UNIVERSITAT POLITÈCNICA DE CATALUNYA

Programa de doctorat:

AUTOMATITZACIÓ AVANÇADA I ROBÒTICA

Tesi doctoral

SOLVING INVERSE KINEMATICS PROBLEMS USING AN INTERVAL METHOD

Albert Castellet

Director: Federico Thomas

Institut d'Organització i Control de Sistemes Industrials

Juny de 1998

A l'Araceli,
el millor
que m'ha passat mai

Acknowledgements

I express my sincere gratitude to Federico Thomas, whose guidance has shaped this thesis. I especially thank him for putting up with my inconstance and my extravagances.

I also thank my friends and colleagues from the Institut de Cibernètica and from the Institut de Robòtica i Informàtica Industrial, for their wholehearted help in all moments and the great times we spent together.

Finally I have to mention the technical and moral support I received from my family, and specially from Araceli.

This work has been partially supported by

- the REST project, from the Spanish CICYT under contract TIC96-0721-C02-01,
- a grant from the *Associació d'Enginyers Industrials de Catalunya* with the aid of the *Agrupació Socio-Cultural dels Enginyers Industrials de Catalunya* and
- a Ph.D. fellowship granted by the Catalan Government, *Direcció General de Recerca* (CIRIT. No. FI/94-3.003).

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The study of motion is very old: the first documents go back more than two millennia, even before Greek science. But it was not until 1834, when Ampère proposed the development of an independent science under the name of *kinematics* in his famous essay on the classification of sciences.

Despite of its "advanced age", some of the basic problems in kinematics are still unsolved and constitute an active focus of research in our days. Among these unsolved problems we can point out the direct kinematics problem for parallel mechanisms or the *inverse kinematics problem* for serial chains. The solution of the latter is the main goal of this work. This problem is difficult due to its inherent computational complexity (i.e., it is NP-complete) and due to the numerical issues involved to guarantee correctness and to ensure termination.

The positional inverse kinematics problem is fundamental, not only in the design of robot manipulators, but also in other applications including computer animation or molecular modeling. The pressing need to have a solution for it in many practical applications, together with the fact that the problem can be stated very simply and that it seems at first sight easy to solve, makes the challenge to face and solve it still more interesting.

Inverse kinematics can be fit into a much more general problem, which constitutes one of the major requirements in fundamental robotics: solving loops of kinematic constraints [58]. This is a basic problem when dealing with task-level robot programming, assembly planning [62], or constraint-based modeling problems.

Kinematics of interconnected rigid bodies may lead to very complex computations and it is important to perform them in the most compact form and to search for their most rational organization. This goal motivates a great deal of research on the fundamental operations and the algebraic structures underlying kinematic methods. Nevertheless, no general satisfactory solution, convenient for practical use, has been found on the general positional inverse kinematics problem.

In the general case, the problem reduces to computing all the solution of a multivariate algebraic system. Their solution has been pursued from a large variety of ways, which will be described and briefly discussed in Section 1.2.2. However, no serious incursion has been done from interval analysis and their associated global methods to solve the systems of nonlinear equations arising in kinematics. Probably because of its fairly youth, interval methods are practically unknown among the robotics and kinematics community.

We are aware only of two papers where interval methods are applied to kinematics problems. In Hentenryck's paper on interval cuts [67], two examples of inverse kinematics problems are solved among many other examples, but no general treatment of the problem is done. Shortly after our first contribution in the area [6], a brief paper describing an initial implementation of an algorithm for solving inverse kinematics problems based on Krawaczyk's method appeared in the Journal of Mechanical Design [48], but no further research has been done by its authors.

1.1 Objectives and Contributions

The main goal of this thesis is to establish the theoretical basis for applying interval methods to the resolution of inverse kinematics problems. To this end, a new general framework for the analysis of spatial kinematic chains has been developed, which constitutes our main theoretical contribution. We have not only obtained the necessary tools required for a general solution of the problem, but also a whole new approach to the kinematic analysis of spatial mechanisms has been deeply exploited and formalized. This approach is based on a deep understanding of the self-motion manifold of the orthogonal spherical mechanism [9]. The main advantage of this approach is its generality. It is based upon a representation of any kinematic chain by two vectors. This representation allows the same treatment of the problem with any mechanism, regardless of its number of links and of its geometry.

Because of the relative novelty of interval methods and the fact that they are still being actively investigated, the bibliography regarding the global solution of nonlinear systems based on interval analysis is still very scarce. There are no comprehensive and up-to-date surveys on the subject. An effort has been made to collect the latest results available and present them in a clarifying way, without unnecessary formalisms, but with continuous references to more detailed bibliography. In this sense, Chapter 5 can be seen as a survey of interval methods for solving systems of nonlinear equations, providing the groundwork necessary for the material to be developed in the following chapters. After briefly reviewing some basics on interval analysis we introduce the central problem of the second part of the thesis: the verified solution of nonlinear systems of equations.

The second main contribution of this thesis is the development of ad hoc interval methods for the inverse kinematics problem; namely, direct cuts and propagation of intervals in spherical mechanisms.

Interval methods are effectively a new tool for the solution of inverse kinematics problems. They obtain all solutions, are general and work for any geometry (also special geometries) and configuration (also singular configurations) of the mechanism. However, for the moment, computation times are still high compared to those obtained using the recently introduced elimination methods [34].

Although our last chapter deals with an implementation and experiments, this thesis should not be seen as an exhaustive experimental study of interval methods for the solution of inverse kinematics problems. That chapter has been included to corroborate the theoretical results, which are the main contribution of this work.

1.2 The Inverse Kinematics Problem

In this section we describe some basic ideas on inverse kinematics problems and present a brief chronological overview of the methods that have been used to solve it. When necessary, the reader is addressed to proper references for more details.

1.2.1 Basic Backgrond

In an informal way, the inverse kinematics problem can be described as the problem of computing all joint positions of a manipulator corresponding to a given pose (position and orientation) of the end-effector. In the general case, the problem boils down to computing all the solutions of a multivariate algebraic system.

The inverse kinematics problem has a wide range of applications in robotics. Most of the high level problem solving about the physical world is posed in Cartesian terms. However, robots are actuated in joint space, which is what we ultimately can control. The mapping from Cartesian space to joint space leads to the inverse kinematics problem.

Formalizing the problem, *forward kinematics* is a mapping from joint space to Cartesian space:

$$\mathbf{F}(\Omega) = W \ ,$$

where Ω is a vector in joint space and W, in Cartesian space. This is a many-to-one mapping, because there is a unique Cartesian configuration for a given set of joint variables, but different sets of joint variables may lead to the same Cartesian configuration.

With *inverse kinematics*, we are referring to the inverse mapping from Cartesian space to joint space:

$$\mathbf{F}^{-1}(W) = \Omega \ .$$

The inverse kinematics is typically a one-to-many mapping. There are usually multiple sets of joint variables that yield a particular Cartesian configuration. For instance, for a robot manipulator with six rotational joints, the direct kinematics would be a mapping from the six dimensional torus, T^6 to the product of $\mathbb{R}^3$ by the rotations space $SO(3)$, that is,

$$\begin{aligned} \mathbf{F} : T^6 &\longrightarrow \mathbb{R}^3 \times SO(3) \\ \mathbf{F}(\theta) &= \mathbf{x} \ . \end{aligned}$$

Then, the inverse kinematic mapping is:

$$\mathbf{F}^{-1}(\mathbf{x}) = \theta .$$

The problem is obvious from the difference between the topological structure of the joint space and the Cartesian space; in this case, the topological difference between T^6 and $\mathbb{R}^3 \times SO(3)$ [3, 32]. This fact, not only leads to the multiplicity of solution, but also causes the existence of singularities (configurations for which the Jacobian of $\mathbf{F}$ has not maximal rank) [4, 16].

For special geometries, such as those characterized by three consecutive intersecting axes or parallel joint axes, the solution of the inverse kinematics problem can be expressed as a closed form function. In [12] and [41] we can find the closed form solutions for most of these manipulators. However, no such formulation is known for the general case and no good practical solutions are available to solve them using iterative methods. As a result, most industrial manipulators are designed so that a closed form solution exists for them. Of course, this imposes important restrictions to the designers.

1.2.2 Historical Development

This section is intended to be an overview of the main methods that have been used to solve the inverse kinematics problem of serial manipulators. For more complete surveys on the topic, refer to Roth's papers [39, 52, 53].

The most commonly used methods for solving sets of nonlinear equations are gradient-based iterative methods. Such methods require an initial guess at a solution and typically only provide a single solution of the problem close to the initial guess. Moreover, they tend to diverge, converge very slowly or even converge to erroneous solutions. In order to overcome these difficulties, two methods have been developed simultaneously. These are continuation and elimination methods, which will be described later on in this section.

Inverse kinematics of 7R closed mechanisms, or the equivalent 6R manipulators, have received the most attention, probably because they are more complex than manipulators with some prismatic joints. In a luncheon talk by Professor F. Freudenstein at the 1972 ASME Mechanisms Conference, he put forward the idea that the position analysis of the closed series-loop seven revolute (7R) chain was the most challenging unsolved problem. He called it "the Mount Everest of Kinematics" [53].

Number of Solutions

In [39] there is a summary of different methods for bounding the number of solutions for various kinematics problems.

The number of solutions of the inverse kinematics problem of a 6R manipulator has been progressively bounded. The first attempt to solve this problem was done by Pieper [42] in 1968, who proposed iterative numerical techniques for manipulators of general geometry. For 6R manipulators of general geometry, Pieper showed that a naive elimination strategy would yield a univariate polynomial of degree 524,288.

In 1972, Roth, Rastegar and Scheinman [55] showed that, in general, this problem has at most 32 solutions. Their proof was based on arguments from synthetic geometry and was non-constructive. Albala and Angeles [1] presented in 1979 the first constructive solution of the problem, in form of a 12×12 determinant. In 1980, Duffy and Crane [14] provided a solution in terms of a 32^{nd} polynomial, which always yield to false roots in addition to the desired values. Five years later, Tsai and Morgan [65] found only 16 solutions for various 6R manipulators and conjectured that this was the maximum.

It was not until 1988, when Lee and Liang [30, 31] obtained a 16^{th} degree polynomial based on Duffy's method and on dyalitic elimination.

Two basic approaches have been used for successfully solving inverse kinematics problems: continuation and elimination methods.

Continuation Methods

In order to overcome the difficulties of classical iteration methods, Roth and Freudenstein developed in 1963 a technique known as *bootstrap method* [54]. This iterative procedure has been improved and is now known as the *homotopy* or *continuation method.* It computes the solutions of the algebraic system by following paths in the complex space. Continuation methods have been widely used in kinematics, for instance by Tsai [65], by Wampler [68, 69] and by Raghavan [46].

The advantages of continuation methods is that they incorporate a *good* initial guess, yield all possible solutions and are robust in practice. However, since they are iterative numerical procedures, they can have numerical instabilities and can require dealing with a large number of unwanted solutions. Moreover, the current algorithms and implementations have to be designed for each different type of mechanism and are too slow for practical applications (around 10 seconds for the best known algorithm [68]).

Elimination Methods

Elimination methods are based on removing variables from the system of equations, and are used along with algorithms for finding roots of univariate polynomials. The elimination methods will, in theory, lead to a solution of any system of multivariate polynomial equations. In practice, the method can only be applied to relatively simple systems of equations. Beyond this, it explodes in complexity and introduces large numbers of extraneous roots.

However, recently, an elimination method known as *Sylvester's dialytic method* has been applied with success to kinematic problems [14, 30]. In 1989, Raghavan and Roth [44] obtained a 16^{th} degree polynomial for 6R manipulators, without containing extraneous roots. One year later they generalized the algorithm for manipulators with some prismatic joints [45].

This approach can be slow in practice due to symbolic expansion and may suffer from numerical instabilities. This is mainly due to the fact that the problem of computing roots of a high degree polynomial can be numerically ill-conditioned. Moreover, these algorithms seldom work for mechanisms with special geometries. Although these cases can be treated separately, there is no general approach for all special geometries.

Lastly, Manocha developed an efficient algorithm based on the use of linear algebra [33, 34]. The algebraic formulation given in [44] is used along with matrix computations and the problem is reduced to computing the eigenvalue decomposition of a numeric matrix. The algorithm is extremely fast, taking on the order of tens of milliseconds. However, it is not a general algorithm, working only for 6R mechanisms.

1.3 General Outlook at the Dissertation

This thesis is divided into two main parts. The first one is devoted to the kinematic analysis of spatial mechanisms, while the second one focuses on interval methods and their use in inverse kinematics. The main results of this thesis are summarized in [8], which can be regarded as an extended 10 page abstract.

Part I: Kinematic Analysis of Spatial Mechanisms

In a spherical mechanism, any point in a moving body is confined to lie within a spherical surface, and all spherical surfaces of motion are concentric. Considering the progress of mechanisms from planar to spherical, and from spherical to spatial, spherical mechanisms can be seen as an intermediate stage between planar and spatial mechanisms. Actually, plane mechanisms are a special case of spherical mechanisms in which the radius of the sphere extends to infinity.

The *orthogonal spherical mechanism* with n degrees of freedom is defined as a single-loop kinematic chain consisting of n rotational pairs, so that all axes of rotation meet at a point and any axis is orthogonal to that of the next pair in the chain. This mechanism becomes redundant for $n > 3$, so that the set of angles that keep it closed can be seen as a variety of dimension $n-3$ embedded in the $n-$fold torus, T^n. This set is called the *self-motion set of the orthogonal spherical mechanism*, SS^{n-3} for short. Chapter 3 focuses on the characterization of this set.

The interest of the pursued characterization in the analysis of spatial mechanisms derives from the analysis of the so-called $n-$bar mechanism (Chapter 2). The factorization of the loop equation of this particular mechanism into a rotation and a translation equation, shows that, while the solution to the former is directly SS^{n-3}, the solution to the latter can be expressed (excluding singular points) in terms of the tangent bundle of SS^{n-3}.

These chapters have been formally structured with definitions and detailed proofs of all its lemmas and theorems. Most of these results have been justified with a combination of geometric, algebraic and visual reasoning, in order to avoid awkward proofs. The result of this first part is a set of general closure equations completely defining any single loop mechanism. Some examples are given in Chapter 4, in order to clarify the basic ideas presented up to this point.

Part II: Interval Methods

In the second part, we first survey the main results in interval methods for solving systems of nonlinear equations (Chapter 5). Here, we have given priority to clearness in front of an excessive formalism, which would hide many of the ideas behind interval methods. References to more detailed bibliography are continuously given.

Chapter 6 is devoted to ad hoc interval methods developed for inverse kinematics problems. Direct cuts are simple procedures based on the closure equations, which considerably speed up general interval methods. We also describe an algorithm for nearly general interval propagation in spherical mechanisms based on geometric reasoning.

To check in practice the theoretical results of this work, an algorithm for the resolution of inverse kinematics problems has been developed and described in Chapter 7. This algorithm is general in the sense that it can be applied directly to any single loop mechanism, which is not the case in continuation and elimination methods. Its implementation is simple and provides highly satisfactory results in terms of reliability and completeness, although much research has still to be done in order to get fast computations.

Part I

Kinematic Analysis of Spatial Mechanisms

Chapter 2

The n-bar Mechanism

The n-bar mechanism used throughout this thesis was introduced first by Enric Celaya in some seminars in 1991 and appeared later in [59] as a tool for analyzing redundant single-loop spatial kinematic chains. In further papers [6, 7, 63, 64], it has been used as an alternative to the Denavit-Hartenberg parameters (D-H parameters) [12, 13, 41] for representing spatial mechanisms. The simpler structure of the n-bar mechanism leads to more regular equations than using the D-H parameters.

In this chapter, we define the n-bar mechanism and describe its relation with the Denavit-Hartenberg parameters, how it can be used to represent any single-loop spatial mechanism and the factorization of its loop equation into a rotation and a translation equation. The solution spaces of both these equations are closely related to the orthogonal spherical mechanism, which is exhaustively analyzed in next chapter.

2.1 Definition

The *n-bar mechanism* is a closed single-loop mechanism composed of n links –from now on *bars*–, each one being orthogonal to the next one. Each bar has two degrees of freedom (d.o.f.): a translational one (d_i), which is the distance along the bar from the previous joint to the next one (it can be negative), and a rotational one (ϕ_i), which corresponds to the angle between bar $(i+1)$ and the plane defined by the two previous bars (i and $i-1$) measured in the direction of bar i (Figure 2.1).

2.2 The Loop Equation

A reference frame can be associated with each bar, representing its spatial position and orientation. It will be taken with its x-axis pointing in the positive direction of the corresponding bar and the y-axis in the negative direction of the previous bar. According to the previous definitions, frame $i+1$ can be expressed in terms of frame i with the

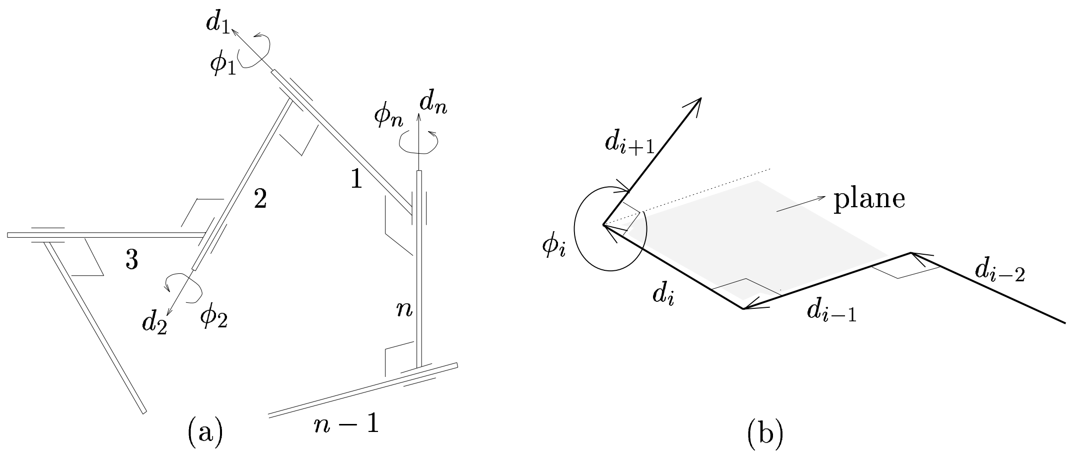


Figure 2.1: The n-bar mechanism (a) and definitions of the involved degrees of freedom (b).

transformation $\mathbf{T}(d_i)\mathbf{R}(\phi_i)\mathbf{Z}$, where $\mathbf{T}(d_i)$ stands for a translation along the x-axis, $\mathbf{R}(\phi_i)$, a rotation around the x-axis and $\mathbf{Z}$, a rotation of $\pi/2$ radians around the z-axis.

Then, the loop equation of the n-bar mechanism can be expressed as

$$\prod_{i=1}^{n} \mathbf{T}(d_i)\mathbf{R}(\phi_i)\mathbf{Z} = \mathbf{I}\,. \tag{2.1}$$

$\boldsymbol{\phi} \triangleq (\phi_1, \phi_2, \ldots, \phi_n)$ will be called the *vector of rotations* and $\mathbf{d} \triangleq (d_1, d_2, \ldots, d_n)$, the *vector of translations.*

Expressing equation (2.1) in terms of homogeneous transforms [41], we get the following matrix equation:

$$\prod_{i=1}^{n} \begin{pmatrix} 0 & -1 & 0 & d_i \\ \cos\phi_i & 0 & -\sin\phi_i & 0 \\ \sin\phi_i & 0 & \cos\phi_i & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} = \mathbf{I}\,. \tag{2.2}$$

2.3 The n-bar Mechanism and the Denavit-Hartenberg Parameters

We can regard each link of a spatial mechanism as two bars of an n-bar mechanism: the odd bars will include the dimensions of each link (the D-H parameters α_i and a_i) and the even ones will describe the connection between two consecutive links (θ_i and t_i) (Figure 2.2).

Remark 2.1. A mechanism described by the Denavit-Hartenberg parameters (α_i, a_i, θ_i and t_i) [12] can always be represented by an n-bar mechanism, where

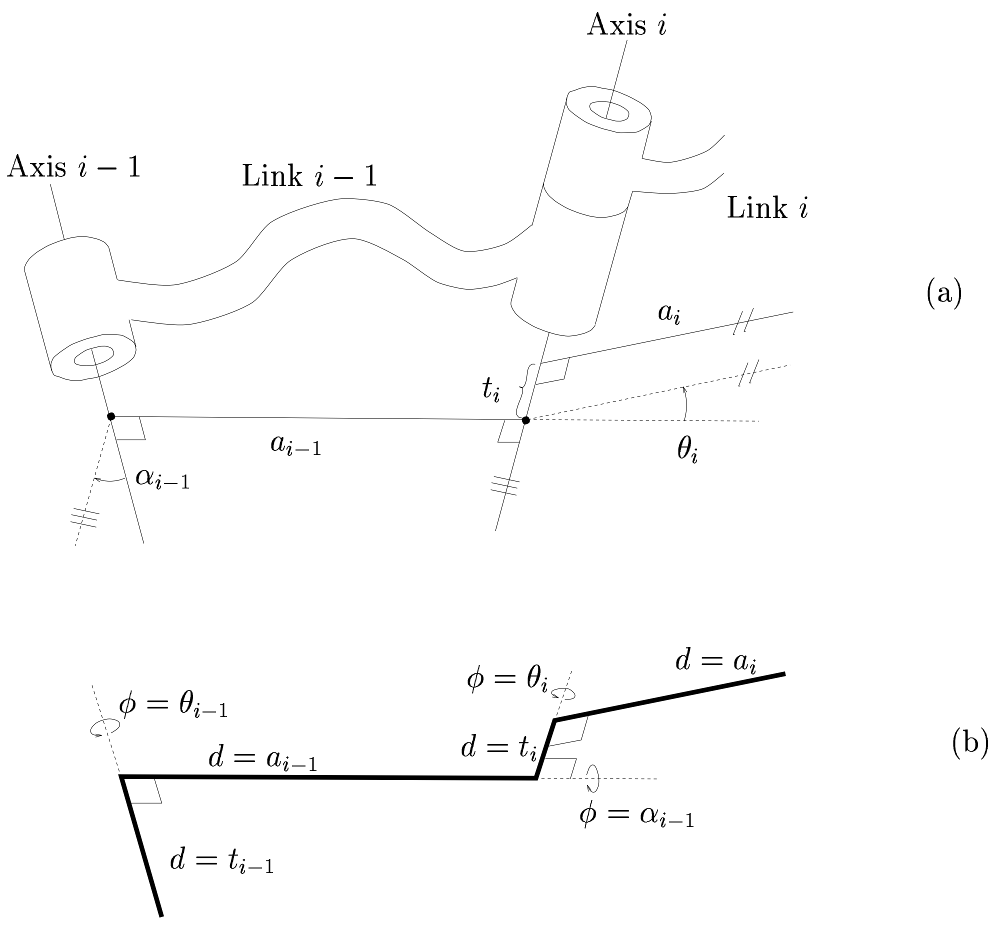


Figure 2.2: The Denavit-Hartenberg parameters of a mechanism (a) and the corresponding ones in an n-bar mechanism (b).

$$\left.\begin{array}{l}\phi_i = \alpha_{(i-1)/2} + \pi \\ d_i = a_{(i-1)/2}\end{array}\right\} \text{ when } i \text{ is odd}$$
$$\left.\begin{array}{l}\phi_i = \theta_{i/2} + \pi \\ d_i = t_{i/2}\end{array}\right\} \text{ when } i \text{ is even} \quad .$$

Three orthogonal bars are enough to reach any point in 3D space with arbitrary orientation. Then, if the mechanism described by the Denavit-Hartenberg parameters is not closed (e.g. a manipulator), we can always close the associated n-bar mechanism with three bars, representing the position of the last bar (the end-effector in a manipulator) with respect to the base (Appendix A).

As an example, let us describe a PUMA 560 with an n-bar mechanism. The Denavit-Hartenberg parameters of the PUMA 560 are given in Table 2.1.

i	α_{i-1}	a_{i-1}	t_i	θ_i
1	0	0	0	θ_1
2	$-\pi/2$	0	0	θ_2
3	0	a_2	t_3	θ_3
4	$-\pi/2$	a_3	t_4	θ_4
5	$\pi/2$	0	0	θ_5
6	$-\pi/2$	0	0	θ_6

Table 2.1: Denavit-Hartenberg parameters of the PUMA 560.

Then, using Remark 2.1, the associated n-bar mechanism will have the following vectors of rotations and translations:

i	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$\boldsymbol{\phi}$	π	$\theta_1+\pi$	$\pi/2$	$\theta_2+\pi$	π	$\theta_3+\pi$	$\pi/2$	$\theta_4+\pi$	$-\pi/2$	$\theta_5+\pi$	$\pi/2$	$\theta_6+\pi$	ϕ_{13}	ϕ_{14}	ϕ_{15}
$\mathbf{d}$	0	0	0	0	a_2	t_3	a_3	t_4	0	0	0	0	d_{13}	d_{14}	d_{15}

Table 2.2: Vectors of rotations and translations of an n-bar mechanism representing a PUMA 560.

ϕ_{13}, ϕ_{14}, ϕ_{15}, d_{13}, d_{14} and d_{15} are used to represent the end-effector's position with respect to the robot base (Appendix A).

Since any single-loop mechanism can be described by their Denavit-Hartenberg parameters, we conclude that:

Remark 2.2. Any single-loop mechanism can be described by an n-bar mechanism by taking as many bars as needed and restricting some of its degrees of freedom.

Note that the description of a mechanism by an n-bar mechanism is not unique. Since the translations are not restricted to be positive, we can always choose between two directions for each bar. This will imply to change the sign of translation d_i and also some changes in the neighboring rotations. These symmetric configurations, which represent the same mechanism, are analyzed in Section 3.2.6.

2.4 Factorization of the Loop Equation

The solution of any kinematic equation can be factored into a solution to both its rotation and translation equations. While the solution to the former can be obtained quite easily, the latter leads, in general, to inextricable formulae. This is why this factorization strategy has had very limited practical application in the past, except for some simple problems arising in the field of geometric reasoning [61]. Herein, this strategy plays a fundamental role.

The loop equation (2.1) can be factored into the following two equations [59]:

$$\mathbf{F}(\boldsymbol{\phi}) \stackrel{\triangle}{=} \prod_{i=1}^{n} \mathbf{R}(\phi_i)\mathbf{Z} = \mathbf{I} \tag{2.3}$$

and

$$\mathbf{T}(\boldsymbol{\phi}, \mathbf{d}) \stackrel{\triangle}{=} \boldsymbol{\mathcal{N}}(\boldsymbol{\phi})\mathbf{d} = \mathbf{0} \ , \tag{2.4}$$

where $\boldsymbol{\mathcal{N}}(\boldsymbol{\phi}) \stackrel{\triangle}{=} (\mathbf{n}_1\ \mathbf{n}_2\ \ldots\ \mathbf{n}_n)$, and $\mathbf{n}_i \stackrel{\triangle}{=} (n_{ix}\ n_{iy}\ n_{iz})^t$ is the unit vector pointing in the positive direction of bar i with respect to the first bar of the n-bar mechanism. The former equation (2.3) will be called the *rotation equation* and the latter one (2.4), the *translation equation.*

The rotation equation can be extracted directly from the original loop equation by simply removing all translations. It only assures that the final orientation of the chain is the same as the first one, without constraining the translation values.

The translation equation, however, involves both translations and rotations. It can also be expressed as

$$\sum_{i=1}^{n} \mathbf{n}_i d_i = 0 \ .$$

It states that the chain really closes, i.e. that the last bar ends at the beginning of the first one without constraining its orientation. This leads to the following remark:

Remark 2.3. The solutions to both the rotation equation (2.3) and the translation equation (2.4) are the solutions to the loop equation (2.1).

Thus, we can regard the solution to the loop equation (2.1) as the intersection of the solutions to equations (2.3) and (2.4). In chapter 3, both are separately analyzed.

2.5 The Generalized n-bar Mechanism

A more general version of the n-bar mechanism can be defined when the bars are not required to be orthogonal. We will call such a mechanism a *generalized n-bar mechanism.*

Each bar of a generalized n-bar mechanism is defined not only by its rotation and translation, but also by a rotation matrix $\mathbf{R}_i$, which turns the reference frame so that the x-axis points in the direction of the next bar.

The loop equation can then be expressed as

$$\prod_{i=1}^{n} \mathbf{T}(d_i)\mathbf{R}(\phi_i)\mathbf{R}_i = \mathbf{I} \ . \tag{2.5}$$

The n-bar mechanism described in Section 2.1 is a special case of this generalized version, where all $\mathbf{R}_i$ are rotations of $\pi/2$ radians around the z-axis. Therefore, any single loop kinematic chain can be described by a generalized n-bar mechanism. Moreover, fewer bars are usually required than using an orthogonal n-bar mechanism.

Here, the loop equation (2.5) can be factorized as for the orthogonal n-bar mechanism, resulting in:

$$\mathbf{F}(\boldsymbol{\phi}) \triangleq \prod_{i=1}^{n} \mathbf{R}(\phi_i)\mathbf{R}_i = \mathbf{I} \tag{2.6}$$

and

$$\mathbf{T}(\boldsymbol{\phi}, \mathbf{d}) \triangleq \boldsymbol{\mathcal{N}}(\boldsymbol{\phi})\mathbf{d} = \mathbf{0} \ , \tag{2.7}$$

where $\boldsymbol{\mathcal{N}}(\boldsymbol{\phi})$ is defined as in equation (2.4).

Next chapter is devoted to the study of the orthogonal spherical mechanism, which is closely related to the orthogonal n-bar mechanism. However, many of the results are more general and can be applied to the generalized n-bar mechanism.

Chapter 3

The Orthogonal Spherical Mechanism

3.1 Definition and Motivation

Definition 3.1 (The orthogonal spherical mechanism). The orthogonal spherical mechanism with n degrees of freedom is defined as a single-loop kinematic chain consisting of n rotational pairs, so that all axes of rotation meet at a point and any axis is orthogonal to that of the next pair in the chain (Figure 3.1).

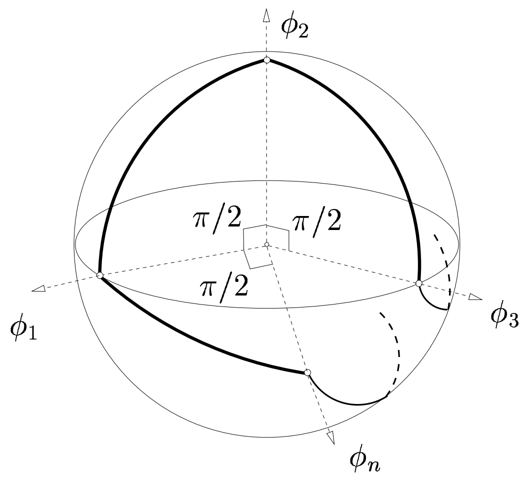


Figure 3.1: The orthogonal spherical mechanism.

