_\mathbf{p} \mid \sum \lambda_\mathbf{p} = 1,\ 0 \leq \lambda_\mathbf{p} \leq 1 \right\}. \tag{3.19}$$

Summarizing the two properties, the evaluation of a multi-affine function at a point within a box can be expressed by interpolation of the values of the function evaluated at the box corners. As a result, these evaluations are always inside the convex region defined by the evaluations at these box corners. Next, these properties are leveraged to come up with an efficient algorithm for solving systems of multi-affine equations.

3.2 A Branch-and-prune solver for multi-affine systems

Let us define the system of equations

$$\mathbf{F}(\mathbf{x}) = 0, \tag{3.20}$$

where $\mathbf{F} = (f_1(\mathbf{x}), \dots, f_m(\mathbf{x}))$, and each function f_j is a multi-affine polynomial in the unknowns $\mathbf{x} = (x_1, \dots, x_n)$.

The proposed method, summarized in Algorithm 3.1, identifies arbitrarily small boxes containing solution candidates of Eq. (3.20) within a given search box $\mathcal{B}_s \subset \mathbb{R}^n$ using a branch-and-prune scheme that iterates two operations, *box reduction* and *box bisection*. Using box reduction (line 6 of Algorithm 3.1), portions of a given box $\mathcal{B} \subseteq \mathcal{B}_s$ containing no solution are ruled out by narrowing some of its defining intervals. Using this process (a) the box is reduced to an empty set, in which case it contains no solution and it can be safely discarded (line 7), or (b) the longest side of the box (denote by SIZE($\mathcal{B}$)) is under a specified threshold σ (line 8), in which case it is considered a solution box candidate, either verified to include a solution point (line 10)

Algorithm 3.1: The trapezoid method.

```
TRAPEZOID($\mathcal{B}_s, \mathcal{F}, \rho, \sigma$)
Input  : $\mathcal{B}_s$: The initial search box.
         $\mathcal{F}$: The evaluation of the functions defining the
            system of equations at the corners of $\mathcal{B}_s$.
         $\rho, \sigma$: Two scalar parameters.
Output: $V$: A set of verified solution boxes.
         $S$: The rest of solution boxes.

1  $L \leftarrow \{\mathcal{B}_s\}$
2  $V \leftarrow \emptyset$
3  $S \leftarrow \emptyset$
4  while $L \neq \emptyset$ do
5      $\mathcal{B} \leftarrow$ EXTRACTFIRST($L$)
6      $(\mathcal{B}, v) \leftarrow$ REDUCEBOX($\mathcal{B}_s, \mathcal{F}, \mathcal{B}, \rho, \sigma$)
7      if not EMPTY($\mathcal{B}$) then
8          if SIZE($\mathcal{B}$) $< \sigma$ then
9              if $v$ then
10                 $V \leftarrow V \cup \{\mathcal{B}\}$
11             else
12                 $S \leftarrow S \cup \{\mathcal{B}\}$
13         else
14             $(\mathcal{B}_1, \mathcal{B}_2) \leftarrow$ BISECTBOX($\mathcal{B}$)
15             $L \leftarrow L \cup \{\mathcal{B}_1, \mathcal{B}_2\}$
16 RETURN($V, S$)
```

or not (line 12), or (c) the box is still larger than σ, in which case it is bisected along its largest side into two equal sub-boxes (line 14). If the later occurs, the two resulting boxes are added to the list of boxes to be processed in subsequent iterations (line 15). If there is only a finite number of solution points in $\mathcal{B}_s$, this method returns (line 16) 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 returned boxes contain the portions of this variety in $\mathcal{B}_s$. The method is thus *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 couple of box reduction procedures. While the first one is already described in [132], the second can be seen as a simplification of a method presented in [159]. The simplification is possible thanks to the multi-affinity of the closure equations derived using dual quaternions.

3.2.1 Pruning using linear programming

As already mentioned, finding the values for which $f_i(\mathbf{x}) = 0$ is equivalent to determining the region $\mathcal{R}_i$ including the points of the form $(\mathbf{x}, 0)$ for the function g_i defined from f_i as in Eq. (3.16). The solution of the system of equations is, thus, the set $\mathcal{R} = \cap_{i=1}^{m} \mathcal{R}_i$. Each $\mathcal{R}_i$ can be bounded by the $n+2$ linear constraints derived from the convex hull property described in Section 3.1, *i.e.*,

$$\sum_{\mathbf{v}\in\mathcal{V}} d^i_{\mathbf{v}}\, \lambda^i_{\mathbf{v}} = (x_1, \ldots, x_n, 0)^T, \tag{3.21}$$

$$\sum_{\mathbf{v}\in\mathcal{V}} \lambda^i_{\mathbf{v}} = 1, \tag{3.22}$$

with $\{d^i_{\mathbf{v}}\}$ the set of control points of f_i, and $\lambda^i_{\mathbf{v}}$, $0 \leq \lambda^i_{\mathbf{v}} \leq 1$, the corresponding linear interpolation parameters. The collection of such constraints for all the equations form a linear program whose feasible set is a convex bound of $\mathcal{R}$. Lower (upper) bounds for this set can be obtained minimizing (maximizing) the value of each variable, x_i, subject to all the previous constraints. If the linear program is unfeasible, the considered box can be eliminated as it includes no solution.

The linear program to be solved has $n_r = \sum_{i=0}^{m}(n_i+2)$ rows and $n_c = n+\sum_{i=0}^{m} 2^{n_i}$ columns, where $n_i \leq n$ is the number of variables in the i-th equation and, according to [77], the computational cost of solving a linear program is at least $O(n_r^{3.5})$. This high cost dominates the whole solving process using this procedure.

3.2.2 Pruning using projections: The trapezoid method

For the sake of clarity, we first describe the procedure when (3.20) consists of just one equation in two variables and then show how it also applies to the general case.

Assume that we want to find all solutions of a multi-affine equation $f(\mathbf{x}) = 0$, for $\mathbf{x} = (x_1, x_2)$ in $\mathcal{B} = [x_1^l, x_1^u] \times [x_2^l, x_2^u] \in \mathbb{R}^2$ (Fig. 3.1). Since $(\mathbf{x}, f(\mathbf{x}))$ must lie within the convex hull $\mathcal{H}$ of the 2^2 points $\{(\mathbf{x}, f(\mathbf{x}))|\ \mathbf{x} \in \{x_1^l, x_1^u\} \times \{x_2^l, x_2^u\}\}$ of $\mathbb{R}^3$, we simply project $\mathcal{H}$ onto each x_i-$f(\mathbf{x})$ coordinate plane, as depicted in the bottom part of Fig. 3.1 (left and right for the planes x_1-$f(\mathbf{x})$ and x_2-$f(\mathbf{x})$, respectively). Each projection defines a trapezoid whose vertices are given by x_i^l and x_i^u with $i \in \{1, 2\}$ in the horizontal axis and by

$$\underline{f_i^l}(\mathcal{B}) = \min\{f(\mathbf{v})|\mathbf{v} \in \mathcal{V}(\mathcal{B}), x_i = x_i^l\}, \tag{3.23}$$

$$\overline{f_i^l}(\mathcal{B}) = \max\{f(\mathbf{v})|\mathbf{v} \in \mathcal{V}(\mathcal{B}), x_i = x_i^l\}, \tag{3.24}$$

$$\underline{f_i^u}(\mathcal{B}) = \min\{f(\mathbf{v})|\mathbf{v} \in \mathcal{V}(\mathcal{B}), x_i = x_i^u\}, \tag{3.25}$$

$$\overline{f_i^u}(\mathcal{B}) = \max\{f(\mathbf{v})|\mathbf{v} \in \mathcal{V}(\mathcal{B}), x_i = x_i^u\}, \tag{3.26}$$

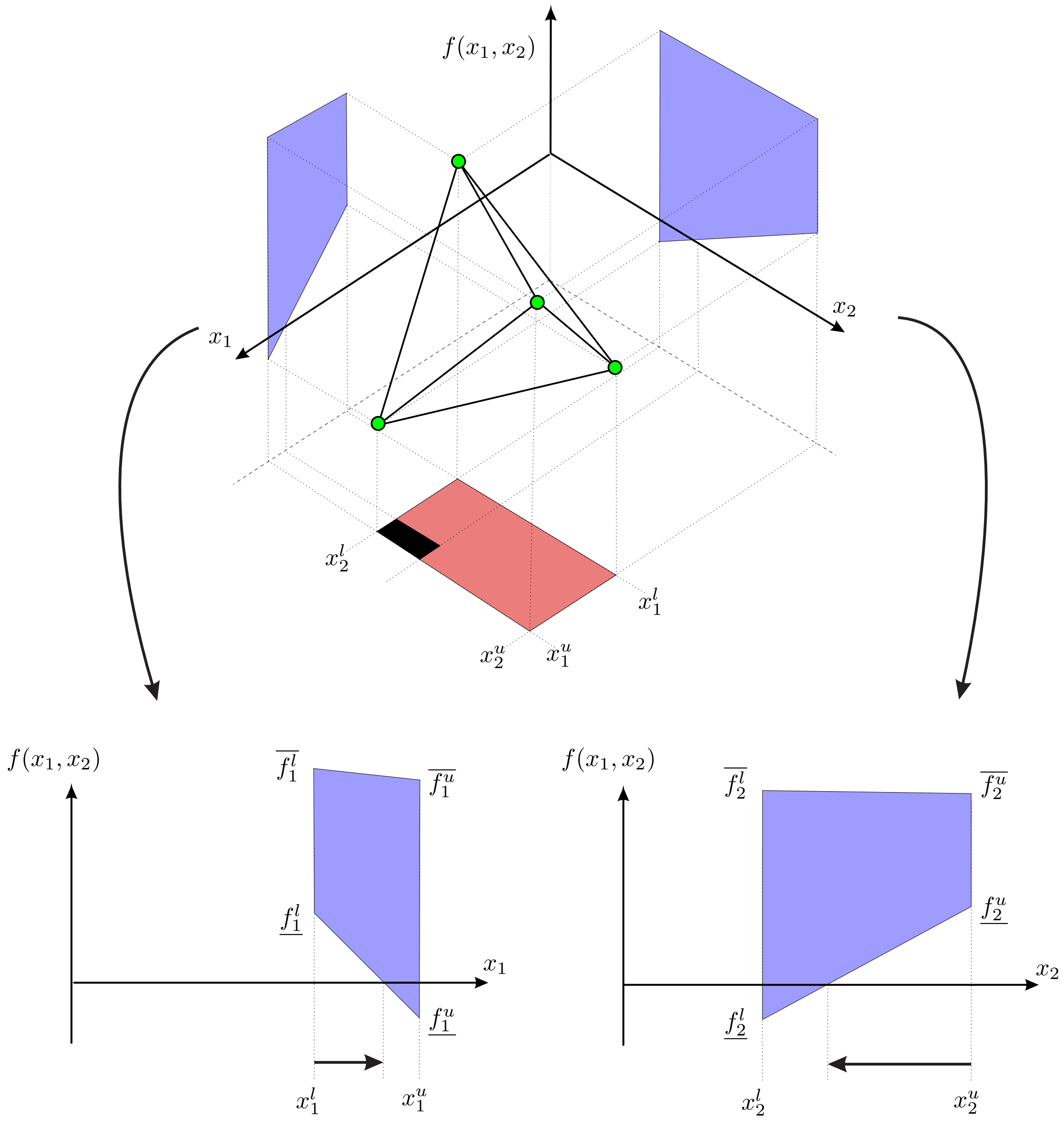


Figure 3.1: The image of the points in the box $[x_1^l, x_1^u] \times [x_2^l, x_2^u]$ (shown in red) for any multi-affine function $f(x_1, x_2)$ necessarily lies inside a tetrahedron. The vertices of the tetrahedron (shown in green) are obtained by evaluating f in the corners of the box. Then, from the 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.

Algorithm 3.2: The ReduceBox procedure.

```
ReduceBox($\mathcal{B}_s, \mathcal{F}, \mathcal{B}, \rho, \sigma$)
Input  : $\mathcal{B}_s$: The initial search box.
         $\mathcal{F}$: The evaluation of the functions defining the
            system of equations at the corners of $\mathcal{B}_s$.
         $\mathcal{B}$: The box to be reduced.
         $\rho, \sigma$: Two scalar parameters.
Output: $\mathcal{B}$: A reduced or empty box.
         $v$: The verification state of the box.
1  $v \leftarrow$ FALSE    // TRUE if the box includes a solution
2  repeat
3      $\nu_1 \leftarrow$ Volume($\mathcal{B}$)
4      $m \leftarrow$ TRUE    // TRUE if the box passes Miranda's test
5      foreach $f_j \in \mathcal{F}$ do
6          $f \leftarrow$ Interpolate($\mathcal{B}_s, f_j, \mathcal{B}$)
7          foreach $i \in \{1, \ldots, n\}$ do
8              $(\underline{f_i^l}, \overline{f_i^l}, \underline{f_i^u}, \overline{f_i^u}) \leftarrow$ Trapezoid($f, i$)
9              $m \leftarrow m$ and [ $(\underline{f_i^l} > 0$ and $\overline{f_i^u} < 0)$ or $(\overline{f_i^l} < 0$ and $\underline{f_i^u} > 0)$ ]
10             $\mathcal{B}_i \leftarrow$ Clipping($\mathcal{B}_i, \underline{f_i^l}, \overline{f_i^l}, \underline{f_i^u}, \overline{f_i^u}$)
11     $v \leftarrow v$ or $m$
12 until Empty($\mathcal{B}$) or Size($\mathcal{B}$) $< \sigma$ or Volume($\mathcal{B}$)$/\nu_1 > \rho$
13 Return($\mathcal{B}$, $v$ or Newton($\mathcal{B}_s$, $\mathcal{F}$, Center($\mathcal{B}$)) $\in \mathcal{B}$)
```

in the vertical one. Clearly, we can prune the values of x_i for which the corresponding trapezoid does not intersect the line defined by $f(\mathbf{x}) = 0$. If the trapezoid and the line do not intersect at all, the considered box includes no solution and it is discarded. Otherwise, the trapezoid-line clippings usually reduce the ranges of some variables defining a box that bounds the sought-after solution set more tightly.

Algorithm 3.2 describes the box reduction procedure for a system of multi-affine equations in $n>2$ variables. The procedure iterates for all the equations in the system (line 5). For all of them (line 6), the values of the function at the corners of the considered box, $\mathcal{B}$, are computed from the values, f_j, of the function at the corners of the initial box, $\mathcal{B}_s$, using the interpolation rule in Eq. (3.15). Then (line 8), these values are projected to the different coordinate planes $x_i - f_j$ to define a trapezoid whose extremes are given in Eqs. (3.23)-(3.26). This trapezoid is clipped with the line $f_j = 0$ to eventually reduce $\mathcal{B}_i$, the *i-th* range of $\mathcal{B}$ (line 10). If the resulting box is not empty, nor small enough, and the reduction for all ranges is significant, i.e., if the volume of the box reduces below a given threshold ρ, the whole process is repeated to obtain tighter bounds for the solution set (line 12). As we will show in Chapter 4, although this strategy produces less pruning than alternative methods, it results advantageous due to its low computational cost. This computational cost can be further reduced parallelizing the algorithm's operations, for instance, processing the different sub-problems defined in Section 2.2.1 independently. The method can also be parallelized at other levels, for instance, when processing the different boxes in the main loop of Algorithm 3.1 or even when processing the different equations and variables (lines 5 or 7 in Algorithm 3.2). Finally, the basic operations of the method, the interpolation (line 6 in Algorithm 3.2) and the projection on coordinate planes (line 8 in Algorithm 3.2), can also be parallelized in single instruction, multiple data (SIMD) architectures.

The computational cost of the trapezoid pruning procedure is $O(m\,n\,2^n)$ with m and n the number of equations and variables in the system, respectively. This is so since the procedure iterates over all equations (line 5 and variables in Algorithm 3.2) and applies two operations, interpolation and projection, which iterate over a set of 2^n control points.

The clustering effect

One undesired feature of the projection method is that it leads to the so-called *clustering effect*. This is a well-know 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 may be returned as a cluster of boxes instead of a single box containing it [112]. Increasing the resolution, *i.e.*, reducing the value of the parameter σ bounding the size of the returned boxes, does not eliminate the clustering effect because it is reproduced just with smaller boxes. Clustering is exacerbated when the solution varieties of at least two equations in the considered system of equations are close to be tangent. In practice, the size of the cluster gives an idea of the loss of rigidity of the mechanism in the corresponding configuration. Indeed, if the effect of manufacturing inaccuracies and/or slight elasticities in the kinematic chain were incorporated as ranges in the coefficients of the closure conditions, clustering-free branch-and-prune methods would also reproduce these 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 of mitigating the clustering effect and in Chapter 5 we will see how redundancy can also be used to reduce this effect.

3.3 Certification of solutions

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.

3.3.1 Certification of isolated solutions

The theorems by Kantorovich [74, 119], Moore [109] or Smale [163] 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 [108] relies only on the values of the function on the different faces of the analyzed box. It can be stated as follows:

Let $\mathbf{F} : \mathcal{S} \subset \mathbb{R}^n \rightarrow \mathbb{R}^n$ 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 $\mathcal{U}_i = \{\mathbf{x} \in \mathcal{B} | x_i = x_i^u\}$ with $i \in \{1, \ldots, n\}$ be the n pairs of opposite faces of $\mathcal{B}$. If $\mathbf{F}(\mathbf{x}) \cdot \mathbf{F}(\mathbf{y}) \leq 0$ for all $\mathbf{x} \in \mathcal{L}_i$, $\mathbf{y} \in \mathcal{U}_i$, $i \in \{1, \ldots, n\}$, then $\mathbf{F}$ has at least one root in $\mathcal{B}$.

Miranda's theorem can be proved relying on Brouwer's fix point theorem [16]. Observe that if Miranda's theorem holds, the box reduction operation based on the trapezoid-line intersection described in Section 3.2.2 will result in a new box contracted in all dimensions. In this case, if $\mathbf{F}$ is continuous and the box reduction procedure is adequately interpreted as a mapping, Brouwer's fix point theorem ensures that the considered box includes a fixed point which is a solution of the considered system.

It has been proven that there is a hierarchy in terms of generality between the different existence theorems [2]. In this hierarchy, Moore's theorem is more general than Kantorovich's theorem which in turn is more general than Smale's theorem [3]. 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 if the norm ball is a box, as it is our case, they are equivalent. Thus, it makes sense to consider Miranda's test as the best option.

Miranda's test can be readily incorporated in the box reduction strategy based on the projection 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 lower and upper 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. This check can be introduced in the box projection-based box reduction procedure with a single line of code using results already computed for the trapezoid-line clipping (see line 9 in Algorithm 3.2).

3.3.2 Certification of higher-dimensional solution sets

None of the existence tests available in the literature apply to problems with higher-dimensional solution sets. 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 will be used in this work, and that is valid 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 inside the box. In this regard, note that the control points can also be used to efficiently evaluate the derivatives of a multi-affine function. For instance, using Eq. (3.15) the derivative of f with respect to x_1 is

$$\frac{\partial f}{\partial x_1} = \frac{f_u(x_2, \ldots, x_n) - f_l(x_2, \ldots, x_n)}{x_1^u - x_1^l}, \tag{3.27}$$

where, as mentioned, f_l and f_u are computed by linear interpolation of the control points. This Newton-based general verification test can be easily included in the box reduction procedure (see line 13 in Algorithm 3.2).

4

Evaluation of the trapezoid solver

In this chapter we evaluate the solver presented in the previous chapter, which we call the trapezoid method, in different problems, namely, spherical mechanisms, 6R loops, overconstrained systems, mobile kinematic loops, and parallel platforms. The results obtained in such a diversity of problems show the generality and the efficiency of the proposed solver.

The solver described in Chapter 3 has been implemented in C and evaluated on different test-cases, including simple and multi-loop spherical mechanisms (Section 4.1), general 6R loops (Section 4.2), overconstrained 6R loops (Section 4.3), kinematotropic mechanisms (Section 4.4), and spatial parallel mechanisms, which define multi-loop systems (Section 4.5).

In all test-cases, we will examine the performance of the method described in Section 3.2.2 based on the trapezoid-line clipping, which we will refer to as the *trapezoid* method. This method is compared with the two alternative methods described, respectively, in Sections 3.2.1 and in [131]. The first one is based on linear programming directly using the convex hull defined in Eq. (3.19) (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). This method reduces the complexity of the problems by formulating them with simple polynomial equations which are directly bounded by hyperplanes defined on the space of their input variables. Thus, 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 fix point theorem. In both the LP and LR methods, the linear programs are solved relying in the CoinOR linear programming library [32]. In contrast, and due to the simplicity of the used 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 sub-problems defined in Section 2.2.1 in a separate thread. 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 positive-dimensional solution sets. The actual implementation and the input files for the test-cases are available for download at [88].

4.1 Position analysis of spherical mechanisms

Three different spherical mechanisms of increasing complexity have been used to evaluate the trapezoid solver (see Fig. 4.1). The first one is a simple spherical wrist whose inverse kinematics involves a single kinematic loop. The second is a 3-$\underline{\text{R}}$RR orienting robot, also known as the agile eye [86], whose forward kinematics includes two independent kinematic loops, which is commonly used to evaluate approaches dealing with spherical robots [13, 25], The third one is a spherical mechanism with four independent kinematic loops [14].

The inverse kinematics of the spherical wrist in Fig. 4.1(top) consists in obtaining the values of $t_i = \tan\left(\frac{\theta_i}{2}\right)$, $i = 1, 2, 3$, which satisfy the following equation:

$$\frac{1}{\sqrt{(t_1^2+1)(t_2^2+1)(t_3^2+1)}}(1 + t_1\,\mathrm{k})(1 + t_2\,\mathrm{i})(1 + t_3\,\mathrm{k})q_o^{-1} = 1, \tag{4.1}$$

where q_0 is the desired orientation for the end-effector. If we set $q_0^{-1} = e_0 + e_1\,\mathrm{i} + e_2\,\mathrm{j} + e_3\,\mathrm{k}$, we derive a set of three multi-affine equations which can be expressed in matrix form as:

$$\begin{pmatrix} e_1 & -e_2 & e_0 & -e_2 & e_3 & -e_1 & -e_3 & e_0 \\ e_2 & e_1 & -e_3 & e_1 & e_0 & -e_2 & -e_0 & -e_3 \\ e_3 & e_0 & e_2 & e_0 & -e_1 & -e_3 & e_1 & e_2 \end{pmatrix} \begin{pmatrix} 1 \\ t_1 \\ t_2 \\ t_3 \\ t_1t_2 \\ t_1t_3 \\ t_2t_3 \\ t_1t_2t_3 \end{pmatrix} = 0 \tag{4.2}$$

The problem of solving the forward kinematics of the 3-$\underline{\text{R}}$RR spherical parallel manipulator

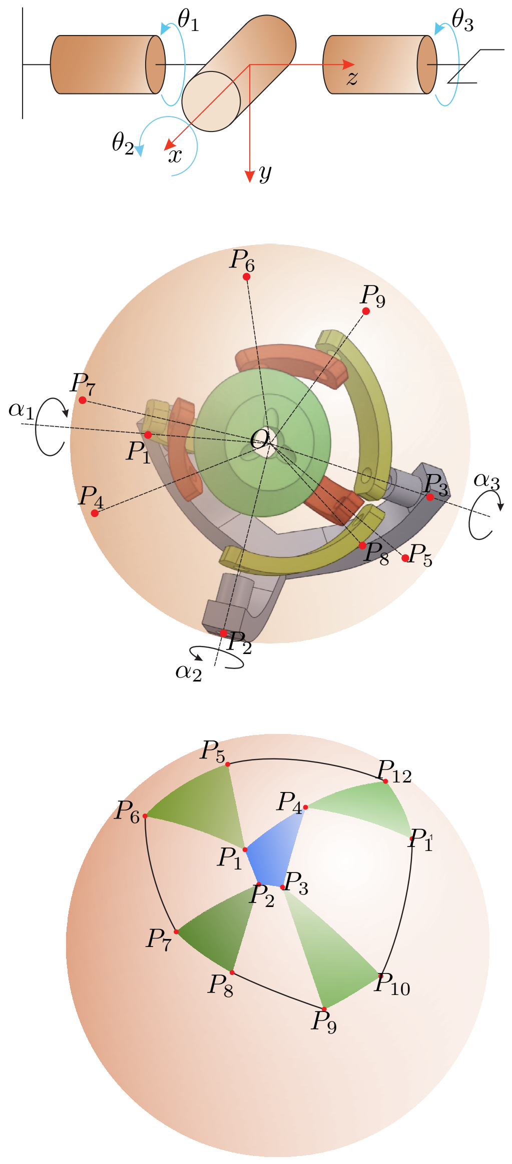


Figure 4.1: The three test cases used to validate the branch-and-prune algorithm presented in this thesis on spherical mechanisms. Top: A spherical wrist. Center: 3-$\underline{\text{R}}$RR parallel spherical manipulator. Bottom: A four loop spherical mechanism.

shown in Fig. 4.1(center) consist in locating the triangle $P_7P_8P_9$ with respect to the base $P_1P_2P_3$ as a function of the motor angles θ_1, θ_2, and θ_3. This mechanism has two independent kinematic loops, which, when formulated in terms of quaternions as described in Section 2.2, give six multi-affine equations in six unknowns.

Finally, the four-loop mechanism in Fig. 4.1(bottom) can be formulated, once represented using quaternions, in terms of 12 multi-affine equations in 12 unknowns.

	LP	**LR**	**Trapezoid**
Processed boxes	20	39	28
Box reductions	34	40	44
Bisected boxes	6	19	10
Empty boxes	12	18	16
Solution boxes	2	2	2
Verified	-	0	2
Execution time [s]	0.004	0.06	0.00009

Table 4.1: Summary of the results for the wrist example.

	LP	**LR**	**Trapezoid**
Processed boxes	584	17	1342
Box reductions	1050	26	2309
Bisected boxes	260	8	639
Empty boxes	320	5	680
Solution boxes	4	4	23
Verified	-	2	4
Execution time [s]	0.38	0.05	0.001

Table 4.2: Summary of the results for the 3-$\underline{\text{R}}$RR parallel spherical manipulator.

	LP	**LR**	**Trapezoid**
Processed boxes	88228	9001	372180
Box reductions	161568	14517	571147
Bisected boxes	42066	4500	184042
Empty boxes	46142	4481	187077
Solution boxes	20	20	1061
Verified	-	7	20
Execution time [s]	190	115	0.32

Table 4.3: Summary of the results for the four loop spherical mechanism.