A body performing a spherical motion has only three degrees of freedom, which can be expressed as rotations about three perpendicular axes. Thus, for $n > 3$ this mechanism becomes redundant, so that the set of angles that keep it closed can be seen as a variety of dimension $n-3$ embedded in the $n-$fold torus, T^n. Herein, this set is called the *self-motion set of the orthogonal spherical mechanism*, SS^{n-3} for short. The self-motion set term is motivated by the fact that any path in it leads to a continuous motion of the spherical mechanism which does not require to open it.

Articles published on the analysis and synthesis of spherical mechanisms are quite scattered. The reader is referred to [11] for a compilation and review of some of the important knowledge and techniques so far developed in this branch of the theory of mechanisms.

The interest of the characterization of the self-motion set of the orthogonal spherical mechanism in the analysis of spatial mechanisms derives from the analysis of the n-bar mechanism introduced in the previous chapter. The factorization of the loop equation of this particular mechanism into a rotation and a translation equation, shows that, while the solution to the former is directly SS^{n-3}, the solution to the latter can be expressed (excluding singular points) in terms of the tangent bundle of SS^{n-3}.

This chapter is structured as follows. The solution space of both the rotation and the translation equations are separately analyzed in Sections 3.2 and 3.3, respectively. Since the rotation equation is equivalent to the loop equation of the orthogonal spherical mechanism, Section 3.2 can also be seen as devoted to the characterization of SS^{n-3}. Its dimension, connectness, singularities and symmetries are fully studied in this section. Section 3.3 can also be seen as a characterization of the tangent space of SS^{n-3}, since it is proved that the solution set of the translation equation of the n-bar mechanism is provided by the tangent bundle of SS^{n-3}.

3.2 The Self-motion Set

3.2.1 The Rotation Equation

There is a spherical kinematic loop associated with any spatial kinematic loop called its *spherical indicatrix* [20]. Its revolute axes are parallel to those in the corresponding spatial kinematic loop, so that they all intersect at the origin. The rotation equation of the n-bar mechanism (2.3) corresponds to the loop equation of its spherical indicatrix which can be simply obtained by removing all translations from the loop equation (2.1).

Since two consecutive axes of rotations of the n-bar mechanism are orthogonal, the rotational equation (2.3) is also the loop equation of the orthogonal spherical mechanism defined above. Therefore, we will indistinctively speak of the orthogonal spherical mechanism or of the rotational component of the n-bar mechanism.

In the previous section it has been intuitively shown that the solution space of the rotation equation (2.3) is of dimension $n-3$ (a proof is given in Section 3.2.2).

Since $\mathbf{F}(\boldsymbol{\phi})$ is a 3×3 matrix, the rotation equation (2.3) consists of nine equations. Considering that there are three rotational degrees of freedom in space, it could be expected that there would be three equations representing this information. However, two equations are enough. Let us suppose that f_{11} (the first row–first column element of

$\mathbf{F}(\boldsymbol{\phi})$) is equaled to 1. Since $\mathbf{F}(\boldsymbol{\phi})$ is an orthogonal matrix, all row and column vectors are unitary and therefore f_{12}, f_{13}, f_{21} and f_{31} are necessarily 0. Likewise, if f_{22} is also 1, f_{23} and f_{32} will be 0. Finally, since the determinant of $\mathbf{F}(\boldsymbol{\phi})$ is 1 (it is a product of proper orthogonal matrices), f_{33} has to be 1. This leads to:

Remark 3.1. Any two elements of the diagonal of $\mathbf{F}(\boldsymbol{\phi})$ equaled to 1 are equivalent to considering the whole rotation matrix equation (2.3).

Lemma 3.1. *The partial derivative of* $\mathbf{F}(\boldsymbol{\phi})$ *with respect to* ϕ_i*, for a value of* $\boldsymbol{\phi}$ *that satisfies the rotation equation* (2.3)*, is a skew-symmetric matrix, whose* vector linear invariant[1] *is the direction of the* i^{th} *bar; that is,*

$$\left.\frac{\partial \mathbf{F}(\boldsymbol{\phi})}{\partial \phi_i}\right|_{\mathbf{F}(\boldsymbol{\phi})=\mathbf{I}} = \mathbf{N}_i(\boldsymbol{\phi}) \ .$$

where $$\mathbf{N}_i(\boldsymbol{\phi}) \triangleq \begin{pmatrix} 0 & -n_{iz} & n_{iy} \\ n_{iz} & 0 & -n_{ix} \\ -n_{iy} & n_{ix} & 0 \end{pmatrix} .$$

Proof. The partial derivatives of the rotation equation (2.3) with respect to ϕ_i are straightforward:

$$\frac{\partial \mathbf{F}(\boldsymbol{\phi})}{\partial \phi_i} = \mathbf{A}_1^{i-1}(\boldsymbol{\phi})\mathbf{Q}\mathbf{A}_i^n(\boldsymbol{\phi}) \ , \tag{3.1}$$

where

$$\mathbf{A}_k^l(\boldsymbol{\phi}) \triangleq \begin{cases} \mathbf{I} & k = l+1 \\ \prod_{j=k}^{l} \mathbf{R}(\phi_j)\mathbf{Z} & k \le l \end{cases} \tag{3.2}$$

and

$$\mathbf{Q} \triangleq \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{pmatrix} .$$

Since $\mathbf{F}(\boldsymbol{\phi}) = \mathbf{A}_1^{i-1}(\boldsymbol{\phi})\mathbf{A}_i^n(\boldsymbol{\phi})$, if $\boldsymbol{\phi}$ satisfies $\mathbf{F}(\boldsymbol{\phi}) = \mathbf{I}$, then

$$\mathbf{A}_i^n(\boldsymbol{\phi}) = \left(\mathbf{A}_1^{i-1}(\boldsymbol{\phi})\right)^{-1} .$$

Moreover, the matrices $\mathbf{A}_j^k(\boldsymbol{\phi})$ are orthogonal, so that $\left(\mathbf{A}_j^k(\boldsymbol{\phi})\right)^{-1} = \left(\mathbf{A}_j^k(\boldsymbol{\phi})\right)^t$. Then,

$$\left.\frac{\partial \mathbf{F}(\boldsymbol{\phi})}{\partial \phi_i}\right|_{\mathbf{F}(\boldsymbol{\phi})=\mathbf{I}} = \mathbf{A}_1^{i-1}(\boldsymbol{\phi})\mathbf{Q}\left(\mathbf{A}_1^{i-1}(\boldsymbol{\phi})\right)^t .$$

[1]The *vector linear invariant* of a 3×3 skew-symmetric matrix $\mathbf{A}$ is the vector $\mathbf{a} = \begin{pmatrix} a_{32} & a_{13} & a_{21} \end{pmatrix}^t$.

Denoting

$$\mathbf{A}_1^{i-1}(\boldsymbol{\phi}) \triangleq (\mathbf{n}_i \; \mathbf{o}_i \; \mathbf{a}_i) \triangleq \begin{pmatrix} n_{ix} & o_{ix} & a_{ix} \\ n_{iy} & o_{iy} & a_{iy} \\ n_{iz} & o_{iz} & a_{iz} \end{pmatrix} ,$$

we can write

$$\mathbf{A}_1^{i-1}(\boldsymbol{\phi})\mathbf{Q}\left(\mathbf{A}_1^{i-1}(\boldsymbol{\phi})\right)^t = \begin{pmatrix} n_{ix} & o_{ix} & a_{ix} \\ n_{iy} & o_{iy} & a_{iy} \\ n_{iz} & o_{iz} & a_{iz} \end{pmatrix} \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} n_{ix} & n_{iy} & n_{iz} \\ o_{ix} & o_{iy} & o_{iz} \\ a_{ix} & a_{iy} & a_{iz} \end{pmatrix} =$$
$$\begin{pmatrix} 0 & (o_{iy}a_{ix} - o_{ix}a_{iy}) & (o_{iz}a_{ix} - o_{ix}a_{iz}) \\ (o_{ix}a_{iy} - o_{iy}a_{ix}) & 0 & (o_{iz}a_{iy} - o_{iy}a_{iz}) \\ (o_{ix}a_{iz} - o_{iz}a_{ix}) & (o_{iy}a_{iz} - o_{iz}a_{iy}) & 0 \end{pmatrix} .$$

Since $\mathbf{A}_1^{i-1}(\boldsymbol{\phi})$ are orthonormal matrices, $\mathbf{n}_i$ is the cross product of $\mathbf{o}_i$ and $\mathbf{a}_i$. Thus,

$$n_{ix} = o_{iy}a_{iz} - o_{iz}a_{iy}$$
$$n_{iy} = o_{iz}a_{ix} - o_{ix}a_{iz}$$
$$n_{iz} = o_{ix}a_{iy} - o_{iy}a_{ix} .$$

Then,

$$\mathbf{A}_1^{i-1}(\boldsymbol{\phi})\mathbf{Q}\left(\mathbf{A}_1^{i-1}(\boldsymbol{\phi})\right)^t = \begin{pmatrix} 0 & -n_{iz} & n_{iy} \\ n_{iz} & 0 & -n_{ix} \\ -n_{iy} & n_{ix} & 0 \end{pmatrix} ,$$

which proves the lemma. □

3.2.2 SS^{n-3} and SM^{n-3}

The solution of the rotation equation (2.3) can be seen as a subset of the space formed by the n-fold product of the rotational variables, that is, the n-torus, T^n. Equation (2.3) has a straightforward *geometric interpretation* as a spherical polygon, which will be useful later. Let us first recall some concepts. A *spherical polygon* is a closed figure on the surface of a sphere composed of arcs of great circles (or *sides*) limited by common points (or *vertices*). If one assigns a circulating direction to the sides of a spherical polygon, the *exterior angle* between two adjacent sides is defined as the angle measured from the prolongation of the first side beyond the common vertex, to the second side.

Equation (2.3) can be interpreted as an n-sided spherical polygon. Consider a unit sphere centered at the coordinate origin. As a result of applying successive rotations, the x-axis will describe a spherical polygon with sides of length $\pi/2$ and exterior angles equal to ϕ_i (Figure 3.2).

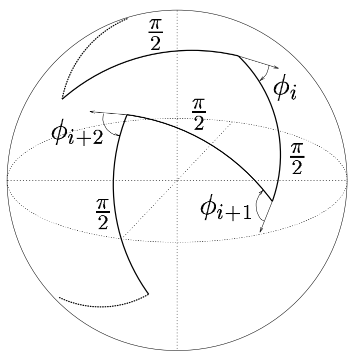


Figure 3.2: Spherical polygon generated by the x-axis with sides of length $\frac{\pi}{2}$ and exterior angles equal to ϕ_i.

Note that by varying the angles between two sides of length $\pi/2$, one can form a triangle whose third side may attain any value between 0 and π. Therefore, one can arbitrarily fix $n-3$ consecutive variables of an orthogonal spherical mechanism. This leads to a spherical chain with $n-2$ sides that can always be closed using two sides. Moreover, if the resulting angle between these two sides is different from 0 and π, then there are two solutions corresponding to the two symmetric placements of the two sides on the sphere (Figure 3.3). As the angle approaches 0 or π, the two solution branches fuse into a single one.

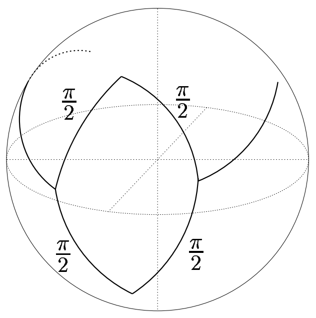


Figure 3.3: Two symmetric placements for two sides to close the spherical polygon.

Definition 3.2 (Self-motion Set, SS^{n-3}). The set of points $\boldsymbol{\phi}$ in T^n that fulfill the rotation equation (2.3), that is $\{\ \boldsymbol{\phi} \in T^n \mid \mathbf{F}(\boldsymbol{\phi}) = \mathbf{I}\ \}$, will be called the *self-motion set*, SS^{n-3} for short.

As it has already been mentioned, the term *self-motion set* is motivated by the fact that any trajectory inside this set corresponds to a continuous motion of the corresponding spherical mechanism, which does not require to open it [5].

Proposition 3.1. *SS^{n-3} is an (n-3)-dimensional algebraic variety embedded in T^n.*

Proof. A spherical mechanism becomes redundant for $n > 3$. Then, if we choose $n-3$ rotations, we will always be able to find some values for the other three rotations. In general, we cannot fix more than $n-3$ rotations, except for some singular points, as shown in next section. Then, since we can locally parameterize SS^{n-3} using $n-3$ parameters, except for some singular points, the self-motion set is an (n-3)-dimensional variety. □

The self-motion set can be seen as smooth hypersurfaces of dimension $n-3$ possibly intersecting themselves. As it will become clear in the next subsection, the stratification of this set leads to a manifold of dimension $n-3$ connected through singular points. See Chapter 1 of Part I in [15] for an introduction to the stratification of algebraic sets.

Definition 3.3 (Self-motion Manifold, SM^{n-3}). The *self-motion manifold*, SM^{n-3}, is the manifold resulting from removing all the singularities from SS^{n-3}.

As it will be proved next, singular points correspond to those situations in which the n-bar mechanism becomes planar, that is, when all axes of rotation lie on the same plane. From a topological point of view, removing from SS^{n-3} all these points yields a (n-3)-manifold, the self-motion manifold (SM^{n-3}); then, the self-motion set can be alternatively seen as a pseudo-manifold, i.e. a punched manifold, since the $(n-3)$-manifold is pinched in a discrete set of points: the singular points.

3.2.3 Singularities of SS^{n-3}

The Jacobian of $\mathbf{F}(\boldsymbol{\phi})$ is defined as

$$\nabla\mathbf{F}(\boldsymbol{\phi}) \triangleq \begin{pmatrix} \frac{\partial\mathbf{F}(\boldsymbol{\phi})}{\partial\phi_1} & \frac{\partial\mathbf{F}(\boldsymbol{\phi})}{\partial\phi_2} & \dots & \frac{\partial\mathbf{F}(\boldsymbol{\phi})}{\partial\phi_n} \end{pmatrix} . \tag{3.3}$$

By Lemma 3.1, the Jacobian in a point $\boldsymbol{\phi}$ of the self-motion set will be

$$\nabla\mathbf{F}(\boldsymbol{\phi})|_{\boldsymbol{\phi}\in SS^{n-3}} = (\ \mathbf{N}_1(\boldsymbol{\phi})\ \ \mathbf{N}_2(\boldsymbol{\phi})\ \ \dots\ \ \mathbf{N}_n(\boldsymbol{\phi})\ ) . \tag{3.4}$$

Definition 3.4 (Global Singularity). *Global singularities* of SS^{n-3} are defined as those values of $\boldsymbol{\phi}$ for which $\nabla\mathbf{F}(\boldsymbol{\phi})|_{\boldsymbol{\phi}\in SS^{n-3}}$ is not of full rank, 3.

If n is even, let us group the rotational variables in couples as follows:

$$(\phi_1\ \phi_2), (\phi_3\ \phi_4), \dots, (\phi_{n-1}\ \phi_n).$$

Now a will be the number of couples that are $(0\ 0)$, b the number of couples $(0\ \pi)$ and c the number of couples $(\pi\ 0)$.

Lemma 3.2. *$\boldsymbol{\phi} = (\phi_1\ \phi_2\ \dots\ \phi_n)$ is a global singularity iff:*

$$\begin{array}{l} 1.\ n \textit{ is even} \\ 2.\ \phi_i(\bmod \pi) = 0, \quad i = 1,\dots,n \quad \textit{and} \\ 3.\ (a+b) \textit{ and } (b+c) \quad \textit{are both even} \quad . \end{array} \tag{3.5}$$

Proof. By analyzing (3.4), it is clear that global singularities have to correspond to those situations where all n bars are on a plane. Since an n-bar mechanism can degenerate to planar only when n is even, we can conclude that there are no global singularities when n is odd. The second condition of the lemma ensures that the mechanism lies on the plane defined by the first and the last bar. The last condition is a simplified version of the closure equation (2.3) when the second condition holds, inspired in the geometric interpretation of the rotation equation as a spherical polygon. Both provide a necessary and sufficient condition for the n-bar mechanism to be in a planar configuration and, hence, for the configuration point to be singular. □

Corollary 3.1. *Global singularities are discrete points of* SS^{n-3}.

Corollary 3.2. *Global singularities are of corank 1.*

Proof. The corank of a singularity is defined as the difference between rank$(\nabla \mathbf{F}(\boldsymbol{\phi})|_{\boldsymbol{\phi}\in SM^{n-3}})$ (in a non-singular point) and the rank in the singularity. Note that the rank in a singularity can never be lower than 2 since all the axes of rotation can never be arranged so that they are aligned. Then the corank cannot be greater than 1. □

Corollary 3.3. *When n is odd,* $SM^{n-3} = SS^{n-3}$.

Corollary 3.4. *When n is even the number of different singularities is* 2^{n-2}.

Proof. We can choose for the $n-2$ first rotations the values 0 or π, which leads to 2^{n-2} possible combinations. Then, using the above lemma, we can choose one, and only one, of the groups $(0\ 0)$, $(0\ \pi)$, $(\pi\ 0)$, $(\pi\ \pi)$ so that the third condition holds. □

3.2.4 The Connectivity of SS^{n-3} and SM^{n-3}

Note that SS^{n-3} might not be fully connected. Bounds on the number of connected components of the self-motion set are discussed in [5], where it is shown that the self-motion set of a redundant chain can have no more connected components than the maximum number of inverse kinematics solutions of a non-redundant kinematic chain of the same class. A discrete closed-form solution exists for spherical mechanisms with up to three degrees of freedom. For $n = 3$, there are two discrete solutions. Therefore, for $n > 3$ the self-motion set of a spherical redundant mechanism can have at most two disconnected components.

Lemma 3.3. SS^{n-3} *is a connected subset of* T^n.

Proof. Let us consider the previous interpretation of the rotation equation as a spherical polygon. To prove that SS^{n-3} has a single connected component, it suffices to show that a reference configuration can be reached from every other configuration. When n is even, let the reference configuration be $\phi_i = \pi$, $\forall i = 1, \ldots, n$ and when n is odd let it be $\phi_1 = \phi_2 = \phi_3 = \pi/2$, $\phi_i = \pi$, $\forall i = 4, \ldots, n$.

Now, from any initial configuration, one can make the first $n-3$ exterior angles attain sequentially their values at the reference configuration. Thanks to the last two sides, the chain will remain closed throughout the process. When these $n-3$ angles have reached the reference values, the other three angles have to be those corresponding to the reference configuration or its symmetric ones (Figure 3.3). If they are the symmetric ones, we can always move the first $n-3$ angles again, so that the two sides corresponding to the last three angles are aligned. Then we move the $n-3$ angles back to the reference configuration, but now bringing the two sides to the right configuration. □

Lemma 3.4. *SM^{n-3} is a connected subset of T^n when $n > 4$.*

Proof. . Given the length of the proof, we will only sketch it out. The idea is to link any two points of SM^{n-3} following a path completely contained in SM^{n-3}. Any point of SM^{n-3} has at least one ϕ_i different from 0 or π. We fix it and make the other angles (except three consecutive ones) attain the values of the other configuration we want to link to it. Then, we make the fixed angle, ϕ_i, attain the corresponding final value. If we were to fall in a singularity, it can be shown that it is avoidable. Finally we have to check if the three consecutive ϕ_i used to close the chain are the ones of the final point. If they are the symmetric ones, we have to proceed similarly to the preceding proof. □

In Chapter 4 we will prove that for $n = 4$, SM^1 is not connected.

3.2.5 Parameterizations of SM^{n-3}

The self-motion manifold is an (n-3)-dimensional manifold of class C^∞, which can be parameterized by a set of $r = n-3$ independent parameters, $\boldsymbol{\psi} = \{\psi_1, \psi_2, \ldots, \psi_r\}$, so that it can be locally generated by continuously sweeping ψ_i through their ranges. Among all possible parameterizations, we can take r coordinates of the surrounding space T^n as local coordinates in the neighborhood of each point $\boldsymbol{\phi} \in SM^{n-3}$. This is, in fact, the implicit function theorem formulated in convenient terms, whose proof can be found in any textbook of differential geometry (see for example [70]).

Definition 3.5 (Trivial Parameterization). In particular, we can take r consecutive variables as parameters. This will be called a *trivial parameterization.*

Let us take a trivial parameterization. Without loss of generality, let the set of parameters $\boldsymbol{\psi} = \{\psi_1, \psi_2, \ldots, \psi_r\}$ be the first r rotational variables: $\psi_i = \phi_i$ $(i = 1, \ldots, r)$. Then, from (2.3) we have

$$\mathbf{A}(\boldsymbol{\psi}) \triangleq \left(\mathbf{Z}\,\mathbf{A}_1^{n-3}(\boldsymbol{\psi})\right)^t = \mathbf{R}(\phi_{n-2})\,\mathbf{Z}\,\mathbf{R}(\phi_{n-1})\,\mathbf{Z}\,\mathbf{R}(\phi_n)\,, \tag{3.6}$$

which in general has two solutions for ϕ_{n-2}, ϕ_{n-1} and ϕ_n given any proper orthogonal matrix $\mathbf{A}(\boldsymbol{\psi})$ encompassing all the parameters; namely,

$$\begin{aligned} \phi_{n-2} &= \text{atan2}(\pm a_{21}, \mp a_{31}) \\ \phi_{n-1} &= \mp\text{acos}(-a_{11}) \\ \phi_n &= \text{atan2}(\mp a_{12}, \mp a_{13})\,, \end{aligned} \tag{3.7}$$

where a_{ij} denotes the element (i,j) of $\mathbf{A}(\boldsymbol{\psi})$. One solution corresponds to the upper row of signs and the other, to the lower one. Note that the two solutions correspond to the two symmetric branches in Figure 3.3.

Definition 3.6 (Singularities of a trivial parameterization). Given a trivial parameterization, the points of SM^{n-3} where $a_{11} = \pm 1$ are called *singularities of the parameterization.*

In a singularity of a trivial parameterization, when $a_{11} = \pm 1$, equation (3.7) has infinite solutions and it can be shown that they correspond to those situations in which the last three bars are coplanar.

These three equations (3.7) *almost* provide the same information as the rotation equation (2.3); actually, they are equivalent but for the singularities of the chosen parameterization.

Proposition 3.2. *The set of all trivial parameterizations provide an atlas for SM^{n-3}.*

Proof. An *atlas* is a set of parameterizations that covers completely the manifold, which means that any point can be parameterized locally by one of these parameterizations. Any point $\boldsymbol{\phi}^0$ in SM^{n-3} has at least an angle ϕ_i^0 whose value is different from 0 or π. Then, we can take a trivial parameterization containing all the angles except ϕ_{i-1}^0, ϕ_i^0 and ϕ_{i+1}^0, which will locally parameterize SM^{n-3} in a neighborhood of $\boldsymbol{\phi}^0$. □

We can also take r non-consecutive variables as parameters, but the formulation becomes a bit more complex. Actually, the rotation equation (2.3) can be written as

$$\mathbf{B}(\boldsymbol{\psi}) = \mathbf{R}(\phi_a)\,\mathbf{C}(\boldsymbol{\psi})\,\mathbf{R}(\phi_b)\,\mathbf{D}(\boldsymbol{\psi})\,\mathbf{R}(\phi_c)\,, \tag{3.8}$$

where ϕ_a, ϕ_b and ϕ_c are the variables that are not parameters and $\mathbf{B}(\boldsymbol{\psi})$, $\mathbf{C}(\boldsymbol{\psi})$ and $\mathbf{D}(\boldsymbol{\psi})$ are rotation matrices that depend on the parameters. There exist two values for ϕ_a, ϕ_b and ϕ_c that satisfy (3.8) iff

$$(1-c_{11}^2)(1-d_{11}^2) - (b_{11} - c_{11}d_{11})^2 \geq 0\,,$$

where b_{ij}, c_{ij} and d_{ij} denote the elements (i,j) of $\mathbf{B}(\boldsymbol{\psi})$, $\mathbf{C}(\boldsymbol{\psi})$ and $\mathbf{D}(\boldsymbol{\psi})$.

Expressions for ϕ_a, ϕ_b and ϕ_c in terms of the elements of $\mathbf{B}(\boldsymbol{\psi})$, $\mathbf{C}(\boldsymbol{\psi})$ and $\mathbf{D}(\boldsymbol{\psi})$ can be found in [61]:

$$\begin{aligned}
\phi_b &= \operatorname{atan2}(ih \mp g\omega, gi \pm h\omega)\\
\phi_c &= \operatorname{atan2}(e_{13}b_{12} - e_{12}b_{13}, e_{12}b_{12} + e_{13}b_{13})\\
\phi_a &= \operatorname{atan2}(f_{32}, f_{22})
\end{aligned}$$

where

$$\begin{aligned}
\omega &= +\sqrt{(1-c_{11}^2)(1-d_{11}^2) - (b_{11} - c_{11}d_{11})^2}\\
g &= c_{12}d_{21} + c_{13}d_{31}\\
h &= c_{13}d_{21} - c_{12}d_{31}\\
i &= b_{11} - c_{11}d_{11}
\end{aligned}$$

and

$$\begin{array}{l}\mathbf{E} = \mathbf{C}\mathbf{R}(\phi_b)\mathbf{D} \\ \mathbf{F} = \mathbf{B}\mathbf{R}^t(\phi_c)\mathbf{E}^t\end{array} \quad .$$

In what follows, when we refer to a parameterization, we mean a trivial parameterization.

3.2.6 Symmetries of SS^{n-3}

The self-motion set of the orthogonal n-bar mechanism is highly symmetric. There are symmetric points with respect to any singularity of the parameterization, which correspond to the two solutions of (3.7).

Definition 3.7 (Symmetric Points). Given a point $\boldsymbol{\phi}^0 = (\phi_1 \; \dots \; \phi_n)^t \in SS^{n-3}$, the points

$$\boldsymbol{\phi}^i = (\phi_1 \quad \dots \quad \phi_{i-1}+\pi \quad \text{-}\phi_i \quad \phi_{i+1}+\pi \quad \dots \quad \phi_n)^t \quad i = 1, 2, \dots, n$$

are also in SS^{n-3} and will be called *symmetric points of* $\boldsymbol{\phi}^0$.

It is obvious that symmetric points are also in SS^{n-3} by analyzing the two solutions of equation (3.7), which correspond to the two symmetric branches in Figure 3.3. It will be said that $\boldsymbol{\phi}^i$ is obtained from $\boldsymbol{\phi}^0$ by *applying a symmetry* to variable ϕ_i.

Proposition 3.3. *Given two vectors* $\boldsymbol{\phi}^0 = (\phi_1 \; \dots \; \phi_n)^t$ *and* $\mathbf{d}^0 = (d_1 \; \dots \; d_n)^t$ *that satisfy loop equation* (2.1)*, the vectors*

$$\begin{array}{l}\boldsymbol{\phi}^i = (\phi_1 \quad \dots \quad \phi_{i-1}+\pi \quad \text{-}\phi_i \quad \phi_{i+1}+\pi \quad \dots \quad \phi_n)^t \quad \textit{and} \\ \mathbf{d}^i = (d_1 \quad \dots \quad d_{i-1} \quad \text{-}d_i \quad d_{i+1} \quad \dots \quad d_n)^t \quad i = 1, 2, \dots, n\end{array}$$

also satisfy the loop equation.

Proof. When a symmetry is applied to ϕ_i, $\mathbf{n}_i$ points in the opposite direction. Then, by changing the sign of d_i, the chain is kept closed. □

Lemma 3.5. *Any point on* SS^{n-3} *has* $2^n - 1$ *different symmetric points, except for:*

1. *Singular points, which have* $2^{n-2} - 1$ *different symmetric points.*
2. *Points that have all its even or odd elements equal to* 0 *or* π*, assuming that* n *is even, which have* $2^{n-1} - 1$ *different symmetric points.*

Proof. It can be seen that the application of two or more symmetries is commutative and that applying a symmetry twice to the same ϕ_i does not change the point. Thus, the iterative computation of all the symmetries of Definition 3.7 leads to 2^n symmetric points. But some of these points can be repeated. Applying symmetries to a rotation that is neither 0 nor π will change its sign and it is impossible to get again its previous value by applying symmetries to its neighboring rotations. However, if we apply a symmetry to an angle that is 0 or π, this value does not change and the neighboring ones are incremented by π. To bring those changed values to their original values, we have to apply a symmetry to its neighboring rotations ($i-2$ and $i+2$) and so on. Then, it can be easily shown that the only points that can have repeated symmetric points are those enumerated in this lemma. □

Corollary 3.5. *All singular points are symmetric.*

Proof. Any symmetric point of a singular point is singular too, since adding π or changing the sign of a variable that is 0 or π leads to 0 or π. Then, all singular points have to be symmetric since there are the same number of different singular points (2^{n-2}) than different symmetric points of a singular one. □

The points of case 2 in Lemma 3.5 are those corresponding to configurations of mechanisms that have the directions of alternating bars in a plane and the other ones orthogonal to it. This is the case of an n-bar mechanism representing a planar mechanism: the odd bars will represent the elements of the planar mechanism and the even bars, of length 0, will be orthogonal to them representing the axes of rotation of the planar mechanism. Note the difference between an n-bar mechanism in a planar configuration (global singularity), where all n bars are contained in a plane, and a planar mechanism represented by an n-bar mechanism.

3.2.7 Differential Approximations of the Rotation Equation

In this section, we obtain the Taylor's approximation of $\mathbf{F}(\boldsymbol{\phi}+\Delta\boldsymbol{\phi})$, when $\boldsymbol{\phi}$ is restricted to SS^{n-3}, in terms of directions of bars.

Lemma 3.6. *Take a point $\boldsymbol{\phi} \in SS^{n-3}$ and a small perturbation of it, $\boldsymbol{\phi}' = \boldsymbol{\phi} + \Delta\boldsymbol{\phi}$. Then, $\boldsymbol{\phi}' \in SS^{n-3}$ iff*

$$\boldsymbol{\mathcal{N}}(\boldsymbol{\phi})\Delta\boldsymbol{\phi} = 0 \ , \tag{3.9}$$

which is called the equation of approximation *in [66].*

Proof. The first order Taylor's expansion of the rotation equation (2.3) around a point $\boldsymbol{\phi}_0$ is

$$\mathbf{F}(\boldsymbol{\phi}_0 + \Delta\boldsymbol{\phi}) \simeq \mathbf{F}(\boldsymbol{\phi}_0) + \nabla\mathbf{F}(\boldsymbol{\phi}_0)\Delta\boldsymbol{\phi} \ .$$

Given $\boldsymbol{\phi} \in SS^{n-3}$, $\boldsymbol{\phi} + \Delta\boldsymbol{\phi}$ will also be another point of the self-motion set if

$$\nabla\mathbf{F}(\boldsymbol{\phi})|_{\boldsymbol{\phi}\in SS^{n-3}}\,\Delta\boldsymbol{\phi} = 0 \ .$$

Using (3.4), we can write this equation in terms of the directions of bars, that is,

$$\sum_{i=1}^{n} \mathbf{n}_i \Delta\phi_i = 0 \ ,$$

which is equivalent to the equation of approximation (3.9). □

Proposition 3.4. *The k^{th} derivative of $\mathbf{F}(\boldsymbol{\phi})$ in $\boldsymbol{\phi} \in SS^{n-3}$ is*

$$\left. \frac{\partial^k \mathbf{F}(\boldsymbol{\phi})}{\partial\phi_{i_1}\partial\phi_{i_2}\dots\partial\phi_{i_k}} \right|_{\boldsymbol{\phi}\in SS^{n-3}} = \mathbf{N}_{i_1}(\boldsymbol{\phi})\mathbf{N}_{i_2}(\boldsymbol{\phi})\dots\mathbf{N}_{i_k}(\boldsymbol{\phi}) \ , \tag{3.10}$$

where $i_1 \leq i_2 \leq \dots \leq i_k$.

Proof. By using the same notation in the proof of Lemma 3.1, the second partial derivative of $\mathbf{F}(\boldsymbol{\phi})$ can be expressed as

$$\frac{\partial^2 \mathbf{F}(\boldsymbol{\phi})}{\partial\phi_i\partial\phi_j} = \mathbf{A}_1^{i-1}(\boldsymbol{\phi})\mathbf{Q}\mathbf{A}_i^{j-1}(\boldsymbol{\phi})\mathbf{Q}\mathbf{A}_j^{n}(\boldsymbol{\phi}) \ .$$

Since $\mathbf{A}_i^{i-1}(\boldsymbol{\phi})\mathbf{A}_i^{j-1}(\boldsymbol{\phi})\mathbf{A}_j^{n}(\boldsymbol{\phi}) = \mathbf{I}$ when $\boldsymbol{\phi} \in SS^{n-3}$, we have that

$$\left. \frac{\partial^2 \mathbf{F}(\boldsymbol{\phi})}{\partial\phi_i\partial\phi_j} \right|_{\boldsymbol{\phi}\in SS^{n-3}} = \mathbf{A}_1^{i-1}(\boldsymbol{\phi})\mathbf{Q}\left(\mathbf{A}_1^{i-1}(\boldsymbol{\phi})\right)^t \mathbf{A}_1^{j-1}(\boldsymbol{\phi})\mathbf{Q}\left(\mathbf{A}_1^{j-1}(\boldsymbol{\phi})\right)^t \ .$$

The second derivatives of $\mathbf{F}(\boldsymbol{\phi})|_{\boldsymbol{\phi}\in SS^{n-3}}$ are then the product of the first two derivatives

$$\left. \frac{\partial^2 \mathbf{F}(\boldsymbol{\phi})}{\partial\phi_i\partial\phi_j} \right|_{\boldsymbol{\phi}\in SS^{n-3}} = \mathbf{N}_i(\boldsymbol{\phi})\mathbf{N}_j(\boldsymbol{\phi}) \ , \tag{3.11}$$

where $i \leq j$. Similarly, higher order partial derivatives in a point of the self-motion set are obtained in terms of the directions of the bars. □

We can express the second order Taylor's approximation of the rotation equation (2.3) for configurations in SS^{n-3} as

$$\nabla\mathbf{F}(\boldsymbol{\phi})|_{\boldsymbol{\phi}\in SS^{n-3}}\,\Delta\boldsymbol{\phi} + \frac{1}{2}\,\Delta\boldsymbol{\phi}\nabla^2\mathbf{F}(\boldsymbol{\phi})|_{\boldsymbol{\phi}\in SS^{n-3}}\,\Delta\boldsymbol{\phi} = 0 \ , \tag{3.12}$$

where

$$\nabla^2\mathbf{F}(\boldsymbol{\phi})|_{\boldsymbol{\phi}\in SS^{n-3}} \triangleq \begin{pmatrix} \mathbf{N}_1^2(\boldsymbol{\phi}) & \mathbf{N}_1(\boldsymbol{\phi})\mathbf{N}_2(\boldsymbol{\phi}) & \dots & \mathbf{N}_1(\boldsymbol{\phi})\mathbf{N}_n(\boldsymbol{\phi}) \\ \mathbf{N}_1(\boldsymbol{\phi})\mathbf{N}_2(\boldsymbol{\phi}) & \mathbf{N}_2^2(\boldsymbol{\phi}) & \dots & \mathbf{N}_2(\boldsymbol{\phi})\mathbf{N}_n(\boldsymbol{\phi}) \\ \vdots & \vdots & & \vdots \\ \mathbf{N}_1(\boldsymbol{\phi})\mathbf{N}_n(\boldsymbol{\phi}) & \mathbf{N}_2(\boldsymbol{\phi})\mathbf{N}_n(\boldsymbol{\phi}) & \dots & \mathbf{N}_n^2(\boldsymbol{\phi}) \end{pmatrix}$$

is the Hessian of $\mathbf{F}(\boldsymbol{\phi})$ in a point of the self-motion set.

Higher order approximations could also be obtained in terms of products of the skew-symmetric matrices $\mathbf{N}_i(\boldsymbol{\phi})$.

3.3 The Tangent Space of SM^{n-3}

3.3.1 The Translation Equation

We have seen that the solution space of the rotation equation –excluding global singularities– is a manifold, the self-motion manifold (SM^{n-3}). The solution space of the translation equation seems to have, however, a much more complicated structure, since it depends on the involved rotations. In this section we will analyze the structure of the solution of the translation equation and its relation with its rotation counterpart.

Most of the results concerning the tangent space of SM^{n-3} are valid for generalized n-bar mechanisms. Only the last equations in Section 3.3.5 are given for orthogonal n-bar mechanisms.

Proposition 3.5. *The set of solutions* $\mathbf{d} = (d_1 \; d_2 \; \dots \; d_n)^t$ *of the translation equation* (2.4)

$$\mathbf{T}(\boldsymbol{\phi}, \mathbf{d}) = \sum_{i=1}^{n} \mathbf{n}_i d_i = 0 \tag{3.13}$$

is an (n-3)-linear variety in $\mathbb{R}^n$ *if* $\boldsymbol{\phi} \in SM^{n-3}$ *and an (n-2)-linear variety if* $\boldsymbol{\phi}$ *is a global singularity.*

Proof. If $\boldsymbol{\phi} \in SM^{n-3}$, the rank of $\boldsymbol{\mathcal{N}}(\boldsymbol{\phi}) = (\mathbf{n}_1 \; \mathbf{n}_2 \; \dots \; \mathbf{n}_n)$ is 3, i.e. the mechanism is not planar. Let $\mathbf{n}_{i_1}, \mathbf{n}_{i_2}, \mathbf{n}_{i_3}$ be linear independent. Any choice of $\{ d_i \, ; \; i \neq i_1, i_2, i_3 \}$ determines the three variables $d_{i_1}, d_{i_2}, d_{i_3}$. When $\boldsymbol{\phi}$ is a global singularity, we can only take two linear independent $\mathbf{n}_i$, and $n-2$ translations can be taken so that the other two are determined. □

Since the solution of the translation equation is a linear variety, it can always be parameterized with n-3 or n-2 translational variables, d_i. For example, $\delta_i = d_i$ ($i = 1, \dots, n-3$) can be used to parameterize this variety outside of a global singularity if the last three bars are not coplanar.

3.3.2 Spatial to Spherical Transference

Theorem 3.1 (Spatial to Spherical Transference). *The solution of the translation equation (outside the global singularities of its rotation equation) is the tangent bundle of* SM^{n-3}*, which can be expressed as*

$$\mathbf{d} = \mathbf{K}\boldsymbol{\lambda}, \qquad \forall \boldsymbol{\lambda} = (\lambda_1 \; \dots \; \lambda_r)^t \in \mathbb{R}^{n-3}, \tag{3.14}$$

where

$$\mathbf{K} = \begin{pmatrix} \dfrac{\partial\phi_1}{\partial\psi_1} & \cdots & \dfrac{\partial\phi_1}{\partial\psi_r} \\ \vdots & & \vdots \\ \dfrac{\partial\phi_n}{\partial\psi_1} & \cdots & \dfrac{\partial\phi_n}{\partial\psi_r} \end{pmatrix}$$

and $\boldsymbol{\psi} = (\psi_1, \psi_2, \ldots, \psi_r)$ *is an arbitrary parameterization of* SM^{n-3}.

Proof. Using Lemma 3.1, the translation equation (3.13) can be rewritten as

$$\sum_{i=1}^{n} \frac{\partial \mathbf{F}(\boldsymbol{\phi})}{\partial \phi_i} d_i = \mathbf{0}$$

or shorter

$$\nabla \mathbf{F}(\boldsymbol{\phi})\mathbf{d} = \mathbf{0} \quad , \tag{3.15}$$

which is another version of the translation equation.

The tangent space of an m-dimensional variety embedded in an n-dimensional space is defined as the orthogonal complement to the vector space spanned by $n-m$ independent gradient vectors of this variety [29, p. 80]. $\nabla\mathbf{F}(\boldsymbol{\phi})$ contains 9 gradient vectors, but the space spanned by these vectors is only of dimension 3 as explained before. Equation (3.15) says that $\mathbf{d}$ has to be orthogonal to these vectors to close the chain and is therefore included in the tangent space of $\mathbf{F}(\boldsymbol{\phi})$. The union of the tangent spaces at all points $\boldsymbol{\phi} \in SM^{n-3}$ is called the *tangent bundle of* SM^{n-3} and is the solution of the translation equation. □

Fixing the rotations of the n-bar mechanism in a non-planar configuration is equivalent to choosing a point of SM^{n-3}. Then, all possible translational solutions, according to Theorem 3.1, will be included in the tangent space of the SM^{n-3} at that point (Figure 3.4). Since the tangent space is a linear space of dimension $r = n-3$, we can find r vectors as a basis of all solutions for the translational equation.

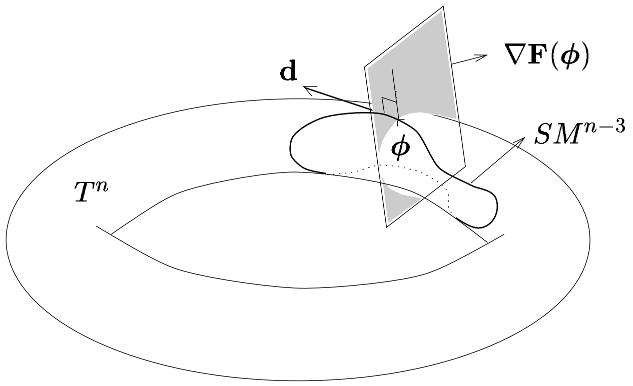


Figure 3.4: **d** must be contained in the tangent space of SM^{n-3} of the corresponding spherical mechanism to keep the n-bar mechanism closed.

From equation (3.13), it follows that $\mathbf{d}$ has to be orthogonal to

$$\mathbf{p}_x = (n_{1x}\; n_{2x}\; \dots\; n_{nx}) \;, \quad \mathbf{p}_y = (n_{1y}\; n_{2y}\; \dots\; n_{ny}) \;\; \text{and} \;\; \mathbf{p}_z = (n_{1z}\; n_{2z}\; \dots\; n_{nz}) \;,$$

which are linearly independent, except for a global singularity.

Then, $\mathbf{d}$ has to be included in the orthogonal space of the space defined by $\mathbf{p}_x$, $\mathbf{p}_y$ and $\mathbf{p}_z$ in $\mathbb{R}^n$. We can get a basis of this space by orthonormalizing, using the Gram-Schmidt algorithm, these three vectors with a basis of $\mathbb{R}^n$, e.g. the canonical one, $[\,(1\;0\;\dots\;0)\;,\;(0\;1\;\dots\;0)\;,\;\dots\;\dots\;(0\;0\;\dots\;1)\,]$. We will get three vectors as a basis of the space generated by $\mathbf{p}_x$, $\mathbf{p}_y$ and $\mathbf{p}_z$, and $n-3$ vectors as a basis of its orthogonal space.

3.3.3 The Tangent Space in Terms of Bar Directions

A basis of the tangent space is also the set of vectors $\dfrac{\partial \boldsymbol{\phi}}{\partial \psi_i}$, where $i = 1, \dots, r$, as stated in Theorem 3.1. Next we obtain these derivatives in terms of bar directions.

Theorem 3.2. *The derivatives of $\boldsymbol{\phi}$, expressed in terms of bar directions, for a parameterization of $n-3$ variables,*

$$\boldsymbol{\psi} = (\psi_1, \psi_2, \dots, \psi_r) = (\phi_1, \phi_2, \dots, \phi_{a-1}, \phi_{a+1}, \dots, \phi_{b-1}, \phi_{b+1}, \dots, \phi_{c-1}, \phi_{c+1}, \dots, \phi_n) \;,$$

are:

- *For the variables taken as parameters, $\phi_i (i \neq a, b, c)$,*

$$\frac{\partial \phi_i}{\partial \psi_j} = 1 \quad \text{if } \psi_j = \phi_i \text{ and}$$
$$\frac{\partial \phi_i}{\partial \psi_j} = 0 \quad \text{if } \psi_j \neq \phi_i \;.$$

- *For variables ϕ_a, ϕ_b and ϕ_c,*

$$\begin{aligned} \frac{\partial \phi_a}{\partial \psi_i} &= -\frac{|\,\mathbf{n}_i\;\mathbf{n}_b\;\mathbf{n}_c\,|}{|\,\mathbf{n}_a\;\mathbf{n}_b\;\mathbf{n}_c\,|} \\ \frac{\partial \phi_b}{\partial \psi_i} &= -\frac{|\,\mathbf{n}_a\;\mathbf{n}_i\;\mathbf{n}_c\,|}{|\,\mathbf{n}_a\;\mathbf{n}_b\;\mathbf{n}_c\,|} \\ \frac{\partial \phi_c}{\partial \psi_i} &= -\frac{|\,\mathbf{n}_a\;\mathbf{n}_b\;\mathbf{n}_i\,|}{|\,\mathbf{n}_a\;\mathbf{n}_b\;\mathbf{n}_c\,|} \;. \end{aligned} \tag{3.16}$$

Proof. The derivatives of the variables taken as parameters are simple to obtain. To get the derivatives of the variables which are not used as parameters (ϕ_a, ϕ_b, ϕ_c), with respect to the parameters, the implicit function theorem is used. To this end, let us define

$$\varphi_a(\boldsymbol{\psi}) \triangleq \phi_a, \quad \varphi_b(\boldsymbol{\psi}) \triangleq \phi_b, \quad \varphi_c(\boldsymbol{\psi}) \triangleq \phi_c,$$

when $\mathbf{F}(\boldsymbol{\phi}) = \mathbf{I}$.

By the implicit function theorem, we can differentiate $\mathbf{F}(\boldsymbol{\psi})$ with respect to any ψ_i as follows:

$$\frac{\partial \mathbf{F}(\boldsymbol{\psi})}{\partial \psi_i} = \frac{\partial \mathbf{F}(\boldsymbol{\phi})}{\partial \phi_i} + \frac{\partial \mathbf{F}(\boldsymbol{\phi})}{\partial \phi_a}\frac{\partial \varphi_a(\boldsymbol{\psi})}{\partial \psi_i} + \frac{\partial \mathbf{F}(\boldsymbol{\phi})}{\partial \phi_b}\frac{\partial \varphi_b(\boldsymbol{\psi})}{\partial \psi_i} + \frac{\partial \mathbf{F}(\boldsymbol{\phi})}{\partial \phi_c}\frac{\partial \varphi_c(\boldsymbol{\psi})}{\partial \psi_i} = 0 \ . \quad (3.17)$$

By Lemma 3.1,

$$\mathbf{N}_a \frac{\partial \varphi_a(\boldsymbol{\psi})}{\partial \psi_i} + \mathbf{N}_b \frac{\partial \varphi_b(\boldsymbol{\psi})}{\partial \psi_i} + \mathbf{N}_c \frac{\partial \varphi_c(\boldsymbol{\psi})}{\partial \psi_i} = -\mathbf{N}_i \quad (3.18)$$

or equivalently,

$$\mathbf{n}_a \frac{\partial \varphi_a(\boldsymbol{\psi})}{\partial \psi_i} + \mathbf{n}_b \frac{\partial \varphi_b(\boldsymbol{\psi})}{\partial \psi_i} + \mathbf{n}_c \frac{\partial \varphi_c(\boldsymbol{\psi})}{\partial \psi_i} = -\mathbf{n}_i \ .$$

By solving this linear system using Cramer's rule, we get the announced result. □

Corollary 3.6. *For the particular case of a trivial parameterization of the first* $n-3$ *variables, matrix* $\mathbf{K}$ *in Equation* (3.14) *can now be easily calculated in terms of bar directions:*

$$\mathbf{K} = \begin{pmatrix} 1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 1 \\ -\frac{|\mathbf{n}_1\ \mathbf{n}_{n-1}\ \mathbf{n}_n|}{|\mathbf{n}_{n-2}\ \mathbf{n}_{n-1}\ \mathbf{n}_n|} & \cdots & -\frac{|\mathbf{n}_r\ \mathbf{n}_{n-1}\ \mathbf{n}_n|}{|\mathbf{n}_{n-2}\ \mathbf{n}_{n-1}\ \mathbf{n}_n|} \\ -\frac{|\mathbf{n}_{n-2}\ \mathbf{n}_1\ \mathbf{n}_n|}{|\mathbf{n}_{n-2}\ \mathbf{n}_{n-1}\ \mathbf{n}_n|} & \cdots & -\frac{|\mathbf{n}_{n-2}\ \mathbf{n}_r\ \mathbf{n}_n|}{|\mathbf{n}_{n-2}\ \mathbf{n}_{n-1}\ \mathbf{n}_n|} \\ -\frac{|\mathbf{n}_{n-2}\ \mathbf{n}_{n-1}\ \mathbf{n}_1|}{|\mathbf{n}_{n-2}\ \mathbf{n}_{n-1}\ \mathbf{n}_n|} & \cdots & -\frac{|\mathbf{n}_{n-2}\ \mathbf{n}_{n-1}\ \mathbf{n}_r|}{|\mathbf{n}_{n-2}\ \mathbf{n}_{n-1}\ \mathbf{n}_n|} \end{pmatrix} . \quad (3.19)$$

Notice that most of the values of $\mathbf{K}$ are 0.

This basis, although comparatively much simpler to get than the one obtained as the orthogonal space of vectors $\mathbf{p}_x$, $\mathbf{p}_y$ and $\mathbf{p}_z$ in Section 3.3.2, has a drawback: it is only valid outside singularities of the parameterization, i.e. when $|\ \mathbf{n}_a\ \mathbf{n}_b\ \mathbf{n}_c\ | \neq 0$. However, this can be easily overcome, since it is always possible to find three non-coplanar bars, provided that the mechanism is not planar.

It can be seen from Figure 2.1 that $|\ \mathbf{n}_{n-2}\ \mathbf{n}_{n-1}\ \mathbf{n}_n\ | = \sin\phi_{n-1}$. Then the parameterization is valid only when $\sin\phi_{n-1} \neq 0$, which is equivalent to say that the last three bars are not coplanar, as already mentioned.

Going back to the general case, the derivatives of ϕ_a, ϕ_b and ϕ_c have a simple geometric interpretation: $\mathbf{n}_a$, $\mathbf{n}_b$ and $\mathbf{n}_c$ constitute a normalized (but non-orthogonal) basis in $\mathbb{R}^3$, because the used parameterization is only valid if $\mathbf{n}_a$, $\mathbf{n}_b$ and $\mathbf{n}_c$ are not coplanar. It can be easily seen that $-\frac{\partial \phi_a}{\partial \psi_i}$, $-\frac{\partial \phi_b}{\partial \psi_i}$ and $-\frac{\partial \phi_c}{\partial \psi_i}$ are just the components of $\mathbf{n}_i$ in the basis defined by $\mathbf{n}_a$, $\mathbf{n}_b$ and $\mathbf{n}_c$ in $\mathbb{R}^3$ (Figure 3.5).

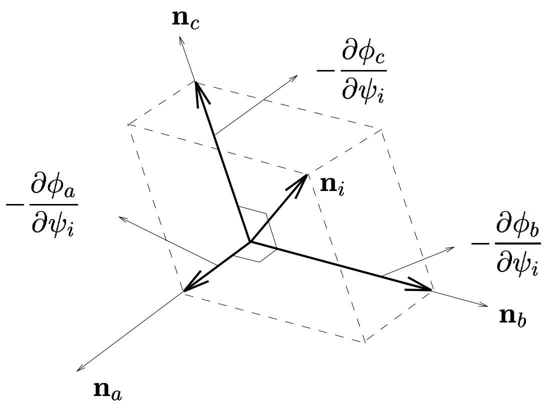

$\mathbf{n}_c$
$-\frac{\partial \phi_c}{\partial \psi_i}$
$-\frac{\partial \phi_a}{\partial \psi_i}$
$\mathbf{n}_i$
$-\frac{\partial \phi_b}{\partial \psi_i}$
$\mathbf{n}_b$
$\mathbf{n}_a$

Figure 3.5: Geometric interpretation of $\phi_a z$, ϕ_b and ϕ_c.

Combining Theorem 3.1 with Theorem 3.2, we obtain:

$$\begin{aligned} d_a &= -\sum_{\substack{i=1\\ i\neq a,b,c}}^{n} \frac{|\ \mathbf{n}_i\ \mathbf{n}_b\ \mathbf{n}_c\ |}{|\ \mathbf{n}_a\ \mathbf{n}_b\ \mathbf{n}_c\ |}\, d_i \\ d_b &= -\sum_{\substack{i=1\\ i\neq a,b,c}}^{n} \frac{|\ \mathbf{n}_a\ \mathbf{n}_i\ \mathbf{n}_c\ |}{|\ \mathbf{n}_a\ \mathbf{n}_b\ \mathbf{n}_c\ |}\, d_i \\ d_c &= -\sum_{\substack{i=1\\ i\neq a,b,c}}^{n} \frac{|\ \mathbf{n}_a\ \mathbf{n}_b\ \mathbf{n}_i\ |}{|\ \mathbf{n}_a\ \mathbf{n}_b\ \mathbf{n}_c\ |}\, d_i\ , \end{aligned} \tag{3.20}$$

which is another version of the translation equation (3.13) for points outside singularities of the parameterization. This is an important result because of its simplicity (compare it to the development in [64] using spherical trigonometry).

It is important to remark that these results have been derived without requiring the n-bar mechanism to be orthogonal and are, therefore, valid for any generalized n-bar mechanism.

3.3.4 Higher Order Derivatives in Terms of Bar Directions

The second derivatives of the variables that are not parameters can also be obtained relatively easily in terms of bar directions.

Lemma 3.7. *The second derivatives of $\boldsymbol{\phi}$ in terms of bar directions for a parameterization of $n-3$ variables,*

$$\boldsymbol{\psi} = (\psi_1, \psi_2, \ldots, \psi_r) = (\phi_1, \phi_2, \ldots, \phi_{a-1}, \phi_{a+1}, \ldots, \phi_{b-1}, \phi_{b+1}, \ldots, \phi_{c-1}, \phi_{c+1}, \ldots, \phi_n)\ ,$$

are:

$$\frac{\partial^2\phi_a}{\partial\psi_i\partial\psi_j} = -\frac{|\ \mathbf{t}\ \mathbf{n}_b\ \mathbf{n}_c\ |}{|\ \mathbf{n}_a\ \mathbf{n}_b\ \mathbf{n}_c\ |}\ , \quad \frac{\partial^2\phi_b}{\partial\psi_i\partial\psi_j} = -\frac{|\ \mathbf{n}_a\ \mathbf{t}\ \mathbf{n}_c\ |}{|\ \mathbf{n}_a\ \mathbf{n}_b\ \mathbf{n}_c\ |}\ , \quad \frac{\partial^2\phi_c}{\partial\psi_i\partial\psi_j} = -\frac{|\ \mathbf{n}_a\ \mathbf{n}_b\ \mathbf{t}\ |}{|\ \mathbf{n}_a\ \mathbf{n}_b\ \mathbf{n}_c\ |}\ .$$

$\mathbf{t}$ *is the* vector linear invariant *(as defined in Lemma 3.1) of the skew-symmetric matrix*

$$\mathbf{T} = \mathcal{T}^i_{abcj} + \mathcal{T}^a_{abcj}\frac{\partial\phi_a}{\partial\psi_i} + \mathcal{T}^b_{abcj}\frac{\partial\phi_b}{\partial\psi_i} + \mathcal{T}^c_{abcj}\frac{\partial\phi_c}{\partial\psi_i}\ , \tag{3.21}$$

where

$$\mathcal{T}^r_{abcj} = \widetilde{\mathbf{N}_r\mathbf{N}_j} + \widetilde{\mathbf{N}_r\mathbf{N}_a}\frac{\partial\phi_a}{\partial\psi_j} + \widetilde{\mathbf{N}_r\mathbf{N}_b}\frac{\partial\phi_b}{\partial\psi_j} + \widetilde{\mathbf{N}_r\mathbf{N}_c}\frac{\partial\phi_c}{\partial\psi_j}\ , \qquad \textit{and} \tag{3.22}$$

$$\widetilde{\mathbf{N}_r\mathbf{N}_s} = \begin{cases} \mathbf{N}_r\mathbf{N}_s & \textit{if } r < s, \\ \mathbf{N}_s\mathbf{N}_r & \textit{if } r > s\ . \end{cases} \tag{3.23}$$

Proof. Derivating (3.17), we get

$$\frac{\partial^2\mathbf{F}(\boldsymbol{\psi})}{\partial\psi_i\partial\psi_j} = \mathbf{T} + \frac{\partial\mathbf{F}(\boldsymbol{\phi})}{\partial\phi_a}\frac{\partial^2\varphi_a(\boldsymbol{\psi})}{\partial\psi_i\partial\psi_j} + \frac{\partial\mathbf{F}(\boldsymbol{\phi})}{\partial\phi_b}\frac{\partial^2\varphi_b(\boldsymbol{\psi})}{\partial\psi_i\partial\psi_j} + \frac{\partial\mathbf{F}(\boldsymbol{\phi})}{\partial\phi_c}\frac{\partial^2\varphi_c(\boldsymbol{\psi})}{\partial\psi_i\partial\psi_j} = 0\ , \tag{3.24}$$

where

$$\begin{aligned}
\mathbf{T} = {} & \frac{\partial^2\mathbf{F}(\boldsymbol{\phi})}{\partial\phi_i\partial\phi_j} + \frac{\partial^2\mathbf{F}(\boldsymbol{\phi})}{\partial\phi_i\partial\phi_a}\frac{\partial\varphi_a(\boldsymbol{\psi})}{\partial\psi_j} + \frac{\partial^2\mathbf{F}(\boldsymbol{\phi})}{\partial\phi_i\partial\phi_b}\frac{\partial\varphi_b(\boldsymbol{\psi})}{\partial\psi_j} + \frac{\partial^2\mathbf{F}(\boldsymbol{\phi})}{\partial\phi_i\partial\phi_c}\frac{\partial\varphi_c(\boldsymbol{\psi})}{\partial\psi_j} + \\
& + [\frac{\partial^2\mathbf{F}(\boldsymbol{\phi})}{\partial\phi_a\partial\phi_j} + \frac{\partial^2\mathbf{F}(\boldsymbol{\phi})}{\partial\phi_a\partial\phi_a}\frac{\partial\varphi_a(\boldsymbol{\psi})}{\partial\psi_j} + \frac{\partial^2\mathbf{F}(\boldsymbol{\phi})}{\partial\phi_a\partial\phi_b}\frac{\partial\varphi_b(\boldsymbol{\psi})}{\partial\psi_j} + \frac{\partial^2\mathbf{F}(\boldsymbol{\phi})}{\partial\phi_a\partial\phi_c}\frac{\partial\varphi_c(\boldsymbol{\psi})}{\partial\psi_j}]\ \frac{\partial\varphi_a(\boldsymbol{\psi})}{\partial\psi_i} + \\
& + [\frac{\partial^2\mathbf{F}(\boldsymbol{\phi})}{\partial\phi_b\partial\phi_j} + \frac{\partial^2\mathbf{F}(\boldsymbol{\phi})}{\partial\phi_b\partial\phi_a}\frac{\partial\varphi_a(\boldsymbol{\psi})}{\partial\psi_j} + \frac{\partial^2\mathbf{F}(\boldsymbol{\phi})}{\partial\phi_b\partial\phi_b}\frac{\partial\varphi_b(\boldsymbol{\psi})}{\partial\psi_j} + \frac{\partial^2\mathbf{F}(\boldsymbol{\phi})}{\partial\phi_b\partial\phi_c}\frac{\partial\varphi_c(\boldsymbol{\psi})}{\partial\psi_j}]\ \frac{\partial\varphi_b(\boldsymbol{\psi})}{\partial\psi_i} + \\
& \quad + [\frac{\partial^2\mathbf{F}(\boldsymbol{\phi})}{\partial\phi_c\partial\phi_j} + \frac{\partial^2\mathbf{F}(\boldsymbol{\phi})}{\partial\phi_c\partial\phi_a}\frac{\partial\varphi_a(\boldsymbol{\psi})}{\partial\psi_j} + \frac{\partial^2\mathbf{F}(\boldsymbol{\phi})}{\partial\phi_c\partial\phi_b}\frac{\partial\varphi_b(\boldsymbol{\psi})}{\partial\psi_j} + \frac{\partial^2\mathbf{F}(\boldsymbol{\phi})}{\partial\phi_c\partial\phi_c}\frac{\partial\varphi_c(\boldsymbol{\psi})}{\partial\psi_j}]\ \frac{\partial\varphi_c(\boldsymbol{\psi})}{\partial\psi_i}\ .
\end{aligned}$$

Using the definition of $\mathcal{T}^r_{abcj}$ (3.22) and $\widetilde{\mathbf{N}_r\mathbf{N}_s}$ (3.22), we can write $\mathbf{T}$ as in (3.21).