Tables 4.1, 4.2, and 4.3 summarize the performance of the algorithm described in Section 3.2 for the three considered testbeds. For each case, the table gives the number of processed boxes, the number of box reduction operations applied (i.e., the number of times lines 3 to 11 in

Algorithm 3.2 are executed), the number of bisected boxes, the number of empty boxes, the number of solution boxes, the number such boxes that can be certified to include a solution, and the execution time in seconds. In the case of the trapezoid method, the certification of solutions is performed combining the Miranda and Newton procedures described respectively in Section 3.3.1 and 3.3.2. The time taken by these certification procedures is included in the given execution time. As it can be seen in the tables of results, the trapezoid method is significantly more efficient that the two other methods in all cases. Although it might return the solutions as a cluster of boxes, the boxes actually including a solution have been readily identified in these test cases.

4.2 The inverse kinematics of general 6R loops

The general inverse kinematics of 6R robots, or equivalently the position analysis of 6R closed loops, is a decades-old problem which still stands as a challenging test-case in Robotics [63, 72, 137]. The problem consist in determining the possible combinations of the joint angles that yield a specific pose for the end effector. This problem is particularly relevant since, in principle, a 6R chain can attain arbitrary positions and orientations inside its workspace by adequately fixing the position of their joints. Thus, a 6R robot has the maximum possible dexterity without redundant actuators. The inverse kinematic problem can be posed as a system of non-linear equations in six variables (one per degree of freedom in the loop).

The problem can be solved in closed-form for decoupled manipulators, i.e., those where the last three rotations intersect at a common point. This includes most of the industrial manipulators, which can have up to 8 solutions. The first solution for this kind of manipulators was proposed in [128]. However, a method able to solve the problem for the general case is desirable. First, such a method will increase the design options for a particular task. For instance, if the robot is to move a high payload, a decoupled design may not reach the task requirements. Moreover, anthropomorphic designs, common in the emerging field of human-robot interaction, often use offset wrists. These cases cannot be solved in closed form.

Numerical algorithms based on variants of the Newton method can be applied to the inverse kinematics of general 6R loops [176]. However, such methods would only identify one of the solutions. The identified solution as well as the computational time of these methods heavily depend their initialization. Continuation methods have also been successfully applied to this problem [180], but their high computational times prevent them to be used in settings with real time requirements.

The most efficient methods up to date are based on elimination theory. The first general methods following this approach can be traced back to [1] and [44] which respectively derived resultant polynomials of degree 48 and 32. At that time, the maximum number of solutions known to be 32 [147] and, thus, the resultant of degree 48 included extraneous factors. However, Primrose in [136] proved for the first time that 16 solutions the 32-degree resultant polynomial always lay at infinity and, therefore, the inverse kinematic of general 6R has at most 16 real solutions. Later, an instance with 16 real distinct solutions was introduced in [102], proving that the bound is tight.

Finally, in 1988 Lee and Liang reformulated the characteristic polynomial of the manipulator as a determinant of 8×8 matrix in square of a joint variable which led to polynomial degree 16 [94]. An alternative solution was proposed by Raghavan and Roth [138] using dialyctic elimination to also obtain a polynomial of degree 16. Both methods use 14 equations in 5 joint's variables, but the latter was more suitable for further improvements. Therefor, the numerical stability, the accuracy, and the computational efficiency of this approach have been improved in subsequent contributions. For instance, the problem has been formulated as the eigenvalue of a 16×16 matrix in [56] and [81]. Afterward, Manocha and Canny proposed an algorithm in term of the general eigenvalue problem of a 24×24 matrix. Despite calculating a surplus of eight imaginary solutions, the approach was able to solve the problem very efficiently thanks to the use of highly-efficient linear algebra software libraries [4].

During the following years, the community explored alternative ways to formulate the problem. For instance, Morton and Elgersma used quaternions to derive a general eigenvalue problem [113] and Gervasi and et.al [55] combined dual quaternions and multivariate elimination [55]. Dual quaternions where also used in [72] and in [51]. where a polynomial of degree 24 was derived using Dixon multi-variate elimination. Later in [202] the eight extraneous roots of this polynomial are factored out.

Most of the approaches described in the literature, though, work well in general, but fail on particular classes of problems. Surprisingly, only limited attempts have been made to compare them on broad classes of problems [6]. However, among all the works, arguably the most influential one is that by Manocha and Canny [101] which provided the first efficient algorithm for the problem. Since we will use it as a reference in the following, we next summarize it.

4.2.1 The Manocha and Canny method

Using the DH convention, the 4×4 homogeneous transform relating link i with link $i+1$ is

$$\mathbf{A}_i = \begin{pmatrix} c_i & -s_i\,\lambda_i & s_i\,\mu_i & a_i\,c_i \\ s_i & c_i\,\lambda_i & -c_i\,\mu_i & a_i\,s_i \\ 0 & \mu_i & \lambda_i & d_i \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{4.3}$$

where

$$s_i = \sin\theta_i, \tag{4.4}$$

$$c_i = \cos\theta_i, \tag{4.5}$$

$$\mu_i = \sin\alpha_i, \tag{4.6}$$

$$\lambda_i = \cos\alpha_i, \tag{4.7}$$

(4.8)

and where $(\theta_i, d_i, \alpha_i, a_i)$ are the DH parameters of the corresponding joint. In 6R chains, only θ_i is variable for $i \in \{1, \ldots, n\}$.

In this formalism, the pose of the end-effector is given by an arbitrary 4×4 homogeneous transform $\mathbf{A}_h$. Thus, the 6R loop defines a matrix equation

$$\mathbf{A}_1\,\mathbf{A}_2\,\mathbf{A}_3\,\mathbf{A}_4\,\mathbf{A}_5\,\mathbf{A}_6 = \mathbf{A}_h, \tag{4.9}$$

which can be re-ordered as

$$\mathbf{A}_3\,\mathbf{A}_4\,\mathbf{A}_5 = \mathbf{A}_1^{-1}\,\mathbf{A}_2^{-1}\,\mathbf{A}_h\mathbf{A}_6^{-1}, \tag{4.10}$$

to reduce the degree of the terms. This equation defines 12 scalar equations, but only 6 of them are independent. The equations corresponding to the last two columns do not involve neither c_6 nor s_6 and, thus, they only depend on θ_1, θ_2, θ_3, θ_4, and θ_5. These equations can be written as

$$\mathbf{Q}\begin{pmatrix} s_1\,s_2 \\ s_1\,c_2 \\ c_1\,s_2 \\ c_1\,c_2 \\ s_1 \\ c_1 \\ s_2 \\ c_1 \end{pmatrix} = \mathbf{P}\begin{pmatrix} s_4\,s_5 \\ s_4\,c_5 \\ c_4\,s_5 \\ c_4\,c_5 \\ s_4 \\ c_4 \\ s_5 \\ c_5 \\ 1 \end{pmatrix}, \tag{4.11}$$

where $\mathbf{Q}$ is a constant matrix and $\mathbf{P}$ depends linearly on c_3 and s_3. Matrix $\mathbf{Q}$ has a particular structure which allows to split Eq. (4.11) as

$$\mathbf{Q}_1 \begin{pmatrix} s_1 \\ c_1 \end{pmatrix} = \mathbf{P} \begin{pmatrix} s_4\, s_5 \\ s_4\, c_5 \\ c_4\, s_5 \\ c_4\, c_5 \\ s_4 \\ c_4 \\ s_5 \\ c_5 \\ 1 \end{pmatrix}, \tag{4.12}$$

and

$$\mathbf{Q}_2 \begin{pmatrix} s_1\, s_2 \\ s_1\, c_2 \\ c_1\, s_2 \\ c_1\, c_2 \\ s_1 \\ s_2 \end{pmatrix} = \mathbf{P} \begin{pmatrix} s_4\, s_5 \\ s_4\, c_5 \\ c_4\, s_5 \\ c_4\, c_5 \\ s_4 \\ c_4 \\ s_5 \\ c_5 \\ 1 \end{pmatrix}, \tag{4.13}$$

Two rows of $\mathbf{Q}_1$ forming a full rank minor are used to remove the left-hand side monomials of Eq. (4.12) and six independent rows of $\mathbf{Q}_2$ are used to remove the left-hand side monomials of Eq. (4.13). In this way we end up with six equations in three unknowns, θ_3, θ_4, and θ_5. After applying the the tangent-half angle substitution $x_i = \tan(\theta_i/2)$ for $i \in \{3, 4, 5\}$ and multiplying by x_4 the system is transformed to

$$\boldsymbol{\Sigma} \begin{pmatrix} x_4^3\, x_5^2 \\ x_4^3\, x_5 \\ x_4^3 \\ x_4^2\, x_5^2 \\ x_4^2\, x_5 \\ x_4^2 \\ x_4\, x_5^2 \\ x_4\, x_5 \\ x_4 \\ x_5^2 \\ x_5 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}. \tag{4.14}$$

Matrix $\boldsymbol{\Sigma}$ is 12×12 and includes quadratic terms in x_3. Thus, it can be decomposed as

$$\boldsymbol{\Sigma} = \mathbf{A}\, x_3^2 + \mathbf{B}\, x_3 + \mathbf{C}, \tag{4.15}$$

where $\mathbf{A}$, $\mathbf{B}$, and $\mathbf{C}$ are constant. It can be seen that the eigenvalue of the 24×24 matrix

$$\mathbf{M} = \begin{pmatrix} \mathbf{0} & \mathbf{I} \\ -\mathbf{A}^{-1}\,\mathbf{C} & -\mathbf{A}^{-1}\,\mathbf{B} \end{pmatrix} \tag{4.16}$$

correspond to the roots of $|\boldsymbol{\Sigma}| = 0$. Moreover, the corresponding eigenvectors have the form

$$\mathbf{v} = \begin{pmatrix} \mathbf{w} \\ x_3\, \mathbf{w} \end{pmatrix} \tag{4.17}$$

where $\mathbf{w}$ are solutions to Eq. (4.14) and, thus, they can be used to determine the values for x_4 and x_5 in the solution points.

The procedure just described woks well in general, but special care must be taken to avoid numerical issues in particular cases. Manocha and Canny used the SVD to adjust matrices $\mathbf{Q}_1$ and $\mathbf{Q}_2$ in Eqs. (4.12) and (4.13) so that small perturbations do not decrease their rank. Moreover, matrix $\mathbf{A}$ in Eq. (4.15) may be bad conditioned if, for instance, one of the solutions of the inverse kinematics has θ_3 close to π. In this case, a random shift on θ_3 similar to the one described in Section 2.2.1, can be applied. Otherwise, a generalized eigenproblem can be used to determine the valid value for x_3, although this is computationally more expensive than solving a standard eigenproblem. Finally, the different numerical processes may introduce inaccuracies in the solutions and, therefore, a Newton procedure is used to refine them.

4.2.2 Comparative analysis

The performance, robustness, and generality of the proposed method has been evaluated on three 6R loops with zero-dimensional solution sets.

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

6R loops with zero-dimensional solution sets

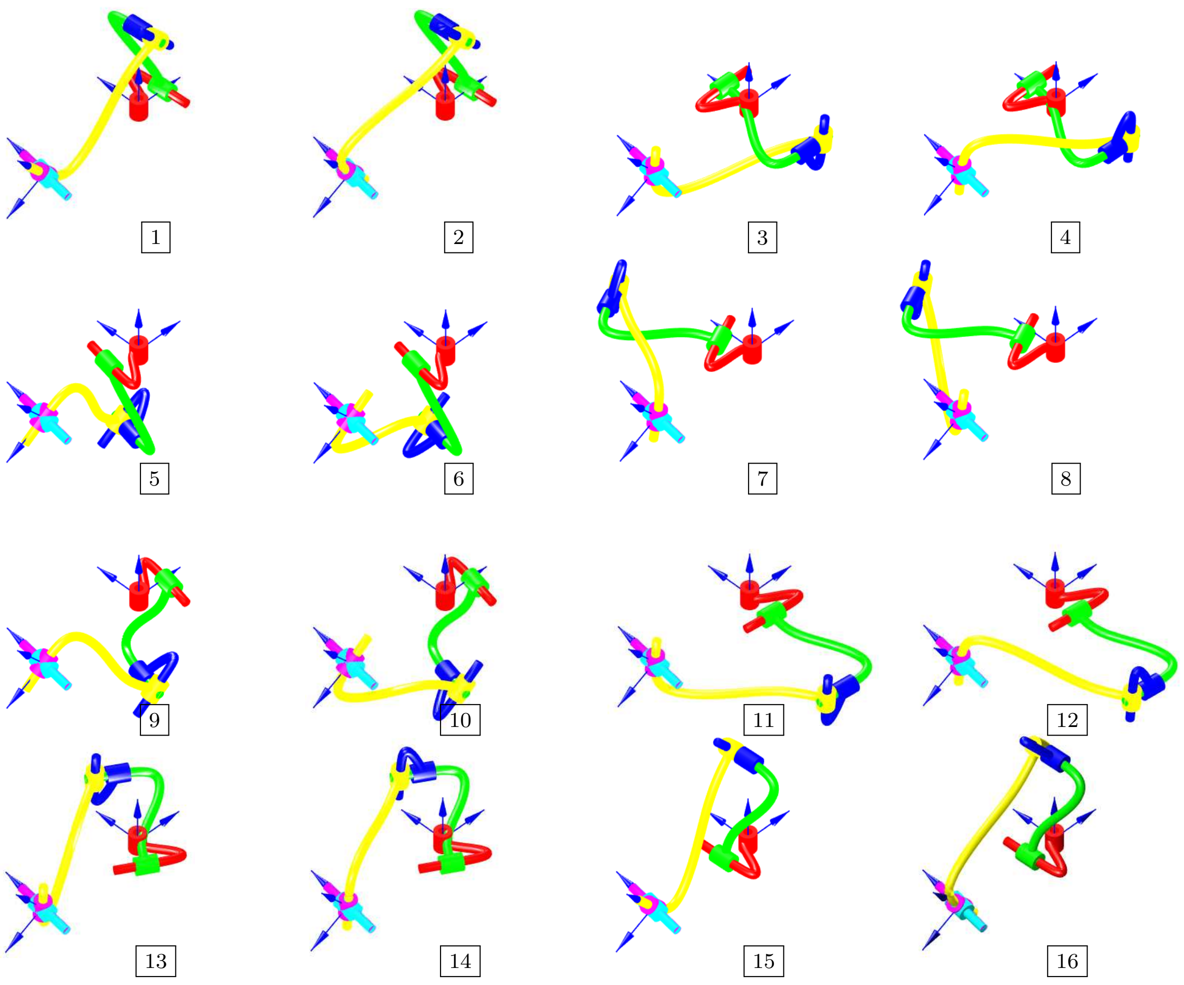
We will use as a first test-case the example appearing in [188] and also in [101][1], which has 16 real solutions. The DH parameters for this test-case appear in Table 4.4. Table 4.5 shows the performance statistics of the compared methods.

The sets of angles for the 16 solutions and the corresponding robot configurations appear in Fig. 4.2. The software described in [175] has been used for the robot representation.

	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	16
Execution time [s]	15	22	0.035

Table 4.5: Performance of the three compared methods for a 6R loop with 16 solutions.

[1]The version of this example given in [101] has a wrong sign in one of the entries of the $\mathbf{A}_{\text{hand}}$ matrix.



solution	θ_1	θ_2	θ_3	θ_4	θ_5	θ_6
1	0.043934	1.886280	1.955521	-0.183661	0.000089	-0.001911
2	0.043934	1.886280	-1.186071	-2.957932	3.141504	3.139684
3	1.547731	-3.084428	0.057089	-2.037814	0.398002	-0.690520
4	1.547731	-3.084428	-3.084504	-1.103779	2.743591	2.451086
5	2.937771	-1.813260	2.558719	-0.300911	-2.999858	1.713237
6	2.937771	-1.813260	-0.582874	-2.840682	-0.141734	-1.428522
7	1.986946	0.092614	0.039370	-2.165532	-2.042299	2.384515
8	1.986946	0.092614	-3.102223	-0.976061	-1.099293	-0.757069
9	-0.225897	-1.834277	1.134894	3.088823	3.012142	1.755402
10	-0.225897	-1.834277	-2.006699	0.052769	0.129451	-1.388927
11	-1.680482	-0.109494	3.141049	0.671707	0.917184	-0.687745
12	-1.680482	-0.109494	0.000544	2.469886	2.224409	2.453854
13	-2.108155	3.007802	0.016182	2.594746	-0.580930	-0.648888
14	-2.108155	3.007802	-3.125411	0.546846	-2.560663	2.492696
15	-3.108889	1.888300	0.563151	-3.042226	-0.267479	-0.007681
16	-3.108889	1.888300	-2.578442	-0.099366	-2.874113	3.134286

Figure 4.2: The 16 solutions to the inverse kinematics of the 6R robot whose DH parameters are given in Table 4.4.

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 includes 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 is smaller, as it has 223 rows and 59 columns (*i.e.*, 13157 entries). Despite defining a larger tableau, the LP method is more efficient due to the particularities of the test-case, since this is not the case in general. The method proposed in this paper (the trapezoid method) only takes 35 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 can be considered solutions if mechanical tolerances are taken into account.

Using the trapezoid method, six of the returned boxes are verified using Miranda's theorem. Since Miranda's 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.

Note that the algorithm described in [101], whose implementation is available at [100], can solve this problem in less than 2 ms. Unfortunately, due to an unreported bug, it fails to provide the correct value of θ_6 for the first solution in Fig. 4.2. This method is based on an eigenproblem which can be efficiently solved relying on highly optimized linear algebra libraries [4]). 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 4.6. In contrast, as shown in Table 4.7 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 in the form of 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 only 8 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 in 98 ms. An alternative strategy to limit the cluster 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.

Numerical issues arise not only for problems with particular arrangements of the rotation axes (e.g., parallel or orthogonal axes), but also in 6R loops with, apparently, generic parameters such as those in Table 4.8, which correspond to Problem 5 in [188]. This is a slightly perturbed version of Problem 4 in the same reference. Problem 4 has two real solutions and

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 4.6: DH parameters and closure condition of the *bad_eg0* example from [100].

	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	8
Execution time [s]	20	19	0.098

Table 4.7: Performance of the three compared methods for the *bad_eg0* example from [100].

two solutions with a tiny imaginary part (of the order of 10^{-2}). The perturbation has the effect of converting the imaginary solutions into ill-conditioned, real solutions. As a consequence, Manocha's method returns four points, but none of them is an actual solution of the problem. Table 4.9 shows the results obtained with the three interval-based method compared in this work. The three of them correctly identify the solutions of the problem, but the trapezoid method does so more efficiently than two other alternatives.

4.3 Computation of the input-output relations of overconstrained mechanisms

A mechanism is overconstrained if it moves when it is expected to be rigid. The degree of mobility of a system is studied in Rigidity Theory, a discipline that can be traced back to Maxwell [104]. In this context, the Laman and Tay Theorems [91, 170] provide necessary and sufficient conditions for rigidity in 2D and 3D, respectively. However, these results only hold for generic or for particular regular structures [79].

θ_i	d_i	α_i	a_i
θ_1	0.9685	0.0175	123.4
θ_2	0.563	0.6109	143.6
θ_3	0.19	0.3491	162.5
θ_4	0.163	1.1519	0.343
θ_5	0.1	2.7925	0.7893
θ_6	0.876	0.6458	0.123

$$\mathbf{A}_{\text{hand}} = \begin{pmatrix} 0.7447 & -0.6667 & -0.02942 & -344.9961 \\ 0.3135 & 0.3884 & -0.86650 & 253.3829 \\ -0.0294 & -0.8665 & 0.49831 & 2.7113 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Table 4.8: DH parameters and closure condition of Problem 5 from [188].

	LP	**LR**	**Trapezoid**
Processed boxes	16510	69	47424
Box reductions	25971	135	62887
Bisected boxes	8223	34	23680
Empty boxes	8283	31	23739
Solution boxes	4	4	5
Verified solutions	-	4	4
Execution time [s]	28.5	1	0.078

Table 4.9: Performance of the three compared methods for Problem 5 from [188].

In Robotics, the degree of mobility of a mechanism is typically estimated following the Grübler–Kutzbach criterion [5, Cap. 6], which only provides a necessary condition for rigidity. According to this criterion, the degree of mobility, M, of a given linkage is given by

$$M = 6\,(N - 1 - J) + \sum_{i=1}^{J} r_i, \tag{4.18}$$

with N the number of rigid bodies in the system, J the number of joint between the bodies, and r_i the degrees of freedom allowed by the i-th joint. For instance, for single degree of freedom joints like revolute or prismatic joints r_i is 1. An intuitive derivation of this formula is given, for instance, in [185].

Despite overconstrained mechanisms can include several kinematic loops (like in deployable structures, for instance), research focus on single loop overconstrained mechanisms including only revolute or, less often, prismatic joints. In these cases $N = J$ and $r_i = 1, \forall i \in \{1, \ldots, J\}$ and, thus, $M = J - 6$. Consequently, any loop with less than 7 links (or joints) is either rigid or overconstrained. In the second case it typically exhibits a one-dimensional solution set. Positive-dimensional solution sets are usually called *self-motion manifolds* [19] because any trajectory embedded in them represents a valid motion of the mechanism which does not require to disassemble it.

From the practical point of view, overconstrainess offers the possibility to transmit motion and forces with less links and joints than the expected ones [29], which is economically advantageous. From a theoretical point of view, several relevant issues are still open in the study of overconstrained mechanisms [103]:

- **The design of overconstrained mechanisms:** The industrial use of overconstrained mechanism is probably limited because the design of their motion trajectories is hard. Although some progresses have been achieved using Bennett mechanisms [123], this issue is mainly open in general.

- **The proof of overconstrainess:** Since rigidity criteria only offers necessary conditions, *ad hoc* geometric reasonings or even the construction of real models if often used to proof the overconstrainess of a given mechanism. Currently, there is a lack of formal tools to provide such proofs for any mechanism from the parameters defining it (e.g., the Denavit-Hartenberg parameters). The difficulty arises due to the complex non-linear relations between such parameters that typically characterize overconstrained mechanisms.

- **Existence of new overconstrained mechanisms:** While the possible overconstrained mechanisms with loops of four and five bodies are well-know, this is not the case for six body loops since new overconstrained mechanisms of this kind are described every now and then. There is a lack of analytic methods to identify new overconstrained mechanisms or even to elucidate if any two overconstrained mechanisms are subsumed by a more general case.

- **The calculation of the input-output relations:** Since typically overconstrained mechanisms with mobility one are studied, the input-output relations describe the evolution of all joint variables (i.e., the output variables) in terms of one of them (i.e., the input variable). Ideally, input-output relations should be derived analytically, but the derivation process is complex and existing methods are valid only for particular geometries. The input-output relations are also difficult to identify using numerical methods since most of the existing methods cannot isolate positive-dimensional solution sets.

Next, we show that the method proposed in this thesis is particularly adequate to isolate the input-output relation of overconstrained mechanisms with different number of joints.

4.3.1 4R overconstrained mechanisms

Bennett proposed the first movable 4R closed loop, i.e., the so-called Bennett linkage [11]. Despite other 4R movable loops have been described [38], Brunnthaler [17] proved that all of them are equivalent to the Bennett linkage.

The family of Bennett linkages fulfill the following conditions between their DH parameters

$$a_1 = a_3 \tag{4.19}$$

$$a_2 = a_4 \tag{4.20}$$

$$\alpha_1 = \alpha_3 \tag{4.21}$$

$$\alpha_2 = \alpha_4 \tag{4.22}$$

$$d_1 = d_2 = d_3 = d_4 = 0 \tag{4.23}$$

$$\frac{a_1}{\sin \alpha_1} = \frac{a_2}{\sin \alpha_2} \tag{4.24}$$

θ_i	d_i	α_i	a_i
θ_1	0	1.2829	20
θ_2	0	0.5	10
θ_3	0	1.2829	20
θ_4	0	0.5	10

Table 4.10: DH parameters of the Bennett linkage used in the experiments.

Table 4.10 gives the DH parameters defining the Bennett linkage used in our tests and Fig. 4.3 shows the input-output relations of this linkage. Finally, Table 4.11 gives the performance statistics of the solver on this problem. In the case of the trapezoid method, the solution boxes are verified with the Newton-based procedure described in Section 3.3.2 and the given execution time includes the time used in this verification procedure.

	LP	**LR**	**Trapezoid**
Processed boxes	2620	2647	5168
Box reductions	3940	2653	7640
Bisected boxes	1302	1323	2576
Empty boxes	42	3	528
Solution boxes	1276	1321	2064
Verified solutions	-	0	596
Execution time [s]	4.03	0.7	0.015

Table 4.11: Summary of the results for the Bennett 4R overconstrained mechanism using the trapezoid box reduction procedure.