Then, from (3.24),

$$\mathbf{T} + \mathbf{N}_a\frac{\partial^2\varphi_a(\boldsymbol{\psi})}{\partial\psi_i\partial\psi_j} + \mathbf{N}_b\frac{\partial^2\varphi_b(\boldsymbol{\psi})}{\partial\psi_i\partial\psi_j} + \mathbf{N}_c\frac{\partial^2\varphi_c(\boldsymbol{\psi})}{\partial\psi_i\partial\psi_j} = 0\ ,$$

which indicates that $\mathbf{T}$ is a skew-symmetric matrix. If we define $\mathbf{t} = \begin{pmatrix} t_{32} & t_{13} & t_{21}\end{pmatrix}^t$ we get the following linear system

$$\mathbf{n}_a\frac{\partial^2\varphi_a(\boldsymbol{\psi})}{\partial\psi_i\partial\psi_j} + \mathbf{n}_b\frac{\partial^2\varphi_b(\boldsymbol{\psi})}{\partial\psi_i\partial\psi_j} + \mathbf{n}_c\frac{\partial^2\varphi_c(\boldsymbol{\psi})}{\partial\psi_i\partial\psi_j} = -\mathbf{t}\ ,$$

which can be solved using Cramer's rule, getting the announced result. □

Note that vector $\mathbf{t}$ depends only on the directions of bars a, b, c, i and j (the derivatives $\partial\phi_r/\partial\psi_i$ can be computed from bar directions according to Theorem 3.2).

Higher order derivatives have the same appearance, the only difference being the expression of $\mathbf{t}$, which becomes more complex as the order increases. The derivatives would still depend only on the directions of bars a, b, c and on the bars associated with the parameters with respect to which we are differentiating.

Corollary 3.7. *The second derivatives of $\boldsymbol{\phi}$ for $i < j < a < b < c$ can be computed as in Lemma 3.7 with vector $\mathbf{t}$ being*

$$\mathbf{t} = (\mathbf{n}_j \wedge \mathbf{n}_a)\frac{|\,\mathbf{n}_i\;\mathbf{n}_b\;\mathbf{n}_c\,|}{|\,\mathbf{n}_a\;\mathbf{n}_b\;\mathbf{n}_c\,|} + (\mathbf{n}_b \wedge \mathbf{n}_c)\frac{|\,\mathbf{n}_a\;\mathbf{n}_b\;\mathbf{n}_j\,||\,\mathbf{n}_a\;\mathbf{n}_i\;\mathbf{n}_c\,|}{|\,\mathbf{n}_a\;\mathbf{n}_b\;\mathbf{n}_c\,|^2} .$$

Proof. Since $i < j < a < b < c$, we can write $\mathcal{T}^i_{abcj}$ (Equation (3.22)) as:

$$\mathcal{T}^i_{abcj} = \mathbf{N}_i\mathbf{N}_j + \mathbf{N}_i\mathbf{N}_a\frac{\partial\phi_a}{\partial\psi_j} + \mathbf{N}_i\mathbf{N}_b\frac{\partial\phi_b}{\partial\psi_j} + \mathbf{N}_i\mathbf{N}_c\frac{\partial\phi_c}{\partial\psi_j} =$$
$$= \mathbf{N}_i(\mathbf{N}_j + \mathbf{N}_a\frac{\partial\phi_a}{\partial\psi_j} + \mathbf{N}_b\frac{\partial\phi_b}{\partial\psi_j} + \mathbf{N}_c\frac{\partial\phi_c}{\partial\psi_j}) .$$

According to Equation(3.18),

$$\mathbf{N}_j + \mathbf{N}_a\frac{\partial\phi_a}{\partial\psi_j} + \mathbf{N}_b\frac{\partial\phi_b}{\partial\psi_j} + \mathbf{N}_c\frac{\partial\phi_c}{\partial\psi_j} = 0 .$$

Thus, $\mathcal{T}^i_{abcj} = 0$. Matrix $\mathcal{T}^a_{abcj}$ can be expressed as the Lie bracket of $\mathbf{N}_j$ and $\mathbf{N}_a$:

$$\mathcal{T}^a_{abcj} = \mathbf{N}_j\mathbf{N}_a + \mathbf{N}_a\mathbf{N}_a\frac{\partial\phi_a}{\partial\psi_j} + \mathbf{N}_a\mathbf{N}_b\frac{\partial\phi_b}{\partial\psi_j} + \mathbf{N}_a\mathbf{N}_c\frac{\partial\phi_c}{\partial\psi_j} =$$
$$= \mathbf{N}_j\mathbf{N}_a - \mathbf{N}_a\mathbf{N}_j + \mathbf{N}_a(\mathbf{N}_j + \mathbf{N}_a\frac{\partial\phi_a}{\partial\psi_j} + \mathbf{N}_b\frac{\partial\phi_b}{\partial\psi_j} + \mathbf{N}_c\frac{\partial\phi_c}{\partial\psi_j}) = \mathbf{N}_j\mathbf{N}_a - \mathbf{N}_a\mathbf{N}_j = [\mathbf{N}_j, \mathbf{N}_a] . \tag{3.25}$$

Similarly, we get:

$$\mathcal{T}^b_{abcj} = [\mathbf{N}_b, \mathbf{N}_c]\frac{|\,\mathbf{n}_a\;\mathbf{n}_b\;\mathbf{n}_j\,|}{|\,\mathbf{n}_a\;\mathbf{n}_b\;\mathbf{n}_c\,|} , \tag{3.26}$$
$$\mathcal{T}^c_{abcj} = 0 .$$

Then,

$$\mathbf{T} = [\mathbf{N}_j, \mathbf{N}_a]\frac{|\,\mathbf{n}_i\;\mathbf{n}_b\;\mathbf{n}_c\,|}{|\,\mathbf{n}_a\;\mathbf{n}_b\;\mathbf{n}_c\,|} + [\mathbf{N}_b, \mathbf{N}_c]\frac{|\,\mathbf{n}_a\;\mathbf{n}_b\;\mathbf{n}_j\,||\,\mathbf{n}_a\;\mathbf{n}_i\;\mathbf{n}_c\,|}{|\,\mathbf{n}_a\;\mathbf{n}_b\;\mathbf{n}_c\,|^2} . \tag{3.27}$$

The Lie brackets of skew-symmetric matrices have a simple formulation in terms of its vector linear invariant. Some tedious, but straightforward, manipulations yield to:

$$\mathrm{vect}([\mathbf{N}_r, \mathbf{N}_s]) = \mathbf{n}_r \wedge \mathbf{n}_s ,$$

where vect($\mathbf{A}$) is the vector linear invariant of matrix $\mathbf{A}$. This is a well-known result derived from the isomorphism between the Lie algebra of the set of vectors with the cross product and the Lie algebra of the skew-symmetric matrices.

Expressing Equation (3.27) with the linear invariants and substituting the Lie brackets by the cross products, we obtain the mentioned expression for $\mathbf{t}$. □

For the orthogonal n-bars mechanism and considering a trivial parameterization of the first $n-3$ variables, the second derivatives of ϕ_{n-2}, ϕ_{n-1} and ϕ_n with respect to the parameters are:

$$\begin{aligned}\frac{\partial^2\phi_{n-2}}{\partial\psi_i\partial\psi_j} &= -\frac{|\,\mathbf{t}_{ij}\;\mathbf{n}_{n-1}\;\mathbf{n}_n\,|}{|\,\mathbf{n}_{n-2}\;\mathbf{n}_{n-1}\;\mathbf{n}_n\,|}\\ \frac{\partial^2\phi_{n-1}}{\partial\psi_i\partial\psi_j} &= -\frac{|\,\mathbf{n}_{n-2}\;\mathbf{t}_{ij}\;\mathbf{n}_n\,|}{|\,\mathbf{n}_{n-2}\;\mathbf{n}_{n-1}\;\mathbf{n}_n\,|}\\ \frac{\partial^2\phi_{n}}{\partial\psi_i\partial\psi_j} &= -\frac{|\,\mathbf{n}_{n-2}\;\mathbf{n}_{n-1}\;\mathbf{t}_{ij}\,|}{|\,\mathbf{n}_{n-2}\;\mathbf{n}_{n-1}\;\mathbf{n}_n\,|}\,,\end{aligned} \tag{3.28}$$

with

$$\mathbf{t}_{ij} = (\mathbf{n}_j \wedge \mathbf{n}_{n-2})\frac{|\,\mathbf{n}_i\;\mathbf{n}_{n-1}\;\mathbf{n}_n\,|}{|\,\mathbf{n}_{n-2}\;\mathbf{n}_{n-1}\;\mathbf{n}_n\,|} + (\mathbf{n}_{n-1}\wedge\mathbf{n}_n)\frac{|\,\mathbf{n}_{n-2}\;\mathbf{n}_i\;\mathbf{n}_n\,|}{|\,\mathbf{n}_{n-2}\;\mathbf{n}_{n-1}\;\mathbf{n}_n\,|}\frac{|\,\mathbf{n}_{n-2}\;\mathbf{n}_{n-1}\;\mathbf{n}_j\,|}{|\,\mathbf{n}_{n-2}\;\mathbf{n}_{n-1}\;\mathbf{n}_n\,|}\,,$$

where $i \leq j$.

3.3.5 The Tangent Space in Terms of Rotations

Equations (3.16), (3.20) and (3.28) are given in terms of bar directions $\mathbf{n}_i$. If we want them in terms of the rotation variables, we should express directions in terms of rotations using the rotation equation (2.3).

In this section we describe two sets of equations for the tangent space of the SM^{n-3} of the orthogonal n-bar mechanism in terms of rotations. Although both sets are straightforwardly differentiable, the second set of equations is specially simple and requires less operations than the first one. This is the reason for using the second set in the second part of this thesis. These parametric rotation and translation equations are summarized in Table 3.1.

First Set

Theorem 3.3. *The derivatives of $\frac{\partial\phi}{\partial\phi_i}$ for a trivial parameterization $\psi_i = \phi_i$ $(i = 1, 2, \ldots, r)$, outside of singularities of that parameterization, can be computed as:*

$$\begin{pmatrix}\frac{\partial\phi_{n-2}}{\partial\psi_i}\\ \frac{\partial\phi_{n-1}}{\partial\psi_i}\\ \frac{\partial\phi_{n}}{\partial\psi_i}\end{pmatrix} = \mathbf{G}\left(\prod_{k=1}^{n-i-1}\mathbf{M}_{n-k}\right)\begin{pmatrix}0\\1\\0\\0\end{pmatrix}, \tag{3.29}$$

where

$$\mathbf{G} \triangleq \begin{pmatrix} -\frac{1}{\sin\phi_{n-1}} & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -\frac{1}{\sin\phi_{n-1}} & 0 \end{pmatrix}$$

and

$$\mathbf{M}_a \triangleq \begin{pmatrix} \cos\phi_a & \sin\phi_a & 0 & 0 \\ 0 & 0 & \sin\phi_{a-1} & -\cos\phi_{a-1} \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix} . \tag{3.30}$$

Note that $\sin\phi_{n-1} \neq 0$, because it is assumed that we are using a proper parameterization.

Proof. Taking three consecutive bars of the n-bar mechanism, it is easy to show (see Figure 2.1) that

$$\begin{aligned} &\mid \mathbf{n}_{a-1}\ \mathbf{n}_a\ \mathbf{n}_{a+1} \mid = \sin\phi_a \ , \\ &\mathbf{n}_{a-1}\cdot\mathbf{n}_{a+1} = -\cos\phi_a \ , \\ &\mathbf{n}_a = -\cos\phi_{a-1}\mathbf{n}_{a-2} + \sin\phi_{a-1}(\mathbf{n}_{a-2}\times\mathbf{n}_{a-1}) \ . \end{aligned} \tag{3.31}$$

Let us also consider the following two relations:

$$\begin{aligned} &(\mathbf{a}\times\mathbf{b})\cdot(\mathbf{c}\times\mathbf{d}) = (\mathbf{a}\cdot\mathbf{c})(\mathbf{b}\cdot\mathbf{d}) - (\mathbf{a}\cdot\mathbf{d})(\mathbf{b}\cdot\mathbf{c}) \ , \\ &|\mathbf{a}\ \mathbf{b}\ \mathbf{c}||\mathbf{a}\ \mathbf{d}\ \mathbf{e}| - |\mathbf{a}\ \mathbf{b}\ \mathbf{d}||\mathbf{a}\ \mathbf{c}\ \mathbf{e}| + |\mathbf{a}\ \mathbf{b}\ \mathbf{e}||\mathbf{a}\ \mathbf{c}\ \mathbf{d}| = 0 \ . \end{aligned} \tag{3.32}$$

While the former is a classic vectorial relation, the latter is a reduced form of the Grassmann-Plücker relations [60].

Using equations (3.31) and (3.32), we can express any determinant of the type

$$|\mathbf{n}_i\ \mathbf{n}_{n-1}\ \mathbf{n}_n| \ , \quad |\mathbf{n}_{n-2}\ \mathbf{n}_i\ \mathbf{n}_n| \ , \ \text{or} \ |\mathbf{n}_{n-2}\ \mathbf{n}_{n-1}\ \mathbf{n}_i| \ ,$$

in terms of the parameters. To this end, if we define

$$\begin{aligned} &v_a^i \triangleq |\mathbf{n}_i\ \mathbf{n}_{a-1}\ \mathbf{n}_a| \quad \text{and} \\ &w_a^i \triangleq \frac{|\mathbf{n}_{a-2}\ \mathbf{n}_i\ \mathbf{n}_a|}{|\mathbf{n}_{a-2}\ \mathbf{n}_{a-1}\ \mathbf{n}_a|} = \frac{|\mathbf{n}_{a-2}\ \mathbf{n}_i\ \mathbf{n}_a|}{\sin\phi_{a-1}}, \end{aligned}$$

the following two recursive expressions can be obtained:

$$\begin{aligned} v_a^i &= v_{a-1}^i\cos\phi_{a-1} + w_{a-1}^i\sin\phi_{a-1} \ , \\ w_a^i &= v_{a-2}^i\sin\phi_{a-2} - w_{a-2}^i\cos\phi_{a-2} \ . \end{aligned} \tag{3.33}$$

Now, let $\mathbf{z}_a^i \triangleq (v_a^i\ w_a^i\ v_{a-1}^i\ w_{a-1}^i)^t$. Then, using the previous definition of $\mathbf{M}_a$ (3.30), we can write Equation (3.33) in matrix form:

$$\mathbf{z}_a^i = \mathbf{M}_{a-1}\mathbf{z}_{a-1}^i \ .$$

Since $\mathbf{z}_{i+1}^i = (0 \;\; 1 \;\; 0 \;\; 0)^t$, it is clear that

$$\mathbf{z}_a^i = \left(\prod_{k=1}^{a-i-1} \mathbf{M}_{a-k}\right) \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix} . \tag{3.34}$$

Considering

$$\mathbf{z}_n^i = \begin{pmatrix} v_n^i \\ w_n^i \\ v_{n-1}^i \\ w_{n-1}^i \end{pmatrix} = \begin{pmatrix} |\mathbf{n}_i \; \mathbf{n}_{n-1} \; \mathbf{n}_n| \\ |\mathbf{n}_{n-2} \; \mathbf{n}_i \; \mathbf{n}_n| / \sin \phi_{n-1} \\ |\mathbf{n}_i \; \mathbf{n}_{n-2} \; \mathbf{n}_{n-1}| \\ |\mathbf{n}_{n-3} \; \mathbf{n}_i \; \mathbf{n}_{n-1}| / \sin \phi_{n-2} \end{pmatrix} ,$$

we realize the similitude between the first three components and the expressions of the derivatives in (3.16) (with $a = n-2$, $b = n-1$ and $c = n$). Note that the third component of $\mathbf{z}_n^i$ is equivalent to $|\mathbf{n}_{n-2} \; \mathbf{n}_{n-1} \; \mathbf{n}_i|$. The first and third components should be divided by $|\mathbf{n}_{n-2} \; \mathbf{n}_{n-1} \; \mathbf{n}_n|$ and we can get rid of the last one. □

Corollary 3.8. *The translational equations* (3.20)*, and its derivatives with respect to* ϕ_i *(*$i = 1, \ldots, n-3$*), can be rewritten as*

$$\begin{pmatrix} d_{n-2} \\ d_{n-1} \\ d_n \end{pmatrix} = \mathbf{G} \sum_{i=1}^{n-3} \left(\prod_{k=1}^{n-i-1} \mathbf{M}_{n-k}\right) \begin{pmatrix} 0 \\ d_i \\ 0 \\ 0 \end{pmatrix} , \tag{3.35}$$

$$\begin{pmatrix} \frac{\partial d_{n-2}}{\partial \psi_j} \\ \frac{\partial d_{n-1}}{\partial \psi_j} \\ \frac{\partial d_n}{\partial \psi_j} \end{pmatrix} = \mathbf{G} \sum_{i=1}^{j-1} \left[\prod_{k=1}^{j} \mathbf{M}_{n-k} \begin{pmatrix} 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \prod_{k=j+1}^{n-i-1} \mathbf{M}_{n-k} \right] \begin{pmatrix} 0 \\ d_i \\ 0 \\ 0 \end{pmatrix} .$$

The proof follows directly from Theorem 3.1 and Equation (3.29).

Second Set

The second set of equations for the tangent space of the SM^{n-3} can be derived from the following theorem:

Theorem 3.4. *The derivatives of* $\frac{\partial \boldsymbol{\phi}}{\partial \phi_i}$ *for a trivial parameterization* $\psi_i = \phi_i$ $(i = 1, 2, \ldots, r)$ *and the derivatives* $\frac{\partial \mathbf{d}}{\partial d_i}$ *for a parameterization* $\delta_i = d_i$ $(i = 1, 2, \ldots, r)$*, can be calculated in terms of rotations as:*

$$\begin{pmatrix} \frac{\partial \phi_{n-2}}{\partial \psi_i} \\ \frac{\partial \phi_{n-1}}{\partial \psi_i} \\ \frac{\partial \phi_n}{\partial \psi_i} \end{pmatrix} = \begin{pmatrix} \frac{\partial d_{n-2}}{\partial \delta_i} \\ \frac{\partial d_{n-1}}{\partial \delta_i} \\ \frac{\partial d_n}{\partial \delta_i} \end{pmatrix} = \mathbf{L}(\phi_{n-1}) \; \left(\mathbf{A}_i^{n-2}(\boldsymbol{\phi})\right)^t \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} ,$$

where $\mathbf{L}(\phi_{n-1}) \triangleq \frac{-1}{\sin\phi_{n-1}} \begin{pmatrix} 0 & -\sin\phi_{n-1} & \cos\phi_{n-1} \\ \sin\phi_{n-1} & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}$.

Proof. The translation equation (3.13) can also be written as:

$$\mathbf{T}(\boldsymbol{\phi}, \mathbf{d}) = \sum_{i=1}^{n} \mathbf{A}_1^{i-1}(\boldsymbol{\phi}) \begin{pmatrix} d_i \\ 0 \\ 0 \end{pmatrix} = \mathbf{0}\ .$$

Applying the implicit function theorem as in Section 3.3.3 to this equation we get the following expression:

$$\mathbf{A}_1^{i-1}(\boldsymbol{\phi}) \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} + \mathbf{A}_1^{n-3}(\boldsymbol{\phi}) \begin{pmatrix} \frac{\partial d_{n-2}}{\partial \delta_i} \\ 0 \\ 0 \end{pmatrix} + \mathbf{A}_1^{n-2}(\boldsymbol{\phi}) \begin{pmatrix} \frac{\partial d_{n-1}}{\partial \delta_i} \\ 0 \\ 0 \end{pmatrix} + \mathbf{A}_1^{n-1}(\boldsymbol{\phi}) \begin{pmatrix} \frac{\partial d_{n}}{\partial \delta_i} \\ 0 \\ 0 \end{pmatrix} = 0\ .$$

Multiplying by $(\mathbf{A}_1^{n-2})^{-1}$ we get

$$\mathbf{I} \begin{pmatrix} \frac{\partial d_{n-2}}{\partial \delta_i} \\ 0 \\ 0 \end{pmatrix} + \mathbf{A}_{n-2}^{n-2}(\boldsymbol{\phi}) \begin{pmatrix} \frac{\partial d_{n-1}}{\partial \delta_i} \\ 0 \\ 0 \end{pmatrix} + \mathbf{A}_{n-2}^{n-1}(\boldsymbol{\phi}) \begin{pmatrix} \frac{\partial d_{n}}{\partial \delta_i} \\ 0 \\ 0 \end{pmatrix} = -\left(\mathbf{A}_i^{n-3}(\boldsymbol{\phi})\right)^t \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}\ .$$

This is,

$$\begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} \frac{\partial d_{n-2}}{\partial \delta_i} + \begin{pmatrix} 0 \\ \cos\phi_{n-2} \\ \sin\phi_{n-2} \end{pmatrix} \frac{\partial d_{n-1}}{\partial \delta_i} + \begin{pmatrix} -\cos\phi_{n-1} \\ -\sin\phi_{n-2}\sin\phi_{n-1} \\ \cos\phi_{n-2}\sin\phi_{n-1} \end{pmatrix} \frac{\partial d_{n-1}}{\partial \delta_i} = -\left(\mathbf{A}_i^{n-3}(\boldsymbol{\phi})\right)^t \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}\ ,$$

which leads to the aforementioned expression for the derivatives of the translations. Likewise, we get the derivatives for the rotations. Note that $\sin\phi_{n-1} \neq 0$, because it is assumed that we are not in a singularity of the chosen trivial parameterization. □

Corollary 3.9. *The translation equations* (3.20) *in terms of the rotational variables and their derivatives can be expressed as shown in Table 3.1.*

A naive algorithm would require about three times more operations for the first set of equation than for this one. However, with an efficient algorithm that would take into account the components of the matrices and vectors that are zero, the difference would decrease significantly.

Table 3.1: Parametric rotation and translation equations and their derivatives.

	rotations	**translations**
equations	$\begin{array}{lcl} \phi_{n-2} & = & \text{atan2}(\pm a_{21}, \mp a_{31}) \\ \phi_{n-1} & = & \mp\text{acos}(-a_{11}) \\ \phi_n & = & \text{atan2}(\mp a_{12}, \mp a_{13}) \end{array}$	$\begin{pmatrix} d_{n-2} \\ d_{n-1} \\ d_n \end{pmatrix} = \mathbf{L}(\phi_{n-1}) \sum_{i=1}^{n-3} \left[\left(\mathbf{A}_i^{n-2}(\boldsymbol{\phi}) \right)^t \begin{pmatrix} d_i \\ 0 \\ 0 \end{pmatrix} \right]$
derivatives with respect to ψ_i (ϕ) $i = 1, \ldots, n-3$	$\begin{pmatrix} \frac{\partial \phi_{n-2}}{\partial \psi_i} \\ \frac{\partial \phi_{n-1}}{\partial \psi_i} \\ \frac{\partial \phi_n}{\partial \psi_i} \end{pmatrix} = \mathbf{L}(\phi_{n-1}) \left(\mathbf{A}_i^{n-2}(\boldsymbol{\psi}) \right)^t j \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}$	$\begin{pmatrix} \frac{\partial d_{n-2}}{\partial \phi_i} \\ \frac{\partial d_{n-1}}{\partial \phi_i} \\ \frac{\partial d_n}{\partial \phi_i} \end{pmatrix} = \mathbf{L}(\phi_{n-1}) \sum_{k=1}^{i} \left[\left(\mathbf{A}_k^{i-1}(\boldsymbol{\phi}) \mathbf{Q} \mathbf{A}_i^{n-2}(\boldsymbol{\phi}) \right)^t \begin{pmatrix} d_k \\ 0 \\ 0 \end{pmatrix} \right]$
derivatives with respect to ϕ_{n-2}, ϕ_{n-1} and ϕ_n.		$\begin{pmatrix} \frac{\partial d_{n-2}}{\partial \phi_{n-2}} \\ \frac{\partial d_{n-1}}{\partial \phi_{n-2}} \\ \frac{\partial d_n}{\partial \phi_{n-2}} \end{pmatrix} = \mathbf{L}(\phi_{n-1}) \sum_{i=1}^{n-2} \left[\left(\mathbf{A}_i^{n-3}(\boldsymbol{\phi}) \mathbf{Q} \mathbf{A}_{n-2}^{n-2}(\boldsymbol{\phi}) \right)^t \begin{pmatrix} d_i \\ 0 \\ 0 \end{pmatrix} \right]$ $\begin{pmatrix} \frac{\partial d_{n-2}}{\partial \phi_{n-1}} \\ \frac{\partial d_{n-1}}{\partial \phi_{n-1}} \\ \frac{\partial d_n}{\partial \phi_{n-1}} \end{pmatrix} = \mathbf{L}'(\phi_{n-1}) \sum_{i=1}^{n-3} \left[\left(\mathbf{A}_i^{n-2}(\boldsymbol{\phi}) \right)^t \begin{pmatrix} d_i \\ 0 \\ 0 \end{pmatrix} \right]$ $\begin{pmatrix} \frac{\partial d_{n-2}}{\partial \phi_n} \\ \frac{\partial d_{n-1}}{\partial \phi_n} \\ \frac{\partial d_n}{\partial \phi_n} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$
derivatives with respect to δ_i $i = 1, \ldots, n-3$		$\begin{pmatrix} \frac{\partial d_{n-2}}{\partial \delta_i} \\ \frac{\partial d_{n-1}}{\partial \delta_i} \\ \frac{\partial d_n}{\partial \delta_i} \end{pmatrix} = \mathbf{L}(\phi_{n-1}) \left(\mathbf{A}_i^{n-2}(\boldsymbol{\phi}) \right)^t \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}$

notation	
$\mathbf{A}_j^k(\boldsymbol{\phi}) \triangleq \begin{cases} \mathbf{I} & \text{if } j = k+1 \\ \prod_{i=j}^{k} \begin{pmatrix} 0 & -1 & 0 \\ \cos\phi_i & 0 & -\sin\phi_i \\ \sin\phi_i & 0 & \cos\phi_i \end{pmatrix} & \end{cases}$	
$\mathbf{A}(\boldsymbol{\psi}) \triangleq \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} \triangleq \left(\mathbf{A}_1^{n-3}(\boldsymbol{\psi}) \right)^t \begin{pmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}$	$\mathbf{Q} \triangleq \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{pmatrix}$
$\mathbf{L}(\phi_{n-1}) \triangleq \frac{-1}{\sin\phi_{n-1}} \begin{pmatrix} 0 & -\sin\phi_{n-1} & \cos\phi_{n-1} \\ \sin\phi_{n-1} & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}$	$\mathbf{L}'(\phi_{n-1}) \triangleq \frac{1}{\sin^2\phi_{n-1}} \begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & \cos\phi_{n-1} \end{pmatrix}$

Chapter 4

Examples

This chapter contains two examples. The first one is the global analysis of the 4-bar mechanism, including the stratification of SS^1. The second one is a local analysis of a 6-bar mechanism in three different situations: a regular configuration, a singularity of a parameterization and a global singularity of SS^3.

4.1 Global Analysis of the 4-bar Mechanism

The loop equation of the 4-bar mechanism is:

$$\prod_{i=1}^{4} \mathbf{T}(d_i)\mathbf{R}(\phi_i)\mathbf{Z} = \mathbf{I}\ .$$

The self-motion set of its spherical indicatrix will be the set of rotations (points in T^4) that fulfill the rotation equation $\displaystyle\prod_{i=1}^{4}\mathbf{R}(\phi_i)\mathbf{Z} = \mathbf{I}$; that is,

$$\begin{pmatrix} \cos\phi_2\cos\phi_4 + \sin\phi_2\cos\phi_3\sin\phi_4 & -\sin\phi_2\sin\phi_4 & -\cos\phi_2\sin\phi_4 + \sin\phi_2\cos\phi_3\cos\phi_4 \\ \sin\phi_1\sin\phi_2\cos\phi_4 + \cos\phi_1\sin\phi_3\sin\phi_4 - \sin\phi_1\cos\phi_2\cos\phi_3\sin\phi_4 & \cos\phi_1\cos\phi_3 + \sin\phi_1\cos\phi_2\sin\phi_3 & \sin\phi_1\sin\phi_2\sin\phi_4 + \cos\phi_1\sin\phi_3\cos\phi_4 - \sin\phi_1\cos\phi_2\cos\phi_3\cos\phi_4 \\ \cos\phi_1\sin\phi_2\cos\phi_4 + \sin\phi_1\sin\phi_3\sin\phi_4 - \cos\phi_1\cos\phi_2\cos\phi_3\sin\phi_4 & \sin\phi_1\cos\phi_3 + \cos\phi_1\cos\phi_2\sin\phi_3 & \cos\phi_1\sin\phi_2\sin\phi_4 + \sin\phi_1\sin\phi_3\cos\phi_4 - \cos\phi_1\cos\phi_2\cos\phi_3\cos\phi_4 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

As stated in Remark 3.1, we need to solve this system of nine equations only for two equations of the diagonal. Let us take the elements $(1,1)$ and $(2,2)$, which lead to a much simpler system:

$$\left\{ \begin{array}{l} \cos\phi_2\cos\phi_4 + \sin\phi_2\cos\phi_3\sin\phi_4 = 1 \\ \cos\phi_1\cos\phi_3 + \sin\phi_1\cos\phi_2\sin\phi_3 = 1 \end{array} \right. .$$

To solve it, let us suppose that $\sin\phi_2 \sin\phi_4 \neq 0$. Then, from the first equation,

$$\cos\phi_3 = \frac{1-\cos\phi_2\cos\phi_4}{\sin\phi_2\sin\phi_4} \quad \Rightarrow \quad \left(\frac{1-\cos\phi_2\cos\phi_4}{\sin\phi_2\sin\phi_4}\right)^2 \leq 1 \, .$$

Changing $\sin^2\phi_i$ for $(1-\cos^2\phi_i)$, we get

$$\cos^2\phi_2 + \cos^2\phi_4 - 2\cos\phi_2\cos\phi_4 = (\cos\phi_2 - \cos\phi_4)^2 \leq 0 \qquad \Rightarrow \qquad \phi_2 = \pm\phi_4 \, .$$

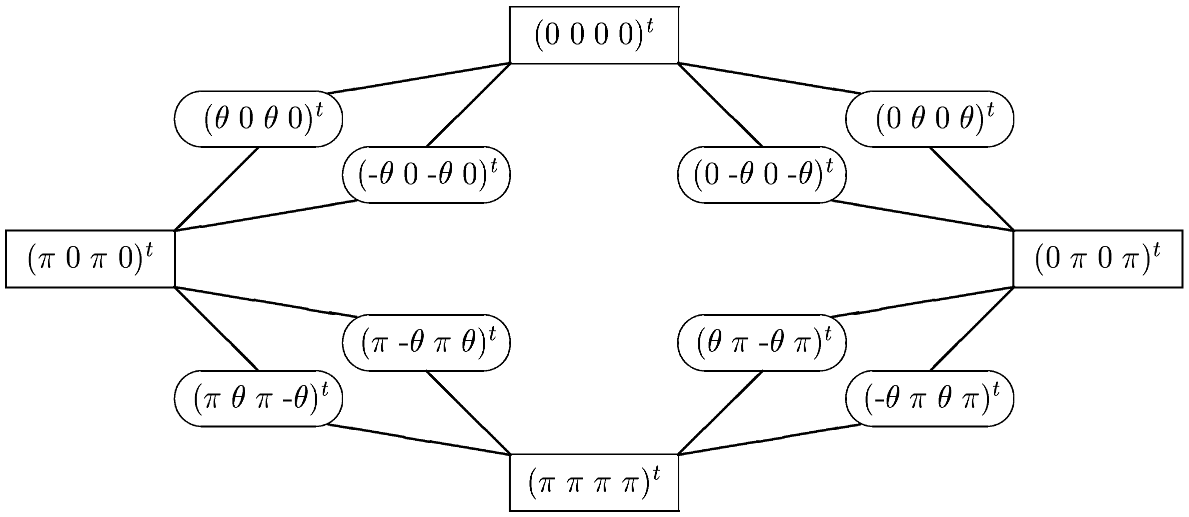
Then, $\cos\phi_3 = \pm 1$ and from the second equation, $\cos\phi_1 = \cos\phi_3 = \pm 1$, leading to the following two solution sets $(\theta \neq 0, \pi)$:

$$\boldsymbol{\phi} = (0\ \theta\ 0\ \theta)^t \quad \text{and} \quad \boldsymbol{\phi} = (\pi\ \theta\ \pi\ \text{-}\theta)^t \, .$$

In a similar way, we can solve the system when $\sin\phi_2 \sin\phi_4 = 0$, leading to:

$$\boldsymbol{\phi} = (\theta\ 0\ \theta\ 0)^t \quad \text{and} \quad \boldsymbol{\phi} = (\theta\ \pi\ \text{-}\theta\ \pi)^t \, .$$

The stratification of this algebraic set leads to 8 strata of dimension 1, whose union is SM^1, and 4 of dimension 0, which are the singular points. As shown in the following diagram, the strata of dimension 1 are connected through those of dimension 0:



$(0\ 0\ 0\ 0)^t$
$(\theta\ 0\ \theta\ 0)^t$
$(\text{-}\theta\ 0\ \text{-}\theta\ 0)^t$
$(0\ \text{-}\theta\ 0\ \text{-}\theta)^t$
$(0\ \theta\ 0\ \theta)^t$
$(\pi\ 0\ \pi\ 0)^t$
$(0\ \pi\ 0\ \pi)^t$
$(\pi\ \text{-}\theta\ \pi\ \theta)^t$
$(\theta\ \pi\ \text{-}\theta\ \pi)^t$
$(\pi\ \theta\ \pi\ \text{-}\theta)^t$
$(\text{-}\theta\ \pi\ \theta\ \pi)^t$
$(\pi\ \pi\ \pi\ \pi)^t$

strata of dim 0
(singularities)
strata of dim 1
$(\theta\ =]0,\pi[)$

Since the number of bars is even, as stated in Corollary 3.4, there are $2^{n-2} = 4$ singular points, which correspond to the strata of dimension 0. Notice that SM^1 is not connected (see Lemma 3.4).

Observe that all the strata are linear, therefore their tangent spaces are constant. For example, the tangent space of the upper left strata on the diagram is $(1\ 0\ 1\ 0)^t$. Thus,

$$\mathbf{d} = \lambda(1\ 0\ 1\ 0)^t, \qquad \lambda \in \mathbb{R} \, ,$$

is the only possible solution of the corresponding equation of translations for any point on these strata. For the other strata, the solutions are:

$$\begin{array}{lll} \mathbf{d} = \lambda(0\ 1\ 0\ 1)^t & \text{upper right strata} & \\ \mathbf{d} = \lambda(1\ 0\ \text{-}1\ 0)^t & \text{lower left strata} & \\ \mathbf{d} = \lambda(0\ 1\ 0\ \text{-}1)^t & \text{lower right strata ,} & \lambda \in \mathbb{R} \end{array}$$

4.2 Local Analysis of the 6-bar Mechanism

The loop equation of the 6-bar mechanism is

$$\prod_{i=1}^{6} \mathbf{T}(d_i)\mathbf{R}(\phi_i)\mathbf{Z} = \mathbf{I} .$$

The loop equation of its spherical indicatrix (the rotation equation) is then

$$\prod_{i=1}^{6} \mathbf{R}(\phi_i)\mathbf{Z} = \mathbf{I} . \tag{4.1}$$

In Section 3.2 we pointed out that, although this expression consists of nine equations, the dimension of the self-motion set is only $n-3=3$. Here, the configuration space is the torus of dimension 6, T^6, and SS^3 is a sub-set of T^6 of dimension 3, with some singular points.

The global singularities correspond to the configurations in which the mechanism is planar. If the number of bars were odd, there would be no global singularities. Since it is even, there are $2^{n-2}=16$ global singularities, which are the points that fulfill conditions (3.5); that is,

$$\begin{array}{ll}
(\pi\ \ \pi\ \ \pi\ \ \pi\ \ \pi\ \ \pi)^t & (\pi\ \ \pi\ \ 0\ \ 0\ \ 0\ \ 0)^t \\
(\pi\ \ \pi\ \ 0\ \ \pi\ \ 0\ \ \pi)^t & (\pi\ \ \pi\ \ \pi\ \ 0\ \ \pi\ \ 0)^t \\
(0\ \ 0\ \ \pi\ \ \pi\ \ 0\ \ 0)^t & (0\ \ \pi\ \ \pi\ \ \pi\ \ 0\ \ \pi)^t \\
(\pi\ \ 0\ \ \pi\ \ \pi\ \ \pi\ \ 0)^t & (0\ \ 0\ \ 0\ \ 0\ \ \pi\ \ \pi)^t \\
(0\ \ \pi\ \ 0\ \ \pi\ \ \pi\ \ \pi)^t & (\pi\ \ 0\ \ \pi\ \ 0\ \ \pi\ \ \pi)^t \\
(0\ \ 0\ \ 0\ \ \pi\ \ \pi\ \ 0)^t & (0\ \ 0\ \ \pi\ \ 0\ \ 0\ \ \pi)^t \\
(0\ \ \pi\ \ 0\ \ 0\ \ \pi\ \ 0)^t & (0\ \ \pi\ \ \pi\ \ 0\ \ 0\ \ 0)^t \\
(\pi\ \ 0\ \ 0\ \ 0\ \ 0\ \ \pi)^t & (\pi\ \ 0\ \ 0\ \ \pi\ \ 0\ \ 0)^t
\end{array}$$

Notice that any of these singularities can be obtained one from any other by applying symmetries (Corollary 3.5).

Excluding these points, the set of trivial parameterizations consisting of 3 consecutive variables provide an atlas for the whole self-motion set (Proposition 3.2).

4.2.1 Finding a General Solution

Now, let us suppose that three rotations and two translations are known: $\phi_1 = 30^o$, $\phi_2 = 45^o$, $\phi_3 = 90^o$, $d_2 = 7$ and $d_3 = 2$. We look for the values of ϕ_4, ϕ_5, ϕ_6, d_1, d_4, d_5 and d_6 that close the mechanism (Figure 4.1).

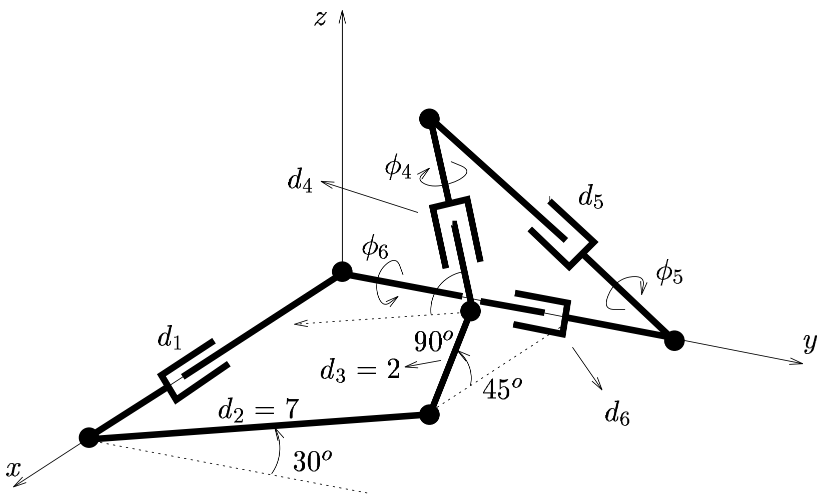


Figure 4.1: The 6-bar mechanism of the example.

Since only three rotations are unknown, they can take only two values, corresponding to the two symmetric closures of the spherical polygon (Figure 3.3).

Let us express loop equation (4.1) as in (3.6):

$$\mathbf{R}(\phi_4)\,\mathbf{Z}\,\mathbf{R}(\phi_5)\,\mathbf{Z}\,\mathbf{R}(\phi_6) \;=\; (\,\mathbf{Z}\,\mathbf{R}(\phi_1)\,\mathbf{Z}\,\mathbf{R}(\phi_2)\,\mathbf{Z}\,\mathbf{R}(\phi_3)\,\mathbf{Z}\,)^{-1} \;=\; \begin{pmatrix} \frac{\sqrt{2}}{4} & \frac{\sqrt{2}}{2} & \frac{\sqrt{3/2}}{2} \\ -\frac{\sqrt{2}}{4} & \frac{\sqrt{2}}{2} & -\frac{\sqrt{3/2}}{2} \\ -\frac{\sqrt{3}}{2} & 0 & \frac{1}{2} \end{pmatrix} .$$

We can calculate ϕ_4, ϕ_5 and ϕ_6 using (3.7):

$$\begin{aligned} \phi_4 &= \operatorname{atan2}(\mp\tfrac{\sqrt{2}}{4}, \pm\tfrac{\sqrt{3}}{2}) = \begin{cases} -22.21^o \\ 157.79^o \end{cases} \\ \phi_5 &= \mp\operatorname{acos}(-\tfrac{\sqrt{2}}{4}) = \mp 110.70^o \\ \phi_6 &= \operatorname{atan2}(\mp\tfrac{\sqrt{2}}{2}, \mp\tfrac{\sqrt{3/2}}{2}) = \begin{cases} -130.89^o \\ 49.11^o \end{cases} . \end{aligned} \tag{4.2}$$

Note that $\phi_4^1 = \phi_4^0 + \pi$, $\phi_5^1 = -\phi_5^0$ and $\phi_6^1 = \phi_6^0 + \pi$ as stated in Section 3.2.6 (superindexes 0 and 1 denote the two different solutions). We arbitrarily take the first solution; we can always get the second one as the symmetric of the first one.

According to Theorem 3.1, the translations have to be included in the tangent space of SM^3 in $\boldsymbol{\phi}^0$. We can get a basis of the tangent space from three different ways:

- As the orthogonal space to vectors $\mathbf{p}_x$, $\mathbf{p}_y$ and $\mathbf{p}_z$ (Section 3.3.2).
- From the derivatives in terms of bar directions (Section 3.3.3).
- From the first and second set of equations for the derivatives in terms of the variables of rotation (Sections 3.3.5 and 3.3.5).

Note that we need a parameterization for the second and third cases, but not for the first one.

Using Vectors $\mathbf{p}_x$, $\mathbf{p}_y$ and $\mathbf{p}_z$

Since

$$\mathbf{n}_i = \left[\prod_{j=1}^{i-1} \mathbf{R}(\phi_j)\mathbf{Z}\right] \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} ,$$

we can calculate the directions of the 6 bars:

$$\mathbf{n}_1 = \begin{pmatrix} 1 & 0 & 0 \end{pmatrix}^t \quad \mathbf{n}_2 = \begin{pmatrix} 0 & \frac{\sqrt{3}}{2} & \frac{1}{2} \end{pmatrix}^t \quad \mathbf{n}_3 = \begin{pmatrix} -\frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{4} & \frac{\sqrt{3/2}}{2} \end{pmatrix}^t$$
$$\mathbf{n}_4 = \begin{pmatrix} \frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{4} & \frac{\sqrt{3/2}}{2} \end{pmatrix}^t \quad \mathbf{n}_5 = \begin{pmatrix} \frac{\sqrt{3}}{\sqrt{7}} & 0 & -\frac{2}{\sqrt{7}} \end{pmatrix}^t \quad \mathbf{n}_6 = \begin{pmatrix} 0 & -1 & 0 \end{pmatrix}^t .$$

We can now obtain a basis of the tangent space of SM^3 in $\boldsymbol{\phi}^0$ as the orthogonal space to vectors $\mathbf{p}_x$, $\mathbf{p}_y$ and $\mathbf{p}_z$ as explained in section 3.3.2. Using the Gram-Schmidt method to orthonormalize these three vectors with the canonical basis of $\mathbb{R}^6$, we get the following basis for the tangent space of the self-motion manifold in $\boldsymbol{\phi}^0$:

$$\mathbf{d} = \begin{pmatrix} d_1 \\ 7 \\ 2 \\ d_4 \\ d_5 \\ d_6 \end{pmatrix} = \begin{pmatrix} 0.7483 & 0 & 0 \\ -0.0926 & 0.6682 & 0 \\ 0.3024 & 0.0327 & 0.7000 \\ -0.5292 & -0.2291 & 0.1000 \\ -0.2449 & 0.2828 & 0.6481 \\ 0 & 0.6481 & -0.2828 \end{pmatrix} \begin{pmatrix} \lambda_1 \\ \lambda_2 \\ \lambda_3 \end{pmatrix} .$$

Solving this linear system, the set of solutions is:

$$\begin{aligned} d_1 &= 4.04 - 1.71\lambda \\ d_4 &= -5.43 + 1.38\lambda \\ d_5 &= 1.85 + 1.12\lambda \\ d_6 &= 7.27 - 0.49\lambda \quad , \quad \text{where} \quad \lambda \in \mathbb{R} . \end{aligned}$$

Assuming that we want to set d_5 to 0, the vectors of rotations and the vector of translations that close the mechanism are

$$\begin{aligned} \boldsymbol{\phi}^0 &= (30 \quad 45 \quad 90 \quad -22.21 \quad -110.70 \quad -130.89)^t \\ \mathbf{d}^0 &= (6.87 \quad 7 \quad 2 \quad -7.72 \quad 0 \quad 8.08)^t . \end{aligned}$$

By applying a symmetry to ϕ_5, as described in Section 3.2.6, we obtain another solution which corresponds to the second solution of (4.2). ϕ_4 and ϕ_6 are incremented by π and ϕ_5 and d_5 change their signs:

$$\begin{array}{lcl} \boldsymbol{\phi}^1 & = & (30 \quad 45 \quad 90 \quad 157.79 \quad 110.70 \quad 49.11)^t \\ \mathbf{d}^1 & = & (6.87 \quad 7 \quad 2 \quad -7.72 \quad 0 \quad 8.08)^t \,. \end{array}$$

Using the Derivatives in Terms of Bar Directions

We could have also used a parameterization to get a basis for the tangent space of SM^3 as described in Section 3.3.3. Taking ϕ_1, ϕ_2 and ϕ_3 as parameters, using (3.16) we have

$$\begin{array}{lll} \frac{\partial\phi_4}{\partial\psi_1} = -\frac{|\mathbf{n}_1\ \mathbf{n}_5\ \mathbf{n}_6|}{|\mathbf{n}_4\ \mathbf{n}_5\ \mathbf{n}_6|} = -0.8081 & \frac{\partial\phi_5}{\partial\psi_1} = -\frac{|\mathbf{n}_4\ \mathbf{n}_1\ \mathbf{n}_6|}{|\mathbf{n}_4\ \mathbf{n}_5\ \mathbf{n}_6|} = -0.6547 & \frac{\partial\phi_6}{\partial\psi_1} = \cdots = 0.2857 \\ \frac{\partial\phi_4}{\partial\psi_2} = -0.3499 & \frac{\partial\phi_5}{\partial\psi_2} = 0.3780 & \frac{\partial\phi_6}{\partial\psi_2} = 0.9897 \\ \frac{\partial\phi_4}{\partial\psi_3} = 0.1429 & \frac{\partial\phi_5}{\partial\psi_3} = 0.9258 & \frac{\partial\phi_6}{\partial\psi_3} = -0.4041 \ . \end{array}$$

Then, a basis of the tangent space is

$$\mathbf{K} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ -0.8081 & -0.3499 & 0.1429 \\ -0.6547 & 0.3780 & 0.9258 \\ 0.2857 & 0.9897 & -0.4041 \end{pmatrix},$$

which yields to the same solution as above. This method to get a basis for the tangent space is faster than the previous one, but we require a valid trivial parameterization.

Using the Derivatives in Terms of Rotations

Finally, we can also get the translation equations directly from the rotations as described in Section 3.3.5, without having to calculate the directions of the bars.

These alternatives also require a trivial parameterization, but not the directions of bars.

First Set. We can rewrite Equation (3.35) for $n = 6$ as:

$$\mathbf{M}_5\mathbf{M}_4 \left[\begin{pmatrix} 0 \\ d_3 \\ 0 \\ 0 \end{pmatrix} + \mathbf{M}_3 \left[\begin{pmatrix} 0 \\ d_2 \\ 0 \\ 0 \end{pmatrix} + \mathbf{M}_2 \begin{pmatrix} 0 \\ d_1 \\ 0 \\ 0 \end{pmatrix} \right] \right] ,$$

which results in the following system:

$$\begin{array}{rcrcr} d_4 & = & -0.81d_1 & - & 2.16 \\ d_5 & = & -0.65d_1 & + & 4.50 \\ d_6 & = & 0.29d_1 & + & 6.12 \end{array} \quad .$$

If we set d_5 to 0, we obtain the same vector of translations as above.

Second set. The parametric translation equation in Table I can also be written as:

$$\begin{pmatrix} d_{n-2} \\ d_{n-1} \\ d_n \end{pmatrix} = \mathbf{L}(\phi_{n-1}) \left(\mathbf{A}_{n-3}^{n-2}(\boldsymbol{\phi})\right)^t \left[\begin{pmatrix} d_{n-3} \\ 0 \\ 0 \end{pmatrix} + \left(\mathbf{A}_{n-4}^{n-4}(\boldsymbol{\phi})\right)^t \left[\begin{pmatrix} d_{n-4} \\ 0 \\ 0 \end{pmatrix} + \cdots + \left(\mathbf{A}_2^2(\boldsymbol{\phi})\right)^t \begin{pmatrix} d_2 \\ -d_1 \\ 0 \end{pmatrix} \right]\right]$$

In our example,

$$\begin{pmatrix} d_4 \\ d_5 \\ d_6 \end{pmatrix} = \mathbf{L}(\phi_5) \left(\mathbf{A}_3^4(\boldsymbol{\phi})\right)^t \left[\begin{pmatrix} d_3 \\ 0 \\ 0 \end{pmatrix} + \left(\mathbf{A}_2^2(\boldsymbol{\phi})\right)^t \begin{pmatrix} d_2 \\ -d_1 \\ 0 \end{pmatrix} \right] ,$$

which leads to the same system as for the first set of equations. However, using this second set, we perform about one third of the operations required by the first set.

4.2.2 Dealing with a Singularity of a Trivial Parameterization

Let us take the same trivial parameterization as above, but now we set

$$\phi_1 = 30^o \; , \quad \phi_2 = 0^o \; , \quad \phi_3 = -150^o \; .$$

Expressing the loop equation as in (3.6), we get the following $\mathbf{A}(\boldsymbol{\psi})$ matrix:

$$\mathbf{A}(\boldsymbol{\psi}) = \begin{pmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{pmatrix} .$$

Since $a_{11} = -1$, we are in a singularity of the parameterization; $|\; \mathbf{n}_4 \; \mathbf{n}_5 \; \mathbf{n}_6 \;| = 0$ and $\sin(\phi_5) = 0$. In other words, taking the first three rotations, there are infinite solutions for the other three rotations. It can be checked that in this case $\phi_5 = 0$ and $\phi_6 = \phi_4 + \pi$.

Now, if we want to calculate a basis for $\mathbf{d}$, we can use the method described in Section 3.3.2 (Gram-Schmidt) or, alternatively, choose another trivial parameterization (e.g. ϕ_2, ϕ_3 and ϕ_4).

4.2.3 Dealing with a Global Singularity

Let us suppose we are on SS^3 in $\boldsymbol{\phi} = \begin{pmatrix} 0 & \pi & 0 & 0 & \pi & 0 \end{pmatrix}$, which is a global singular point according to Lemma 3.2, i.e. all bars are coplanar. This means that no parameterization can be taken and therefore we cannot obtain a basis for **d** using the results of Sections 3.3.3 and 3.3.5. Instead, we can use the method of Section 3.3.2 (Gram-Schmidt). The directions of the bars for this point are

$$\begin{array}{lll} \mathbf{n}_1 = \begin{pmatrix} 1 & 0 & 0 \end{pmatrix}^t & \mathbf{n}_2 = \begin{pmatrix} 0 & 1 & 0 \end{pmatrix}^t & \mathbf{n}_3 = \begin{pmatrix} 1 & 0 & 0 \end{pmatrix}^t \\ \mathbf{n}_4 = \begin{pmatrix} 0 & -1 & 0 \end{pmatrix}^t & \mathbf{n}_5 = \begin{pmatrix} -1 & 0 & 0 \end{pmatrix}^t & \mathbf{n}_6 = \begin{pmatrix} 0 & -1 & 0 \end{pmatrix}^t \ . \end{array}$$

d has to be orthogonal to $\mathbf{p}_x$, $\mathbf{p}_y$ and $\mathbf{p}_z$. Now these three vectors are linearly dependent and therefore its orthogonal space is of dimension $n - 2 = 4$. Imposing that **d** has to be orthogonal to $\mathbf{p}_x$ and $\mathbf{p}_y$ we obtain

$$d_2 = d_4 + d_6 \text{ and } d_5 = d_1 + d_3 \ ,$$

which is clearly the solution (Figure 4.2).

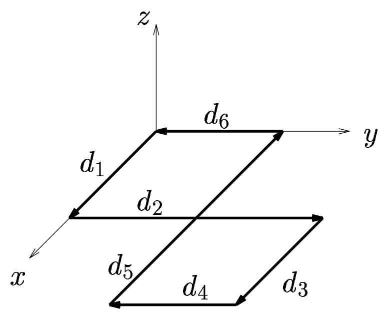


Figure 4.2: Singular configuration of the 6-bar mechanism.

Part II

Interval Methods

Chapter 5

Preliminaries

5.1 Introduction

Interval arithmetic generalizes ordinary arithmetic by using intervals in addition to real numbers. Originally, the purpose of interval analysis was to provide upper and lower bounds on the effect of rounding errors on a computed quantity. Interval mathematics can be said to have begun with the appearance of R. E. Moore's book *Interval Analysis* [36] in 1966. Moore extended the use of interval analysis to bound the effect of errors from all sources, including approximation errors and errors in data.

E. Hansen, in his book *Global Optimization Using Interval Analysis* [17], highlights the importance of interval analysis:

> "*...the introduction of interval analysis is the single most important advance in numerical analysis other than the advent of modern computer and high-level programming languages. The success of its original purpose of error analysis is enough to make this true.*"

However, interval analysis has had other successful applications. The most important one was to solve (at least to a large extent) a previously unsolved problem: the general optimization problem. But interval analysis is also used for solving systems of nonlinear equations and, in general, for improving algorithms by making them more reliable and robust.

In this chapter we focus on a rather narrow part of interval mathematics. Our goal is to introduce the basic tools to solve a system of nonlinear equations, which is the problem we face in inverse kinematics. We shall describe the well known *interval Newton methods* which will be used together with the *specific methods* developed in Chapter 6.

Interval Newton methods for solving systems of nonlinear equations can be used to find and bound *all* solutions in a given region. The bounds are guaranteed to be correct despite errors from rounding, approximation and uncertain data. Moreover, proof of existence and/or uniqueness follows as a by-product of the algorithm and no extra computing is required.

The theory exposed in this chapter is based on Hansen's book [17] and on Kearfott's book *Rigorous Global Search: Continuous Problems*[1] [24]. We present the results in a plain form, avoiding complicated mathematic formalism in order to give the reader a quick idea about interval methods for solving systems of nonlinear equations. References to more detailed bibliography are also given.

5.2 Interval Arithmetic

5.2.1 Interval Numbers and Notation

An *interval number* is a closed real interval $\underline{x} = [a, b]$, consisting of the set $\{x : a \leq x \leq b\}$ of real numbers between and including the endpoints a and b. The set of intervals will be denoted $\mathbb{IR}$, the set of n-dimensional interval vectors, also referred as *boxes*, will be denoted by $\mathbb{IR}^n$, and the set of m by n matrices whose entries are intervals will be denoted by $\mathbb{IR}^{m\times n}$.

We shall use standard notation for noninterval elements: lower case will denote scalar quantities, boldface lower case will denote vectors and boldface upper case will denote matrices. Underscores will be used for the corresponding interval quantities, vectors and matrices. Brackets "[·]" will delimit intervals while parentheses "(·)"will delimit vectors and matrices. We will use left-pointing arrows over an interval quantity for its lower bound and right-pointing ones for its upper bound. Left and right-pointing arrows over interval vectors and matrices will denote the vector or matrix whose components are respectively lower or upper bounds of corresponding components (Table 5.1).

Other symbols and real-valued functions over intervals which will be used are:

- $\check{\underline{x}}$ will denote a representative point in $\underline{x}$; often its center. Similarly, $\check{\underline{\mathbf{x}}}$ and $\check{\underline{\mathbf{X}}}$ will denote a representative point for the box $\underline{\mathbf{x}}$ and interval matrix $\underline{\mathbf{X}}$ respectively.
- *The midpoint* or *center* of an interval $\underline{x}$ is $\mathrm{m}(\underline{x}) = \frac{\overleftarrow{\underline{x}} + \overrightarrow{\underline{x}}}{2}$. The vector or matrix whose entries are midpoints of the entries of the vector $\underline{\mathbf{x}}$ or matrix $\underline{\mathbf{X}}$ will be denoted by $\mathrm{m}(\underline{\mathbf{x}})$ or $\mathrm{m}(\underline{\mathbf{X}})$.
- The *width* of an interval $\underline{x}$ is denoted by $\mathrm{w}(\underline{x}) = \overrightarrow{\underline{x}} - \overleftarrow{\underline{x}}$. The width of a box or an interval matrix is defined component-wise and is denoted by $\mathrm{w}(\underline{\mathbf{x}})$ or $\mathrm{w}(\underline{\mathbf{X}})$.
- The *absolute value* or *magnitude* of an interval is defined as $|\underline{x}| = \max\{|\overleftarrow{\underline{x}}|, |\overrightarrow{\underline{x}}|\}$. The magnitude of a box will be interpreted component-wise, $|\underline{\mathbf{x}}| = (|\underline{x_1}|\ |\underline{x_2}|\ \ldots\ |\underline{x_n}|)$, and similarly for an interval matrix.

[1]The first edition of this book has many flaws. Some of them are corrected in the erratas available at `http://interval.usl.edu/kearfott.html`.

	real	interval
scalars	x	$\underline{x} = [\overleftarrow{\underline{x}}, \overrightarrow{\underline{x}}]$
vectors	$\mathbf{x}$	$\underline{\mathbf{x}} = [\overleftarrow{\underline{\mathbf{x}}}, \overrightarrow{\underline{\mathbf{x}}}] = (\underline{x}_1, \underline{x}_2, \ldots, \underline{x}_n)$, where $\underline{x}_i = [\overleftarrow{\underline{x}}_i, \overrightarrow{\underline{x}}_i]$ $\overleftarrow{\underline{\mathbf{x}}} = (\overleftarrow{\underline{x}}_1, \overleftarrow{\underline{x}}_2, \ldots, \overleftarrow{\underline{x}}_n)$ and $\overrightarrow{\underline{\mathbf{x}}} = (\overrightarrow{\underline{x}}_1, \overrightarrow{\underline{x}}_2, \ldots, \overrightarrow{\underline{x}}_n)$
matrices	$\mathbf{X}$	$\underline{\mathbf{X}} = [\overleftarrow{\underline{\mathbf{X}}}, \overrightarrow{\underline{\mathbf{X}}}] = \begin{pmatrix} \underline{x}_{11} & \underline{x}_{12} & \cdots & \underline{x}_{1n} \\ \underline{x}_{21} & \underline{x}_{22} & \cdots & \underline{x}_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ \underline{x}_{11} & \underline{x}_{12} & \cdots & \underline{x}_{1n} \end{pmatrix}$, where $\underline{x}_{ij} = [\overleftarrow{\underline{x}}_{ij}, \overrightarrow{\underline{x}}_{ij}]$ $\overleftarrow{\underline{\mathbf{X}}} = \begin{pmatrix} \overleftarrow{\underline{x}}_{11} & \overleftarrow{\underline{x}}_{12} & \cdots & \overleftarrow{\underline{x}}_{1n} \\ \overleftarrow{\underline{x}}_{21} & \overleftarrow{\underline{x}}_{22} & \cdots & \overleftarrow{\underline{x}}_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ \overleftarrow{\underline{x}}_{11} & \overleftarrow{\underline{x}}_{12} & \cdots & \overleftarrow{\underline{x}}_{1n} \end{pmatrix}$ and $\overrightarrow{\underline{\mathbf{X}}} = \begin{pmatrix} \overrightarrow{\underline{x}}_{11} & \overrightarrow{\underline{x}}_{12} & \cdots & \overrightarrow{\underline{x}}_{1n} \\ \overrightarrow{\underline{x}}_{21} & \overrightarrow{\underline{x}}_{22} & \cdots & \overrightarrow{\underline{x}}_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ \overrightarrow{\underline{x}}_{11} & \overrightarrow{\underline{x}}_{12} & \cdots & \overrightarrow{\underline{x}}_{1n} \end{pmatrix}$

Table 5.1: Adopted interval notation.

- The *mignitude* of an interval $\underline{x}$ will be defined by $< \underline{x} >= \min_{x \in \underline{x}} |x|$.
- The *norm* of an interval vector is defined as $\|\underline{\mathbf{x}}\| = \big\| |\underline{\mathbf{x}}| \big\|$.
- The *convex hull* of two intervals $\underline{x}$ and $\underline{y}$ is the smallest interval including $\underline{x}$ and $\underline{y}$ and will be denoted by $\underline{x} \cup \underline{y}$.
- If F is a function $F : \mathbb{R}^n \to \mathbb{R}^m$, $\underline{F}^u(\underline{\mathbf{x}})$ denotes[2] the range of F over the box $\underline{\mathbf{x}}$.