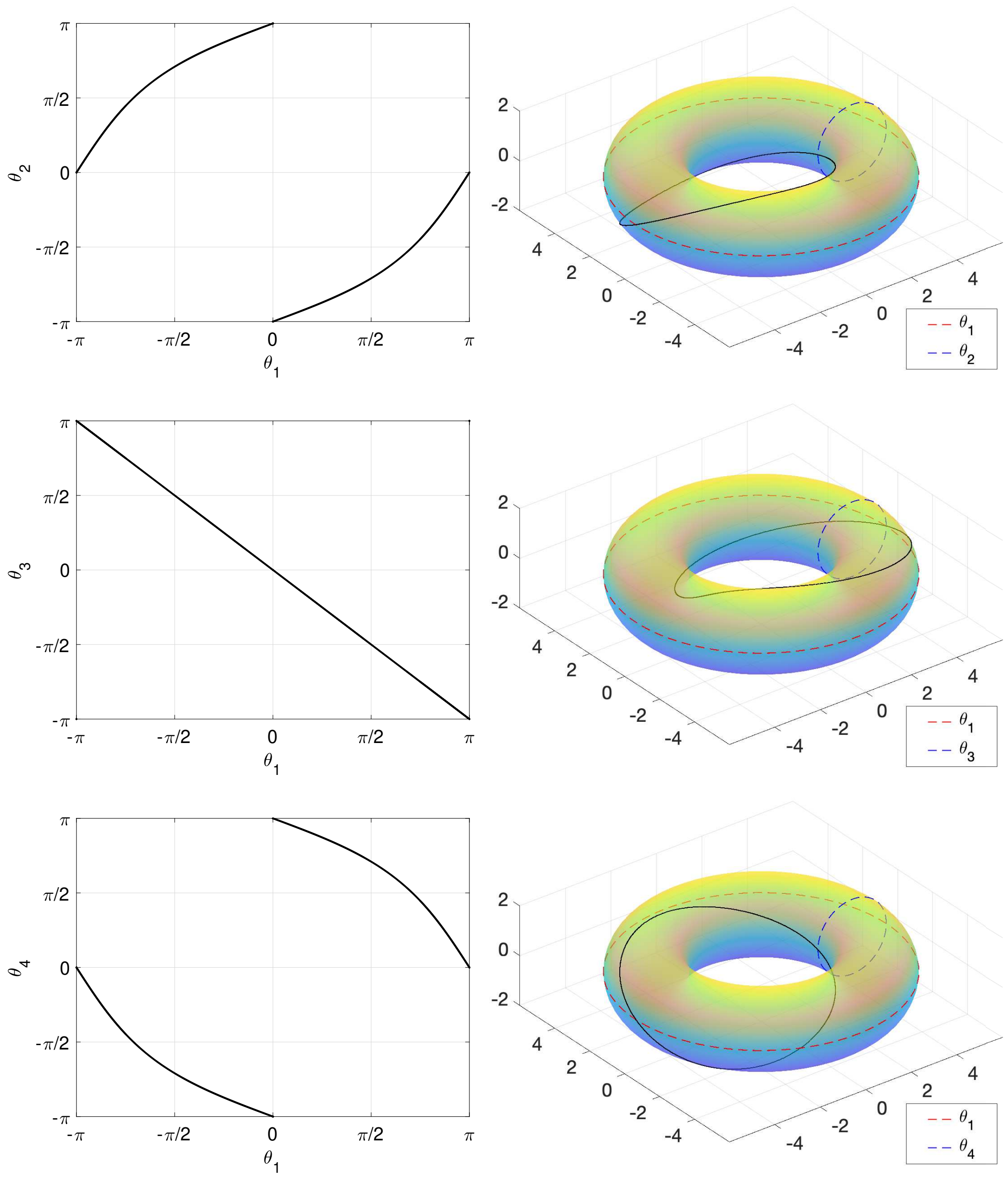


Figure 4.3: The input-output relations of the Bennett linkage, with θ_1 the input variable.

4.3.2 5R overconstrained mechanisms

Two families of 5R overconstrained mechanisms exists [76]. Both of them essentially combine Bennett linkages and the difference between them are in the offset in the links. The family with zero offsets was proposed by Goldberg [58] and it subsumes the Myard linkage [114]. In its general form, the Goldberg linkage is defined by the constraints for the two combined Bennett linkages [194]

$$a_1 = a_3 \tag{4.25}$$

$$\alpha_1 = \alpha_3 \tag{4.26}$$

$$d_3 = d_4 = d_5 = 0 \tag{4.27}$$

$$\frac{a_1}{\sin\alpha_1} = \frac{a_4}{\sin\alpha_4} = \frac{a_5}{\sin\alpha_5} \tag{4.28}$$

and the constraints on the connector link

$$\alpha_2 = \pm\arccos\left(\cos\alpha_4\cos\alpha_5 - \cos\epsilon\sin\alpha_4\sin\alpha_5\right), \tag{4.29}$$

$$a_2 = \frac{a_1}{\sin\alpha_1}\left(\cos\alpha_4 + \cos\alpha_5\right)\,\tan\frac{\alpha_2}{2}, \tag{4.30}$$

$$d_1 = d_2 = \frac{a_1}{\sin\alpha_1}\,\sin\epsilon\,\frac{\sin\alpha_4\sin\alpha_5}{1+\cos\alpha_2}, \tag{4.31}$$

where ϵ is the so-called kink angle. Fixing the different parameters overconstrained mechanisms arise. For instance, $\epsilon = \pi$, $\alpha_1 = \pi/2$, and $\alpha_1 = \pi - \alpha_4$ the Myard mechanism is obtained. For $\epsilon = 0$ the conditions for the original 5R Goldberg mechanism are obtained:

$$a_1 = a_3 \tag{4.32}$$

$$a_2 = a_4 + a_5, \tag{4.33}$$

$$\alpha_1 = \alpha_3 \tag{4.34}$$

$$\alpha_2 = \alpha_4 + \alpha_5, \tag{4.35}$$

$$d_1 = d_2 = d_3 = d_4 = d_5 = 0 \tag{4.36}$$

$$\frac{a_1}{\sin\alpha_1} = \frac{a_4}{\sin\alpha_4} = \frac{a_5}{\sin\alpha_5} \tag{4.37}$$

Table 4.12 gives the DH parameters defining the Goldberg linkage used in our tests, which is defined with $\epsilon = 0$, and Fig. 4.4 shows the input-output relations of this linkage. Finally, Table 4.13 gives the performance statistics of the solver on this problem.

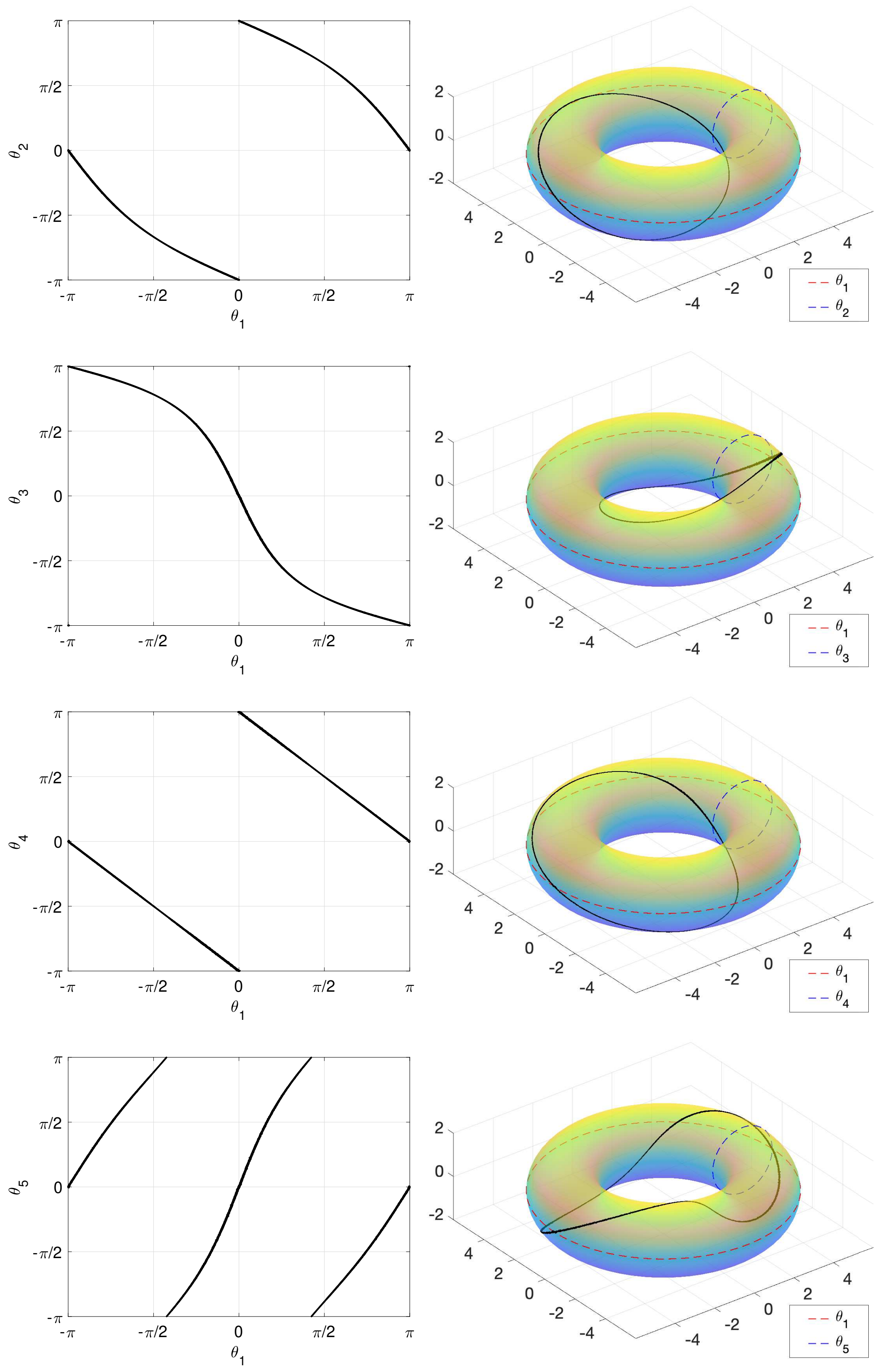


Figure 4.4: The input-output relations of the Goldberg linkage, with θ_1 the input variable.

θ_i	d_i	α_i	a_i
θ_1	0	0.8727	0.3711
θ_2	0	2.5307	0.6243
θ_3	0	0.8727	0.3711
θ_4	0	0.4363	0.2047
θ_5	0	2.0944	0.4195

Table 4.12: DH parameters of the Goldberg 5R linkage used in the experiments using the trapezoid box reduction procedure.

	LP	**LR**	**Trapezoid**
Processed boxes	5032	4305	41480
Box reductions	7676	5661	60184
Bisected boxes	2500	2152	20724
Empty boxes	662	225	9898
Solution boxes	1870	1928	10858
Verified solutions	-	0	1778
Execution time [s]	15.05	4	0.14

Table 4.13: Summary of the results for the Goldberg 5R overconstrained mechanism using the trapezoid box reduction procedure.

4.3.3 6R overconstrained mechanisms

While all the possible 4R and 5R overconstrained mechanism are rather well determined, currently there is no a characterization of all possible overconstrained 6R mechanisms. Therefore, new designs are derived every now and then, using either geometric reasonings or algebraic approaches. A list of the described 6R overconstrained linkages up to 2002 can be found in [9]. Three of them have been widely used in industrial applications so far (see Fig. 4.5, for few examples). The first one is the double-Hooke's-joint linkage (also known as double Cardan linkage), which was proposed as early as 1683. It connects two universal joints by an intermediate link to obtain a uniform rotational movement, improving the motion obtained with a single universal joint [9]. This mechanism is widely-used for transmission coupling. Later, in 1853 Sarrus proposed a mechanism that obtains a linear motion using only revolute joints [152]. Due to its simplicity, it is used in many engineering applications [99, 201]. Finally, Schatz in 1942 [155] patented a mechanism that is the base of the Turbula mixing machine. This mechanism was later shown to be a variation of the Bricard's trihedral linkage [127].

Next, we will apply the solver proposed in this thesis to six 6R overconstrained mechanisms. First we will analyze the Bricard's trihedral linkage [15], a case that was later studied and generalized by Wohlhart in [193]. This mechanisms has a particular simple set of parameters, which allowed us to explicitly give the set of multi-lineal equations formalizing its position

Figure 4.5: Industrial applications of overconstrained 6R mechanisms. From left to right: A double-Hooke's-joint transmission, a Sarrus linkage used in the positioning of a Bosch saw, and a Turbula mixing machine based on the overconstrained mechanism proposed in [155].

analysis problem in detail in Chapter 2. Next we will give the input-output relations for the three 6R overconstrained linkages with industrial application mentioned in the previous paragraph and, finally, the input-output relations for two representative linkages deeply studied in the literature, namely, the classical Bricard's plane symmetric linkage [15] and the more recent linkage in this class, proposed by Dietmaier in [43].

Bricard trihedral linkage

Let start, thus, analyzing in detail the performance of the presented method for solving the position analysis of the Bricard's overconstrained 6R closed-loop mechanism shown in Fig. 4.6. The conditions defining this family of overconstrained mechanisms is:

$$a_1 = a_2 \tag{4.38}$$

$$a_3 = a_4 \tag{4.39}$$

$$a_5 = a_6 \tag{4.40}$$

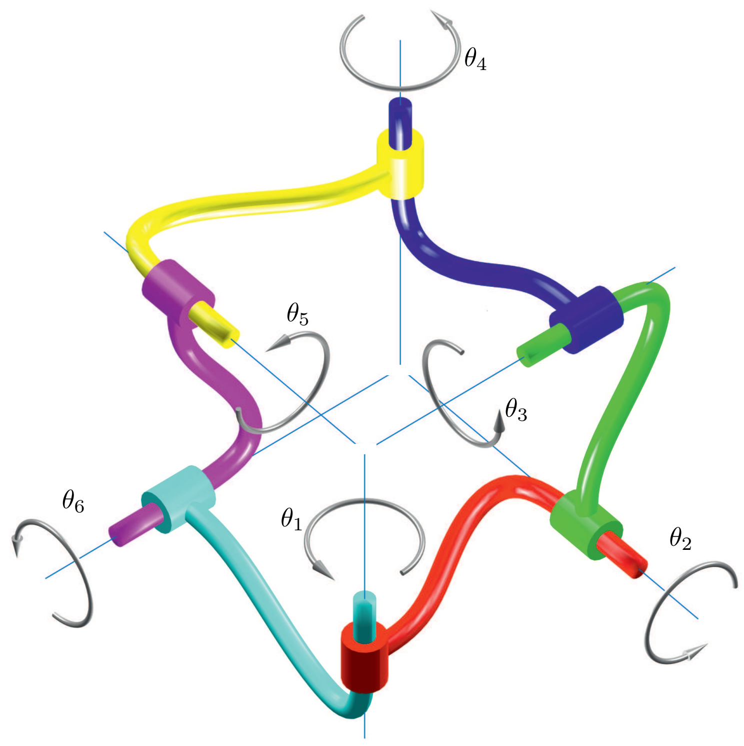
$$\alpha_1 = 2\,\pi - \alpha_2 \tag{4.41}$$

$$\alpha_3 = 2\,\pi - \alpha_4 \tag{4.42}$$

$$\alpha_5 = 2\,\pi - \alpha_6 \tag{4.43}$$

$$d_1 = d_3 = d_5 = 0 \tag{4.44}$$

$$d_6 = -d_2 - d_4 \tag{4.45}$$



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 4.6: Bricard's trihedral overconstrained 6R closed-loop mechanism and its DH parameters. All joint angles in the shown mechanism configuration are $\pi/2$.

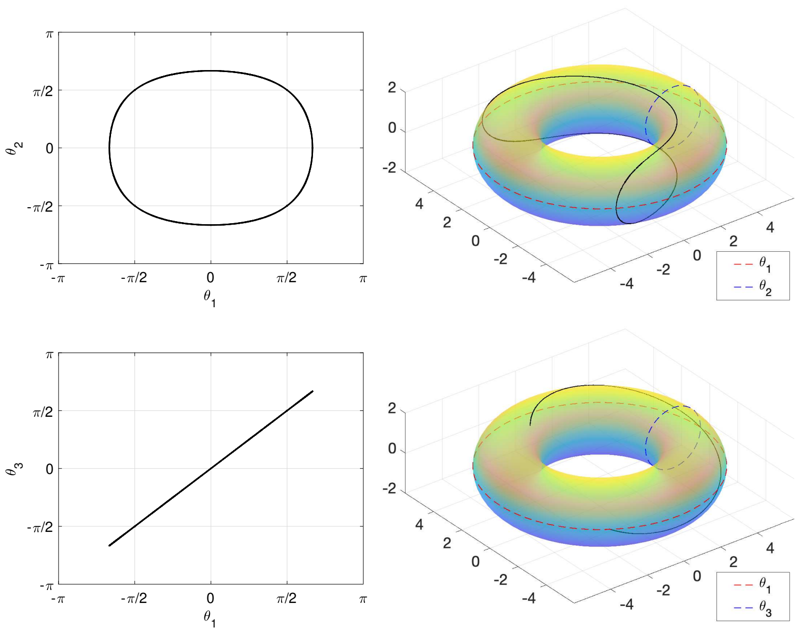


Figure 4.7: The input-output relations of the Bricard trihedral linkage, with θ_1 as the input variable. Only the relations with θ_2, and θ_3 are shown since the relations with the rest of variables are the same as the displayed ones.

In particular, the Bricard's trihedral linkage fulfills:

$$a_1^2 + a_3^2 + a_5^2 = a_2^2 + a_4^2 + a_6^2 \tag{4.46}$$

$$\alpha_1 = \alpha_3 = \alpha_5 = \frac{\pi}{2} \tag{4.47}$$

$$\alpha_2 = \alpha_4 = \alpha_5 = \frac{3\,\pi}{2} \tag{4.48}$$

$$d_1 = d_2 = d_3 = d_4 = d_5 = d_6 = 0 \tag{4.49}$$

The set of DH parameters used in our case are included in Fig. 4.6, Fig 4.7 shows the solution obtained with the solver proposed in this work using the trapezoid box reduction method, and Table 4.14 presents the corresponding performance statistics. Since the solution set is one-dimensional in this problem, Miranda's theorem can not identify which boxes include part of this set. As an alternative, we can post-process each box using the Newton-Raphson method taking as initial guess the box center. If this method converges inside the considered box, it includes part of the sought-after one-dimensional solution set. Using this approach, 1372 solution boxes are verified, taking only 7 ms to the overall execution time.

	LP	**LR**	**Trapezoid**
Processed boxes	12264	7459	31312
Box reductions	18148	9767	45464
Bisected boxes	6100	3729	15624
Empty boxes	3921	1322	12068
Solution boxes	2243	2408	3620
Verified solutions	-	0	1372
Execution time [s]	33.09	6	0.06

Table 4.14: Performance in the characterization of the self-motion manifold of the mechanism in Fig. 4.6.

Double-Hooke's-joint

The double-Hooke's-joint (or Cardan) linkage fulfills:

$$a_2 = a_3 = a_5 = a_6 = 0 \tag{4.50}$$

$$\alpha_2 = \alpha_3 = \alpha_5 = \alpha_6 = \frac{\pi}{2} \tag{4.51}$$

$$d_1 = d_2 = d_3 = d_6 = 0 \tag{4.52}$$

θ_i	d_i	α_i	a_i
θ_1	0	0.1745	2.2361
θ_2	0	$\pi/2$	0
θ_3	0	$\pi/2$	0
θ_4	1	0	1
θ_5	1	$\pi/2$	0
θ_6	0	$\pi/2$	0

Table 4.15: DH parameters of the double-Hooke's-joint linkage used in the experiments.

The particular parameters of the double-Hooke's linkage used in our experiments are included in Table 4.15, and Fig. 4.8 shows the input output parameters of this mechanism obtained with the solver introduced in this work. The statistics of the solution process using the trapezoid box reduction procedure are given in Table 4.16. In this case, the Newton-Raphson box certification procedure identifies 3422 boxes including part of the solution, which is in the same order of magnitude as solution boxes identified by the LR and LP methods. This means that the trapezoid method can provide a good coverage of the self-motion manifold with certified boxes much faster than the alternative methods.

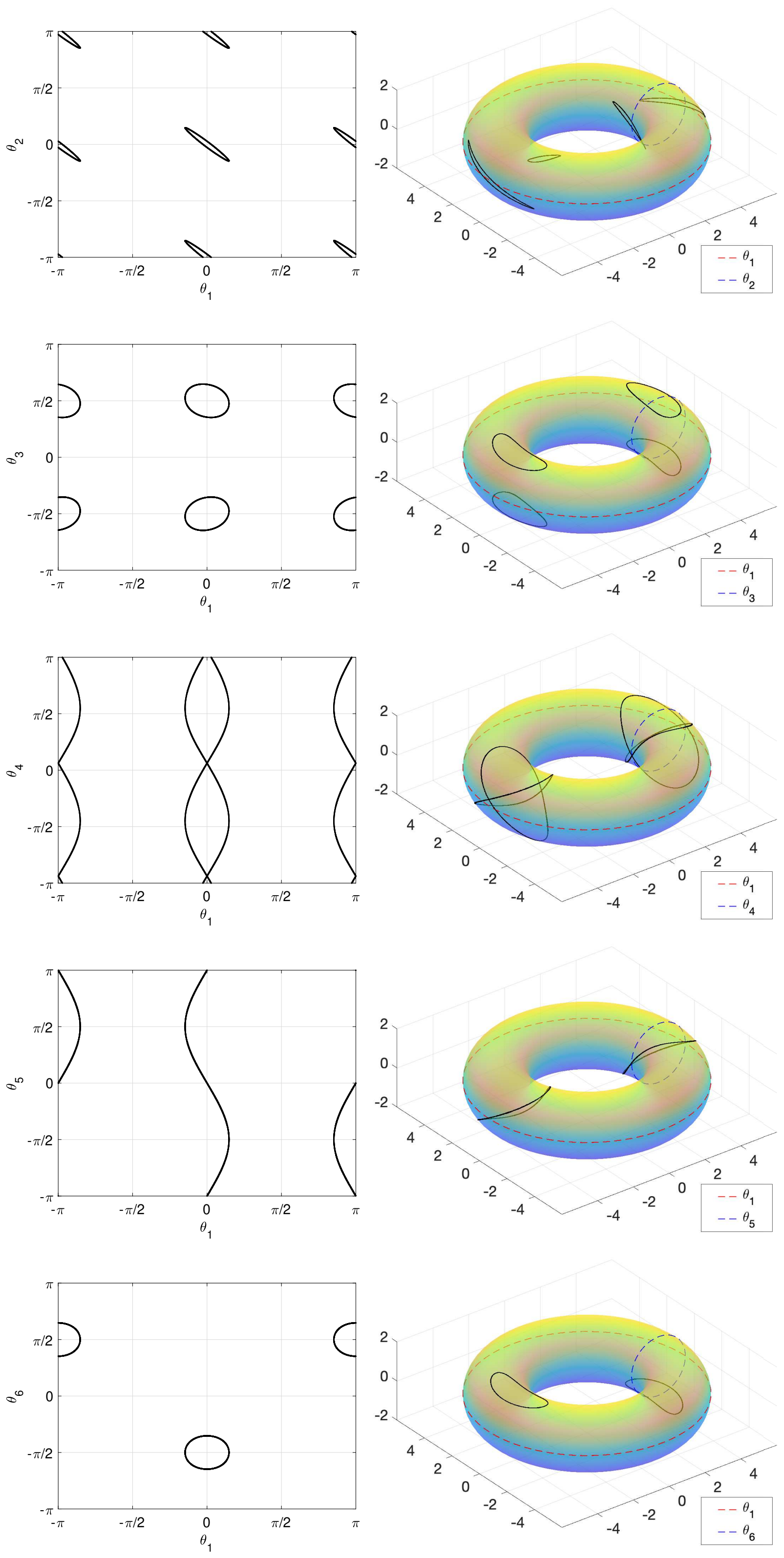


Figure 4.8: The input-output relations of the double-Hooke's-joint linkage, with θ_1 the input variable.

	LP	LR	Trapezoid
Processed boxes	17168	6461	43642
Box reductions	26808	9327	64841
Bisected boxes	8552	3230	21789
Empty boxes	5352	12	15190
Solution boxes	3264	3219	6663
Verified solutions	-	0	3422
Execution time [s]	39.60	11	0.08

Table 4.16: Performance in the characterization of the self-motion manifold of the double-Hooke's-joint linkage.

Sarrus linkage

The Sarrus linkage fulfills:

$$a_1 = a_5 \tag{4.53}$$

$$a_2 = a_4 \tag{4.54}$$

$$a_3 = a_6 = 0 \tag{4.55}$$

$$\alpha_1 = \alpha_2 = \alpha_4 = \alpha_5 = 0 \tag{4.56}$$

$$\alpha_3 = \alpha_6 \tag{4.57}$$

$$d_1 = d_2 = d_3 = d_4 = d_5 = d_6 = 0 \tag{4.58}$$

Note that since all axes are parallel, the d_i parameters for $i \in \{1, \ldots, 6\}$ are irrelevant. Different designs can be defined by changing them.

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

Table 4.17: DH parameters of the Sarrus linkage used in the experiments.

The particular parameters of the Sarrus linkage used in our experiments are included in Table 4.17, and Fig. 4.9 shows the input output parameters of this mechanism obtained with the trapezoid method. The statistics of the solution process using the trapezoid box reduction procedure are given in Table 4.18. In this particular case, the Newton-Raphson certification procedure can not identify solution points in any of the boxes bounding the self-motion manifold.

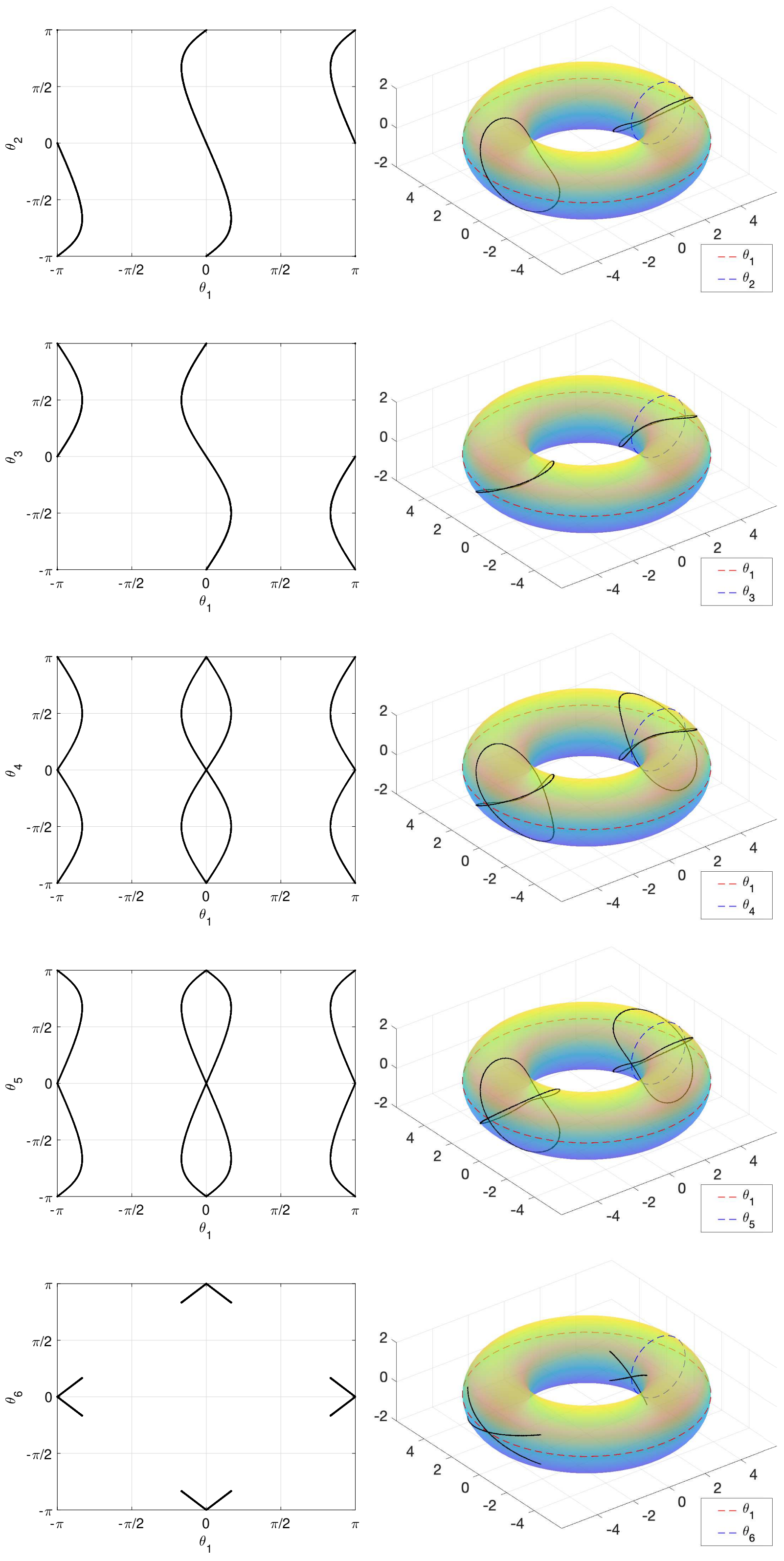


Figure 4.9: The input-output relations of the Sarrus linkage, with θ_1 the input variable.