Comparison of intervals will be similar to set comparison: $\underline{x} < \underline{y}$ will mean that every element of $\underline{x}$ is smaller than every element of $\underline{y}$, i.e., $\overrightarrow{\underline{x}} < \overleftarrow{\underline{y}}$. Comparison of interval vectors $\underline{\mathbf{x}}, \underline{\mathbf{y}} \in \mathbb{IR}^n$ will be component-wise: $\underline{\mathbf{x}} \leq \underline{\mathbf{y}}$ will mean that $\underline{x}_i \leq \underline{y}_i$ for i between 1 and n, but note that $\underline{\mathbf{x}} < \underline{\mathbf{y}}$ will mean that $\underline{x}_i \leq \underline{y}_i$ for i ranging from 1 to n, and $\underline{x}_i \neq \underline{y}_i$ for at least one i.

5.2.2 Real Interval Arithmetic

The four elementary operations for the arithmetic of real numbers, $+, -, \times$ and $\div$, can be generalized into the interval case. We will use the same symbols for both interval and exact-value operations. If op denotes any one of these operators, then the corresponding operations for interval numbers obey

$$\underline{x}\ \mathsf{op}\ \underline{y} = \{x\ \mathsf{op}\ y \mid x \in \underline{x} \text{ and } y \in \underline{y}\} \quad \text{for } \mathsf{op} \in \{+, -, \times \text{ and } \div\}\ . \tag{5.1}$$

[2]The notation is suggestive of “united” interval extension, a term first introduced by Moore.

Thus, the image of each of the four basic interval operations is the *exact-range* of the corresponding real operation. The following rules can be derived from this definition:

$$\begin{aligned}
\underline{x}+\underline{y} &= [\overleftarrow{x}+\overleftarrow{y}, \overrightarrow{x}+\overrightarrow{y}]\,, && (5.2)\\
\underline{x}-\underline{y} &= [\overleftarrow{x}-\overrightarrow{y}, \overrightarrow{x}-\overleftarrow{y}]\,, && (5.3)\\
\underline{x}\times\underline{y} &= [\min\{\overleftarrow{x}\overleftarrow{y}, \overleftarrow{x}\overrightarrow{y}, \overrightarrow{x}\overleftarrow{y}, \overrightarrow{x}\overrightarrow{y}\}, \max\{\overleftarrow{x}\overleftarrow{y}, \overleftarrow{x}\overrightarrow{y}, \overrightarrow{x}\overleftarrow{y}, \overrightarrow{x}\overrightarrow{y}\}]\,, && (5.4)\\
\underline{x}\div\underline{y} &= \underline{x}\times[1/\overrightarrow{y}, 1/\overleftarrow{y}] \quad \text{if } 0\notin\underline{y}\,. && (5.5)
\end{aligned}$$

The interval quotient $\underline{x}\div\underline{y}$ is undefined when $0\in\underline{y}$. An extension to interval arithmetic for this case based on infinite intervals was first introduced by Hanson [18] and Kahan [22] independently and later corrected by Novoa [40] and Ratz [49] also separately. This arithmetic is called *extended interval arithmetic*, also known as *Kahan-Novoa-Ratz arithmetic*. Here, the set of real intervals $[a,b]\in\mathbb{IR}$ is augmented with the set of complements $]a,b[=[-\infty,a]\cup[b,\infty]$. Then, the division of two ordinary intervals $\underline{x}$ and $\underline{y}$ with $0\in\underline{y}$ is still defined according to (5.1) and can be computed with the following rules:

$$\frac{[\overleftarrow{x},\overrightarrow{x}]}{[\overleftarrow{y},\overrightarrow{y}]} = \begin{cases}
\underline{x}\times[1/\overrightarrow{y}, 1/\overleftarrow{y}] & \text{if } 0\notin\underline{y}\,,\\
[-\infty,\infty] & \text{if } 0\in\underline{x} \text{ and } 0\in\underline{y}\,,\\
[\overrightarrow{x}\overleftarrow{y},\infty] & \text{if } \overrightarrow{x}<0 \text{ and } \overleftarrow{y}<\overrightarrow{y}=0\,,\\
[-\infty,\overrightarrow{x}\overrightarrow{y}]\cup[\overrightarrow{x}\overleftarrow{y},\infty] & \text{if } \overrightarrow{x}<0 \text{ and } \overleftarrow{y}<0<\overrightarrow{y}\,,\\
[-\infty,\overrightarrow{x}\overrightarrow{y}] & \text{if } \overrightarrow{x}<0 \text{ and } 0=\overleftarrow{y}<\overrightarrow{y}\,,\\
[-\infty,\overleftarrow{x}\overleftarrow{y}] & \text{if } 0<\overleftarrow{x} \text{ and } \overleftarrow{y}<\overrightarrow{y}=0\,,\\
[-\infty,\overleftarrow{x}\overleftarrow{y}]\cup[\overleftarrow{x}\overrightarrow{y},\infty] & \text{if } 0<\overleftarrow{x} \text{ and } \overleftarrow{y}<0<\overrightarrow{y}\,,\\
[\overleftarrow{x}\overrightarrow{y},\infty] & \text{if } 0<\overleftarrow{x} \text{ and } 0=\overleftarrow{y}<\overrightarrow{y}\,,\\
\emptyset & \text{if } 0\notin\underline{x} \text{ and } \underline{y}=0\,.
\end{cases} \qquad (5.6)$$

For our purposes, i.e. when solving systems of nonlinear equations, it will suffice to intersect the result of Formula (5.6) with an ordinary interval, to obtain either a single interval, two intervals, or the empty set. Thus, we will not have to face intervals extending to infinity.

5.2.3 Interval Dependency

Although the ranges of interval arithmetic operations are exactly the ranges of the corresponding real operations, this is not the case if the operations are composed. Combined interval expressions, such as $\underline{x}^2-\underline{x}$, can be evaluated applying the primitive operations in sequence. By doing so, we are implicitly assuming that the variable $\underline{x}$ varies independently in the term $\underline{x}^2$ and the term $\underline{x}$, and the resulting value may have excess width. For example, if

$$f(x)=x^2-x\,,$$

evaluating f over the interval $[0,1]$ with interval arithmetic gives

$$[0,1]^2-[0,1]=[0,1]-[0,1]=[-1,1]\,,$$

while the actual range is $f^u([0,1]) = [-1/4, 0]$. When a given variable occurs more than once in an interval computation, it is treated as a different variable and causes widening of computed intervals. In general, one should try to minimize the number of different instances of the same variable in an interval expression in order to avoid an excessive expansion of the resulting interval[3].

A consequence of interval dependency is that algebraic expressions that are equivalent in real arithmetic lead to different results when evaluated in interval arithmetic. For instance, if f above is written as $x(x-1)$, the evaluation over the same interval $[0,1]$ gives a tighter bound:

$$[0,1]\big([0,1]-1\big) = [0,1][-1,0] = [-1,0]\ .$$

Note that real interval arithmetic always computes *rigorous bounds*, in the sense that the actual range is always included.

5.2.4 Rounded Interval Arithmetic

In the previous sections we assumed infinitely precise computations. However, when interval arithmetic is implemented on a computer, the default rounding may result in erroneous results. For example, consider the functions given in [21]:

$$\begin{aligned} a &= x + y - x && \text{where } x = 1\text{e}34 \text{ and } y = 2\ , \\ b &= 9x^4 - y^4 + 2y^2 && \text{where } x = 10864 \text{ and } y = 18817\ . \end{aligned}$$

The correct values are $a = 2$ and $b = 1$. However, by using the IEEE standard double precision arithmetic [57], the results are completely wrong, that is $a = 0$ and $b = 2$.

Nonetheless, if instead of rounding to the nearest machine number, the lower bound of the intervals is rounded down to the largest machine number less than or equal to the exact result and the upper bound is rounded up to the smallest machine number greater or equal than the actual result, then the resulting interval definitely includes all possible values even in cases where rounding errors degenerate results. This process is called *outward rounding.* In the previous example, using double precision interval arithmetic of BIAS/PROFIL interval libraries [28] with outward rounding, the resulting intervals are:

$$\begin{aligned} a &= [0,\ 1.15292\text{e}18] \ \text{ and} \\ b &= [-14,\ 2]\ . \end{aligned}$$

This example is highly degenerate and the resulting intervals are extremely wide, but indeed bound the correct values. Rounded interval arithmetic may sometimes give completely useless results, but at least it does never "lie"! Usually, rounded interval evaluation returns essentially similar results as exact-value evaluation. If it returns wide intervals, it warns the user about degenerate situations.

[3]This is true in most cases, but not always!

Another interesting example can be found in Hansen's book [17]. Evaluating the function

$$f(x,y) = 333.75y^6 + x^2(11x^2y^2 - y^6 - 121y^4 - 2) + 5.5y^8 + \frac{x}{2y}$$

for $x = 77617$ and $y = 33096$, using single, double and extended precision gives:

$f = 1.172603\ldots$	for single precision,
$f = 1.1726039400531\ldots$	for double precision, and
$f = 1.172603940053178\ldots$	for extended precision.

It is a common mistake to believe that if two results agree to some number of digits using different precisions, then those digits are correct. In this example all three results agree in the first seven decimal digits. However, they all are completely incorrect; the right value is $f = -0.827396\ldots$. If we evaluate f using BIAS/PROFIL [28] we obtain a wide interval containing the correct result: $[-8.26414e21, 7.08355e21]$. Here again, the fact that the interval is wide suggests an underlying roundoff problem.

5.3 Interval Functions

An *interval function* is an interval-valued function of one or more interval arguments:

$$\underline{F} : \mathbb{IR}^n \to \mathbb{IR}^m \,.$$

Interval functions are usually related to real-valued functions. For example, the interval function $\sin \underline{x}$ includes all possible values of the real-valued function $\sin x$ over the interval $\underline{x}$. We will call $\sin \underline{x}$ an extension of $\sin x$. In some special cases, we will underline the extension in order to prevent confusion, but we will often use the same symbol to denote both the real valued function and any extension of it. Whether the function is real-valued or an interval function can be distinguished by the nature of its arguments.

5.3.1 Exact-range Functions

For many simple functions $F : \mathbb{R}^n \to \mathbb{R}^m$, its exact-range function $\underline{F}^u : \mathbb{IR}^n \to \mathbb{IR}^m$ can be obtained directly using monotonicity and deriving simple formulas as for the elementary operators in Section 5.2.2. For example, e^x evaluated over an interval $\underline{x}$ gives $[e^{\overleftarrow{\underline{x}}}, e^{\overrightarrow{\overline{\underline{x}}}}]$, since the exponential function is monotonic increasing. Integer powers can also be computed exactly and the same happens for trigonometric functions, although here the resulting formulas are more complicated. A list of the exact-range functions we have used (with $m = n - 1$), together with its formulas, are given in Appendix B.

Ranges of functions can also be computed with any desired accuracy –within the limits of the floating point system– from Taylor series, or any other expansion, that has an explicit formula for its error [24, p. 12].

5.3.2 Interval Extensions

In the example of Section 5.2.3, we showed how to obtain bounds on the range of a function using interval arithmetic. The function was evaluated as a sequence of the four basic operations, and we stated the dependency problem, which prevents to get tight bounds on the result when a variable is repeated in the function. In this section, we introduce different methods, or *interval extensions*, for obtaining the range of functions.

Consider a real-valued function $f : \mathbb{R}^n \to \mathbb{R}$ of real variables $\mathbf{x} = (x_1, x_2, \ldots, x_n)$. An interval function $\underline{f} : \mathbb{IR}^n \to \mathbb{IR}$ is said to be an *interval extension* of f if

$$\{f(\mathbf{x}) \mid \mathbf{x} \in \underline{\mathbf{x}}\} \subseteq \underline{f}(\underline{\mathbf{x}}) \qquad \forall \underline{\mathbf{x}} \in \mathbb{IR}^n .$$

The basic interval Newton method for solving systems of nonlinear equations uses two interval extensions, the *natural interval extension* and the *Taylor interval extension.* Other implementations of the Newton method (e.g. [67]) make also use of the so-called *distributed interval extension.*

Natural Interval Extension

In the example of Section 5.2.3, we evaluated the function f over an interval $\underline{x}$ by simply replacing each occurrence of the variable x by the interval $\underline{x}$ and by executing all operations according to formulas (5.2) through (5.5). This evaluation is obviously an interval extension and we will call it *natural interval extension.*

In fact, we can extend the concept of natural interval extension to any function $F : \mathbb{R}^n \to \mathbb{R}^m$ composed of the four operations $\{+, -, \times \text{ and } \div\}$ and of any exact-range function.

It is important to note that the natural extension evaluated in a box $\underline{\mathbf{x}}$ corresponds to the actual range of the real-valued function F over that box if each variable appears only once in the expression and if real interval arithmetic and exact-range functions are used.

We will refer to natural extensions even when rounded interval arithmetic is used. Then, the function range when each variable appears only once is merely an enclosure of the actual range that is tight to within roundout error.

Taylor Interval Extension

Using natural extensions in functions with repeated variables, the bounds on the ranges so obtained are often too wide to be of any practical interest. The *Taylor interval extension*, or *mean value extension*, gives sometimes tighter bounds and is more desirable in some contexts, as for example in solving systems of nonlinear equations and in optimization. Taylor extension is based on bounding the range of the derivatives in series expansions.

Suppose $f : \mathbb{R}^n \to \mathbb{R}$ has continuous derivatives and $\check{\underline{\mathbf{x}}}$ is a representative point of box

$\underline{\mathbf{x}} \in \mathbb{R}^n$. The *Taylor interval extension* for f over $\underline{\mathbf{x}}$ and centered at $\check{\underline{\mathbf{x}}}$ is defined as

$$\underline{f}_T(\underline{\mathbf{x}}, \check{\underline{\mathbf{x}}}) \triangleq f(\check{\underline{\mathbf{x}}}) + \underline{f}'(\underline{\mathbf{x}})^t(\underline{\mathbf{x}} - \check{\underline{\mathbf{x}}}) \ ,$$

where $\underline{f}'(\underline{\mathbf{x}})$ is a component-wise interval enclosure for the range of the gradient of f over $\underline{\mathbf{x}}$. It can be proved from the mean value theorem and properties of interval arithmetic that the Taylor extension is an enclosure for the range of f over $\underline{\mathbf{x}}$.

Taylor interval extensions give tighter bounds on the range of a function over an interval when its width is small *enough*, but not necessarily when the interval is wide. For example, suppose the function from the previous example $f(x) = x^2 - x$ evaluated in the interval $\underline{x} = [-1/2, 1]$. Taking $\check{\underline{x}} = 1/2$ as a representative point in $\underline{x}$ and being $f'(x) = 2x - 1$ the exact range of the derivative, the natural and the Taylor interval extensions are

$$\begin{aligned} \underline{f}_n\big([-1/2,1]\big) &= [-1/2,1]^2 - [-1/2,1] = [0,1] - [-1/2,1] = [-1,3/2] \text{ and} \\ \underline{f}_T\big([-1/2,1],1/2\big) &= -1/4 + \big(2[-1/2,1]-1\big)\big([-1/2,1]-1/2\big) = \\ &= -1/4 + [-2,1][-1,1/2] = [-5/4,7/4] \text{ , respectively.} \end{aligned}$$

Thus, $\mathrm{w}\Big(\underline{f}_n\big([-1/2,1]\big)\Big) = 2.5$ and $\mathrm{w}\Big(\underline{f}_T\big([-1/2,1],1/2\big)\Big) = 3$, whereas the width of the actual range is $\mathrm{w}\big(\underline{f}^u\big([-1/2,1]\big)\big) = 1$. This illustrates that a Taylor extension may not be superior to a natural extension when the width of the arguments are *large*. If we take a narrower interval, for instance $\underline{x} = [0.4, 0.6]$, and $\check{\underline{x}} = 0.5$, then the natural and Taylor extensions are

$$\begin{aligned} \underline{f}_n\big([0.4,0.6]\big) &= [-0.44,-0.04] \text{ and} \\ \underline{f}_T\big([0.4,0.6],0.5\big) &= [-0.27,-0.23] \text{ , respectively.} \end{aligned}$$

Here, the width of the natural extension is much wider than the width of the Taylor extension. This exemplifies an unsolved problem in interval analysis: it is not known how small an interval has to be for the result from the Taylor expansion to be tighter than the one obtained using the natural evaluation.

5.4 Interval Linear Systems

Interval linear systems are the basis of interval Newton methods for solving systems of nonlinear equations.

In this section we will first describe the solution set of interval linear systems. Then we will explain the necessity to *precondition* the system and finally introduce directly the *preconditioned interval Gauss-Seidel method*, the most suitable solution algorithm for our purposes.

5.4.1 Solution Set

Consider an interval linear system described by

$$\underline{\mathbf{A}}\,\underline{\mathbf{x}} = \underline{\mathbf{b}}\,, \tag{5.7}$$

where $\underline{\mathbf{A}} \in \mathbb{IR}^{m\times n}$, $\underline{\mathbf{b}} \in \mathbb{IR}^{m}$ and $\underline{\mathbf{x}} \in \mathbb{IR}^{n}$.

Different solution sets may be defined depending on whether a solution $\mathbf{x}$ requires that $\mathbf{Ax} = \mathbf{b}$ for every matrix $\mathbf{A} \in \underline{\mathbf{A}}$ and vector $\mathbf{b} \in \underline{\mathbf{b}}$ or not.

Here, we will consider the solution set to be the set $\Sigma(\underline{\mathbf{A}}, \underline{\mathbf{b}}) \subseteq \mathbb{R}^n$ such that, if $\mathbf{x} \in \Sigma(\underline{\mathbf{A}}, \underline{\mathbf{b}})$, there exists a matrix $\mathbf{A} \in \underline{\mathbf{A}}$ and a vector $\mathbf{b} \in \underline{\mathbf{b}}$ such that $\mathbf{Ax} = \mathbf{b}$. $\Sigma(\underline{\mathbf{A}}, \underline{\mathbf{b}})$ is sometimes called the *united solution set*, but we will simply refer to it as the *solution set*.

In general, the solution set is not an interval vector, but a non-convex polygonal region, which can be quite complicated in high-dimensional problems. In fact, the computation of $\Sigma(\underline{\mathbf{A}}, \underline{\mathbf{b}})$ is in general an NP-complete problem [51]. The next example from Hansen [17] illustrates that the solution set $\Sigma(\underline{\mathbf{A}}, \underline{\mathbf{b}})$ is not simple. Let us consider

$$\underline{\mathbf{A}} = \begin{pmatrix} [2,3] & [0,1] \\ [1,2] & [2,3] \end{pmatrix} \quad \text{and} \quad \underline{\mathbf{b}} = \begin{pmatrix} [0,120] \\ [60,240] \end{pmatrix}\,. \tag{5.8}$$

The actual solution set to $\underline{\mathbf{A}}\,\underline{\mathbf{x}} = \underline{\mathbf{b}}$, i.e. $\Sigma(\underline{\mathbf{A}}, \underline{\mathbf{b}})$, is shown in Figure 5.1.

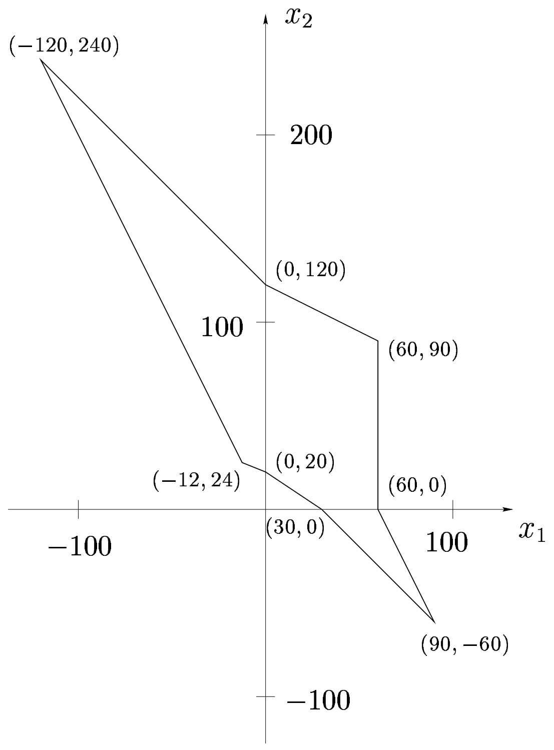


Figure 5.1: The solution set $\Sigma(\underline{\mathbf{A}}, \underline{\mathbf{b}})$ for the linear system (5.8)

Instead of working with the solution set, we shall consider the *solution hull* denoted by $\square\Sigma(\underline{\mathbf{A}}, \underline{\mathbf{b}})$; that is, the box formed from bounds on the coordinates of $\Sigma(\underline{\mathbf{A}}, \underline{\mathbf{b}})$. However, computing the solution hull is also NP-complete [51] and we will have to work with *outer estimates* to $\Sigma(\underline{\mathbf{A}}, \underline{\mathbf{b}})$ (Figure 5.2).

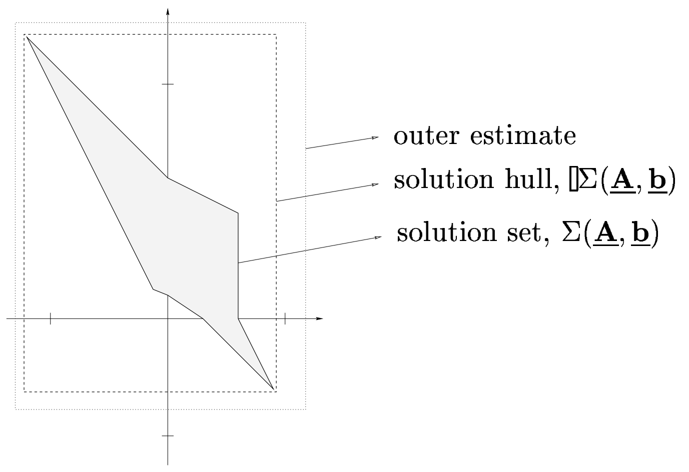

outer estimate
solution hull, $\square\Sigma(\underline{\mathbf{A}}, \underline{\mathbf{b}})$
solution set, $\Sigma(\underline{\mathbf{A}}, \underline{\mathbf{b}})$

Figure 5.2: Solution set, solution hull and outer estimates of interval linear systems

The three most common methods for computing outer estimates to $\Sigma(\underline{\mathbf{A}}, \underline{\mathbf{b}})$ are the *interval Gaussian elimination method*, the *interval Gauss-Seidel method* and the *Krawczyk method*.

Interval Gaussian elimination and the interval Gauss-Seidel method are similar to their corresponding real standard versions. Although the interval Gauss-Seidel method requires an initial guess box, it sometimes leads to tighter enclosures of the solution set than Gaussian elimination. Moreover, since the Gauss-Seidel method proceeds coordinate by coordinate, it may produce tighter bounds on certain variables when Gaussian elimination fails, even when the interval matrix $\underline{\mathbf{A}}$ contains singular matrices, or when the system is not square; i.e. $\underline{\mathbf{A}} \in \mathbb{IR}^{m\times n}$ with $m \neq n$.

Much of the theoretical literature concerns the Krawczyk method, whose convergence properties are particularly simple to analyze. However, Gauss-Seidel method always give tighter results than the Krawczyk method, so the Gauss-Seidel method is preferable in practice.

5.4.2 Preconditioning the System

The interval versions of Gaussian elimination and the Gauss-Seidel method obtained by simply replacing the algorithm by an interval one cannot be recommended in practice. It is usually necessary to *precondition* the system by a point matrix $\mathbf{Y} \in \mathbb{R}^{n\times m}$ for these methods to be effective.

When preconditioning a system, the algorithms are applied, not to the original system 5.7, but to

$$\underline{\mathbf{M}}\,\underline{\mathbf{x}} = \underline{\mathbf{c}} \qquad \text{where } \underline{\mathbf{M}} = \mathbf{Y}\underline{\mathbf{A}} \text{ and } \underline{\mathbf{c}} = \mathbf{Y}\underline{\mathbf{b}}\ .$$

The preconditioning matrix $\mathbf{Y}$ is chosen to make the solution set of $\underline{\mathbf{M}}\,\underline{\mathbf{x}} = \underline{\mathbf{c}}$ easier to bound. Usually, $\mathbf{Y}$ is taken as the inverse of $\mathrm{m}(\underline{\mathbf{A}})$ in order to have a matrix $\underline{\mathbf{M}}$ that somehow approximates a diagonal matrix. In Section 5.7 we will describe the different preconditioners we can use.

An important result is that the solution set of the preconditioned system always includes the solution set of the original system (a proof can be found in [38], Section 4.1):

$$\Sigma(\mathbf{Y}\underline{\mathbf{A}}, \mathbf{Y}\underline{\mathbf{b}}) \supseteq \Sigma(\underline{\mathbf{A}}, \underline{\mathbf{b}})\ .$$

Therefore, when solving a preconditioned system, we do not miss any solution of the original interval linear system. Even though preconditioning in general increases the size of the actual solution set, it nonetheless allows the Gauss-Seidel, Gaussian elimination and Krawczyk methods to compute tighter bounds for the components of the solution set. Thus, preconditioning should be in general used.

5.4.3 Preconditioned Interval Gauss-Seidel Method

We will use the preconditioned interval Gauss-Seidel method to solve systems of linear equations for three reasons:

1. It gives tighter bounds than either Gaussian elimination or the Krawczyk method [37].

2. It works even when Gaussian elimination fails because $\underline{\mathbf{A}}$ contains singular matrices.

3. It can be used for non-square systems ($\underline{\mathbf{A}} \in \mathbb{R}^{m\times n}$ with $m \neq n$). Even when the system is underdetermined, the solution set can still be bounded.

An interval version of the Gauss-Seidel method was first introduced by Alefeld and Herzbergin in 1970 [2]. The Gauss-Seidel method proceeds coordinate by coordinate and needs an initial guess box. Each iteration works only on a single column of $\underline{\mathbf{A}}$ and requires a single row of the preconditioner $\mathbf{Y}$. As a result, we obtain a new bound for one of the variables. Thus, the preconditioner can be computed row by row with some of the variables already bounded. This results in a gain in efficiency when using preconditioners which depend on the variables (e.g. the optimal preconditioners of Section 5.7.2). The computation of the preconditioner rows will be analyzed in Section 5.7.

The algorithm can be summarized in the following steps:

```
DO for $i = 1$ to $n$ (number of variables)
    1. Compute the $i$-th row $\mathbf{y}_i$ of the preconditioner
    2. $\tilde{\underline{x}}_i = \underline{x}_i \cap \dfrac{\mathbf{y}_i\underline{\mathbf{b}} - \sum_{j=1}^{i-1}(\mathbf{y}_i\underline{\mathbf{a}}_j)\tilde{\underline{x}}_j - \sum_{j=i+1}^{n}(\mathbf{y}_i\underline{\mathbf{a}}_j)\underline{x}_j}{\mathbf{y}_i\underline{\mathbf{a}}_i}$
END DO
```

where $\underline{\mathbf{a}}_i$ denotes the i-th column of $\underline{\mathbf{A}}$, $\underline{x}_i$ is the i-th component of box $\underline{\mathbf{x}}$ and $\tilde{\underline{x}}_i$ the corresponding bounded variable.

Algorithm 5.1: Preconditioned interval Gauss-Seidel algorithm for a system $\underline{\mathbf{A}}\,\underline{\mathbf{x}} = \underline{\mathbf{b}}$.

If the intersection in Step 2 is empty, the system does not have a solution in the original box $\underline{\mathbf{x}}$. This is true even when outward rounding is used.

Note that the denominator $\mathbf{y}_i\underline{\mathbf{a}}_i$ may contain the origin. Then, the division using extended interval arithmetic (5.6) yields a result which is not finite. However, when intersecting it with $\underline{x}_i$, the new bounds for the variable will be finite and may be either empty or consist of one interval or the union of two intervals. As before, if the intersection is empty the system does not have a solution. If the intersection leads to two intervals, we may consider only its convex hull or we may decide to split the box in that dimension in order to reduce the solution space.

Usually, a single iteration of the Gauss-Seidel method does not give tight results and multiple sweeps are required to get useful bounds.

We will denote the result of i iterations of the preconditioned interval Gauss-Seidel algorithm by

$$\tilde{\underline{\mathbf{x}}} = \mathbf{GS}_i(\underline{\mathbf{x}}) \ .$$

It is important to point out that the sequential nature of Algorithm 5.1 is not critical. Any order of the variables may be chosen and the algorithm will lead to similar results.

Finally, note that $\lim_{i\to\infty} \mathbf{GS}_i(\underline{\mathbf{x}})$ does usually not converge to the convex hull of the solution set, but to an outer estimate.

5.4.4 Existence and Uniqueness

Most of the automatic verification procedures are based on the following result concerning the computational existence and uniqueness of the solutions:

- The intersection of the solution hull of an interval linear system and a box $\underline{\mathbf{x}}$ is always contained in the resulting box from the preconditioned interval Gauss-Seidel method. In other words,

$$(\square\Sigma(\underline{\mathbf{A}}, \underline{\mathbf{b}}) \cap \underline{\mathbf{x}}) \subset \mathbf{GS}_i(\underline{\mathbf{x}}) \ .$$

- If the resulting box from the preconditioned interval Gauss-Seidel method is strictly included in the original box, i.e., if $\mathbf{GS}_i(\underline{\mathbf{x}}) \subset \text{int}(\underline{\mathbf{x}})$, then for each $\mathbf{A} \in \underline{\mathbf{A}}$ and $\mathbf{b} \in \underline{\mathbf{b}}$, the system $\mathbf{Ax} = \mathbf{b}$ has a unique solution in $\underline{\mathbf{x}}$.

Considerations related to these results and their proofs can be found in [38].

5.5 Interval Derivatives

Interval derivatives are mainly used in bounding variations of functions in boxes. We have already introduced them in the Taylor extensions in Section 5.3.2 and we will use them extensively in interval Newton methods.

Consider the real function $F : \mathbb{R}^n \to \mathbb{R}^m$. The *interval derivative* or *interval Jacobian matrix* of F will refer to a component-by-component interval extension over a box $\underline{\mathbf{x}}$ of the Jacobian matrix of F, which will be denoted by $F'(\underline{\mathbf{x}})$.

5.5.1 Slopes

Often, it is not necessary to use interval extensions of the Jacobian matrix. For example, interval Newton methods as explained below, are based in a Taylor extension of a multivariate function. Then, we only need a matrix $\underline{\mathbf{A}}$, such that given a box $\underline{\mathbf{x}}$ and a point in it $\check{\underline{\mathbf{x}}} \in \underline{\mathbf{x}}$, for every $\mathbf{x} \in \underline{\mathbf{x}}$,

$$F(\mathbf{x}) - F(\check{\underline{\mathbf{x}}}) = \mathbf{A}(\mathbf{x} - \check{\underline{\mathbf{x}}}) \qquad \text{for some } \mathbf{A} \in \underline{\mathbf{A}} \ . \tag{5.9}$$

A matrix $\underline{\mathbf{A}}$ that satisfies (5.9) is said to be a *slope matrix*. Actually, the interval Jacobian of F is a slope matrix, but with the term slope matrix we will refer to a *good* outer estimate of the smallest set of matrices satisfying (5.9). Slopes lead generally to tighter bounds in Taylor extensions of $F(\underline{\mathbf{x}})$ than using interval derivatives. We will also use the term slope matrix when $\check{\underline{\mathbf{x}}}$ in (5.9) is not a point, but an interval, as for instance in Theorem 5.3.

The idea behind slopes is most easily introduced with a one-dimensional example. Suppose a Taylor expansion of $f(x) = x^2$ in the interval $[0, 2]$, centered at 1. Its derivative $f'(x) = 2x$ evaluated in $[0, 2]$ is $f'([0, 2]) = [0, 4]$. The mean value extension is then

$$f_{Td}([0, 2], 1) = f(1) + f'([0, 2])([0, 2] - 1) = [-3, 5] \ .$$

A slope for f in the interval $[0,2]$ and centered in 1 is

$$\underline{s} = \frac{f(x) - f(\check{\underline{x}})}{x - \check{\underline{x}}} = \frac{x^2 - 1}{x - 1} = x + 1 \quad \text{for } x \in [0,2] \, , \tag{5.10}$$
$$\underline{s} = [1,3] \, .$$

The mean value extension using this slope is now

$$f_{Ts}([0,2],1) = f(1) + \underline{s}([0,2] - 1) = [-2,4] \, .$$

The graphical interpretation of this results can be seen in Figure 5.3. The shaded cone is the region where the Taylor extension assumes that the function plot is contained. The difference between interval slope and the derivative range is clearly seen.

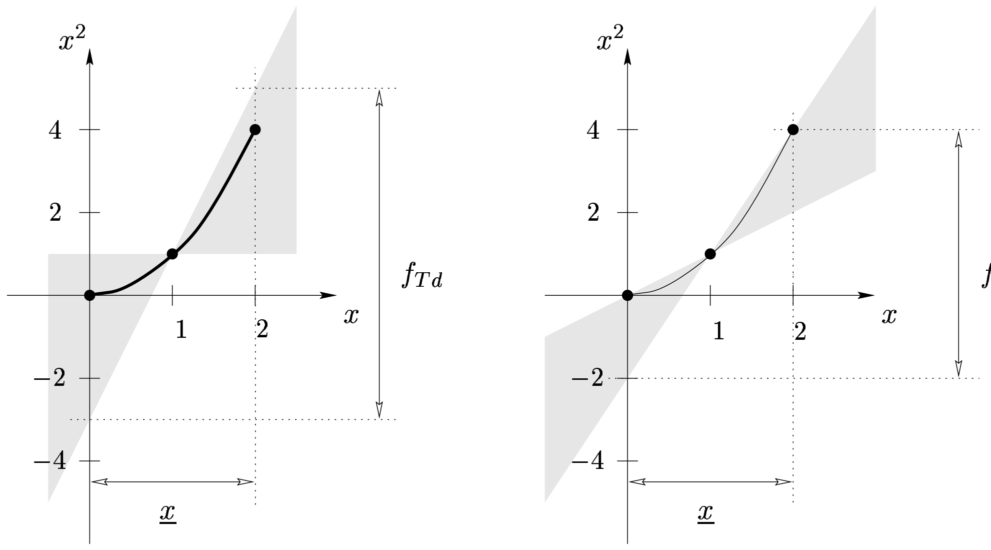


Figure 5.3: Taylor extension of x^2 using interval derivative (left) and using a slope (right).

This figure illustrates the fact that usually the bounds obtained using slopes are tighter than the ones using extensions of derivatives. In the limit, when $\mathrm{w}(\underline{x})$ decreases to zero, slope-based evaluation would give bounds whose width is half the width obtained using derivatives [24, pp. 28–29][4].

Multivariate slopes may be obtained by setting, for each component of the slope matrix, all variables other than the active one to their interval and not to point values (Algorithm 3 in [24]). This algorithm requires a method for computing a slope of a univariate function. However, slopes of a univariate function are not always easy to compute. We can get them as in the previous example, but the resulting fraction can usually not be simplified as in (5.10). Then, both the numerator and the denominator include the zero and the division using extended interval arithmetic (5.6) is unbounded.

Slopes can also be implemented in a process similar to automatic differentiation, first described by Neumaier (see Section 6.7 in [17] and [50]).

[4]The proof given in the book has some mistakes. A correct proof can be found in the erratas (available at `http://interval.usl.edu/kearfott.html`).

5.5.2 Hansen Slopes

In Taylor extensions of multivariate functions, the derivative matrix can be computed using point values for some of the variables. This technique, due to Hansen (Sections 6.3 and 6.4 in [17]), is based on decomposing the variation in $f : \mathbb{R}^n \to \mathbb{R}$ into changes in displacements along the coordinate directions.

The mean value extension of a multivariate function $f : \mathbb{R}^n \to \mathbb{R}$ is

$$f_T(\underline{\mathbf{x}}, \check{\underline{\mathbf{x}}}) = f(\check{\underline{\mathbf{x}}}) + f'(\underline{\mathbf{x}})^t(\underline{\mathbf{x}} - \check{\underline{\mathbf{x}}}) \tag{5.11}$$

where $f'(\underline{\mathbf{x}})$ is an extension of the gradient of f evaluated in $\underline{\mathbf{x}}$. Note that all arguments of $f'(\underline{\mathbf{x}})$ are intervals. But some of these arguments can be replaced by noninterval quantities expanding the function coordinate by coordinate.

We regard first $f(x_1, x_2, \ldots, x_n)$ only as a function of x_1 and expand it about $\check{\underline{x}}_1$:

$$f(\underline{x}_1, \underline{x}_2, \ldots, \underline{x}_n) = f(\check{\underline{x}}_1, \underline{x}_2, \ldots, \underline{x_n}) + f'_1(\underline{x}_1, \underline{x}_2, \ldots, \underline{x}_n)(\underline{x}_1 - \check{\underline{x}}_1) \ ,$$

where f'_i is the derivative of f with respect to variable x_i.

We now expand $f(\check{\underline{x}}_1, \underline{x}_2, \ldots, \underline{x_n})$ about $\check{\underline{x}}_2$ as a function of $\underline{x}_2$ and obtain

$$f(\check{\underline{x}}_1, \underline{x}_2, \ldots, \underline{x_n}) = f(\check{\underline{x}}_1, \check{\underline{x}}_2, \underline{x}_3, \ldots, \underline{x_n}) + f'_2(\check{\underline{x}}_1, \underline{x}_2, \ldots, \underline{x_n})(\underline{x}_2 - \check{\underline{x}}_2) \ .$$

We expand successively f with some variables fixed to point values. At the end we obtain

$$\begin{aligned} f(\underline{x}_1, \underline{x}_2, \ldots, \underline{x}_n) \quad = \quad & f(\check{\underline{x}}_1, \check{\underline{x}}_2, \ldots, \check{\underline{x}}_n) + \\ & + f'_1(\underline{x}_1, \underline{x}_2, \ldots, \underline{x}_n)(\underline{x}_1 - \check{\underline{x}}_1) + \\ & + f'_2(\check{\underline{x}}_1, \underline{x}_2, \ldots, \underline{x_n})(\underline{x}_2 - \check{\underline{x}}_2) + \\ & + f'_3(\check{\underline{x}}_1, \check{\underline{x}}_2, \underline{x}_3, \ldots, \underline{x_n})(\underline{x}_3 - \check{\underline{x}}_3) + \\ & + \cdots + \\ & + f'_n(\check{\underline{x}}_1, \check{\underline{x}}_2, \ldots, \check{\underline{x}}_{n-1}, \underline{x_n})(\underline{x}_n - \check{\underline{x}}_n) \ . \end{aligned} \tag{5.12}$$

In (5.11) all the arguments of f' were intervals, while in (5.12), some are real. Thus, a bound on $f(\underline{\mathbf{x}})$ using this technique is generally tighter than the corresponding one evaluating the gradient over $\underline{\mathbf{x}}$. Hansen extensions clearly correspond to the definition of slope matrices. Moreover, we can change f'_i by a method for computing slopes of univariate functions, getting still tighter bounds.

Observe that in (5.12) any sequence of the variables could be taken and the order of the indices would change. Particular orders may give better inclusions of f than others. Several heuristics can be used to order the variables [24, p. 34]. In any case, for tighter inclusions, results corresponding to several different orders can be computed and intersected.

5.6 Interval Newton Methods

Multivariate interval Newton methods for solving systems of nonlinear equations are closely tied to the methods for linear systems discussed in Section 5.4. In particular, they require the iterative execution of the Gauss-Seidel algorithm to bound the solution sets of linear systems of the form

$$\underline{\mathbf{S}}(\underline{\mathbf{x}} - \check{\underline{\mathbf{x}}}) = -F(\check{\underline{\mathbf{x}}}) \;, \tag{5.13}$$

where $\underline{\mathbf{S}}$ is either $F'(\underline{\mathbf{x}})$, the Jacobian matrix of F in $\underline{\mathbf{x}}$, or a slope matrix as explained in the previous section. Since $\check{\underline{\mathbf{x}}}$ is usually an approximation to a root of F, (5.13) approximates an homogeneous system of equations and there is more structure and symmetry than in the general systems of Section 5.4. For instance, the computation of the LP preconditioners (Section 5.7) is based on the assumption that $F(\check{\underline{\mathbf{x}}})$ is small.

Assume that $F : \underline{\mathbf{x}} \subset \mathbb{R}^n \to \mathbb{R}^m$, $\check{\underline{\mathbf{x}}} \in \underline{\mathbf{x}}$ and $\mathbf{x}^* \in \underline{\mathbf{x}}$ is a solution of system F, that is, $F(\mathbf{x}^*) = 0$. Then the mean value theorem says that

$$F(\check{\underline{\mathbf{x}}}) + \mathbf{S}(\mathbf{x}^* - \check{\underline{\mathbf{x}}}) = 0$$

for some matrix $\mathbf{S} \in \underline{\mathbf{S}}$. Thus, $\mathbf{x}^* - \check{\underline{\mathbf{x}}}$ must be in the solution set of the system (5.13). Any root $\mathbf{x}^*$ of F within $\underline{\mathbf{x}}$ must be within $\check{\underline{\mathbf{x}}} + \tilde{\underline{\mathbf{x}}}$, where $\tilde{\underline{\mathbf{x}}}$ is any bounding interval for the solution set of Equation (5.13). If we use the interval Gauss-Seidel algorithm described in Section 5.4.3, replacing $\underline{\mathbf{b}}$ by $-F(\check{\underline{\mathbf{x}}})$, $\underline{\mathbf{A}}$ by $\underline{\mathbf{S}}$, and $\underline{x}_i$ by $\underline{x}_i - \check{\underline{x}}_i$, we can change Step 2 of Algorithm 5.1 by

$$\tilde{\underline{x}}_i = \underline{x}_i \cap \left(\check{\underline{x}}_i - \frac{\mathbf{y}_i F(\check{\underline{\mathbf{x}}}) + \sum_{j=1}^{i-1} (\mathbf{y}_i \underline{\mathbf{s}}_j)(\tilde{\underline{x}}_j - \check{\underline{x}}_j) + \sum_{j=i+1}^{n} (\mathbf{y}_i \underline{\mathbf{s}}_j)(\underline{x}_j - \check{\underline{x}}_j)}{\mathbf{y}_i \underline{\mathbf{s}}_i} \right) \;, \tag{5.14}$$

where $\underline{\mathbf{s}}_i$ denotes the i-th column of $\underline{\mathbf{S}}$. Rigorously, all of the computations, including evaluation of $F(\check{\underline{\mathbf{x}}})$, should be done with outward rounded interval arithmetic. As explained in 5.4.3, extended interval arithmetic can be used when $0 \in \mathbf{y}_i \underline{\mathbf{s}}_i$.

An important fact making interval Newton methods efficient is its *local quadratic convergence* (see Theorem 1.14 in [24]). However, Newton methods without any branching are often not able to isolate all solutions of the system. The basic ideas of a branch-and-bound algorithm for a verified solution of nonlinear systems are given in Section 5.8.

5.6.1 Existence and Uniqueness

The existence/uniqueness theory and the quadratic convergence properties of interval Newton methods are analogous to the Kantorovich theorem for the classic Newton method (see for example [47]). Theorems regarding existence and uniqueness are given in [24, pp. 59–65]. Here, we will focus only on the results concerning the Gauss-Seidel method. We will have to introduce some theorems (the only ones in this chapter!).

The most general theorem regarding existence and uniqueness is:

Theorem 5.1 (Derivative Based Existence and Uniqueness). *Suppose a function $F : \underline{\mathbf{x}} \rightarrow \mathbb{R}^n$, $\underline{\mathbf{F}}'(\underline{\mathbf{x}})$ is an interval Jacobian matrix for F over $\underline{\mathbf{x}}$, $\check{\underline{\mathbf{x}}} \in \underline{\mathbf{x}}$, and $\tilde{\underline{\mathbf{x}}} = \check{\underline{\mathbf{x}}} + \underline{\mathbf{v}}$, where $\underline{\mathbf{v}} \in \mathbb{IR}^n$ is any interval vector such that*

$$\Sigma\big(\underline{\mathbf{F}}', -F(\check{\underline{\mathbf{x}}})\big) \subset \underline{\mathbf{v}}\,.$$

Then, if $\tilde{\underline{\mathbf{x}}} \subseteq \underline{\mathbf{x}}$, it follows that there exists a unique solution of $F(\mathbf{x})$ within $\underline{\mathbf{x}}$.

Theorem 5.1 encompasses any method for bounding the solution set $\Sigma\big(\underline{\mathbf{F}}', -F(\check{\underline{\mathbf{x}}})\big)$; in particular, the interval Gauss-Seidel method. Note that it is assumed that $\underline{\mathbf{F}}'$ is an interval Jacobian matrix and not a slope matrix.

A similar existence theorem is possible with a slope matrix:

Theorem 5.2 (Slope Based Existence). *Suppose $\tilde{\underline{x}}_i$ is defined by the second part of Formula (5.14), that is,*

$$\tilde{\underline{x}}_i = \check{\underline{x}}_i - \frac{\mathbf{y}_i F(\check{\underline{\mathbf{x}}}) + \sum_{j=1}^{i-1} (\mathbf{y}_i \underline{\mathbf{s}}_j)(\tilde{\underline{x}}_j - \check{\underline{x}}_j) + \sum_{j=i+1}^{n} (\mathbf{y}_i \underline{\mathbf{s}}_j)(\underline{x}_j - \check{\underline{x}}_j)}{\mathbf{y}_i \underline{\mathbf{s}}_i}$$

for i ranging from 1 to n, where $F : \underline{\mathbf{x}} \rightarrow \mathbb{R}^n$, $\underline{\mathbf{S}}$ is a slope matrix for F over $\underline{\mathbf{x}}$ at $\check{\underline{\mathbf{x}}}$, and $\mathbf{Y}$ is non-singular.

If $\tilde{\underline{x}}_i \subseteq \underline{x}_i$ for i between 1 and n, then there exists an $\mathbf{x}^* \in \underline{\mathbf{x}}$ such that $F(\mathbf{x}^*) = 0$.

This result can be extended to non-square systems. Then, $\mathbf{Y}$ must have at least one maximal minor which is not singular.

Uniqueness using slopes requires additional consideration. In [24, Figure 1.6], there is an illustrating example of non-uniqueness with slopes .

In [56], Rump pointed out that uniqueness verification with slope matrices is still possible in certain context, such as with the ϵ-inflation technique, as explained in Section 4.2 in [24]. This technique is based on:

Theorem 5.3. *Let $F : \underline{\mathbf{x}} \subset \mathbb{R}^n \rightarrow \mathbb{R}^n$, and let $\underline{\mathbf{z}} \subseteq \underline{\mathbf{x}}$ be such that there exists an $\mathbf{x}^* \in \underline{\mathbf{z}}$ with $F(\mathbf{x}^*) = 0$. Let $\underline{\mathbf{S}}(F, \underline{\mathbf{x}}, \underline{\mathbf{z}})$ be a slope matrix for F over $\underline{\mathbf{x}}$ at $\underline{\mathbf{z}}$. If $\underline{\mathbf{S}}(F, \underline{\mathbf{x}}, \underline{\mathbf{z}})$ is regular (it does not contain any singular matrix), then $\mathbf{x}^*$ is unique within $\underline{\mathbf{x}}$.*

Theorem 5.3 allows verification of uniqueness within larger boxes than when an interval Jacobian $F'(\underline{\mathbf{x}})$ is used, provided that the box $\underline{\mathbf{z}}$ can be found. This is because slope bounds $\underline{\mathbf{S}}(F, \underline{\mathbf{x}}, \underline{\mathbf{z}})$ are generally narrower than interval Jacobians $F'(\underline{\mathbf{x}})$.

5.7 Preconditioners

As mentioned in Section 5.4.2, it is usually necessary to precondition an interval linear system by a point matrix for the interval Gauss-Seidel method to be effective. Two families of preconditioners have been mainly used: the inverse midpoint preconditioners and the optimal linear-programming preconditioners. The latter can be computed as a linear-programming problem using some heuristics based on assuming the linear system to be homogeneous. This is the case in nonlinear systems of equations solved using an interval Newton method, but not in general linear systems of equations.

5.7.1 The Inverse Midpoint Preconditioner

If $\underline{\mathbf{A}}\,\underline{\mathbf{x}} = \underline{\mathbf{b}}$ is a square interval linear system, the inverse midpoint preconditioner is

$$\mathbf{Y}^{mid} = \mathrm{m}(\underline{\mathbf{A}})^{-1} .$$

The inverse midpoint preconditioner was first introduced by Hansen. It is the most commonly used in the literature and has often been assumed to be the best possible preconditioner [38]. However, alternate preconditioners are better in some circumstances.

In [56], Rump observed that much of the computation related to preconditioning and bounding the solution sets of the preconditioned systems can be simplified when the preconditioner is $\mathbf{Y}^{mid}$. These shortcuts make use of floating point arithmetic in much of the computations and are detailed in [24, pp. 115–118].

One of the drawbacks of the inverse midpoint preconditioner is that it requires the system to be square.

5.7.2 Optimal Preconditioners

When $\underline{\mathbf{A}}$ has some rows with much wider entries than others, preconditioners other than the inverse midpoint preconditioner can be more effective. Kearfott introduced for the interval Gauss-Seidel method preconditioners that optimize some criteria, and whose computation is usually based on some heuristics.

A first review on optimal preconditioners for the interval Gauss-Seidel method can be found in [26]. For more detail, take a look at the unpublished Novoa's work [40] (available via ftp).

Optimal LP preconditioners optimize some aspect of the reduced box in the Gauss-Seidel method ($\tilde{\underline{x}}_i$ in equation (5.14)) at each step. In fact, they are computed row by row and are sometimes termed *row-wise preconditioners*. Heuristics enable such preconditioners to be computed as solutions to linear programming problems.

LP preconditioners usually optimize an aspect of $\tilde{\underline{x}}_i$ before its intersection with the initial range of the variable $\underline{x}_i$. That is,

$$\tilde{\underline{x}}_i = \check{\underline{x}}_i - \frac{\mathbf{y}_i F(\check{\underline{\mathbf{x}}}) + \sum_{j=1}^{i-1}(\mathbf{y}_i \underline{\mathbf{s}}_j)(\tilde{\underline{x}}_j - \check{\underline{x}}_j) + \sum_{j=i+1}^{n}(\mathbf{y}_i \underline{\mathbf{s}}_j)(\underline{x}_j - \check{\underline{x}}_j)}{\mathbf{y}_i \underline{\mathbf{s}}_i} . \tag{5.15}$$

We denote the numerator as $\underline{n}_i(\mathbf{y}_i)$ and the denominator as $\underline{d}_i(\mathbf{y}_i)$:

$$\underline{\tilde{x}}_i = \underline{\check{x}}_i - \frac{\underline{n}_i(\mathbf{y}_i)}{\underline{d}_i(\mathbf{y}_i)} = \underline{\check{x}}_i - \frac{[\overleftarrow{\underline{n}}_i(\mathbf{y}_i),\, \overrightarrow{\underline{n}}_i(\mathbf{y}_i)]}{[\overleftarrow{\underline{d}}_i(\mathbf{y}_i),\, \overrightarrow{\underline{d}}_i(\mathbf{y}_i)]}\,.$$

C- and E-preconditioners

Two preconditioners are defined depending upon whether $0 \in \underline{d}_i(\mathbf{y}_i)$:

- A preconditioning row $\mathbf{y}_i$ is a *C-preconditioner* [5] if $0 \notin \underline{d}_i(\mathbf{y}_i)$. Furthermore, $\mathbf{y}_i$ is a *normal C-preconditioner* provided $\overleftarrow{\underline{d}}_i(\mathbf{y}_i) = 1$.

- A preconditioning row $\mathbf{y}_i$ is an *E-preconditioner* [6] if $0 \in \underline{d}_i(\mathbf{y}_i)$ and $0 \notin \underline{n}_i(\mathbf{y}_i)$.

Observe that preconditioners which are not C- neither E-preconditioners will necessarily have $0 \in \underline{d}_i(\mathbf{y}_i)$ and $0 \in \underline{n}_i(\mathbf{y}_i)$. Such preconditioning rows are useless, since $\underline{\tilde{x}}_i$ is unbounded.

An interesting property of C-preconditioners is that they always exist, provided that at least one element of the corresponding row of $\underline{\mathbf{S}}$ does not contain 0 [19].

Optimality Criteria

Different optimality criteria can be defined for both preconditioners. However, most practical experience has been done with *width-optimal* C-preconditioners, which minimize the width of $\underline{\tilde{x}}_i$ over all C-preconditioners. They are commonly denoted by $\mathrm{C^W}$-preconditioners.

Other optimality criteria are based on maximizing the lower bound of $\underline{\tilde{x}}_i$, minimizing its upper bound, or minimizing the magnitude of $\underline{\tilde{x}}_i - \underline{\check{x}}_i$. There are equivalent optimizing criteria for E-preconditioners, but there has been few practical experience in its use. Note that the E-preconditioners generally produce two subboxes. The proliferation of subboxes reduces in general the overall efficiency of Newton methods.

Depending on the optimality criteria, preconditioners are better to prove existence and uniqueness or to reduce the box. For instance, the $\mathrm{C^W}$-preconditioner is usually a good general-purpose preconditioner. However, the magnitude-optimal C-preconditioner is better in existence and uniqueness tests and an E-preconditioner can be efficient for bisection. A detailed discussion about the different characteristics of the optimal preconditioners can be found in [24, pp. 123–128 and 141].

[5]The C comes from contraction.
[6]The E comes from extension.

5.7.3 Width-optimal C-preconditioners

Width-optimal preconditioners tend to minimize the width of the i^{th} variable under the interval Gauss-Seidel method. Ideally, they would minimize $\mathrm{w}(\tilde{\underline{x}}_i \cap \underline{x}_1)$, but it is easier to optimize $\mathrm{w}(\tilde{\underline{x}}_i)$.

The computation of optimal preconditioners is a nonlinear optimization problem. However, heuristic considerations allow it to be reformulated as a linear programming problem.

A normal $\mathrm{C^W}$−preconditioner $(\mathbf{y}_i)$ is the solution of

$$\min_{\overleftarrow{\underline{d}}_i(\mathbf{y}_i)=1} \mathrm{w}\left(\frac{\underline{n}_i(\mathbf{y}_i)}{\underline{d}_i(\mathbf{y}_i)}\right) . \tag{5.16}$$

The formulation as a linear programming problem involves rewriting each of the components of $\mathbf{y}_i$ as a difference of positive and negative parts. The *positive part* of a real number a will be denoted by $a^+ = \max\{a, 0\}$ and the *negative part*, by $a^- = \max\{-a, 0\}$.

We will need some elementary properties of interval arithmetic. Suppose that $a, b \in \mathbb{R}$ and $\underline{x}, \underline{y} \in \mathbb{IR}$. Then,

1. $a\underline{x} = [\overleftarrow{\underline{x}}\, a^+ - \overrightarrow{\underline{x}}\, a^-, \overrightarrow{\underline{x}}\, a^+ - \overleftarrow{\underline{x}}\, a^-] = a\mathrm{m}(\underline{x}) + \frac{1}{2}|a|\mathrm{w}(\underline{x})[-1, 1]$, (5.17)
2. If $\mathrm{m}(\underline{y}) = 0$, then $\underline{x}\underline{y} = \frac{1}{2}|\underline{x}|\mathrm{w}(\underline{y})[-1, 1]$, (5.18)
3. $|\underline{x}| = \max\{-\overleftarrow{\underline{x}}, \overrightarrow{\underline{x}}\}$, (5.19)
4. $\max\{a, b\} = a + (b - a)^+$. (5.20)

From now on, we will take $\check{\underline{\mathbf{x}}}$ as the center of the box $\underline{\mathbf{x}}$. After each step of the Gauss-Seidel method (5.14), the interval $\underline{x}_i$ may be substituted by the new one $\tilde{\underline{x}}_i$. Now, consider the numerator and denominator in equation (5.15), but denoting by $\underline{x}_i$ instead of $\tilde{\underline{x}}_i$ the new intervals for the variables that have already been reduced:

$$\underline{n}_i(\mathbf{y}_i) = \mathbf{y}_i F(\mathrm{m}(\underline{\mathbf{x}})) + \sum_{\substack{j=1 \\ j\neq i}}^{n} (\mathbf{y}_i \underline{\mathbf{s}}_j)(\underline{x}_j - \mathrm{m}(\underline{x}_j)) . \tag{5.21}$$

$$\underline{d}_i(\mathbf{y}_i) = \mathbf{y}_i \underline{\mathbf{s}}_i = \sum_{k=1}^{m} y_{i,k}\, \underline{s}_{k,i} .$$

Here we introduce the first approximation: let us assume that $\mathrm{m}(\underline{\mathbf{x}})$ is a *good* approximation of a root of $F(\mathbf{x})$ and $||F(\mathrm{m}(\underline{\mathbf{x}}))||$ is small enough, so that $0 \in \underline{n}_i(\mathbf{y}_i)$. Then, if $\overleftarrow{\underline{d}}_i(\mathbf{y}_i) = 1$,

$$\frac{\underline{n}_i(\mathbf{y}_i)}{\underline{d}_i(\mathbf{y}_i)} = \underline{n}_i(\mathbf{y}_i)$$

and (5.16) is reduced to

$$\min_{\overleftarrow{\underline{d}}_i(\mathbf{y}_i)=1} \mathrm{w}(\underline{n}_i(\mathbf{y}_i)) .$$

The width of the numerator is given only by the second term of equation (5.21), since the first term $\mathbf{y}_i F(\mathrm{m}(\underline{\mathbf{x}}))$ has zero width. Using property (5.18), the second term of the numerator can be rewritten as

$$\tfrac{1}{2}\sum_{\substack{j=1\\ j\neq i}}^{n} \mathrm{w}\big(\underline{x}_j\big)\left|\sum_{k=1}^{m} y_{i,k}\,\underline{s}_{k,j}\right|[-1,1]$$

and the numerator's width as

$$\mathrm{w}(\underline{n}_i(\mathbf{y}_i)) = \sum_{\substack{j=1\\ j\neq i}}^{n} \mathrm{w}\big(\underline{x}_j\big)\left|\sum_{k=1}^{m} y_{i,k}\,\underline{s}_{k,j}\right| . \tag{5.22}$$

Now, let us define the following variables:

$$\begin{aligned} \underline{r}_{i,j} &\triangleq \sum_{k=1}^{m} y_{i,k}\,\underline{s}_{k,j} \qquad \text{and} \\ v_j &\triangleq \overrightarrow{\underline{r}}_{i,j} + \overleftarrow{\underline{r}}_{i,j} . \end{aligned} \tag{5.23}$$

Variables v_j will be used to represent $\overrightarrow{\underline{n}}_i(\mathbf{y}_i)$, $\overleftarrow{\underline{n}}_i(\mathbf{y}_i)$ and $\mathrm{w}(\underline{n}_i(\mathbf{y}_i))$ in the linear programming problems.

We can now rewrite (5.22) as

$$\mathrm{w}(\underline{n}_i(\mathbf{y}_i)) = \sum_{\substack{j=1\\ j\neq i}}^{n} \mathrm{w}\big(\underline{x}_j\big)\left|\underline{r}_{i,j}\right| .$$

Using the previous properties (5.19) and (5.20), $\left|\underline{r}_{i,j}\right|$ can be expressed as

$$\left|\underline{r}_{i,j}\right| = \max\{-\overleftarrow{\underline{r}}_{i,j}, \overrightarrow{\underline{r}}_{i,j}\} = -\overleftarrow{\underline{r}}_{i,j} + (\overrightarrow{\underline{r}}_{i,j} + \overleftarrow{\underline{r}}_{i,j})^{+} = \overrightarrow{\underline{r}}_{i,j} + (\overrightarrow{\underline{r}}_{i,j} + \overleftarrow{\underline{r}}_{i,j})^{-} . \tag{5.24}$$

Property (5.17) allows us to write

$$\overrightarrow{\underline{r}}_{i,j} = \overrightarrow{\sum_{k=1}^{m} y_{i,k}\,\underline{s}_{k,j}} = \sum_{k=1}^{m}\big(y_{i,k}^{+}\overrightarrow{\underline{s}}_{k,j} - y_{i,k}^{-}\overleftarrow{\underline{s}}_{k,j}\big) \qquad \text{and} \tag{5.25}$$

$$\overleftarrow{\underline{r}}_{i,j} = \overleftarrow{\sum_{k=1}^{m} y_{i,k}\,\underline{s}_{k,j}} = \sum_{k=1}^{m}\big(y_{i,k}^{+}\overleftarrow{\underline{s}}_{k,j} - y_{i,k}^{-}\overrightarrow{\underline{s}}_{k,j}\big) . \tag{5.26}$$

Substituting in (5.24), we get

$$\left|\underline{r}_{i,j}\right| = \sum_{k=1}^{m}\big(y_{i,k}^{-}\overrightarrow{\underline{s}}_{k,j} - y_{i,k}^{+}\overleftarrow{\underline{s}}_{k,j}\big) + v_j^{+} \qquad \text{and} \tag{5.27}$$

$$\left|\underline{r}_{i,j}\right| = \sum_{k=1}^{m}\big(y_{i,k}^{+}\overrightarrow{\underline{s}}_{k,j} - y_{i,k}^{-}\overleftarrow{\underline{s}}_{k,j}\big) + v_j^{-} . \tag{5.28}$$

The width of the numerator can now be expressed as

$$\mathrm{w}(\underline{n}_i(\mathbf{y}_i)) = \delta \sum_{\substack{j=1\\ j\neq i}}^{n} \mathrm{w}(\underline{x}_j) \left(\sum_{k=1}^{m} \left(y^-_{i,k}\,\overrightarrow{\underline{s}}_{k,j} - y^+_{i,k}\,\overleftarrow{\underline{s}}_{k,j}\right) + v^+_j \right) +$$

$$+ (1-\delta) \sum_{\substack{j=1\\ j\neq i}}^{n} \mathrm{w}(\underline{x}_j) \left(\sum_{k=1}^{m} \left(y^+_{i,k}\,\overrightarrow{\underline{s}}_{k,j} - y^-_{i,k}\,\overleftarrow{\underline{s}}_{k,j}\right) + v^-_j \right) , \qquad (5.29)$$

which constitutes the objective function of the linear programming problem. Here, both equations (5.27) and (5.28) are added not only for flexibility but also for possible adjustments of the linear programming problems to attain numerical stability in their solution.