	LP	**LR**	**Trapezoid**
Processed boxes	16432	12587	63136
Box reductions	27861	13924	100848
Bisected boxes	8184	6293	31536
Empty boxes	3864	2101	22088
Solution boxes	4384	4193	9512
Verified solutions	-	0	0
Execution time [s]	47.57	26.0	0.22

Table 4.18: Performance in the characterization of the self-motion manifold of the Sarrus linkage.

Schatz linkage

The Schatz linkage fulfills:

$$a_1 = a_5 = 0 \tag{4.59}$$

$$a_2 = a_3 = a_4 \tag{4.60}$$

$$a_6 = \sqrt{3}\, a_2 \tag{4.61}$$

$$\alpha_1 = \alpha_2 = \alpha_3 = \alpha_4 = \alpha_5 = \frac{\pi}{2} \tag{4.62}$$

$$\alpha_6 = 0 \tag{4.63}$$

$$d_1 = -d_6 \tag{4.64}$$

$$d_2 = d_3 = d_4 = d_5 = 0 \tag{4.65}$$

θ_i	d_i	α_i	a_i
θ_1	2	$\pi/2$	0
θ_2	0	$\pi/2$	3
θ_3	0	$\pi/2$	3
θ_4	0	$\pi/2$	3
θ_5	0	$\pi/2$	0
θ_6	-2	0	5.1962

Table 4.19: DH parameters of the Schatz linkage used in the experiments.

The particular parameters of the Schatz linkage used in our experiments are included in Table 4.19, and Fig. 4.10 shows the input output parameters of this mechanism obtained with the solver introduced in this work. The statistics of the solution process using the trapezoid box reduction procedure are given in Table 4.20.

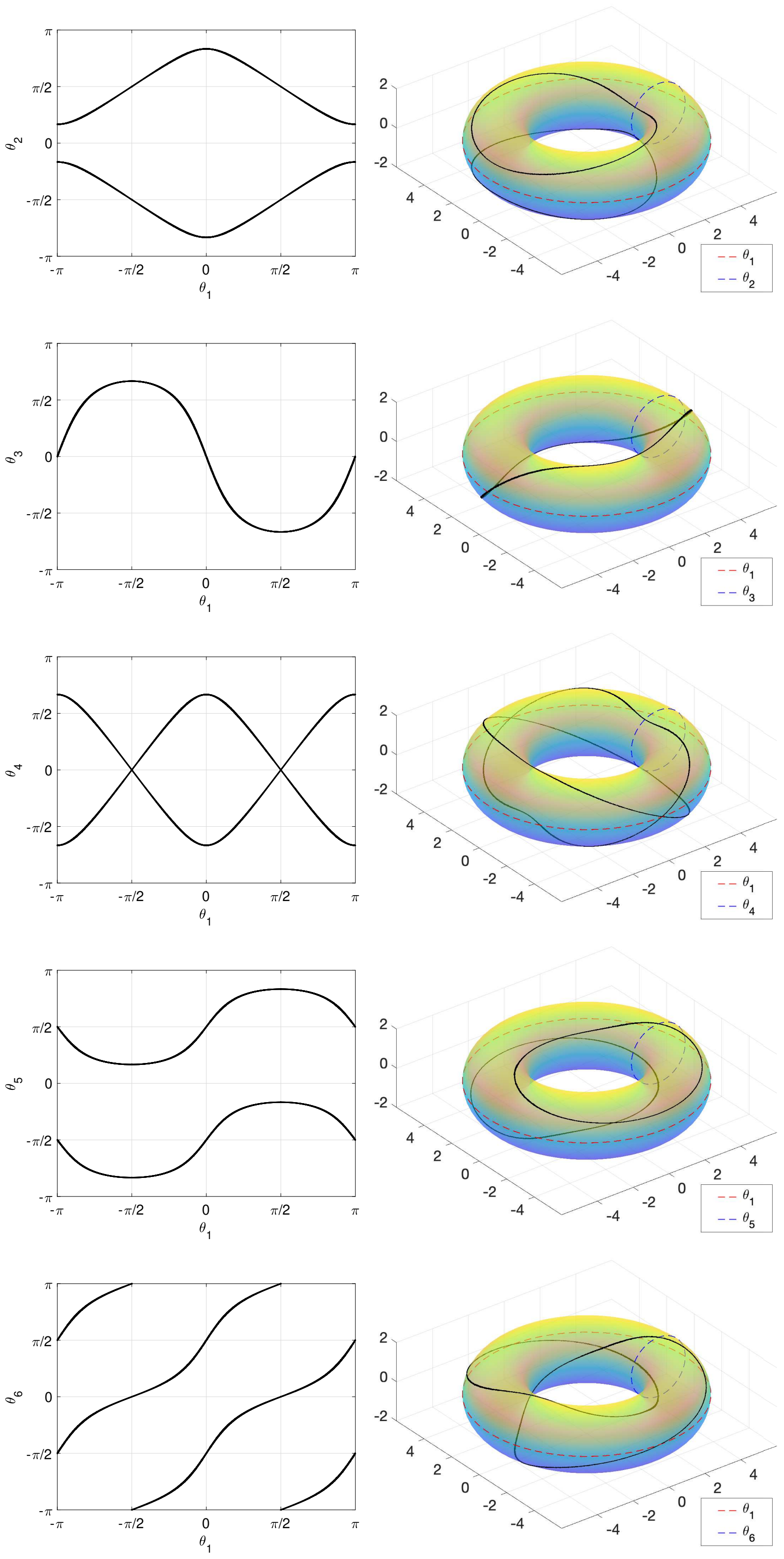


Figure 4.10: The input-output relations of the Schatz linkage, with θ_1 the input variable.

	LP	**LR**	**Trapezoid**
Processed boxes	17912	9889	81700
Box reductions	28184	13498	124639
Bisected boxes	8924	4944	40818
Empty boxes	5024	919	24942
Solution boxes	3964	4026	15940
Verified solutions	-	0	3188
Execution time [s]	55.10	8	0.21

Table 4.20: Performance in the characterization of the self-motion manifold of the Schatz linkage.

Plane symmetric Bricard's linkage

The general plane symmetric Bricard's linkage fulfills:

$$a_1 = a_6, a_2 = a_5, a_3 = a_4 \tag{4.66}$$

$$\alpha_1 = 2\pi - \alpha_6 \tag{4.67}$$

$$\alpha_2 = 2\pi - \alpha_5 \tag{4.68}$$

$$\alpha_3 = 2\pi - \alpha_4 \tag{4.69}$$

$$d_1 = d_4 = 0 \tag{4.70}$$

$$d_2 = -d_6 \tag{4.71}$$

$$d_3 = -d_5 \tag{4.72}$$

θ_i	d_i	α_i	a_i
θ_1	0	$2\pi/3$	1
θ_2	0	$-2\pi/3$	1
θ_3	0	$2\pi/3$	1
θ_4	0	$-2\pi/3$	1
θ_5	0	$2\pi/3$	1
θ_6	0	$-2\pi/3$	1

Table 4.21: DH parameters of the plane symmetric Bricard's linkage used in the experiments.

The particular parameters of the plane symmetric Bricard's linkage used in our experiments are included in Table 4.21, and Fig. 4.11 shows the input output parameters of this mechanism obtained with the solver introduced in this work. The statistics of the solution process are given in Table 4.22.

	LP	**LR**	**Trapezoid**
Processed boxes	15008	8441	98564
Box reductions	23680	10975	151255
Bisected boxes	7472	4220	49250
Empty boxes	4871	1431	29686
Solution boxes	2665	2790	19628
Verified solutions	0	0	1948
Execution time [s]	59.90	6	0.32

Table 4.22: Performance in the characterization of the self-motion manifold of the plane symmetric Bricard's linkage.

Dietmaier linkage

The 6R overconstrained linkage proposed by Dietmaier fulfills

$$a_1 = a_4 \tag{4.73}$$

$$\alpha_1 = \alpha_4 \tag{4.74}$$

$$d_1 = d_3 \tag{4.75}$$

$$d_4 = d_6 \tag{4.76}$$

$$d_2 = d_5 = 0 \tag{4.77}$$

$$\frac{a_2}{\sin\alpha_2} = \frac{a_3}{\sin\alpha_3} \tag{4.78}$$

$$\frac{a_2\,(\cos\alpha_2 + \cos\alpha_3)}{\sin\alpha_2} = \frac{a_5\,(\cos\alpha_5 + \cos\alpha_6)}{\sin\alpha_5} \tag{4.79}$$

$$\frac{a_5}{\sin\alpha_5} = \frac{a_6}{\sin\alpha_6} \tag{4.80}$$

θ_i	d_i	α_i	a_i
θ_1	0	0.6981	1
θ_2	0	0.8727	1.3
θ_3	0	1.2217	1.5947
θ_4	0	0.6981	1
θ_5	0	1.9199	-9.3273
θ_6	0	1.3963	-9.7751

Table 4.23: DH parameters of the Dietmaier's linkage used in the experiments.

The particular parameters of the Dietmaier linkage used in our experiments are included in Table 4.23, and Fig. 4.12 shows the input output parameters of this mechanism obtained with the solver introduced in this work. The statistics of the solution process are given in Table 4.24. As in all the previous case, the trapezoid method is significantly faster than the

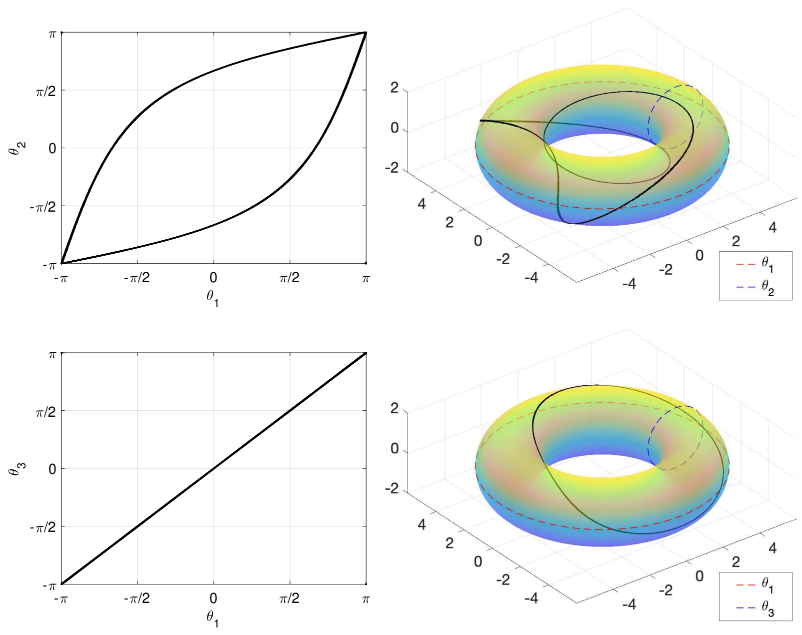


Figure 4.11: The input-output relations of the plane symmetric Bricard's linkage, with θ_1 the input variable. Only the relations with θ_2, and θ_3 are shown since the relations with the rest of variables are the same as the displayed ones.

two other methods. It is typically two orders of magnitude faster than the LP method and one order of magnitude faster than the LR method. It isolates the self-motion manifold with some clustering (specially noticeable in this example) but, in general it produces a dense set of certified boxes. These results qualify the methods proposed in this thesis as a the most effective and efficient tool to derive the input-output relations of overconstrained linkages.

	LP	**LR**	**Trapezoid**
Processed boxes	19996	9871	218620
Box reductions	34892	14232	345607
Bisected boxes	9966	4935	109278
Empty boxes	6380	1380	57244
Solution boxes	3650	3556	52098
Verified solutions	0	0	2768
Execution time [s]	64.64	9	0.58

Table 4.24: Performance in the characterization of the self-motion manifold of the Dietmaier's linkage.

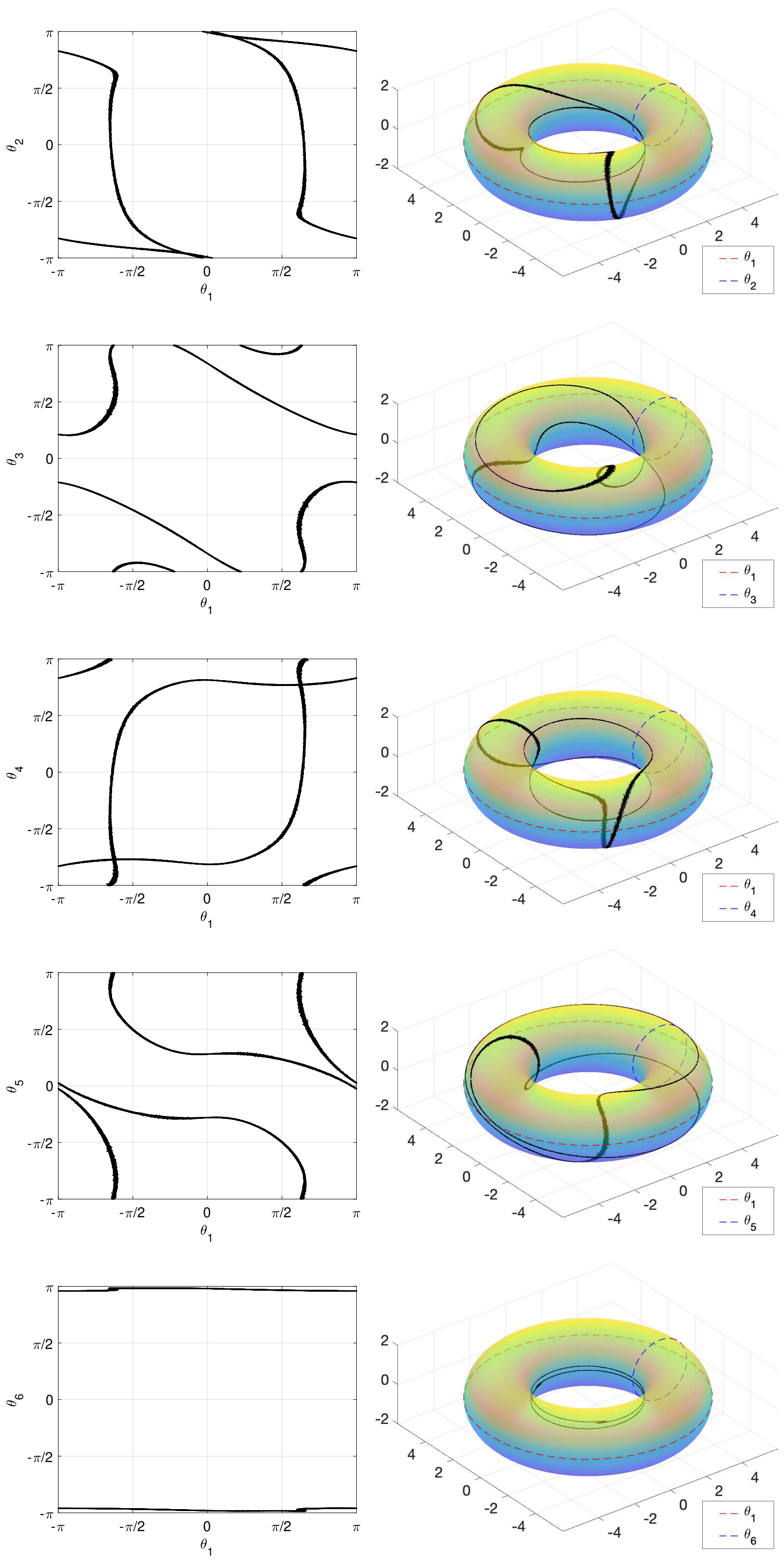


Figure 4.12: The input-output relations of the Dietmaier's linkage, with θ_1 the input variable.

4.4 Computation of the input-output relationships of mobile closed-loop mechanisms

In this section we analyze the input-output relations of several closed-loop mechanisms with seven joints, which can be either revolute of prismatic joints. Such mechanisms are mobile and, in general, they exhibit a one-dimensional motion. Kinematotropic mechanisms, thought, can have variable degrees of freedom, depending on the configuration. Examples of these two classes of mechanisms are considered herein to validate the solver proposed in this thesis and to show its robustness and generality.

A C5R loop with a singular 1-dimensional solution set

Fig. 4.13-top shows a C5R closed-loop mechanism resulting from substituting one revolute joint of Wohlhart's 6R mechanism with a cylindrical joint. The self-motion manifold of this mechanism is also one-dimensional, but singular (it contains nodes). When the translation in the cylindrical joint is null, we obviously have the standard Bricard's mechanism. Therefore, the solution set appearing in Fig. 4.7 will appear as a subset of the new solution set.

	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
Verified	-	0	9892
Execution time [s]	1237	101	4

Table 4.25: Performance of the three compared methods for the self-motion manifold computation of the mechanism in Fig. 4.13.

Figure 4.14 shows the characterization of the self-motion manifold of this mechanism obtained using the trapezoid method. The nodes appearing on this self-motion manifold are given in Table 4.26. Although hardly noticeable to the naked eye in the plot, clustering appears, specially around the nodes. This is clear from the results in the table since the LP method bounds the solutions set with 138884 whereas the trapezoid method delivers 271828 solution boxes. Using the Newton-based procedure 9892 of these boxes can be verified to include a solution. Thus, approximation obtained with the LP method is more accurate, but obtaining it is about 300 times more expensive in terms of computational time. One possibility is to use the trapezoid method first and then filter out the possible solutions using the LP method. This procedure

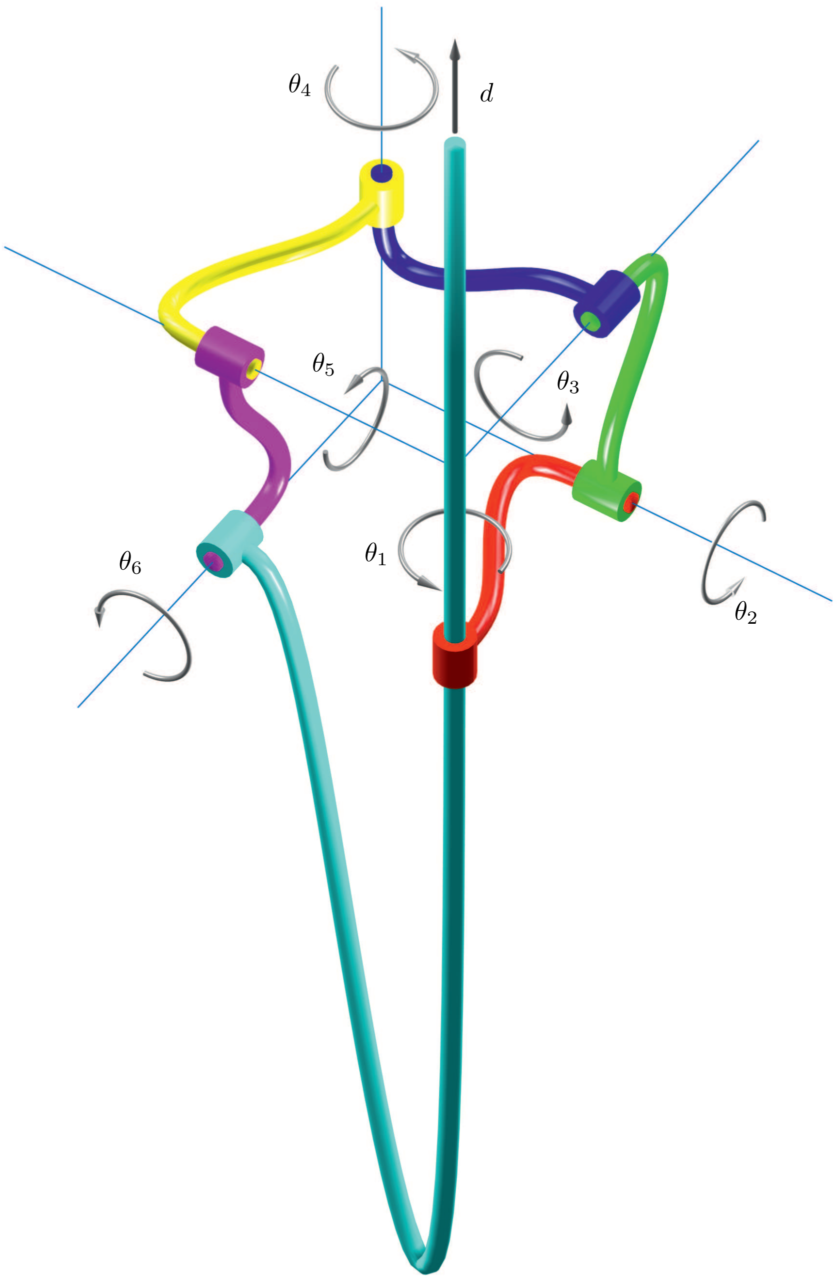
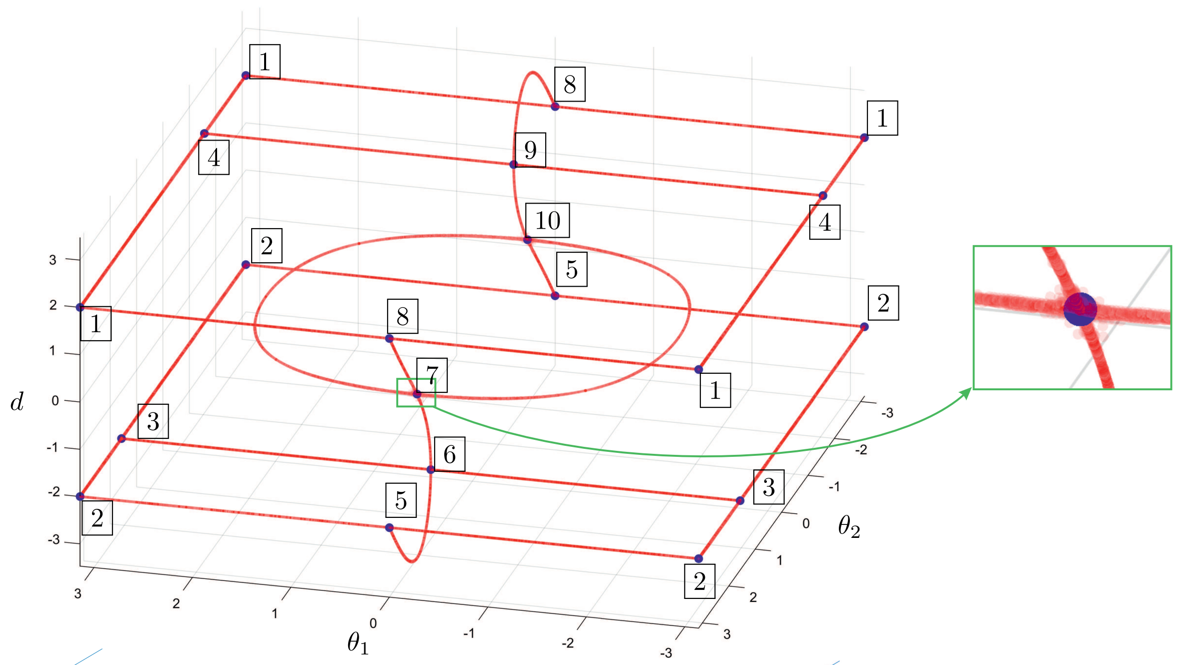


Figure 4.13: C5R closed-loop mechanism resulting from substituting one revolute joint in the Bricards's overconstrained mechanism with a cylindrical joint. Contrarily to one might expect, its self-motion manifold remains one-dimensional after this substitution.



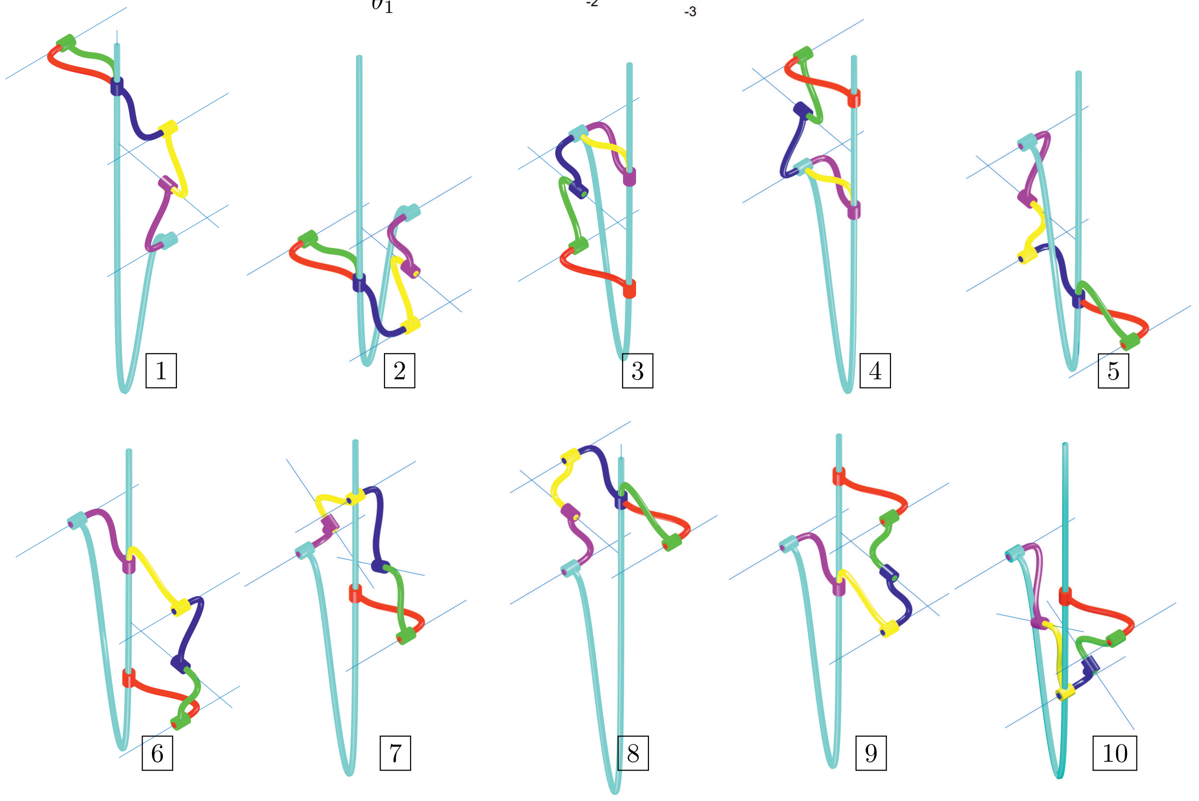


Figure 4.14: Top: self-motion manifold of the C5R closed-loop mechanism in Fig. 4.13 projected on the subspace defined by θ_1, θ_2, and d. The solution boxes are represented as semitransparent spots to better appreciate the accumulation of boxes. Here, we only represent the result obtained using the trapezoid method. The only noticeable difference with respect to the solutions obtained using the LP or the LR methods is the cluster effect around the nodes, as shown on the right zoom-in. Bottom: mechanism configurations at the nodes of the self-motion whose coordinates are given in Table 4.26. Some nodes are repeated at the boundaries of the represented region because this self-motion is periodic in θ_1 and θ_2.

isolates the solution set with 192362 solution boxes in about 466 s, still faster than when using the LP method directly.

Node	d	θ_1	θ_2	θ_3	θ_4	θ_5	θ_6
1	2	π	π	0	$\pi/2$	0	$\pi/2$
2	-2	π	π	0	$-\pi/2$	0	$-\pi/2$
3	-2	π	$\pi/2$	0	$\pi/2$	π	π
4	2	π	$-\pi/2$	0	$-\pi/2$	π	π
5	-2	0	π	0	$-\pi/2$	0	$-\pi/2$
6	-2	0	$\pi/2$	0	$\pi/2$	0	π
7	0	0	$2\pi/3$	0	$2\pi/3$	0	$2\pi/3$
8	2	0	π	0	$\pi/2$	0	$\pi/2$
9	2	0	$-\pi/2$	0	$-\pi/2$	0	π
10	0	0	$-2\pi/3$	0	$-2\pi/3$	0	$-2\pi/3$

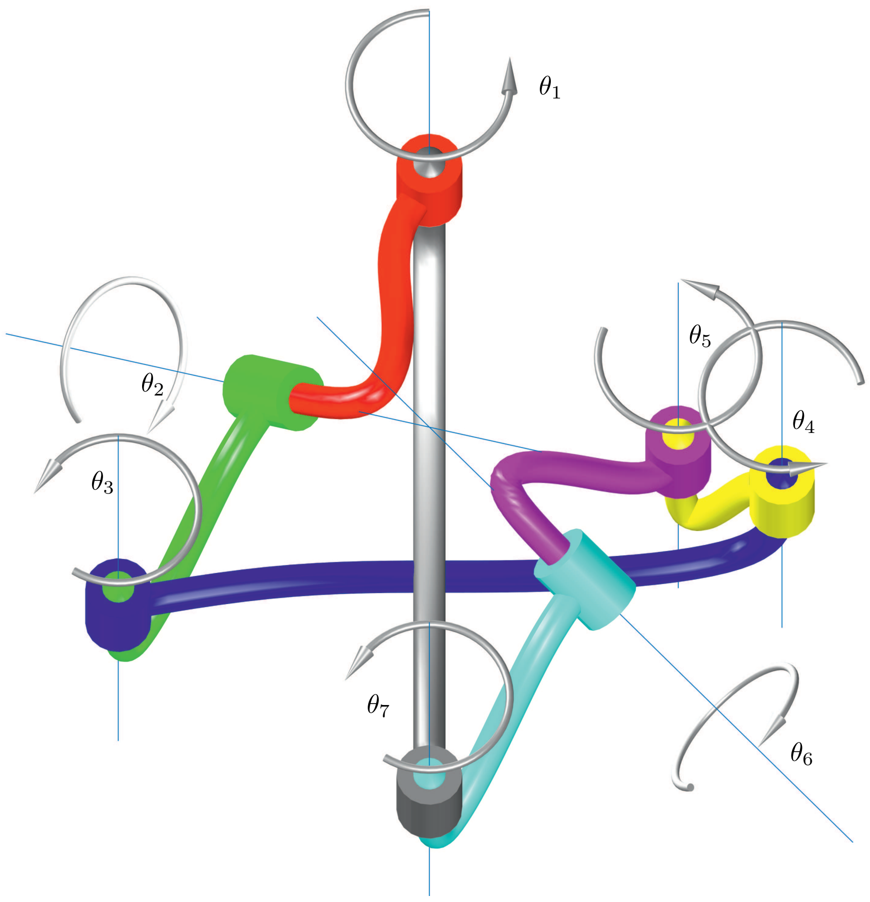
Table 4.26: Coordinates of the nodes of the self-motion manifold represented in Fig. 4.14.

A 7R closed-loop kinematotropic mechanism

Mechanisms with variable degrees of freedom (or kinematotropic) are a class of reconfigurable mechanisms motion modes with variable degrees of freedom [64,87,125,195], despite its theoretical mobility computed as described in Section 4.3 should be constant. The kinematic analysis of these mechanisms is of particular interest since their configuration space includes algebraic components of different dimension. Therefore, there has been a significant amount of work devoted to derive their input-output relations using both algebraic [84,126] of continuation [68] methods.

Among all possible 7R kinematotropic closed-loop mechanisms, the one studied in [85] (see Fig. 4.15), which is a particular case of the one proposed in [87], is of particular interest for us because it has five motion modes, the largest number of modes known to date for such a kind of mechanism. According to the results presented in [87], 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 nice challenging example in which we can check the correctness of our method and, if possible, attain a greater insight into this peculiar mechanism.

The performance of the three compared methods to compute the self-motion manifold of this mechanism appear in Table 4.27. The representation of the obtained solutions using the LP and the trapezoid methods appear in Fig. 4.16-top. In this example, while the LP and the trapezoid 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



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 4.15: 7R closed-loop mechanism studied in [85], and its DH-parameters.

suggests the idea that the trapezoid method is very good at characterizing the components of higher 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.

In Fig. 4.16-top, we have identified the ten transition configurations with the same indices as in [85] and the corresponding mechanism configurations are represented in Fig. 4.16-bottom. They perfectly match those reported in [85]. The situation is not so clear when trying to identify the five 1-dimensional components reported in [85]. Although their identification from Fig. 4.16-top might be puzzling due to the $2\pi-$periodicity of the represented set in the three coordinate axes, a simple visual inspection reveals that this number should be higher than five. Indeed, if we remove the transition points from the self-motion manifold, we obtain one 2-dimensional and 16 1-dimensional connected smooth manifolds, not five. The reason for this discrepancy is rather subtle: the decomposition in motion modes given in [87] does not correspond to a stratification of the self-motion manifold. According to our analysis, the self-motion manifold can be decomposed into the union of one 2-dimensional, 16 1-dimensional, and 10 0-dimensional smooth submanifolds. Roughly speaking, since these submanifolds are disjoint, and their union is the whole self-motion manifold, this decomposition is a *stratification* and each submanifold is a *stratum*. Nevertheless, strictly speaking, some additional conditions on the way in which the parts fit together must also be satisfied to be stratification, which we omit here for the sake of simplicity (see [186] for details). On the contrary, the characterization given in [87] is not a stratification. Two strata of dimension 1 accepting the same parameterization is presented as a single motion mode. As a consequence, the connectivity graph between the different motion modes given in [87] should not be confused with a topological description of the corresponding self-motion manifold. In our case, a correct topological description can be directly inferred from Fig. 4.16-top and simply stated as follows: the 0-dimensional strata indexed from 1 to 8 are directly connected to the 2-dimensional stratum. The 0-dimensional strata 9 and 10 are connected to these eight strata through two disjoint sets of eight 1-dimensional strata each.

Using screw theory, it can be readily proved that in each transition configuration, the variable-DOF 7R mechanism has generally three instantaneous DOF.

4.5 Position analysis of spatial parallel mechanisms

A parallel mechanism is typically composed of a moving platform connected to a base by means of serial kinematic chains, which are called legs. Only few of the joints in each leg are actuated and the rest are passive. The actuators tend to be mounted near the base so that the platform has a low inertia and, thus, it can be efficiently moved. Parallel platforms have high speed, payload,

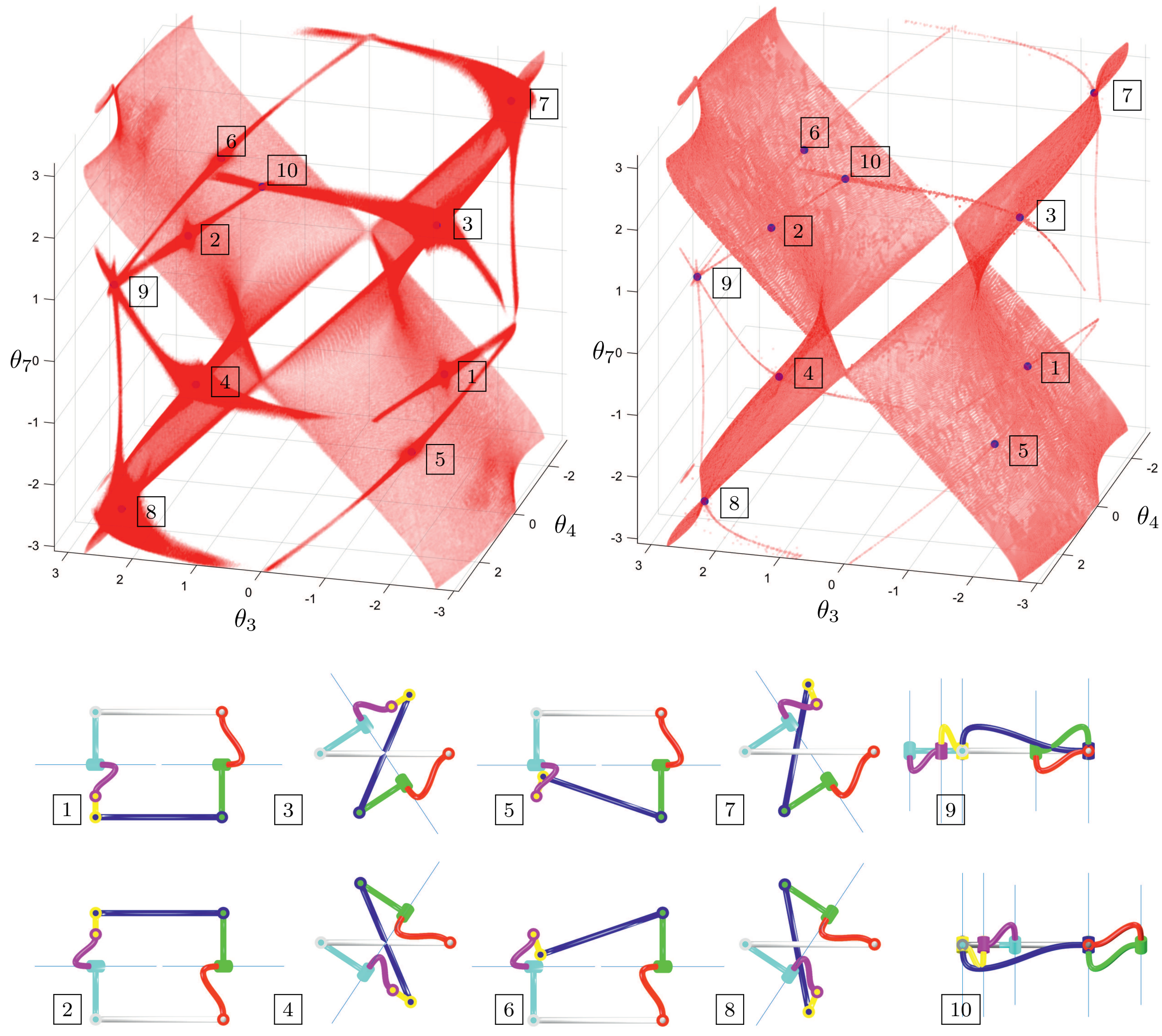


Figure 4.16: Top: Approximation of the self-motion manifold of the mechanism in Fig. 4.15 using the trapezoid (left) and the LP (right) methods. The results have been projected onto the subspace defined by θ_3, θ_4, and θ_7. Bottom: The strata of dimension 0 have been identified using an index and the corresponding mechanism configurations represented in two rows. They represent the transition points between the different motion modes.

	LP	**LR**	**Trapezoid**
Processed boxes	706658	679935	12699994
Box reductions	1090058	733929	18416851
Bisected boxes	353265	339967	6349933
Empty boxes	115952	61423	3279703
Solution boxes	237441	278545	3070358
Verified solutions	-	0	238261
Execution time [s]	7000	339	89

Table 4.27: Performance of the three compared methods for the computation of the self-motion manifold of the mechanism in Fig. 4.15.

stiffness, and accuracy. Their main disadvantage is that due to leg collision and specially due to the low mobility ranges of the spherical joints typically used in the legs, their workspace is usually small. Such issues are highly-relevant in the most know architecture, the Gough-Steward platform [36]. This platform is composed by two bodies connected by six legs each formed by a universal, a prismatic and a spherical joint. In this case, only the prismatic joint is actuated and the platform can move with six degrees of freedom. Many industrial applications, however, do not require of six degrees of freedom. Thus, architectures with less than six DOF have been proposed [135]. Most of them have 3 DOF and their added advantage is that they are simpler and cheaper to construct [52]. They may exhibit spherical (discussed in Section 4.1), translational [22], or mixed motion models [46]. In the later, position and orientation displacement are coupled and their accuracy is typically low due to the presence of parasitic motions. An alternative is to use hybrid architectures where position and orientation are decoupled [161, 179]. Yet another way to alleviate the shortcomings general parallel platforms, is to keep the number of legs low, but increase the number of actuated joints at each leg [8, 20, 31, 33, 160, 181]. Such structures are more complex to construct, but keep the advantages of parallel robots with larger workspaces. In any case and whatever the design used, parallel platforms have at least three legs and, thus, they define a multi-loop system.

As in the case of spherical mechanisms (see Section 4.1), spatial multi-loop mechanisms do not offer any extra conceptual complexity once formalized as follows. First, a graph is defined where the vertices are the links of the mechanism and the edges are given by the joints connecting the links. 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 a independent loop [27]. Finally, the equations for each one of these independent loops are generated using the procedure defined in Section 2.3. In a mechanism with m links and n joints, we have $m - n + 1$ independent loops. However, different sets of independent loops can be extracted from a given graph. All of them encode the same problem, but not all of them are equally adequate from a computational point of view.

We use the Tripteron mechanism shown in Fig. 4.17 to illustrate this procedure. This is a patented [59] 3-$\underline{P}$RRR 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 representing the displacement of the slider joint of the first leg.

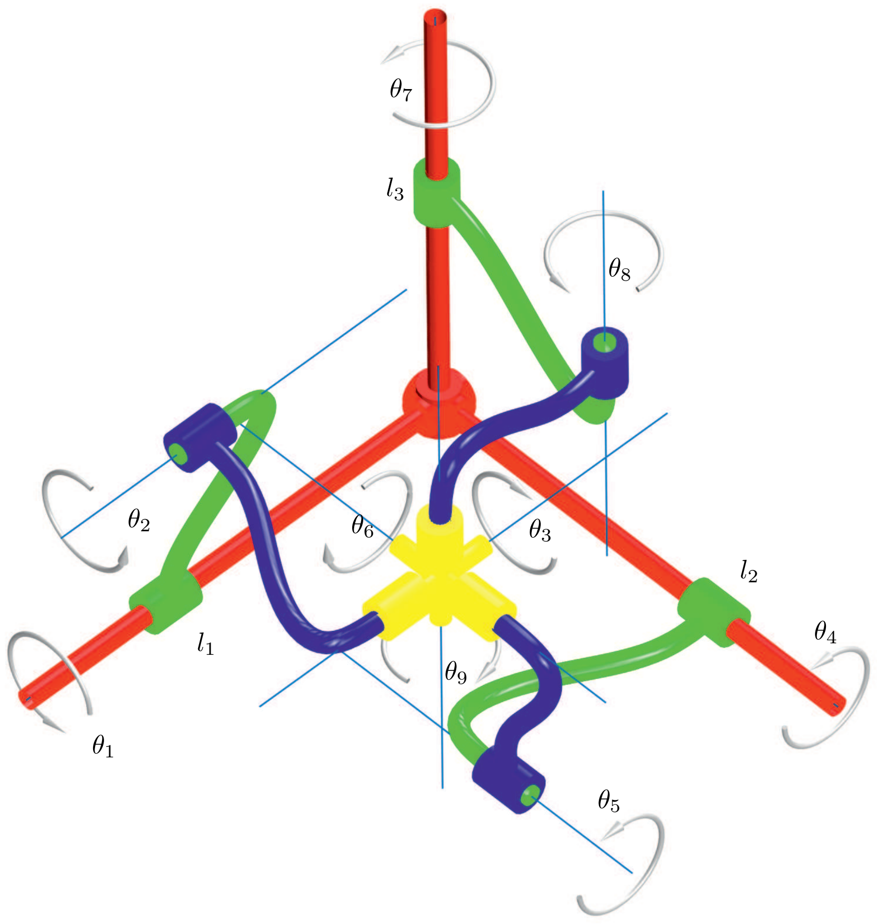
The three legs of the robot define three loops, but only two of them are independent (i.e., the third one can be deduced from the other two). In this case, we use the loop formed by legs one and two and the loop formed by legs one and three.

	LP	**LR**	**Trapezoid**
Processed boxes	1084304	14611	7980140
Box reductions	1735675	18310	8915118
Bisected boxes	541896	7305	3976104
Empty boxes	535483	20	4656253
Solution boxes	6925	7305	14222
Verified	-	0	8994
Execution time [s]	10302	14	32

Table 4.28: Performance of the three compared methods for the self-motion manifold computation of the 3-$\underline{P}$RRR parallel robot in Fig. 4.17.

Table 4.28 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 so because the LR method incorporates some basic symbolic simplification procedures which, due to the regularity of the parameters for this robot, results in a particularly trivial set of equations. 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 we will see in the next chapters, the basic trapezoid method can be improved significantly to the point of outperform the LR method taking into account redundant equations and variable eliminations. Without such improvements, though, the trapezoid method bounds the solution set with 14222 small boxes. Using the Newton-based verification procedure, 8994 of them are verified to include a point of the sought solution set. This is more than the number of solution boxes returned by the LP method, meaning that the trapezoid method identifies a dense set of verified solutions.

Figure 4.18 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



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

Figure 4.17: 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.

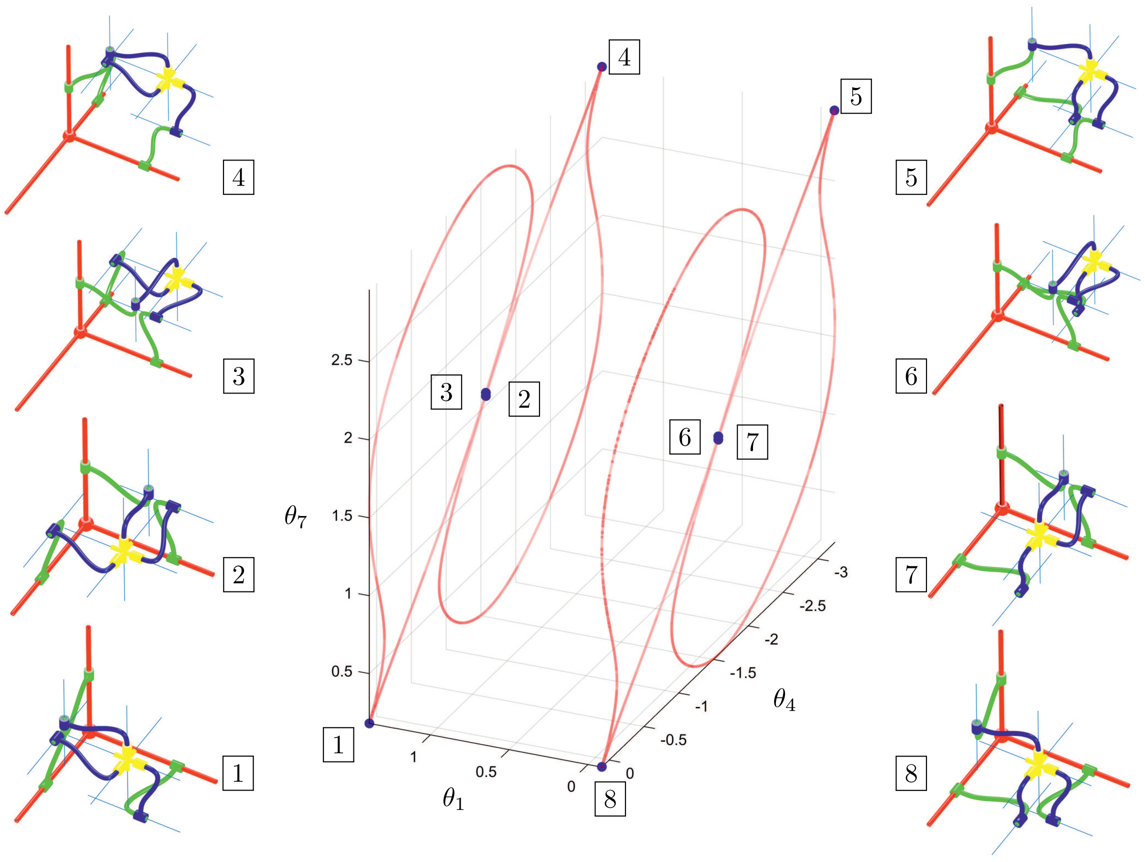


Figure 4.18: The self-motion manifold of the 3-$\underline{P}$RRR parallel robot and some representative configurations.

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.

5 Multi-affine redundancy

In this chapter we show how we take advantage of multi-affine redundant equations to reduce the clustering effect. We describe how such redundant equations can be generated and analyze experimentally which ones are more beneficial in the solving process.

Redundancy is typically used in engineering to obtain robust solutions to a given problem. This robustness, though, is obtained at a higher cost (e.g., at the cost of implementing two complementary and concurrent solutions for the same problem). In the context of solving systems of equations, redundancy manifest in the form of a system with more equations than those strictly needed to represent a given problem. It is well known that in linear systems, redundancy creates overdetermined problems, which, if the redundancy is properly defined, are compatible, i.e. some points exist that fulfill all the equations simultaneously. The same happens in non-linear systems of equations.

The different approaches to solve systems of non-linear equations reviewed in Chapter 1 deal with redundancy in different ways. In some approaches, like interval Newton [80] or some continuation methods [62], redundant equations are identified and discarded. In some cases, redundancy is not just tolerated, but explicitly favored. This is the case of many elimination methods that augment the system of equations before condensing it. Finally, branch-and-prune methods like linear relaxations or the trapezoid method introduced in this work, can be directly applied to redundant systems. The former uses a linear solver where the active set of constraints (i.e., a non-redundant subset of linear inequalities) is considered at each step. The trapezoid method, instead, processes each equation independently and, thus, any number of equations can be considered. The only effect is a linear increment in the computational cost. This increment, though, is often compensated by the additional box reduction provided by the redundant equations. Moreover, this additional reduction mitigates the clustering effect (see Section 3.2.2).

This effect is caused by the fact of proceeding equation-wise since each equation is just a necessary condition for a point being a solution. Therefore, taking into account just one equation many points can be wrongly characterized as being valid solutions. However, by considering a large number of redundant, sufficient conditions, we are approximating a necessary condition, i.e., the probability of a point being wrongly identified as a solution decreases with every new redundant equation added to the system.

Taking into account the potential benefits of using redundant equations, we next present different options to generate redundant multi-affine equations and then we evaluate their effect in several representative problems.

5.1 A redundant equation for each loop

Using the notation introduced in Chapter 2, the closure equations of a kinematic loop can be compactly represented from its DH parameters $(\theta_i, d_i, \alpha_i, a_i)$ for $i = 1 \ldots n$ as

$$\mathbf{r_z}(t_1)\,\mathbf{t_z}(d_1)\,\mathbf{r_x}(u_1)\,\mathbf{t_x}(a_1)\cdots\mathbf{r_z}(t_n)\,\mathbf{t_z}(d_n)\,\mathbf{r_x}(u_n)\,\mathbf{t_x}(a_n)\,\mathbf{h} = c, \tag{5.1}$$

where $t_i = \frac{\theta_i}{2}$, $u_i = \tan\frac{\alpha_i}{2}$, $\mathbf{h}$ is a constant dual quaternion representing the pose of the end-effector with respect to the origin, and c is a non-null scalar depending on the configuration. This equation can be more compactly represented as

$$\mathbf{t}_1 \ldots \mathbf{t}_n = c, \tag{5.2}$$

where $\mathbf{t}_i$, $i = 1, \ldots, n$ is each one of the individual transforms in Eq. (5.1). This equation generates eight independent multi-linear equations, four in the rotational part and four in the translational (or dual) part. The equation given by the real term in the rotational part can not be used unless the scalar c, which is non-linearly dependent on the configuration, is known. The other seven equations are multi-linear, but only six of them are independent. The remaining one can be used to increase the redundancy of the system.

5.2 Cyclic permutations of loops

The formulation of a loop closure condition can be arbitrarily started at any point in the loop. Each starting point results in an equivalent, but different system of loop closure equations. This property can be leveraged to generate redundant equations for each kinematic loop.

Formally, if the loops closure is represented as

$$\mathbf{t}_1 \ldots \mathbf{t}_n = c. \tag{5.3}$$

with $\mathbf{t}_i$ $i = 1, \ldots, n$ each individual constant or variable transform in the loop, then

$$\begin{aligned} \mathbf{t}_2 \ldots \mathbf{t}_n\, \mathbf{t}_1 &= c, \\ \mathbf{t}_3 \ldots \mathbf{t}_n\, \mathbf{t}_1\, \mathbf{t}_2 &= c, \\ &\ldots \\ \mathbf{t}_n\, \mathbf{t}_1 \ldots \mathbf{t}_{n-1} &= c \end{aligned}$$

are alternative formulations of the same condition. Each of them express the same solution variety, but with different systems of equations. The clustering effect is caused by using one equation at a time and it is exacerbated if the solution varieties of the individual equations are near tangent. Thus, having more alternative equations decreases the probability of being affected by this undesired effect.

If the transform moved from one end of the loop equation to the other is constant, we essentially have the same system of equations, but with a different preconditioner. A preconditioner is the application of a transformation to a problem to obtain a form that is more suitable for a given solving method. If the moved transform is variable, we end up with a transform sequence where the order of variables changes. Our experiments show that adding redundant equations by changing the order of variables in the loop has notable effects on the solving process, but using the ones obtained just changing the constant preconditioners has a modest effect.

5.3 Redundant loops

As mentioned in Section 4.5, in a multi-loop mechanism, a set of independent loops is enough to properly encode the problem since the rest of loops can be generated by composition of the independent ones. Thus, in multi-loop mechanism, a way to generate redundant equations is to compose two or more overlapping loops to obtain a redundant loop constraint.

Formally, the composition of two loop equations

$$\mathbf{t}_1 \ldots \mathbf{t}_i\, \mathbf{t}_k \ldots \mathbf{t}_{k+r} = c_1, \tag{5.4}$$

and

$$\mathbf{t}_1 \ldots \mathbf{t}_i\, \mathbf{t}_l \ldots \mathbf{t}_{l+s} = c_2, \tag{5.5}$$

generates a loop where the common part between the two composed links vanishes

$$\mathbf{t}_k \ldots \mathbf{t}_{k+r}\, \mathbf{t}_{l+s}^{-1} \ldots \mathbf{t}_l^{-1} = c_3. \tag{5.6}$$

This operation can be repeated for each pair of overlapping loops, independently of whether they are basic or already composed ones.

5.4 Examples

We will evaluate the effects of the redundant equations described in Sections 5.1 and 5.2 on three problems with solution sets of different dimension. First, we will consider the 6R loop with 16 solutions whose parameters are given in Table 4.4 in Section 4.2.2. Then, we will use the C5R loop described in Section 4.4 which has the one-dimensional solution set shown in Fig. 4.14. Finally, we will examine the effects of redundancy in the 7R kinematotropic mechanism with the 2-dimensional solution shown in Fig. 4.16. In all three cases we will solve the problem with the trapezoid method and:

- The basic set of six non-redundant equations per loop described in Chapter 2 that will be used as a reference;
- The same basic set, adding the extra equation per loop described in Section 5.1;
- All the equations that can be generated cycling each loop as described in Section 5.2, including the circulation of constant transforms;
- The equations with different cyclic permutations of variables that are obtained cycling the loops.

These four systems will be denoted, respectively, as *Basic non-redundant*, *Redundancy in loops*, *Cyclic permutation (all)*, and *Cyclic permutation (variables)*.

Tables 5.1, 5.2, and 5.3 show the results of the experiments in the different problems. In all cases, we observe that adding a new equation in each loop results in a moderate reduction of the clustering effect and of the number of processes boxes. This reduction does not compensate the extra time required to process an additional equation. Therefore, the overall execution time typically increases when taking into account this redundant constraint.

A similar increment in the computational cost is observed when considering the redundant equations resulting from cycling all the transforms in the loop equations, including the constant ones. However, in this case there is a notable mitigation of the clustering effect to the point that the number of solution boxes is closer to the one obtained with the LP method (see Chapter 4),

	Basic non-redundant	**Redundancy in loop**	**Cyclic permutations (all)**	**Cyclic permutations (variables)**
Processed boxes	20270	17922	4668	5820
Box reductions	27698	24546	5168	6664
Bisected boxes	10103	8929	2302	2878
Empty boxes	10149	8975	2350	2926
Solution boxes	18	18	16	16
Verified solutions	16	16	16	16
Execution time [s]	0.035	0.033	0.054	0.040

Table 5.1: Performance of the trapezoid method without and with different sources of redundancy for a 6R loop with 16 solutions. See the text for details.

	Basic non-redundant	**Redundancy in loop**	**Cyclic permutations (all)**	**Cyclic permutations (variables)**
Processed boxes	926804	856200	390368	399592
Box reductions	1408882	1280852	453548	471564
Bisected boxes	463370	428068	195152	199764
Empty boxes	191606	188716	19596	23308
Solution boxes	271828	239416	175620	176520
Verified solutions	9892	0	11092	8051
Execution time [s]	4	8.8	23.7	13.8

Table 5.2: Performance of the trapezoid method without and with different sources of redundancy for the C5R loop. See the text for details.

which applies necessary and sufficient conditions and not just necessary ones. When considering only the cyclic permutations that change the order of the variables in the loop, we obtain a similar reduction on the clustering effect and, in some cases like in the 7R kinematotropic mechanism, there is also a significant reduction in the overall execution time (see Table 5.3). Figure 5.1 shows the approximation of the solution set obtained when considering this last set of redundant equations compared to the approximation obtained when considering the basic, non-redundant set of equations. This approximation is noticeable closer to the one obtained when using the LP method (shown in Fig. 4.16), which, as mentioned, applies necessary and sufficient conditions.

The effect of using redundant loops in multi-loop mechanisms will be illustrated with the Tripteron parallel mechanism shown in Fig. 4.17. The three legs of this parallel mechanism define two independent loops. Table 5.4 includes the performance of the trapezoid method when considering the two independent loops formed by leg one and leg two and by leg one and leg three. The table also includes the results obtained when additionally considering the loop formed by combining these two loops, i.e., the loop formed by legs two and three. Finally, the

	Basic non-redundant	**Redundancy in loop**	**Cyclic permutations (all)**	**Cyclic permutations (variables)**
Processed boxes	12699994	12605922	685486	742550
Box reductions	18416851	18286384	932976	1041208
Bisected boxes	6349933	6302897	342679	371211
Empty boxes	3279703	3235703	68243	86472
Solution boxes	3070358	3067322	274564	284867
Verified solutions	238261	0	202132	204187
Execution time [s]	89	132	62	35

Table 5.3: Performance of the trapezoid method without and with different sources of redundancy for the 7R kinematotropic mechanism. See the text for details.

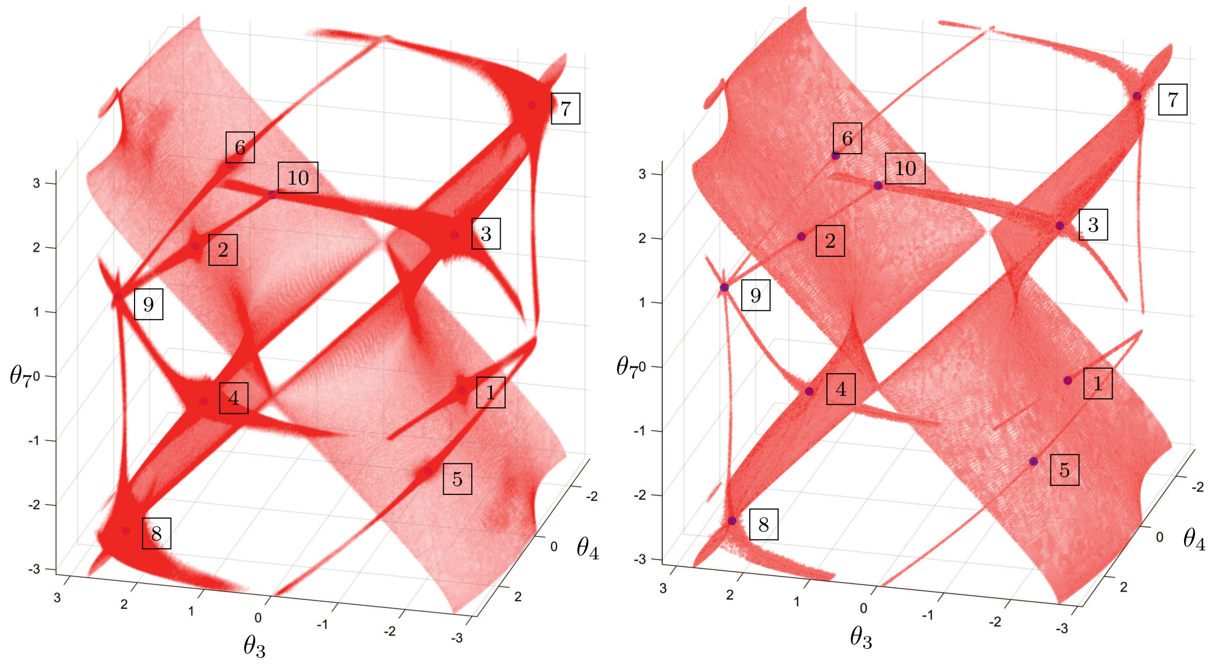


Figure 5.1: Approximation of the self-motion manifold of the 7R kinematotropic mechanism using the trapezoid method on the basic set of non-redundant equations (left) and on a set of redundant equations obtained with cyclic permutation of the loop equations, considering only the permutations where the order of variable change. The results have been projected onto the subspace defined by θ_3, θ_4, and θ_7. Using the redundant set of equations, the solution set is approximated more accurately. This improved accuracy is particularly noticeable in the solution components of dimension one. The labels refer to the transition points identified in Fig. 4.16.

table includes the results obtained when using the three loops and all their cyclic permutations that change the order of the variables. Again, we observe that the redundancy does not decrease the execution time, but it has a strong effect in minimizing the clustering effect. In the last experiment, the solution set is bounded with 6905 boxes, which is about the same number

	Basic non-redundant	**Redundant loop**	**Redundant loop and cyclic permutations (variables)**
Processed boxes	7980140	6537976	1022104
Box reductions	8915118	7115126	1078703
Bisected boxes	3989814	3268732	510796
Empty boxes	3976104	3256931	504403
Solution boxes	14222	12313	6905
Verified solutions	8994	6490	6905
Execution time [s]	32	31	33

Table 5.4: Performance of the trapezoid method without and with different sources of redundancy for the Tripteron parallel robot. See the text for details.

of boxes returned when using the LP method. Moreover, all of them are verified to include a solution point. Therefore, in this case, the trapezoid method with redundancy is completely free of clustering.

Summarizing, in this chapter we have presented several methods to generate multi-affine, redundant equations. When considering such redundant equations, the clustering effect, one of the principal drawbacks of the trapezoid method, is consistently mitigated. The effect on the overall execution time is more problem dependent, but in many cases the improvement in the accuracy in bounding the solution set is obtained with no extra computational cost or even reducing it. In these cases, the additional burden of processing more equations is compensated by the additional pruning provided by the redundant equations.

6 Multi-affine elimination

In this chapter we show how we can eliminate variables and equations of the systems of obtained with the formulation described in Chapter 2 while still keeping the multi-affinity of the system. Therefore, the resulting systems can still be managed by the solved described in Chapter 3, but with significant speed ups. We exemplify such speed ups in the inverse kinematics of general 6R loops and on several parallel robots with challenging direct kinematics.

Elimination methods has been used to produce outstanding results in position analysis. The idea behind these methods is to reduce the number of equation and variables transforming the initial set of equations into a minimum degree polynomial in a single variable. Typically, every elimination step reduces the number of variables, but increases the degree of the new equations. So, if we apply a typical elimination procedure to the equations formulated in Chapter 2, we will generate equations of high degree and, thus, not adequate for the solving method described in Chapter 3. However, we next present an elimination procedure that keeps the multi-affinity of the system of equations. Since the computational cost of the trapezoid method scales with $O(m\ n\ 2^n)$, with m and n the number of equations and variables in the system (see 3.2.2), reducing the number of equations and, specially, the number of variables results in significant speed ups in the process of each candidate box.

We next describe the proposed elimination procedure in a 6R loop, although the procedure can in principle be applied to any loop equation.

6.1 Variable elimination

Typically, elimination processes involve the generation of redundant equations and the use of linear algebra tools (e.g., determinants) on matrices including variables to generate resultant equations. Herein, we will use the same tools, but on matrices of constant coefficients so that the multi-affinity of the manipulated equations is preserved.

Using the notation in Chapter 2, the closure condition of a 6R loop can be formalized as

$$\mathbf{s_z}(t_1, d_1)\, \mathbf{s_x}(u_1, a_1) \cdots \mathbf{s_z}(t_6, d_6)\, \mathbf{s_x}(u_6, a_6)\, \mathbf{h} = c, \tag{6.1}$$

where

$$\mathbf{s_z}(t_i, d_i) = \left(1 + t_i\,\mathrm{k} + \varepsilon\,\frac{d_i}{2}\,(-t_i + \mathrm{k})\right), \tag{6.2}$$

and

$$\mathbf{s_x}(u_i, a_i) = \left(1 + u_i\,\mathrm{i} + \varepsilon\,\frac{a_i}{2}\,(-u_i + \mathrm{i})\right). \tag{6.3}$$

and where $\mathbf{h}$ specifies the pose of the base with respect to the end-effector. Since

$$\mathbf{s_z}(t_i, d_i) = \mathbf{r_z}(t_i)\, \mathbf{t_z}(d_i), \tag{6.4}$$

Eq. (6.1) can be re-arranged as

$$\mathbf{t_z}(d_6)\, \mathbf{s_x}(u_6, a_6)\, \mathbf{h}\, \mathbf{s_z}(t_1, d_1)\, \mathbf{s_x}(u_1, a_1) \cdots \mathbf{s_z}(t_5, d_5)\, \mathbf{s_x}(u_5, a_5) = c \begin{pmatrix} 1 \\ 0 \\ 0 \\ -t_6 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}, \tag{6.5}$$

where the second, third, fifth, sixth, seventh, and eighth equations define a system of six equations involving only 5 variables, $t_1, \ldots, t_5$. Thus, using them, t_6 is effectively eliminated from the problem. Since the resulting system still has more equations than variables, we can eliminate one more variable to obtain a simpler system of four equations in four unknowns. To this end, we generalize the previous straightforward algebraic procedure. First, Eq. (6.1) is re-arranged as

$$\mathbf{h}\,\mathbf{s}_\mathbf{z}(t_1,d_1)\,\mathbf{s}_\mathbf{x}(u_1,a_1)\cdots\mathbf{s}_\mathbf{z}(t_4,d_4)\,\mathbf{s}_\mathbf{x}(u_4,a_4) = \\ c\,\mathbf{s}_\mathbf{x}(-u_5,-a_5)\,\mathbf{s}_\mathbf{z}(-t_5,-d_5)\,\mathbf{s}_\mathbf{x}(-u_6,-a_6)\,\mathbf{s}_\mathbf{z}(-t_6,-d_6), \tag{6.6}$$

where the left-hand side involves four variables and the right-hand side only two. Since both sides of the equation define multi-linear polynomials, we can represent Eq. (6.6) as

$$\mathbf{C}_l\,\mathbf{t}_l = c\,\mathbf{C}_r\,\mathbf{t}_r, \tag{6.7}$$

where $\mathbf{C}_l \in \mathbb{R}^{8\times16}$ and $\mathbf{C}_r \in \mathbb{R}^{8\times4}$ are matrices of constant coefficient and $\mathbf{t}_l \in \mathbb{R}^{16}$ and $\mathbf{t}_r \in \mathbb{R}^{4}$ are vectors containing all the monomials involving the variables in the left- and right-hand side, respectively, i.e.,

$$\mathbf{t}_l = (1, t_1, t_2, t_1\,t_2, t_3, t_1\,t_3, t_2\,t_3, t_1\,t_2\,t_3, \\ t_4, t_1\,t_4, t_2\,t_4, t_1\,t_2\,t_4, t_3\,t_4, t_1\,t_3\,t_4, t_2\,t_3\,t_4, t_1\,t_2\,t_3\,t_4)^T, \tag{6.8}$$

$$\mathbf{t}_r = (1, t_5, t_6, t_5\,t_6)^T. \tag{6.9}$$

Matrix $\mathbf{C}_r$ can be decomposed as

$$\mathbf{C}_r = \mathbf{Q}\begin{bmatrix}\mathbf{R}\\ \mathbf{0}\end{bmatrix}, \tag{6.10}$$

with $\mathbf{Q} \in \mathbb{R}^{8\times8}$ an orthonormal matrix, i.e., a matrix such that $\mathbf{Q}^T\,\mathbf{Q} = \mathbf{I}$, and $\mathbf{R} \in \mathbb{R}^{4\times4}$ a upper triangular matrix. Therefore, we can write

$$\mathbf{Q}^T\,\mathbf{C}_l\,\mathbf{t}_l = \begin{bmatrix}\mathbf{R}\\ \mathbf{0}\end{bmatrix}\mathbf{t}_r, \tag{6.11}$$

which can be split in two matrix equations

$$\mathbf{Q}_1^T\,\mathbf{C}_l\,\mathbf{t}_l = c\,\mathbf{R}\,\mathbf{t}_r, \tag{6.12}$$

$$\mathbf{Q}_2^T\,\mathbf{C}_l\,\mathbf{t}_l = \mathbf{0}, \tag{6.13}$$

where $\mathbf{Q}_1$ and $\mathbf{Q}_2$ include, respectively, the first and last four columns of $\mathbf{Q}$. Matrix $\mathbf{Q}_2$ encodes the Gaussian elimination of the variables in $\mathbf{t}_r$ from the last four equations in the system and, thus, Eq. (6.13) defines sought after system of four equations in four unknowns, t_1, t_2, t_3, and t_4.

When the 6R chain includes a wrist, we can go one step further and eliminate the three rotation variables with co-punctual rotation axis. In this case, the loop in Eq.(6.1) can be split as

$$\begin{aligned}&\mathbf{h}\,\mathbf{s_z}(t_1,d_1)\,\mathbf{s_x}(u_1,a_1)\,\mathbf{s_z}(t_2,d_2)\,\mathbf{s_x}(u_2,a_2)\,\mathbf{s_z}(t_3,d_3)\,\mathbf{s_x}(u_3,a_3)=\\&c\,\mathbf{s_x}(-u_6,-a_6)\,\mathbf{s_z}(-t_6,-d_6)\,\mathbf{s_x}(-u_5,-a_5)\,\mathbf{s_z}(-t_5,-d_5)\,\mathbf{s_x}(-u_4,-a_4)\,\mathbf{s_z}(-t_4,-d_4),\end{aligned} \tag{6.14}$$

where we assume that t_4, t_5, and t_6 form the wrist. This equation can be rewritten as

$$\mathbf{C}_l\,\mathbf{t}_l = c\,\mathbf{C}_r\,\mathbf{t}_r, \tag{6.15}$$

where, in this case, $\mathbf{C}_l \in \mathbb{R}^{8\times 8}$, $\mathbf{C}_r \in \mathbb{R}^{8\times 8}$, and

$$\mathbf{t}_l = (1, t_1, t_2, t_1\,t_2, t_3, t_1\,t_3, t_2\,t_3, t_1\,t_2\,t_3)^T, \tag{6.16}$$

$$\mathbf{t}_r = (1, t_4, t_5, t_4\,t_5, t_6, t_4\,t_6, t_5\,t_6, t_4\,t_5\,t_6)^T. \tag{6.17}$$

The decomposition of $\mathbf{C}_r$ as

$$\mathbf{C}_r = \mathbf{Q}\begin{bmatrix}\mathbf{R}\\ \mathbf{0}\end{bmatrix}, \tag{6.18}$$

gives an orthonormal matrix $\mathbf{Q} \in \mathbb{R}^{8\times 8}$ and an upper triangular matrix $\mathbf{R} \in \mathbb{R}^{4\times 8}$. In this case, Eq. (6.13) defines a system of equations involving only the three non-eliminated variables, t_1,t_2, and t_3.

Independently of the number of eliminated variables, since Eq. (6.1) defines necessary conditions for the problem at hand, Eq. 6.13 also defines necessary conditions for this problem, but involving only the unknowns defining $\mathbf{t}_l$. Depending on the parameters of the system, the simpler necessary condition may have the same solution set as the original ones or they may have a larger solution set.

If all the variables in the original problem are present in the reduced, redundant system, the trapezoid method already takes care of propagating the ranges between subsets with shared variables. If some variables are not present in the reduced system we can use the original system, but only on the solution boxes identified with the reduced system. Alternatively, when removing two variables, Eq. (6.12) can be used to determine the monomials in $\mathbf{t}_r$ from those in $\mathbf{t}_l$ as

$$\mathbf{t}_r = \frac{1}{c}\,\mathbf{R}^{-1}\,\mathbf{Q}_1^T\,\mathbf{C}_r\,\mathbf{t}_l, \tag{6.19}$$

assuming that matrix $\mathbf{R}$ is full rank. Actually, there is no need to compute the scalar c since we can compute

$$\tilde{\mathbf{t}}_r = \mathbf{R}^{-1}\,\mathbf{Q}_1^T\,\mathbf{C}_r\,\mathbf{t}_l \tag{6.20}$$

	Trapezoid Standard	**Trapezoid Elimination**
Processed boxes	20270	338
Box reductions	27698	771
Bisected boxes	10103	161
Empty boxes	10149	156
Solution boxes	18	21
Verified solutions	16	16
Execution time [s]	0.035	0.0004

Table 6.1: Performance of the trapezoid method with and without elimination for a 6R loop with 16 solutions.

from where

$$\mathbf{t}_r = \frac{\tilde{\mathbf{t}}_r}{\tilde{\mathbf{t}}_{r,1}} \tag{6.21}$$

with $\tilde{\mathbf{t}}_{r,1}$ the first component of $\tilde{\mathbf{t}}_r$.

Note that the solver presented in Chapter 3 bounds the solutions with arbitrarily small ranges in each of the variables in the problem. Consequently, Eq. (6.20) has to be evaluated using interval arithmetic [110] to determine valid bounds for the variables in $\mathbf{t}_r$. Such evaluation would over-estimate the bounds due to the dependencies between the terms intervening in Eq. (6.20). If more accuracy is necessary, affine arithmetic [37] may be used to evaluate this expression.

When removing three variables, though, Eq. (6.12) defines a under-determined linear system where the is not a one-to-one relation between $\mathbf{t}_l$ and $\mathbf{t}_r$. Thus, in this case, the interval arithmetic procedure just described can not be used to recover the ranges for the eliminated variables.

Summarizing, the elimination procedure just described can be used to obtain a new set of necessary conditions in a reduced set of variables. In some cases, these necessary conditions are enough to solve the problem since they have the same solution set as the initial problem. In others, they define larger solution sets that fully include the sought after solutions. Thus, considering the equations resulting from the elimination might not be enough to solve the problem at hand. However, as we have seen in Chapter 5, redundant, necessary conditions can be combined to mitigate this issue.

	Trapezoid Standard	**Trapezoid Elimination**	
		Phase 1	Phase 2
Processed boxes	45802	196	306
Box reductions	74481	416	541
Bisected boxes	22869	94	101
Empty boxes	22277	89	168
Solution boxes	656	13	37
Verified solutions	8	0	6
Execution time [s]	0.098	0.0002	0.001

Table 6.2: Performance of the trapezoid method with and without elimination for the *bad_eg0* example from [100].

6.2 Examples

The elimination procedure just described will be illustrated in the two 6R loops with zero-dimensional solution sets described in Section 4.2 and in several parallel robots with challenging kinematics.

6.2.1 Elimination in 6R loops

The first considered 6R loop has a general structure and 16 solutions, and the second one includes a wrist and it only has 8 solutions. Their DH parameters are given in Tables 4.4 and 4.6, respectively.

Table 6.1 compares the performance of the trapezoid method with and without the elimination procedure. In the reported experiment variables t_1 and t_6 are eliminated and the problem is solved in variables $t_2, \ldots, t_5$. This results in a significant reduction in the execution time. Actually, with this elimination, the problem is solved faster than with specialized algorithms such as the Manocha-Canny method described in Section 4.2.1. Variables $t_2, \ldots t_5$ are bounded with intervals whose size is below σ which is set to 10^{-4}. The range propagation described in the previous section allows bounding the valid values for variables t_1 and t_6 with ranges whose size is typically below 10^{-2}. As mentioned even more accurate ranges would be obtained using affine arithmetic.

Different results can be obtained depending on the pair of eliminated variables. In this particular problem, for instance, eliminating rotation axis 5 and 6 defines a system of necessary conditions with a one-dimensional solution set. In this case, redundancy may be used to identify the sought after solution points of the original problem. Redundancy can be obtained cyclically permuting the loops, as described in Section 5.2. In this case, a different pair of variables can be eliminated from each redundant loop equation. As a consequence, the problem is formalized with a larger set of equations, where each equation only involves four variables. From the point of view of the computational cost, removing variables and adding redundancy we reduce the exponential factor (2^n with n the number of variables in the equation), but increase the linear factor (the number of equations m). The obtained results show that this strategy pays off in general.

In the second considered 6R, i.e., the one with 8 solutions, we can take advantage of the fact that the three last axes are copunctual (i.e., they form an spherical wrist) to eliminate three variables. In this case, the problem has to be solved in two phases. In the first one we consider the reduced system of three equation involving only three variables. In the second one the solutions identified in the first phase are considered with the full set of variables and equations. Table 6.2 gives the performance statistics for each one of these two phases and as well as those for the standard trapezoid method directly applied to the original problem. Overall, using the elimination procedure the problem can be solved in about 1 ms, i.e., about two orders of magnitude faster than with the standard trapezoid method. Moreover, in this particular testbed, the elimination procedure reduces the clustering effect.

6.2.2 Elimination in parallel robots

The formulation of a parallel robot using the formalism described in Chapter 2 typically involves a large number of variables. Therefore, their position analysis is harder when addressed using the trapezoid method. Thus, in this kind of robots, the role of variable elimination is crucial. We illustrate the performance improvements achieved with this procedure in three parallel platforms: the Tripteron, the 3U$\underline{\text{P}}$U and the 3R$\underline{\text{P}}$S.

The Tripteron parallel platform

This parallel platform has already been described in Section 4.5, where the trapezoid method has been shown to be able to solve its direct kinematic, but with a relatively low quality (i.e., with a high degree of clustering) and less efficiently than alternative methods. In particular the trapezoid method returned more than twice solution boxes than the LP method and duplicated the execution time of the LR method. In Section 5.4 we have shown that the the quality of the approximation of the solution set can be clearly improved using redundant equations. Actually, we have shown that, in this particular robot, the clustering effect can be completely canceled. Next, we show that using variable elimination the computational time taken by the trapezoid method can be drastically reduced, outperforming alternative methods.

With the non-redundant formulation, this problems has two loop equations. Two variables

	Standard	Variable elimination	Variable elimination and three loops	Variable elimination, three loops, and cyclic permutations (variables)
Processed boxes	7980140	432762	68296	43902
Box reductions	8915118	663201	122664	62119
Bisected boxes	3989814	216349	34116	21695
Empty boxes	3976104	120908	11392	14777
Solution boxes	14222	95505	22788	7430
Verified solutions	8994	5605	8233	7368
Execution time [s]	32	1.1	0.3	0.9

Table 6.3: Performance of the trapezoid method without and with variable elimination and redundancy for the Tripteron parallel robot.

can be eliminated from each one of them. Assuming that we eliminate a different pair of variables for each loop, the total number of variables would be reduced form ten to six.

Note, however, that eliminating the two parallel rotation axis of each leg would produce a system of necessary equations with a 2D solution set instead of the 1D expected one. To avoid this issue, we cyclically permute the loop equations using the procedure described in Section 5.2 so that the left-hand side of Eq. (6.6) includes pairs of variables from different legs.

Table 6.3 shows the results obtained eliminating variables and compares them with the results using the trapezoid method in the full problem. Clearly, thanks to the variable elimination the execution time of the trapezoid method reduces from 32 seconds to just 1.1, outperforming alternative methods. However, this improvement comes at the cost of exacerbating the clustering effect. As described in the previous chapter, redundancy can be used to palliate this negative side effect. Thus, Table 6.3, includes the results when using a redundant loop equation. In this case, the clustering significantly reduces and the execution time is even lower than before since the problem is solved in merely 0.3 seconds. To further reduce the clustering effect, we can add the redundant equations obtained by cyclically permuting loops. In this case the execution time is still below one second and the solution is almost free of clustering.

Thanks to the efficiency improvements obtained with the proposed variable elimination procedure, the trapezoid method can now be applied to more complex parallel robots, as the one described next.

The 3-U$\underline{\text{P}}$U parallel platform

The 3-U$\underline{\text{P}}$U is a widely studied parallel robot consisting of a fixed base and moving platform connected by three serial chains, or legs, each of them having a universal-prismatic-universal joint arranged in sequence. Fig. 6.1 shows one of these legs. The universal joints are passive

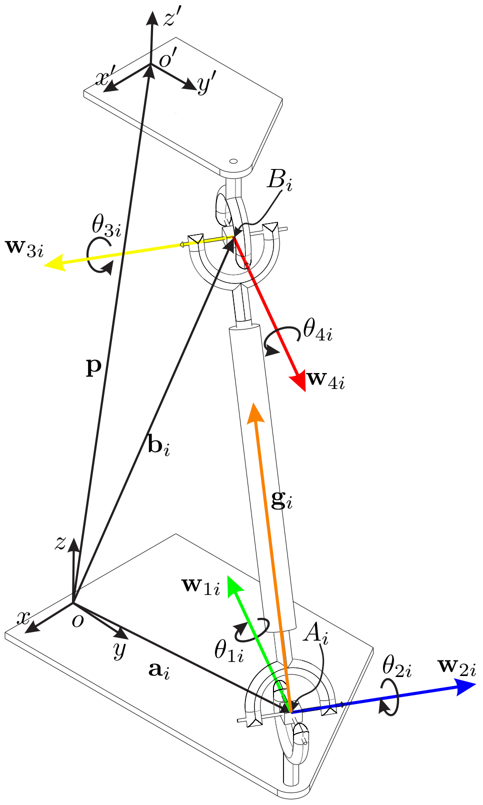


Figure 6.1: Notation associated with the i^{th} leg of a general 3-U$\underline{\text{P}}$U robot.

and only the prismatic joint are actuated.

With reference to Fig. 6.1, $\mathbf{w}_{1i}$ and $\mathbf{w}_{2i}$ are two mutually orthogonal unit vectors defined by the revolute axes of the universal joint centered at A_i. Likewise, $\mathbf{w}_{3i}$ and $\mathbf{w}_{4i}$ are the two mutually orthogonal unit vectors of the axes of the two revolute pairs constituting the universal joint centered at B_i. $\mathbf{a}_i$ and $\mathbf{b}_i$ are the position vectors of A_i and B_i, respectively, in a generic Cartesian reference fixed to the base, whereas $\mathbf{p}$ is the position vector of the origin, O', of the reference frame associated with the moving platform. θ_{ji}, $j = 1, \ldots, 4$, is a joint variable denoting a rotation angle around the joint-axis defined by $\mathbf{w}_{ji}$, $j = 1, \ldots, 4$, using the right-hand rule. The length of the $i - th$ leg is equal to $\|\mathbf{b}_i - \mathbf{a}_i\|$, and it will be denoted l_i. Moreover, we define

$$\begin{aligned}
\mathbf{g}_i &= (\mathbf{b}_i - \mathbf{a}_i)/l_i, \\
\mathbf{h}_i &= \mathbf{w}_{3i} \times \mathbf{w}_{4i}, \\
\mathbf{r}_i &= \mathbf{w}_{1i} \times \mathbf{w}_{2i}, \\
\mathbf{s}_i &= \mathbf{h}_i \times \mathbf{r}_i - [\mathbf{g}_i \cdot (\mathbf{h}_i \times \mathbf{r}_i)]\mathbf{g}_i.
\end{aligned}$$

Observe that $\mathbf{s}_i$ is just the component of $\mathbf{h}_i \times \mathbf{r}_i$ perpendicular to $\mathbf{g}_i$.

In 1996, Tsai proposed a 3-U$\underline{\text{P}}$U parallel robot with three translational degrees of freedom in [177]. The axes of the universal joints of this particular robot, henceforth called Tsai manipulator, are arranged as follows (see Fig. 6.2a):

(a1) the axes of the three revolute joints embedded in the base/platform (shown in green/red Fig. 6.2a) form a triangle.

(a2) the two triangles are similar.

(a3) for each leg, the axes of the intermediate revolute pairs are parallel to each other and perpendicular to the axis of the prismatic pair.

The sensitivity of this robot to geometric parameter variations and manufacturing tolerances was analyzed in [122], where it was shown that small torsions in the legs generate large deviations in the position of the moving platform. Therefore, applications of the Tsai's robot are limited by this pseudo-singular behavior. The sensitivity of this robot to other manufacturing errors is studied in [28,61]. Di Gregorio studied its singularities in [42]. The same analysis was later preformed by Joshi and Tsai in [73] using screw calculus.

In 1998, Di Gregorio and Parenti-Castelli [41] studied the more general 3-RRPRR architecture and, from this analysis, they arrived at the important conclusion that the geometric conditions for a 3-U$\underline{\text{P}}$U robot to have three translational DOFs can algebraically be expressed as:

(b1) $|\mathbf{w}_{1,1} \cdot \mathbf{w}_{1,2}| = |\mathbf{w}_{4,1} \cdot \mathbf{w}_{4,2}|$.

(b2) $|\mathbf{w}_{1,1} \cdot \mathbf{w}_{1,3}| = |\mathbf{w}_{4,1} \cdot \mathbf{w}_{4,3}|$.

(b3) $|\mathbf{w}_{1,2} \cdot \mathbf{w}_{1,3}| = |\mathbf{w}_{4,2} \cdot \mathbf{w}_{4,3}|$.

(b4) $\mathbf{w}_{2,i} = \pm\mathbf{w}_{3,i},\ i = 1,2,3$.

(b5) $\mathbf{w}_{1,i} = \pm\mathbf{w}_{4,i},\ i = 1,2,3$.

Another important conclusion in [41] is that the pure translation of the moving platform does not only depend on the leg topology, but also on specific mounting conditions. In this sense, while the above conditions (b1), (b2), (b3), and (b4) are *manufacturing conditions*, (b5) is a *mounting condition*.

As a result of this analysis, Tsai's robot can be seen as a particular case of a large family of 3-U$\underline{\text{P}}$U translational robots. Another particular translational 3-U$\underline{\text{P}}$U robot results if all the revolute-pair axes at the leg endings converge, while remaining coplanar, toward a single point and every leg has the two intermediate revolute-pair axes parallel to each other and perpendicular to the

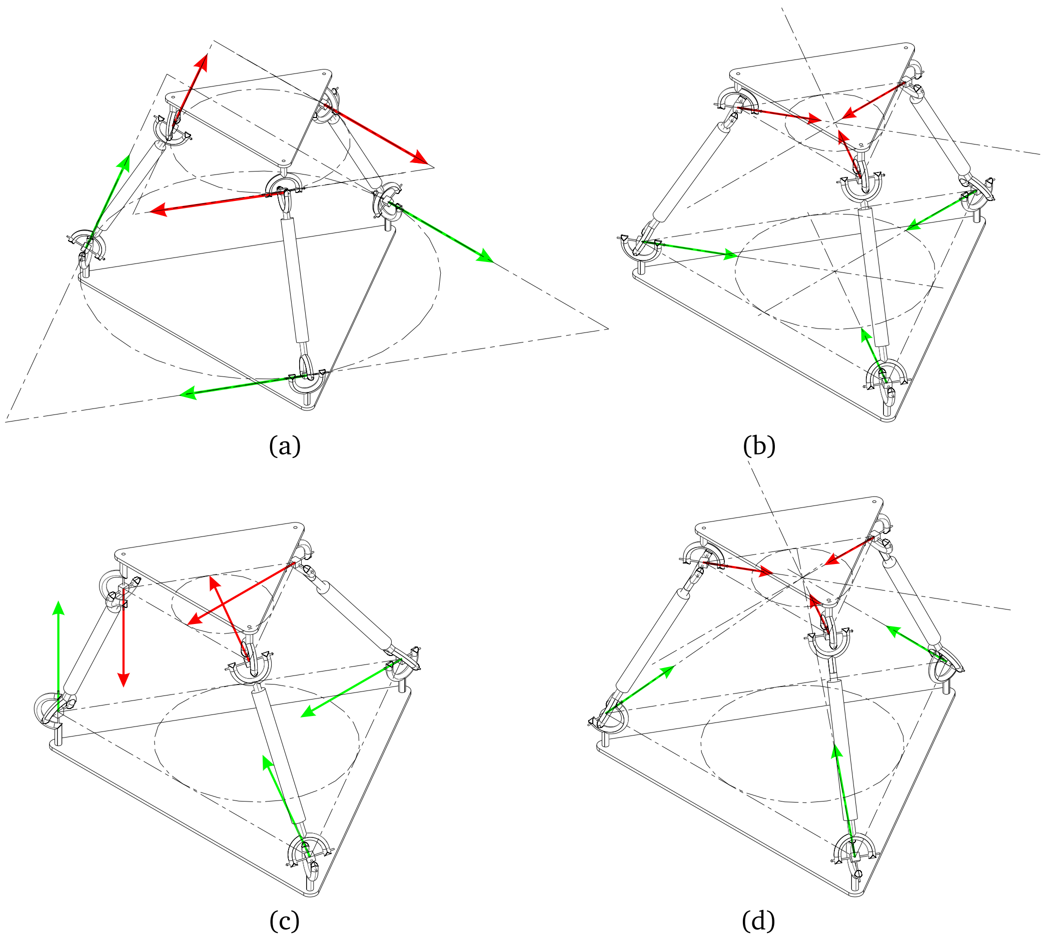


Figure 6.2: Four 3-U$\underline{P}$U robots: (a) Tsai's robot, (b) central robot, (c) Lu-Hu's robot, and (d) Hervé's robot.

straight line through the universal joint centers (see Fig. 6.2b). This particular 3-U$\underline{P}$U robot, which we will call *central robot*, was studied in [187] and [30]. In [187], Walter *et al.* showed that the translational motion of this robot is rather doubtful due to the presence of at least 16 different assembly modes including the pure translational one. Thus, it is important to highlight that for a given set of leg lengths a translational 3-U$\underline{P}$U manipulator have, in general, different assembly modes and, only if the platform is properly assembled, it can have a pure translational motion.

In 2006, Lu and Hu proposed a family of asymmetrical 3-U$\underline{P}$U robots [98]. This family of robots included a translational design (see Fig. 6.2c). Lu and Hu argued that, contrarily to what happens with the above two symmetrical designs, condition (b5) is easier to satisfy due to the peculiar joint disposal of their design, thus concluding that it provides a significant advantage with respect to Tsai's robot.

At this point, we have three 3-U$\underline{P}$U robots with identical pure translational DOFs and an identical actuator arrangement. Nevertheless, they necessarily differ in terms of singularity configuration and stiffness due to the different arrangement of their universal joints. For example, as it is proved in [67], Tsai's robot has a singularity plane and a singularity cylindrical surface, while Lu-Hu's robot has two singularity planes.

Observe that condition (b5) means that the axes defined by the first and the fourth revolute axes in each leg should be parallel, and, if this condition is satisfied, (b1), (b2) and (b3) are also satisfied. Thus, the geometric conditions for a 3-U$\underline{P}$U robot to have three translational DOFs can be simply expressed as the conjunction of (b4) and (b5). There is no need that the axes from different legs intersect in finite points. Actually, these unnecessary extra geometric constraints seem to be the ultimate reason for the poor behavior of Tsai's and the central 3-U$\underline{P}$U robots.

In 2000, Karouia and Hervé showed that a 3-U$\underline{P}$U robot, under some mounting and manufacturing conditions, can provide its moving platform with spherical motions [78]. These conditions are as follows (see Fig. 6.2d):

(c1) The three revolute pairs axes fixed to the platform (base) must converge at a point fixed in the platform (base).

(c2) In each leg, the intermediate revolute pair axes must be parallel to each other and perpendicular to the leg axis which is the line through the universal joints' centers.

(c3) The point located at the intersection of the platform's revolute pair axes must coincide with the point located at the intersection of the base's revolute pair axes.

In this case, (c1) and (c2) are manufacturing conditions, and (c3) is a mounting condition. Different aspects of the kinematics of this robot were studied in [39, 40, 178].

Finally, in [149] a 3-U$\underline{P}$U robot is presented that can be reconfigured to work either as a translational or as a rotational robot by simply flipping upside down its moving platform.

Despite all these research efforts, the direct kinematics of the 3-U$\underline{P}$U is not yet available in closed form. Thus, numerical methods as the one presented in this thesis have a fundamental role in this problem.

In the 3-U$\underline{P}$U, eliminating a pair of variables appearing consecutively in the loop equations produces a higher-dimensional solution set in almost all cases. However, eliminating different pairs of variables defines solution sets that only intersect in the solution points of the original problem. Thus, the redundant equations described in Section 5.2 can be used to generate equations where variables appear in different order. Then, we can use the procedure described in Section 6.1 to eliminate different pairs of variables. In this way, redundancy can be used to

	Trapezoid	**Trapezoid**
	Standard	**Elimination**
Processed boxes	36296674	122686
Box reductions	51169733	165651
Bisected boxes	18148305	61311
Empty boxes	18124818	61130
Solution boxes	23551	245
Verified solutions	4	4
Execution time [s]	893	3.9

Table 6.4: Performance of the trapezoid method with and without elimination on the Tsai's 3-U$\underline{\text{P}}$U parallel robot.

	Trapezoid	**Trapezoid**
	Standard	**Elimination**
Processed boxes	28741056	95380
Box reductions	39988171	127165
Bisected boxes	14370496	47658
Empty boxes	14361082	47597
Solution boxes	9478	125
Verified solutions	8	8
Execution time [s]	645	2.6

Table 6.5: Performance of the trapezoid method with and without elimination on the central 3-U$\underline{\text{P}}$U parallel robot.

complement the proposed variable elimination procedure. Table 6.4 shows that using this strategy the problem is solved more than two orders of magnitude faster than when considering the original system. As shown in tables 6.5, 6.6, and 6.7, similar performance improvements are obtained in the central, the Lu-Hu's and the Hervé's 3-U$\underline{\text{P}}$U variants.

The 3-R$\underline{\text{P}}$S parallel platform

The 3-R$\underline{\text{P}}$S is a parallel platform proposed by [70] that is composed of three identical legs connecting the base and the platform. Each leg includes a revolute joint, a prismatic actuated joint, and a spherical joint.

The 3-R$\underline{\text{P}}$S has been used in telescope applications [21], in milling machines [66], and in human-machine interfaces in the context of medical applications [183]. Its kinematic analysis including the number of solutions of its direct kinematics [115] or its self-motions [153] has been addressed using different formalisms such as screw theory [50, 69], quaternions [26], and the Study's kinematic mapping [154].

Different variants of this platform are obtained depending on the arrangement of the revolute

	Trapezoid Standard	**Trapezoid Elimination**
Processed boxes	45961484	105140
Box reductions	72005865	165735
Bisected boxes	22980764	52538
Empty boxes	22644137	51915
Solution boxes	336583	687
Verified solutions	9	9
Execution time [s]	2483	4.7

Table 6.6: Performance of the trapezoid method with and without elimination on the Lu-Hu's 3-U$\underline{\text{P}}$U parallel robot.

	Trapezoid Standard	**Trapezoid Elimination**
Processed boxes	30878080	86078
Box reductions	43763219	112465
Bisected boxes	15439008	43007
Empty boxes	15416668	43032
Solution boxes	22404	39
Verified solutions	4	4
Execution time [s]	628	1.9

Table 6.7: Performance of the trapezoid method with and without elimination on the Hervé's 3-U$\underline{\text{P}}$U parallel robot.

axis in the base. In the most popular designs, they are either tangential to a circle, parallel, or copunctual, but they can also be arranged arbitrarily. Whatever the configuration, a 3-R$\underline{\text{P}}$S parallel platform is fully characterized by the anchor points of the legs at the base and the platform and the director vectors of the axis of the revolute joints at the base. For our tests, we use the parameters given in [116, Section 5.3] which are

$$\begin{aligned}
\mathbf{a}_1 &= (0,0,0)^T, & \mathbf{b}_1 &= (0,0,0)^T, & \mathbf{w}_{1,1} &= (-3,5,0)^T,\\
\mathbf{a}_2 &= (\tfrac{18}{13},0,0)^T, & \mathbf{b}_2 &= (-\tfrac{135}{91},0,0)^T, & \mathbf{w}_{1,2} &= (-3,2,0)^T,\\
\mathbf{a}_3 &= (2,2,0)^T, & \mathbf{b}_3 &= (-\tfrac{12}{7},3,0)^T, & \mathbf{w}_{1,3} &= (2,1,0)^T,
\end{aligned}$$

where, using the same notation as in Fig. 6.1, $\mathbf{a}_i$ and $\mathbf{b}_i$ denote, respectively, the coordinates of the anchor points in the base and the platform in corresponding the local frame, and $\mathbf{w}_{1,i}$ is the (non-normalized) director vector of the rotation axis of the joints connecting the base and the i-th leg. Using the same convention, the leg lengths are denoted as l_1, l_2, and l_3.

As reported in Table 6.8, the process of solving the forward kinematic of this platform for $l_1 = 2$, $l_2 = 2$, and $l_3 = 3$ using the trapezoid method without any simplification takes more than 5 hours. However, using the procedure described at the end of Section 6.1 we can eliminate

	Trapezoid Standard	**Trapezoid Elimination**	
		Phase 1	Phase 2
Processed boxes	1525170926	14306	36380
Box reductions	2068753294	28067	54843
Bisected boxes	762585207	7137	14934
Empty boxes	762574255	6762	15993
Solution boxes	11464	407	5453
Verified solutions	8	0	8
Execution time [s]	18639	0.043	1.25

Table 6.8: Performance of the trapezoid method with and without elimination for the 3-R$\underline{\text{P}}$S parallel platform.

the spherical joints from each loop equations before solving the forward kinematics. In this case, the first phase (i.e., the one using the simplified system of equations) is solved in less than 0.05 seconds. The second phase (i.e, the one to recover the values of the eliminated variables using the full system of equations) takes about 1.25 seconds. Thus, the overall process is solved more than three orders of magnitude faster thanks to the use of the variable elimination procedure described in this chapter.

7 Conclusions

The multi-affinity property of the closure equations of multi-loop mechanisms has been exploited in this thesis to its latest consequences. As a result, we have derived a branch-and-prune method —which we have named *trapezoid method*— for solving sets of this kind of equations which is much simpler than all the related algorithms 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 multi-affine systems, *i.e.*, to closure equations in their simplest form.

We have presented a variety of examples including single-loop and multiple-loop mechanisms and, in all of them, the trapezoid method is up to two orders of magnitude faster that the alternative methods using a similar branch-and-prune approach. The obtained results for 6R closed loops with 0-dimensional solution sets are even comparable, in terms of performance, with a specialized numerical approach. In the case of higher-order sets, commonly known as self-motion manifolds, the obtained results provide good approximations for the strata of the highest dimension, but those of lower dimension are degraded by the clustering effect. In these cases, the trapezoid method benefits from the introduction of redundancy and elimination.

The simplicity of the proposed algorithm comes with three main disadvantages; namely:

- The use of a single equation at a time makes the convergence to the roots be linear while other standard interval-based algorithms exhibit quadratic convergence [159]. Although our algorithm compares unfavorably in terms of execution time, it should be borne in mind that we are actually facing a trade-off between cost of each iteration and convergence rate.

- The use of one equation at a time leads to the so-called clustering problem, that is, each solution is obtained as a cluster of boxes instead of a single box containing it [112]. All the solution boxes, though, are close to the actual solutions and the error (*i.e.*, the norm of the residue of the equations) evaluated at the center of the solution boxes is small in all cases. Indeed, if the effect of manufacturing inaccuracies and/or slight elasticities in the kinematic chain were incorporated as ranges in the coefficients of the closure conditions, clustering-free branch-and-prune methods would also reproduce these 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 the loss of rigidity. Nevertheless, the clustering effect can be mitigated taking into account redundant equations. In this thesis, we have shown how to generate multi-affine redundant equations and how they reduce the clustering effect to the point of completely eliminating it in some cases. Thus, we have shown that the use of multiple necessary conditions can produce the same result as when using necessary and sufficient conditions. The derivation of necessary conditions has been one of the central topics of research in numerical methods for position analysis. However, the results presented in this thesis call into question whether they are actually required at all.

- The approach is inherently exponential with the number of variables in the problem and it would sooner or later be unpractical as the problems become more complex. This should not came as a surprise since the position analysis problem is known to be NP-hard [141]. We have shown that the approach can successfully be applied to many problems of practical interest and we have presented elimination techniques which significantly extend the applicability of the method. However, to address even more complex problems we would need to explore alternative, more compact formulations. Otherwise the number of initial boxes to consider can be overwhelming. Note, though, that in practical applications the ranges of the joints are typically limited. For instance, the spherical joints used in parallel platforms have significantly small operational ranges. Thus, despite its theoretical limitation, the trapezoid method is a good alternative to solve complex, but realistic problems.

Thus, despite its limitations, we have presented an algorithm remarkable for its simplicity, efficiency, and easy parallelization.

Given the benefits of multi-affine formulations in the context of branch-and-bound methods, it seems worth exploring other numerical alternatives to solve multi-affine systems. For example, the method presented in [113] 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. It would be also interesting to derive the Gröbner basis of the systems of multi-affine equations since these basis pave the way for practical tools to solve the problem. Our preliminary results show that the redundant equations introduced in this work may facilitate the computation of such basis.

A
List of publications

The following is a list of publications related to the research reported in this thesis:

1. **A. Shabani**, S. Sarabandi, J. M. Porta and F. Thomas. A fast branch-and-prune algorithm for the position analysis of spherical mechanisms, 15th IFToMM World Congress on Mechanism and Machine Science, Krakow, Poland, in Advances in Mechanism and Machine Science, Vol 73 of Mechanism and Machine Science, pp. 549-558, Springer, 2019.

2. S. Sarabandi, **A. Shabani**, J. M. Porta and F. Thomas. On closed-form formulas for the 3D nearest rotation matrix problem. IEEE Transactions on Robotics, 36(4):1333-1339, 2020.

3. **A. Shabani**, J. M. Porta and F, Thomas. Solving closure equations in dual quaternions using a specific interval-based method, Mechanism and Machine Theory, In press, 2021.

References

[1] H. Albala and J. Angeles. Numerical solution to the input output displacement equation of the general 7R spatial mechanism. In *Proceedings of the Fifth world congress on theory of machines and mechanisms*, pages 1008–1011, 1979.

[2] G. E. Alefeld, A. Frommer, G. Heindl, and J. Mayer. On the existence theorems of Kantorovich, Miranda and Borsuk. *Electronic Transactions on Numerical Analysis*, 17:102–111, 2004.

[3] G. E. Alefeld, F. A. Potra, and Z. Shen. On the existence theorems of Kantorovich, Moore and Miranda. In *Topics in numerical analysis*, pages 21–28. Springer, 2001.

[4] E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen. *LAPACK Users' Guide*. Society for Industrial and Applied Mathematics, third edition, 1999.

[5] J. Angeles. *Rational Kinematics*. Springer, 1989.

[6] A. Angerer and M. Hofbaur. Industrial versatility of inverse kinematics algorithms for general 6R manipulators. In *International Conference on Advanced Robotics*, pages 1–7, 2013.

[7] N. A. Aspragathos and J. K. Dimitros. A comparative study of three methods for robot kinematics. *IEEE Transactions on Systems, Man, and Cybernetics*, 28(2):135–145, 1998.

[8] Z. Baigunchekovl and M. Kalimoldaev. Inverse kinematics of six-DOF three-limbed parallel manipulator. In A. Rodic and T. Borangiu, editors, *Advances in Robot Design and Intelligent Control*, pages 171–17, 2017.

[9] J. E. Baker. Displacement–closure equations of the unspecialised double-Hooke's-joint linkage. *Mechanism and Machine Theory*, 37(10):1127 – 1144, 2002.

[10] C. Belta and L. C. G. J. M. Habets. Controlling a class of nonlinear systems on rectangle. *IEEE transactions on Automatic Control*, 51(11):1749–1759, 2006.

[11] G. T. Bennett. A new mechanism. *Engineering*, 76:777–778, 1903.

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

[13] C. Bombín, L. Ros, and F. Thomas. On the computation of the direct kinematics of parallel spherical mechanisms using bernstein polynomials. In *IEEE International Conference on Robotics and Automation*, 2001.

[14] J. Borràs and R. Di Gregorio. Direct position analysis of a large family of spherical and planar parallel manipulators with four loops. In *International Workshop on Fundamental Issues and Future Research Directions for Parallel Mechanisms and Manipulators*, pages 19–29, 2008.

[15] R. Bricard. *Leçons de cinématique*, page 7–12. Gauthier-Villars, Paris, 1927.

[16] L. E. J. Brouwer. Über Abbildungen von Mannigfaltigkeiten. *Mathematische Annalen*, 71:97–115, 1911.

[17] K. Brunnthaler, H.-P. Schröcker, and M. Husty. A new method for the synthesis of bennet mechanisms. In *International Workshop on Computational Kinematics*, 2005.

[18] B. Buchberger. *An Algorithm for Finding the Basis Elements of the Residue Class Ring Modulo a Zero Dimensional Polynomial Ideal*. PhD thesis, Universitat Innsbruck, 1966.

[19] J. W. Burdick. On the inverse kinematics of redundant manipulators: Characterization of the self-motion manifolds. In K.J. Waldron, editor, *Advanced Robotics: 1989*, pages 25–34. Springer, Berlin, Heidelberg, 1974.

[20] Y. K. Byun and H. S. Cho. Analysis of a novel 6-DOF, 3-PPSP parallel manipulator. *The International Journal of Robotics Research*, 16(6):859–872, 1997.

[21] J. A. Carretero, M. Nahon, B. Buckham, and C. M. Gosselin. Kinematic analysis of a three-dof parallel mechanism for telescope applications. In *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, 1997.

[22] M. Carricato and V. Parenti-Castelli. A family of 3-DOF translational parallel manipulators. *Journal of Mechanical Design*, 125:302–307, 2003.

[23] A. Castellet. Towards an efficient interval method for solving inverse kinematic problems. In *Towards an efficient interval method for solving inverse kinematic problems*, volume IV, pages 615–3620, 1997.

[24] A. Castellet and F. Thomas. An algorithm for the solution of inverse kinematic problems based on an interval method. In *Advances in Robot Kinematics*, pages 393–403. Kluwer Academic Publishers, Dordrecht, 1998.

[25] E. Celaya. Interval propagation for solving parallel spherical mechanisms. In J. Lenarcic and F. Thomas, editors, *Advances in Robot Kinematics*. Springer, 2002.

[26] D. Chablat, R. Jha, F. Rouillier, and G. Moroz. Workspace and joint space analysis of the 3-RPS parallel robot. In *ASME International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, 2014.

[27] G. Chartrand and L. Lesniak. *Graphs and Digraphs*. Chapman and Hall, 3rd edition edition, 1996.

[28] A.-H. Chebbi, Z. Affi, and L. Romdhane. Prediction of the pose errors produced by joints clearance for a 3-UPU parallel robot. *Mechanism and Machine Theory*, 44(9):1768–1783, 2009.

[29] Y. Chen and Z. You. Spatial overconstrained linkages—The lost jade. In Teun Koetsier and Marco Ceccarelli, editors, *Explorations in the History of Machines and Mechanisms*, pages 535–550. Springer Netherlands, 2012.

[30] Z. Chen, Y. Zhang, K. Huang, H. Ding, and Z. Huang. Mobility and motion analysis of a special 3-UPU parallel mechanism. In *IFToMM World Congress*, pages 256–263, 2015.

[31] K. Cleary and M. Uebel. Jacobian formulation for a novel 6-dof parallel manipulator. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 1994.

[32] COIN. The COIN linear program code (CLP). https://github.com/coin, Last accessed 2020.

[33] C. L. Collins and G. L. Long. The singularity analysis of an in-parallel hand controller for force-reflected teleoperation. *IEEE Transactions on Robotics and Automation*, 11(5):661–669, 1995.

[34] D. Cox, J. Little, and D. O'Shea. *An Introduction to Computational Algebraic Geometry and Commutative Algebra*. Springer, 2nd edition edition, 1997.

[35] J. Craig. *Introduction to Robotics. Mechanics and Control*. Addison-Wesley Publishing Company, 1986.

[36] B. Dasgupta and T. Mruthyunjaya. The Stewart platform manipulator: a review. *Mechanism and Machine Theory*, 35:15–40, 2000.

[37] L. H. De Figueiredo and J. Stolfi. Affine arithmetic: concepts and applications. *Numerical Algorithms*, 37(1):147–158, 2004.

[38] E. Delassus. The closed and deformable linkage chains with four bars. *Bulletin of Science and Mathematics*, 46:283–304, 1922.

[39] R. Di Gregorio. Kinematics of the 3-UPU wrist. *Mechanism and Machine Theory*, 38(3):253–263, 2003.

[40] R. Di Gregorio. Statics and singularity loci of the 3-UPU wrist. *IEEE Transactions on Robotics*, 20(4):630–635, 2004.

[41] R. Di Gregorio and V. Parenti-Castelli. A translational 3-DOF parallel manipulator. In *Advances in Robot Kinematics: Analysis and Control*, pages 49–58, 1998.

[42] R. Di Gregorio and V. Parenti-Castelli. Influence of leg flexibility on the kinetostatic behaviour of a 3-DOF fully-parallel manipulator. In *World Congress on the Theory of Machines and Mechanisms*, volume 3, pages 1091–1098, 1999.

[43] P. Dietmaier. A new 6R space mechanism. In *World Congress on the Theory of Machines and Mechanisms*, volume 1, pages 52–56, 1995.

[44] J. Duffy and C. Crane. A displacement analysis of the general spatial 7-link, 7R mechanism. *Mechanism and Machine Theory*, 15(3):153–169, 1980.

[45] I. Z. Emiris. Toric resultants and applications to geometric modelling. In *Solving polynomial equations*, pages 269–300. Springer, 2005.

[46] Y. Fang and L. Tsai. Structure synthesis of a class of 4-DoF and 5-DoF parallel manipulators with identical limb structures. *The International Journal of Robotics Research*, 21(9):799–810, 2002.

[47] G. Farin. *Curves and Surfaces for Computer Aided Geometric Design - A Practical Guide*. Academic Press, 1990.

[48] I. S. Fischer. *Dual-Number Methods in Kinematics, Statics and Dynamics*. CRC Press, 1999.

[49] J. Funda, R. Taylor, and R. P. Pau. On homogeneous transforms, quaternions, and computational efficiency. *IEEE Transactions on Robotics and Automation*, 6:382–388, 1990.

[50] J. Gallardo, H. Orozco, and J. M. Rico. Kinematics of 3-RPS parallel manipulators by means of screw theory. *The International Journal of Advanced Manufacturing Technology*, 36(5-6):598–605, 2008.

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

[52] F. Gao, W. Li, X. Zhao, Z. Jin, and H. Zhao. New kinematic structures for 2-, 3-, 4-, and 5-DOF parallel manipulator designs. *Mechanism and Machine Theory*, 37:1395–1411, 2002.

[53] 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.

[54] C. B. Garcia and W. I. Zangwill. *Pathways to Solutions, Fixed Points, and Equilibria*. Prentice-Hall, 1981.

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

[56] M. Ghazvini. Reducing the inverse kinematics of manipulators to the solution of a generalized eigen problem. In Kovacs J. Angeles, G.Hommel, editor, *Computational Kinematics, Solid Mechanics and its Applications*, volume 28, pages 15–26, 1992.

[57] G. Gogu. *Structural Synthesis of Parallel Robots*, volume 149 of *Solid Mechanics and its Application*. Springer, 2008.

[58] M. Goldberg. New five-bar and six-bar linkages in three dimensions. *Transactions of the ASME*, 65:649, 1943.

[59] C. M. Gosselin and X. Kong. Cartesian parallel manipulators. US patent no. US6729202B2, 2011.

[60] W. R. Hamilton. On quaternions or a new system of imaginaries in algebra. *Philosophical Magazine*, 25:489–495, 1844.

[61] C. Han, J. Kim, J. Kim, and F. C. Park. Kinematic sensitivity analysis of the 3-UPU parallel mechanism. *Mechanism and Machine Theory*, 37(8):787–798, 2002.

[62] J. Hauenstein, A. Sommese, and C. Wampler. Regeneration homotopies for solving systems of polynomials. *Mathematics of Computation*, 80(273):345–377, 2011.

[63] K. Hauser. Learning the problem-optimum map: Analysis and application to global optimization in robotics. *IEEE Transactions on Robotics*, 33(1):141–152, 2017.

[64] X. He, X. Kong, D. Chablat, S. Caro, and G. Hao. Kinematic analysis of a single-loop reconfigurable 7R mechanism with multiple operation modes. *Robotica*, 32:1171–1188, 2014.

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

[66] A. Hernández, O. Altuzarra, C. Pinto, and E. Amezua. Transitions in the velocity pattern of lower mobility parallel manipulators. *Mechanism and machine theory*, 43(6):738–753, 2008.

[67] B. Hu, Y. Yao, P. Wu, and Y. Lu. A comparison study of two 3-upu translational parallel manipulators. *International Journal of Advanced Robotic Systems*, 10(4):190, 2013.

[68] C. Huang, X. Kong, and T. Ou. Position analysis of a Bennett-based multiple-mode 7R linkage. pages 1231–1236. ASME International Design Engineering Technical Conferences and Computers and Information in Engineering, 2009.

[69] Z. Huang and Y. Fang. Motion characteristics and rotational axis analysis of three DOF parallel robot mechanisms. In *IEEE International Conference on Systems, Man and Cybernetics*, volume 1, pages 67–71, 1995.

[70] K. H. Hunt. Structural kinematics of in-parallel-actuated robot arms. *ASME Journal of Mechanisms, Transmission and Automation in Design*, 105(2):705–712, 1983.

[71] M. Hunt, G. Mullineux, R. J. Cripps, and B. Cross. Characterizing isoclinic matrices and the cayley factorization. *ASME Journal of Mechanical Engineering*, 30:18, 2016.

[72] M. Husty, M. Pfurner, and H.-P. Schröcker. A new and efficient algorithm for the inverse kinematics of a general serial 6R manipulator. *Mechanism and Machine Theory*, 42(1):66–81, 2007.

[73] S. Joshi and L. Tsai. Jacobian analysis of limited-DOF parallel manipulators. *Journal of Mechanical Design*, 124(2):254–258, 2002.

[74] L. V. Kantorovich. On Newton's method for functional equations. *Proceedings of the USSR Academy of Sciences*, 59:1237–1240, 1948.

[75] D. Kapur and T. Saxena. Comparison of various multivariate resultant formulations. In *International Symposium on Symbolic and Algebraic Computation*, pages 187–194, 1995.

[76] A. Karger. Classification of 5R closed kinematic chains with self mobility. *Mechanism and Machine Theory*, 33(1/2):213–222, 1998.

[77] N. Karmarkar. A new polynomial-time algorithm for linear programming. In *Proceedings of the sixteenth annual ACM symposium on Theory of computing*, pages 302–311, 1984.

[78] M. Karouia and J. M. Hervé. A three-dof tripod for generating spherical rotation. In *Advances in Robot Kinematics*, pages 395–402, 2000.

[79] N. Katoh and S. Tanigawa. A proof of the molecular conjecture. *Discrete & Computational Geometry*, 45(4):647–700, 2011.

[80] R. B. Kearfott. *Rigorous Global Search: Continuous Problems*. Kluwer Academic Publishers, Dordrecht, 1996.

[81] D. Kohli and M. Osvatic. Inverse kinematics of general 6r and 5r,p serial manipulators. *ASME J. Mech. Design*, 115:922–931, 1993.

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

[83] R. Kolodny, L. Guibas, M. Levitt, and P. Koehl. Inverse kinematics in biology: the protein loop closure problem. *International Journal of Robotics Research*, 24:151–163, 2005.

[84] X. Kong. Reconfiguration analysis of multimode single-loop spatial mechanisms using dual quaternions. *Journal of Mechanisms and Robotics*, 9(5), 2017.

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

[86] X. Kong and C. M. Gosselin. Type synthesis of 3-dof spherical parallel manipulators based on screw theory. *ASME Journal Mechanical Design*, 126(1):101–108, 2004.

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

[88] KRD. KRD Research Group: A solver for systems of multi-afffine equations. http://www.iri.upc.edu/people/porta, Last accessed 2020.

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

[90] K. Kutzbach. Mechanische leitungsverzweigung maschinenbau. *Der Betrieb*, 8:710–716, 1929.

[91] G. Laman. On graphs and the rigidity of plane skeletal structures. *Journal of Engineering Mathematics*, 4(4):331–340, 1970.

[92] 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.

[93] E. Lee and C. Mavroidis. Solving the geometric design problem of spatial $3r$ robot manipulator using polynomial homotopy continuation. *ASME Journal of Mechanical Design*, 124(4):652–661, 2002.

[94] H.-Y. Lee and C.-G. Liang. Displacement analysis of the general spatial 7-link 7R mechanism. *Mechanism and Machine Theory*, 23(3):219–226, 1988.

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

[96] 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.

[97] Z. Li and 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*, 94:1–8, 2015.

[98] Y. Lu and B. Hu. Analysis of kinematics and solution of active/constrained forces of asymmetric 2UPU+X parallel manipulators. *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*, 220(12):1819–1830, 2006.

[99] Z. Lu, G. Yang, Q. Wang, and Q. Nan. Research on mechanism of actuated active reflectors for FAST. *Journal of Beijing University of Aeronautics and Astronautics*, 32:233–238, 2006.

[100] D. Manocha. GAMMA Research Group: Inverse kinematics code for serial manipulators. http://gamma.cs.unc.edu/software/downloads/IK/ik-1.0.tar.gz, Last accessed 2020.

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

[102] R. Manseur and K. L. Doty. A robot manipulator with 16 real inverse kinematic solution set. *International Journal of Robotics Research*, 8:75–79, 1989.

[103] C. Mavroidis and B. Roth. Analysis of overconstrained mechanisms. *Journal of Mechanical Design*, 117(1):69–74, 1995.

[104] J. C. Maxwell. On reciprocal figures and diagrams of forces. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 27(182):250–261, 1864.

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

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

[107] J.-P. Merlet. *Parallel Robots*. Springer, 2006.

[108] C. Miranda. Un'osservazione su un teorema di Brouwer. *Bollettino dell'Unione Matematica Italiana*, 2(3):5–7, 1941.

[109] R. Moore. A test for existence of solutions to nonlinear systems. *ACM Journal of Numerical Analysis*, 14:611–615, 1977.

[110] R. Moore, R. B. Kearfott, and M. J. Cloud. *Introduction to interval analysis*. Society for Industrial Mathematics, 2009.

[111] A. P. Morgan. A homotopy for solving polynomial systems. *Applied Mathematics and Computation*, 18(1):87–92, 1986.

[112] A. P. 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.

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

[114] F. E. Myard. Contribution a la geometrie des systemes articules. *Bulletin de la Societe Mathematique de France*, 59:183–210, 1931.

[115] P. Nanua, K. J. Waldron, and V. Murthy. Direct kinematic solution of a Stewart platform. *IEEE Transactions on Robotics and Automation*, 6(4):438–444, 1990.

[116] A. Nayak, T. Stigger, M. Husty, P. Wenger, and S. Caro. Operation mode analysis of 3-RPS parallel manipulators based on their design parameters. *Computer Aided Geometric Design*, 63:122–134, 2018.

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

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

[119] J. Ortega and W. Rheinboldt. *Iterative Solution of Nonlinear Equations in Several Variables*. Academic Press, 1970.

[120] J. Owen and S. Power. The nonsolvability by radicals of generic 3-connected planar laman graphs. *Transactions of the American Mathematical Society*, 359:2269–2303, 2007.

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

[122] V. Parenti-Castelli, R. Di Gregorio, and J. Lenarčič. Sensitivity to geometric parameter variation of a 3-dof fully-parallel manipulator. In *International Conference on Advanced Mechatronics*, pages 364–369, 1998.

[123] A. Perez and J. M. McCarthy. Dimensional synthesis of bennett linkages. *Journal of Mechanical Design*, 125(1):98–104, 2003.

[124] M. Pfurner. A new family of overconstrained 6R-mechanisms. In *EUCOMES 08*, 2009.

[125] M. Pfurner and X. Kong. Algebraic analysis of a new variable-DOF 7R mechanism. In P. Wenger and P. Flores, editors, *New Trends in Mechanism and Machine Science, Theory and Industrial Applications*, pages 71–79. Springer, 2016.

[126] Martin Pfurner and Xianwen Kong. Algebraic analysis of a new variable-DOF 7R mechanism. In *New Trends in Mechanism and Machine Science*, pages 71–79. Springer, 2017.

[127] J. Phillips. *Freedom in machinery: Volume 2, Screw theory exemplified*. Cambridge University Press, 1984.

[128] D. Pieper. *The Kinematics of Manipulators under Computer Control*. PhD thesis, Stanford University, 1968.

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

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

[131] J. M. Porta, L. Ros, and F. Thomas. A linear relaxation technique for the position analysis of multi-loop linkages. *IEEE Transactions on Robotics*, 25(2):225–140, 2009.

[132] 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*, pages 144–155, 2008.

[133] J. M. Porta, L. Ros, F. Thomas, and C. Torras. A branch-and-prune solver for distance constraints. *IEEE Transactions on Robotics*, 21:176–187, 2005.

[134] J. M. Porta, S. Sarabandi, and F. Thomas. Angle-bound smoothing with applications in kinematics. In *IFToMM Asian Mechanism and Machine Science*. Springer, 2018.

[135] N. Pouliot, C. M. Gosselin, and M. Nahon. Motion simulation capabilities of three-degree-of-freedom flight simulators. *Journal of Aircraft*, 35:9–17, 1998.

[136] E. J. Primrose. On the input-output equation of the general 7R-mechanism. *Mechanism and Machine Theory*, 21(6):509–510, 1986.

[137] S. Qiao, Q. Liao, S. Wei, and H.-J. Su. Inverse kinematic analysis of the general 6R serial manipulators based on double quaternions. *Mechanism and Machine Theory*, 45(2):193–199, 2010.

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

[139] A. Rikun. A convex envelope formula for multilinear functions. *Journal of Global Optimization*, 10(4):425–437, 1970.

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

[141] J. Rohn. Np-hardness results for linear algebraic problems with interval data. In *Topics on Validated Computation*, Studies in Computationl Mathematics, pages 463–471. 1994.

[142] N. Rojas. *Distance-Based Formulations for the Position Analysis of Kinematic Chains*. PhD thesis, Universitat Politècnica de Catalunya, 2012.

[143] N. Rojas and F. Thomas. Closed-form solution to the position analysis of watt-baranov trusses using the bilateration method. *ASME Journal of Mechanisms and Robotics*, 3:3, 2011.

[144] C. Rosales, J. M. Porta, R. Suárez, and L. Ros. Finding all valid hand configurations for a given precision grasp. In *IEEE International Conference on Robotics and Automation*, pages 1634–1640, 2008.

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

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

[147] B. Roth, J. Rastegar, and V. Scheinman. On the design of computer controlled manipulators. In J. Lenarcic and M.L. Husty, editors, *On Theory and Practice of Robots and Manipulators*, volume 201. Springer, Vienna, 1974.

[148] L. Rubbert, I. Charpentier, S. Henein, and P. Renaud. Higher-order continuation method for the rigid-body kinematic design of compliant mechanisms. *Journal of Precision Engineering*, 50:455–466, 2017.

[149] S. Sarabandi, P. Grosch, J. M. Porta, and F. Thomas. A reconfigurable asymmetric 3-UPU parallel robot. In *International Conference on Reconfigurable Mechanisms and Robots*, pages 1–8, 2018.

[150] S. Sarabandi, A. Shabani, J. M. Porta, and F. Thomas. On closed-form formulas for the 3-d nearest rotation matrix problem. *IEEE Transactions on Robotics*, 36(4):1333–1339, 2020.

[151] E. Sariyildiz, E. Cakiray, and H. Temeltas. A comparative study of three inverse kinematic methods of serial industrial robot manipulators in the screw theory framework. *International Joural of Advadnces Robotic System*, 8:9–24, 2011.

[152] P. T. Sarrus. Note sur la transformation des mouvements rectilignes alternatifs, en mouvements circulaires; et reciproquement. *Academie des sciences, comtes rendus hebdomataires des seances*, 36:1036–1038, 1853.

[153] J. Schadlbauer, M. Husty, S. Caro, and P. Wengery. Self-motions of 3-RPS manipulators. *Frontiers of Mechanical Engineering*, 8(1):62–69, 2013.

[154] J. Schadlbauer, D. R. Walter, and M. Husty. The 3-RPS parallel manipulator from an algebraic viewpoint. *Mechanism and Machine Theory*, 75:161–176, 2014.

[155] P. Schatz. Mechanism producing wavering and rotating movevements of receptacles. U.S. Pat. No. 2,302,804, 1942.

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

[157] J. M. Selig. Exponential and Cayley maps for dual quaternions. *Advances in Applied Clifford Algebras*, 20(3):923–936, 2010.

[158] A. Sharkawy and N. A. Aspragathos. Comparative study of two methods for forward kinematics and Jacobian matrix determination. In *International Conference on Mechanical, System and Control Engineering*, pages 179–183, 2017.

[159] E. Sherbrooke and N. Patrikalakis. Computation of the solutions of nonlinear polynomial systems. *Computer Aided Geometric Design*, 10:5, 1993.

[160] J. H. Shim, D. S. Kwon, and H. S. Cho. Kinematic analysis and design of a six D.O.F. 3-PRPS in-parallel manipulator. *Robotica*, 17(3):269–281, 1999.

[161] B. Siciliano. The Tricept robot: Inverse kinematics, manipulability analysis and closed-loop direct kinematics algorithm. *Robotica*, 17:437–445, 1999.

[162] B. Siciliano and O. Khatib. *Springer Handbook of Robotics*. Springer, 2008.

[163] S. Smale. Algorithms for solving equations. In *International Congress of Mathematicians*, pages 172–195, 1986.

[164] A. Sommese, J. Verschelde, and C. Wampler. Advances in polynomial continuation for solving problems in kinematics. *ASME Journal of Mechanical Design*, 126:262–268, March 2004.

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

[166] E. Study. Von den bewegungen und umlegungen. *Mathematische Annalen*, 39:441–565, 1891.

[167] B. Sturmfels. Solving systems of polynomial equations. *Journal of Computational Chemistry*, 97:152, 2002.

[168] B. Sturmfels. What is a Gröbner basis? *Notices of the American Mathematical Society*, 52(10):1199, 2005.

[169] H. D. Taghirad. *Parallel Robot. Mechanics and Control*. CRC Press, 2013.

[170] T.-S. Tay. Rigidity of multi-graphs, linking rigid bodies in n-space. *Journal Combinatorial Theory, B*, 36:95–112, 1984.

[171] R. Testylier and T. Dang. Analysis of parametric biological models with non-linear dynamics. *arXiv preprint arXiv:1208.3849*, 2012.

[172] 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*, volume I, pages 403–408, 1992.

[173] F. Thomas. Approaching dual quaternions from matrix algebra. *IEEE Transactions on Robotics*, 30(5):1037–1048, 2014.

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

[175] F. Thomas and A. Benito Martínez. Automatic meaningful representation of serial kinematic chains from their DH parameters. in preparation, 2020.

[176] S. C. A. Thomopoulos and R. Y. J. Tam. An iterative solution to the inverse kinematics of robotic manipulators. *Mechanism and Machine Theory*, 26(4):359 – 373, 1991.

[177] L. Tsai. Kinematics of a three-dof platform with three extensible limbs. In *Advances in Robot Kinematics*, pages 401–410, 1996.

[178] L. Tsai and S. Joshi. Kinematics and optimization of a spatial 3-UPU parallel manipulator. *Journal of Mechanical Design*, 122(4):439–446, 2000.

[179] L. Tsai and S. Joshi. Kinematic analysis of 3-DOF position mechanisms for use in hybrid kinematic machines. *Journal of Mechanical Design*, 124(2):245–253, 2002.

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

[181] L. Tsai and F. Tahmasebi. Synthesis and analysis of a new class of six-degree-of-freedom parallel minimanipulators. *Journal of Robotic Systems*, 10(5):561–580, 1993.

[182] S. M. Varedi, H. M. Daniali, and D. D. Gangi. Kinematics of an offset 3-UPU translational parallel manipulator by the homotopy continuation method. *Journal of Nonlinear Analysis*, 10:1468–1218, 2008.

[183] D. Verde, S. Stan, M. Manic, R. Balan, and V. Matie. Kinematics analysis, workspace, design and control of 3-RPS and TRIGLIDE medical parallel robots. In *Conference on Human System Interactions*, pages 103–108, 2009.

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

[185] K. J. Waldron. Overconstrained linkages. *Environment and Planning B: Planning and Design*, 6(4):393–402, 1979.

[186] C. T. C. Wall. Regular stratifications. In Anthony Manning, editor, *Proceedings of a Symposium on Dynamical Systems held at the University of Warwick*, pages 332–344. Springer, 1974.

[187] D. R. Walter, M. Husty, and M. Pfurner. A complete kinematic analysis of the SNU 3-UPU parallel robot. *Contemporary Mathematics*, 496:331, 2009.

[188] 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.

[189] C. Wampler, A. P. Morgan, and A. Sommese. Numerical continuation methods for solving polynomial systems arising in kinematics. *ASME Journal of Mechanical Design*, 112(1):59–68, 1990.

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

[191] E. Weisstein. *Algebraic function*. MathWorld - A Wolfram Web Resource, Last accessed 2020.

[192] E. Weisstein. Sylvester matrix. MathWorld - A Wolfram Web Resource, Last accessed 2020.

[193] K. Wohlhart. A new 6R space mechanism. In *World Congress on the Theory of Machines and Mechanisms*, volume 1, pages 193–198, 1987.

[194] K Wohlhart. Merging two general goldberg 5R linkages to obtain a new 6R space mechanism. *Mechanism and Machine Theory*, 26(7):659–668, 1991.

[195] K. Wohlhart. Kinematotropic linkages. In J. Lenarcic and V. Parenti-Castelli, editors, *Recent Advances in Robot Kinematics*, pages 359–368. Springer, Dordrecht, 1996.

[196] T.-M. Wu. The inverse kinematics problem of spatial 4P3R robot manipulator by the homotopy continuation method with an adjustable auxiliary homotopy function. *Journal of Nonlinear Analysis*, 64:2373–2380, 2006.

[197] 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*, 17:951–958, 2001.

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

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

[200] L. Yaohui. New method to extend macaulay resultant. In *International Conference on Intelligent Computation Technology and Automation*, pages 562–565, 2009.

[201] J.-S. Zhao, F. Chu, and Z.-J. Feng. Synthesis of rectilinear motion generating spatial mechanism with application to automotive suspension. *Journal of Mechanical Design*, 130(6), 2008.

[202] Z. Zhao, T. Wang, and D. Wang. Inverse kinematic analysis of the general 6R manipulator based on unit dual quaternion and dixon resultant. In *Chinese Automation Congress*, pages 2646–2650, 2017.

[203] N. D. Zoric, M. P. Lazarevic, and A. M. Simonovic. Multi-body kinematics and dynamics in terms of quaternions: Lagrange formulation in covariant form - Rodriguez approach. *FME Transactions*, 38(1):19–28, 2010.

Index of Authors

Index of concepts