In the linear programming problem, $y^+_{i,k}$, $y^-_{i,k}$, v^+_j and v^-_j will be the $2(m+n-1)$ domain variables, subjected to non-negative constraints. Additionally, we should consider the constraints derived from he definition of variables v_j. Combining (5.25) and (5.26) with the definition of v_j (5.23), we have

$$v_j = \sum_{k=1}^{m} \left(y^+_{i,k} - y^-_{i,k}\right) \left(\overleftarrow{\underline{s}}_{k,j} + \overrightarrow{\underline{s}}_{k,j}\right) = 2 \sum_{k=1}^{m} \left(y^+_{i,k} - y^-_{i,k}\right) \mathrm{m}(\underline{s}_{k,j}) \; .$$

Since $v_j = v^+_j - v^-_j$, the constraints can be written as

$$v^+_j - v^-_j = 2 \sum_{k=1}^{m} \left(y^+_{i,k} - y^-_{i,k}\right) \mathrm{m}(\underline{s}_{k,j}) \; . \qquad (5.30)$$

The last constraint corresponds to the normalization condition. Using (5.26), the lower bound of the denominator is

$$\overleftarrow{\underline{d}}_i(\mathbf{y}_i) = \overleftarrow{\underline{r}}_{i,i} = \sum_{k=1}^{m} \left(y^+_{i,k}\,\overleftarrow{\underline{s}}_{k,i} - y^-_{i,k}\,\overrightarrow{\underline{s}}_{k,i}\right)$$

and the constraint results in

$$\sum_{k=1}^{m} \left(y^+_{i,k}\,\overleftarrow{\underline{s}}_{k,i} - y^-_{i,k}\,\overrightarrow{\underline{s}}_{k,i}\right) = 1 \; . \qquad (5.31)$$

Summarizing, a C^{W}-LP-preconditioner for the i-th variable is any solution of the linear programming problem with the $2(m+n-1)$ variables

$$(y^+_{i,1}, \ldots y^+_{i,m}, y^-_{i,1}, \ldots, y^-_{i,m}, v^+_1, \ldots, v^+_{i-1}, v^+_{i+1}, \ldots, v^-_m, \ldots, v^-_{i-1}, v^-_{i+1}, \ldots, v^-_m)$$

with the objective function (5.29), and with the n constraints (5.30) and (5.31), and subjected to non-negativity constraints on all the variables.

In fact, there is an abuse of notation here, since $y^+_{i,k}$, $y^-_{i,k}$, v^+_j and v^-_j are not necessarily the positive and negative parts of numbers. The preconditioner is actually formed by taking

$$y_{i,k} = y^+_{i,k} - y^-_{i,k}, \qquad 1 \leq k \leq m \; .$$

In [40], Novoa develops a general theory which clarifies in which circumstances the solution of such a linear programming problem corresponds to an optimal preconditioner. The adopted treatment is general; other optimal preconditioners and cases where $\check{\underline{\mathbf{x}}}$ is not the center of the box are included. The main result is that the linear programming problem is feasible if and only if the corresponding preconditioner exists. As stated above, a C-preconditioners exists, provided that at least one element of the corresponding row of $\underline{\mathbf{S}}$ does not contain 0 [19].

Structure of the Linear-programming Problem

The linear programming problem as given above can be written as

$$
\begin{aligned}
&\text{minimize } \mathcal{C}^t\mathcal{X}\\
&\text{subject to}\\
&\qquad \mathcal{A}\mathcal{X} = \mathcal{B}\,,\\
&\qquad \mathcal{X} \geq 0\,,
\end{aligned}
$$

where

$$
\begin{aligned}
\mathcal{A} &= \begin{pmatrix} 2\check{\underline{\mathbf{S}}}^t_{\neg i} & -2\check{\underline{\mathbf{S}}}^t_{\neg i} & -\mathbf{I}_{n-1} & \mathbf{I}_{n-1} \\ \overleftarrow{\underline{\mathbf{S}}}^t_i & -\overleftarrow{\underline{\mathbf{S}}}^t_i & \mathbf{0}_{1\times(n-1)} & \mathbf{0}_{1\times(n-1)} \end{pmatrix}\,,\\
\mathcal{B} &= \begin{pmatrix} \mathbf{0}_{(n-1)\times 1} \\ 1 \end{pmatrix}\,,\\
\mathcal{C} &= \left(-\mathrm{w}(\underline{\mathbf{x}}_{\neg i})^t\,\overleftarrow{\underline{\mathbf{S}}}^t_{\neg i}\,,\quad \mathrm{w}(\underline{\mathbf{x}}_{\neg i})^t\,\overrightarrow{\underline{\mathbf{S}}}^t_{\neg i}\,,\quad \mathrm{w}(\underline{\mathbf{x}}_{\neg i})^t\,,\quad \mathbf{0}_{1\times(n-1)}\right)\,,\\
\mathcal{X} &= (y^+_{i,1},\ldots y^+_{i,m}, y^-_{i,1},\ldots,y^-_{i,m}, v^+_1,\ldots,v^+_{i-1},v^+_{i+1},\ldots,v^-_m,\ldots,v^-_{i-1},v^-_{i+1},\ldots,v^-_m)\,,
\end{aligned}
$$

and

$\check{\underline{\mathbf{S}}}_{\neg i} \in \mathbb{R}^{m\times(n-1)}$	is the midpoint matrix of $\underline{\mathbf{S}}$ with the i-th column removed,
$\overleftarrow{\underline{\mathbf{S}}}_{\neg i} \in \mathbb{R}^{m\times(n-1)}$	is the lower bound matrix of $\underline{\mathbf{S}}$ with the i-th column removed,
$\overrightarrow{\underline{\mathbf{S}}}_{\neg i} \in \mathbb{R}^{m\times(n-1)}$	is the upper bound matrix of $\underline{\mathbf{S}}$ with the i-th column removed,
$\overleftarrow{\underline{\mathbf{S}}}^t_i$	is the i-th column of the lower bound matrix of $\underline{\mathbf{S}}$,
$\overrightarrow{\underline{\mathbf{S}}}^t_i$	is the i-th column of the upper bound matrix of $\underline{\mathbf{S}}$,
$\mathrm{w}(\underline{\mathbf{x}}_{\neg i})$	is the vector $\mathrm{w}(\underline{\mathbf{x}})$ with the i-th component removed,
$\mathbf{I}_{n-1} \in \mathbb{R}^{(n-1)\times(n-1)}$	is the identity matrix and
$\mathbf{0}_{p,q} \in \mathbb{R}^{p\times q}$	is a matrix of zeros.

5.8 Nonlinear Systems of Equations

In this section we combine the techniques explained in the previous sections into an overall algorithm for finding all solutions of a nonlinear system of equations $F(\mathbf{x}) = 0,\ F : \mathbb{R}^n \to$

$\mathbb{R}^m$. First, we will give an idea of the basic steps of the algorithm. In the subsequent sections, we will describe some heuristic-based variants that can improve its efficiency. Most of the ideas of these improved algorithms were introduced in [25] and [56] and are extensively developed in [24].

In what follows, $\|\mathrm{w}(\underline{\mathbf{x}})\|_{rel}$ will denote the *relative diameter* of box $\underline{\mathbf{x}}$, which is defined as

$$\|\mathrm{w}(\underline{\mathbf{x}})\|_{rel} = \max_{1\le i\le n} \left\{ \frac{\mathrm{w}(\underline{x}_i)}{\max\{1, \mathrm{m}(\underline{x}_i)\}} \right\} .$$

5.8.1 The Basic Branch-and-bound Algorithm

The basic algorithm is based on a branch-and-bound technique, where the interval Gauss-Seidel method is used to reduce the boxes.

```
Put $\underline{\mathbf{x}}_0$ into an empty list $\mathcal{L}$.

DO $k = 1$ to $M$ WHILE $\mathcal{L} \neq \emptyset$.

    1. Remove the first box from $\mathcal{L}$ and place it in the current box $\underline{\mathbf{x}}_c$.

    2. DO WHILE $\|\mathrm{w}(\underline{\mathbf{x}}_c)\|_{rel} > \epsilon_d$:
       a. Try to verify that $0 \notin F^u(\underline{\mathbf{x}}_c)$ using a natural extension.
          IF $0 \notin F^u(\underline{\mathbf{x}}_c)$ is verified, THEN EXIT loop 2 and CYCLE main loop.
       b. Compute the interval derivative $F'(\underline{\mathbf{x}}_c)$ or the slope matrix of $F$ over $\underline{\mathbf{x}}_c$
          at $\mathrm{m}(\underline{\mathbf{x}}_c)$ and compute $F(\mathrm{m}(\underline{\mathbf{x}}_c))$.
       c. Perform a Gauss-Seidel sweep using Equation (5.14) for $i = 1, \ldots, n$.
       d. IF the Gauss-Seidel sweep proved that $\underline{\mathbf{x}}_c$ could not contain any roots,
          THEN EXIT loop 2 and CYCLE main loop.
       e. IF the Gauss-Seidel sweep did not result in a change in $\underline{\mathbf{x}}_c$,
          THEN bisect $\underline{\mathbf{x}}_c$ and insert one of the boxes into $\mathcal{L}$, while the other one
          becomes $\underline{\mathbf{x}}_c$.
       END DO

    3. Insert $\underline{\mathbf{x}}_c$ into list $\mathcal{U}$.

END DO
```

Algorithm 5.2: Basic branch-and-bound algorithm for finding solutions of nonlinear systems of equations.

The algorithm takes as input a function $F : \mathbb{R}^n \to \mathbb{R}^m$, its interval derivative F' or a method to compute a slope matrix of this function, an initial box $\underline{\mathbf{x}}_0$, the maximum number of subboxes M to be processed, and a domain tolerance ϵ_d. As output we will get a list of boxes $\mathcal{U}$ containing all solutions to $F(\mathbf{x}) = 0$.

If we exit the algorithm properly, that is, we exit the main loop because $\mathcal{L}$ is empty without exceeding the maximum number of processed boxes, then all the solutions to $F(\mathbf{x}) = 0$ will be in the boxes of list $\mathcal{U}$. However, a box in $\mathcal{U}$ will contain a solution only if the existence condition of Theorem 5.2 holds. Usually, when the processed box is small enough, either existence can be proved or the box vanishes after a Gauss-Seidel sweep. The case in which existence cannot be proved in small boxes, and the Gauss-Seidel method does not eliminate it, seldom occurs in practice. Actually, we have never found it in our experiments. Note that the proof of existence does not require any extra computing.

Note also that Step 2.b of Algorithm 5.2, $F(\mathrm{m}(\underline{\mathbf{x}}_c))$ should be computed using interval arithmetic to bound roundoff errors.

In the following sections, we will briefly describe the variants of the basic algorithm that can improve its performance.

5.8.2 Bisection

In Step 2.e of Algorithm 5.2, the current box has to be split into two subboxes when the Gauss-Seidel algorithm is not able to make any improvement. The idea of generalized bisection is to replace a single box by two smaller boxes. This allows either the overestimation in interval extensions to be reduced, so that $0 \notin F^u(\underline{\mathbf{x}})$ can be verified (Step 2.a of Algorithm 5.2), or the interval Gauss-Seidel method to converge in the split boxes.

Generalized bisection divides a box by one of its coordinates, k: $\underline{\mathbf{x}}$ will be divided into $\underline{\mathbf{x}}^{(1)}$ and $\underline{\mathbf{x}}^{(2)}$, where $\underline{x}_i^{(1)} = \underline{x}_i^{(2)} = \underline{x}_i$ if $i \neq k$, $\underline{x}_k^{(1)} = [\overleftarrow{\underline{x}}_k, (\overleftarrow{\underline{x}}_k + \overrightarrow{\underline{x}}_k)/2]$ and $\underline{x}_k^{(2)} = [(\overleftarrow{\underline{x}}_k + \overrightarrow{\underline{x}}_k)/2, \overrightarrow{\underline{x}}_k]$.

A natural choice for the bisection coordinate k would be the coordinate for which the width $\mathrm{w}(\underline{x})_k$ is maximum. However, k corresponding to maximum width is not always the most effective at reducing overestimation or producing boxes in which the Gauss-Seidel method will converge.

In [27], Kearfott and Novoa proposed an heuristic choice of k, which they named *maximum smear heuristic.* This heuristic is similar to the one proposed in [17] (Section 8.8). A coordinate of maximum smear k satisfies

$$s_k = \max_{1 \leq j \leq n} \left\{ \max_{1 \leq i \leq m} \{ |\underline{s}_{i,j}| \} \mathrm{w}(\underline{x}_j) \right\} .$$

This heuristic attempts to choose a coordinate such that the function components vary the most across $\underline{x}_k$.

5.8.3 Introducing Uniqueness Verification

We can introduce a new output list of boxes, where uniqueness has been verified according to Theorems 5.1 or 5.3. Then, the output of the algorithm will consists of a list $\mathcal{R}$ of boxes, each of them containing a unique root, and a list $\mathcal{U}$ of boxes, whose relative diameter is smaller than ϵ_d, such that all roots of F in $\underline{\mathbf{x}}_0$, not in boxes in $\mathcal{R}$, are in boxes in $\mathcal{U}$.

5.8.4 Approximate Roots and ϵ-inflation

Existence and uniqueness verification is usually easier when a root is approximately centered within the box. This does not usually occur while iterating the interval Gauss-Seidel method with a width-optimal preconditioner.

Algorithm 5.2 can be improved using approximate roots when they are available. Then, boxes can be centered around the approximate root, so that $\|F(\check{\underline{\mathbf{x}}})\|$ is small. Centering and small norm make existence and uniqueness verification easier (see Lemma 6.2 in [24]).

On the other hand, if the Jacobian matrix is singular or ill-conditioned at a root, interval Newton methods may not be able to reduce any box containing it. In such cases, efficiency of the search process is increased if a small box is constructed, centered at the root, and increasing the widths until it is possible to verify existence or uniqueness. This process is named *ϵ-inflation* [56].

An algorithm for finding an approximate root, and verifying uniqueness within a box as large as possible around that root, is detailed in Section 4.2 in [24]. This algorithm will be introduced in Algorithm 5.2 between Steps 1 and 2 and also after the Gauss-Seidel sweep, if uniqueness could not be proved for the current box.

5.8.5 Box Complementation

ϵ-inflation shall be used together with a box complementation process. This allows us to remove boxes from the search region that are difficult to analyze. A box complementation algorithm produces a list of boxes whose union is the complement of a box in the union of an original list of boxes. For instance, in Figure 5.4, box complementation of box $\underline{\mathbf{x}}$ in the list of boxes $\underline{\mathbf{x}}_1$, $\underline{\mathbf{x}}_2$, $\underline{\mathbf{x}}_3$, results in $\underline{\mathbf{y}}_1$, $\underline{\mathbf{y}}_2$, $\underline{\mathbf{y}}_3$, $\underline{\mathbf{y}}_4$.

Observe that boxes are usually of higher dimension than this two-dimensional example, so that box complementation will generate still more boxes. Fortunately, in practice, box complementation does usually not increase significantly the total number of processed boxes. Experiments with the overall root finding algorithm in [25] indicate that it leads to a net advantage with approximate roots.

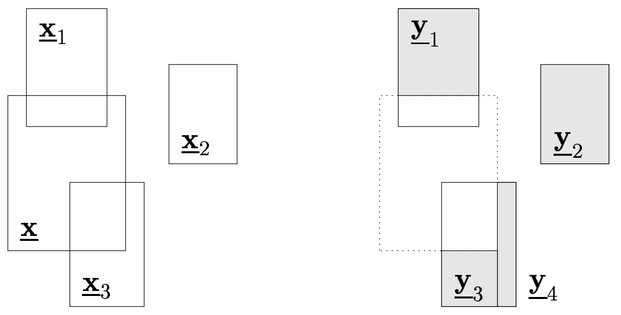

$\underline{\mathbf{x}}_1$
$\underline{\mathbf{x}}_2$
$\underline{\mathbf{x}}$
$\underline{\mathbf{x}}_3$
$\underline{\mathbf{y}}_1$
$\underline{\mathbf{y}}_2$
$\underline{\mathbf{y}}_3$
$\underline{\mathbf{y}}_4$

Figure 5.4: Box complementation.

5.9 Interval Cuts

The problem of finding solutions to systems of nonlinear equations has also been approached from a different point of view, based on refinements of constraint propagation techniques that have been developed in the AI and logic programming communities. Van Hentenryck, McAllester and Kapur described and implemented a system called Newton, which is exhaustively described in [67]. A latter technical report [35] gives a clarifying abstract presentation of the three *cuts* which are the base of the pruning phase of their algorithm.

The heart of Newton is a propagation process which iteratively improves known bounds on variables using a set of inference rules called *interval cuts*. An interval cut is a method of inferring new bounds on a variable so that every constraint appears to be locally consistent. The local consistency condition of Newton is called *box-consistency*, an approximation of arc-consistency, a well-known notion in artificial intelligence and used in many systems to solve discrete combinatorial search problems.

The three interval cuts used in Newton are:

- *Newton Cuts.* Newton cuts are based on a natural evaluation of an equation for a representative value of the variable to cut. Evaluating the derivative of that equation over the whole box, some values of the variable can be eliminated. An intuitive description of Newton cuts can be found in [35].
- *Snake Cuts.* Snake cuts are similar to Newton cuts, but they use a distributed evaluation of the function. They are more useful in very large boxes, while Newton cuts become more useful as the boxes get smaller.
- *Combination Cuts.* Combination cuts use a Taylor evaluation, which is more accurate for small boxes, over a linear combination of the constraint equations.

Let us briefly describe how a Newton cut proceeds.

A Newton cut is schematically represented in Figure 5.5. Let $e = 0$ be a constraint, $\underline{\mathbf{b}}$, a given box for the variables, x, a variable appearing in e (the one we want to reduce),

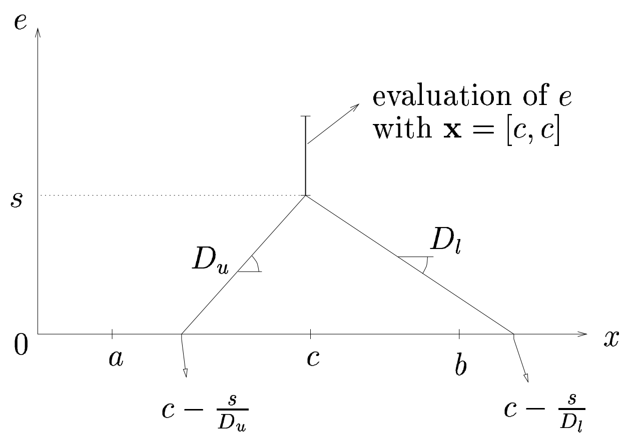


Figure 5.5: A Newton cut. Here the interval $[a, b]$ will be reduced to $[a, c - \frac{s}{D_u}]$.

and $[a, b]$, the interval of x in $\underline{\mathbf{b}}$. We choose c in the interval $[a, b]$. We evaluate e for a box identically to $\underline{\mathbf{b}}$ except that the variable x is fixed to c. Let us assume that the lower bound, s, is positive. Now we evaluate the derivative of e with respect to x in $\underline{\mathbf{b}}$ and denote the resulting interval $[D_l, D_u]$. Note that the values of slopes D_u and D_l represent the fastest possible rate of descent of the value of expression e. We can ensure that there is no solution with $x \in [c, b]$ when either $D_l \geq 0$ or $D_l < 0$ and $c - \frac{s}{D_l} > b$. If $D_l < 0$ and $c - \frac{s}{D_l} < b$, we can reduce interval $[c, d]$ to $[c - \frac{s}{D_l}, b]$. Likewise, there is no solution in $[a, c]$ if $D_u \leq 0$ or if $D_u > 0$ and $c - \frac{s}{D_u} < a$ and if $D_u > 0$ and $c - \frac{s}{D_u} > a$, we can reduce interval $[a, c]$ to $[a, c - \frac{s}{D_u}]$. It can be shown that we can choose c such that we always achieve some reduction of the bounds. Many heuristics can be used to choose c, but usually the middle point of the interval $[a, b]$ is taken.

Although Newton and snake cuts are less expensive to compute, at the end we nearly always end up with combination cuts, which are much more efficient in small boxes.

For further details on these three cuts refer to [35, 67].

It is important to note that combination cuts are quite similar to solving the system using the interval Newton method described in Section 5.6. The linearization is equivalent to the Taylor evaluation and preconditioning the system is equivalent to a linear combination of the constraint equations. Therefore, we are not going to use these interval cuts, but the interval Newton method described in Section 5.6. However, we have introduced interval cuts here, since we have used the ideas behind them to develop some specific cuts for our constraint equations (the closure equations), which are described in next chapter.

Chapter 6

Specific Interval Methods

In the previous chapter we have presented interval methods for solving systems of non-linear equations. In this chapter we exploit the special features of the inverse kinematics problem and the closure equations developed in Chapter 2: in Section 6.1 three domain-dependent cuts are presented, in Section 6.2 a nearly complete interval propagation for spherical mechanisms based on spherical geometry is described and, finally, in Sections 6.3 and 6.4 we develop a tighter interval evaluation for matrix $\mathbf{A}_i^j(\boldsymbol{\phi})$ defined in (3.2) and for other vectors required by direct cuts, respectively.

6.1 Direct Cuts

Following the idea behind interval cuts described in Section 5.9, we have developed a family of cuts specifically for the rotation equation (2.3) and the translation equation (2.4). In those equations, it is possible to isolate the variable we want to cut and evaluate directly its range of possible values; this is the reason for calling them *direct cuts*.

By using direct cuts we are reducing the overestimation of Newton, snake and combination cuts, since we are evaluating directly the variable we want to cut using a natural evaluation. Thus, the error due to the evaluation of its derivative is not introduced.

However, direct cuts are usually not able to reduce the interval of a variable to its actual range of possible values. This is due to interval dependency, that leads to overestimation (see Section 5.2.3).

Direct cuts also suffer from another drawback, which is common to Newton and snake cuts. All these cuts work only for a set of equations at a time. Usually they will not be able to reduce the range of the variables to the minimum considering the whole system. This fact can be seen more clearly with an example. Consider the system of equations:

$$\begin{cases} x - y = 0 \\ x + y = 1 \,. \end{cases}$$

If the initial box is $\underline{x} = [0,1], \underline{y} = [0,1]$, we cannot reduce it by applying cuts to the equations of the system independently. We will have to perform combination cuts or, equivalently, the interval Newton method described in Section 5.6, to get the solution ($x = 0.5$, $y = 0.5$).

Basically, direct cuts are used to accelerate the pruning process, which actually is the interest of domain-dependent cuts. But they are not enough by themselves to converge to a solution and require to be complemented with an interval Newton method or with combination cuts.

We can divide direct cuts into three groups: two cuts for the variables of rotation –derived from both the rotation and translation equation– and one cut for the variables of translation –derived from the translation equation.

6.1.1 Cutting ϕ_i through the Rotation Equation

The rotation equation (2.3) can be written as

$$\mathbf{A}_1^{i-1}(\boldsymbol{\phi})\mathbf{R}(\phi_i)\mathbf{Z}\mathbf{A}_{i+1}^{n}(\boldsymbol{\phi}) = \mathbf{I}\ .$$

In this equation, the only matrix that involves ϕ_i is $\mathbf{R}(\phi_i)$, which can be isolated:

$$\mathbf{R}(\phi_i) = (\mathbf{Z}\mathbf{A}_{i+1}^{n}(\boldsymbol{\phi})\mathbf{A}_1^{i-1}(\boldsymbol{\phi}))^t\ .$$

Let us define the interval matrix $\underline{\mathbf{V}}^i(\underline{\boldsymbol{\phi}})$ as:

$$\underline{\mathbf{V}}^i(\underline{\boldsymbol{\phi}}) \triangleq (\mathbf{Z}\mathbf{A}_{i+1}^{n}(\underline{\boldsymbol{\phi}})\mathbf{A}_1^{i-1}(\underline{\boldsymbol{\phi}}))^t\ . \tag{6.1}$$

All possible values of ϕ_i that close the spherical mechanism for the box $\underline{\boldsymbol{\phi}}$ must fulfill

$$\mathbf{R}(\phi_i) \in \underline{\mathbf{V}}^i(\underline{\boldsymbol{\phi}})\ .$$

Thus, the chain can be effectively closed if

$$1 \in \underline{v}_{11}^i \quad \text{and} \quad 0 \in \underline{v}_{12}^i, \underline{v}_{13}^i, \underline{v}_{21}^i, \underline{v}_{23}^i\ ,$$

where $\underline{v}_{jk}^i$ denotes the (j, k) element of matrix $\underline{\mathbf{V}}^i(\underline{\boldsymbol{\phi}})$.

If the previous conditions hold, the possible values for ϕ_i will be:

$$\underline{\phi}_i = \arccos(\underline{v}_{22}^i \cap \underline{v}_{33}^i) \cap \arcsin(\underline{v}_{32}^i \cap -\underline{v}_{23}^i)\ . \tag{6.2}$$

The interval of possible values for ϕ_i can be cut by intersecting the initial range of ϕ_i with the interval obtained from (6.2).

Note that the arcsin function gives only values between $-\pi/2$ and $+\pi/2$. However, we need to evaluate this function in an interval and get all possible values between 0 and 2π. In Appendix B it is shown that the result might encompass up to three intervals. Something similar happens with the arccos function, but there only two intervals are possible. Then, the intersection of intervals in (6.2) might lead to as much as four disjoint intervals. For the sake of simplicity, we take as the final interval for ϕ_i the convex hull of the resulting intervals. However, branching in these cases could be an important improvement in the efficiency of the overall algorithm.

6.1.2 Cutting ϕ_i through the Translation Equation

We can also isolate ϕ_i from the translation equation (2.4):

$$\sum_{k=1}^{i} \mathbf{A}_1^{k-1}(\boldsymbol{\phi}) \begin{pmatrix} d_k \\ 0 \\ 0 \end{pmatrix} + \mathbf{A}_1^{i}(\boldsymbol{\phi}) \left[\sum_{k=i+1}^{n} \mathbf{A}_{i+1}^{k-1}(\boldsymbol{\phi}) \begin{pmatrix} d_k \\ 0 \\ 0 \end{pmatrix} \right] = \mathbf{0} \, .$$

Multiplying by $(\mathbf{A}_1^{i-1}(\boldsymbol{\phi}))^t$ we get

$$\left(\mathbf{A}_1^{i-1}(\boldsymbol{\phi})\right)^t \sum_{k=1}^{i} \mathbf{A}_1^{k-1}(\boldsymbol{\phi}) \begin{pmatrix} d_k \\ 0 \\ 0 \end{pmatrix} + \mathbf{R}(\phi_i)\mathbf{Z} \sum_{k=i+1}^{n} \mathbf{A}_{i+1}^{k-1}(\boldsymbol{\phi}) \begin{pmatrix} d_k \\ 0 \\ 0 \end{pmatrix} = \mathbf{0} \, .$$

Defining

$$\mathbf{w}_0^i(\boldsymbol{\phi}, \mathbf{d}) \triangleq -\left(\mathbf{A}_1^{i-1}(\boldsymbol{\phi})\right)^t \sum_{k=1}^{i-1} \mathbf{A}_1^{k-1}(\boldsymbol{\phi}) \begin{pmatrix} d_k \\ 0 \\ 0 \end{pmatrix} \quad \text{and} \tag{6.3}$$

$$\mathbf{w}_1^i(\boldsymbol{\phi}, \mathbf{d}) \triangleq \sum_{k=i+1}^{n} \mathbf{A}_{i+1}^{k-1}(\boldsymbol{\phi}) \begin{pmatrix} d_k \\ 0 \\ 0 \end{pmatrix} \, , \tag{6.4}$$

we can write

$$\begin{pmatrix} d_k \\ 0 \\ 0 \end{pmatrix} - \mathbf{w}_0^i(\boldsymbol{\phi}, \mathbf{d}) = \mathbf{R}(\phi_i)\mathbf{Z}\mathbf{w}_1^i(\boldsymbol{\phi}, \mathbf{d}) \, .$$

In other words,

$$\begin{pmatrix} 0 & -1 & 0 \\ \cos\phi_i & 0 & -\sin\phi_i \\ \sin\phi_i & 0 & \cos\phi_i \end{pmatrix} \begin{pmatrix} w_{11}^i \\ w_{12}^i \\ w_{13}^i \end{pmatrix} = \begin{pmatrix} w_{01}^i - d_i \\ w_{02}^i \\ w_{03}^i \end{pmatrix} \, , \tag{6.5}$$

where w_{0j}^i and w_{1j}^i stand for the (j) element of vectors $\mathbf{w}_0^i(\boldsymbol{\phi}, \mathbf{d})$ and $\mathbf{w}_0^i(\boldsymbol{\phi}, \mathbf{d})$ respectively.

Let us extend the definitions of $\mathbf{w}_0^i(\boldsymbol{\phi}, \mathbf{d})$ and $\mathbf{w}_1^i(\boldsymbol{\phi}, \mathbf{d})$ (Equations (6.3) and (6.4)) to the interval case. When evaluated over a box of rotations and translations $(\underline{\boldsymbol{\phi}}, \underline{\mathbf{d}})$, we will denote both vectors with an underscore. $\underline{w}_{0j}^i$ and $\underline{w}_{1j}^i$ will stand for the (j) element of these vectors.

From Equation (6.5) it can be seen that the chain can be effectively closed if

$$0 \in (\underline{w}_{01}^i + \underline{w}_{12}^i - \underline{d}_i) \, .$$

Solving the linear system derived from (6.5), we obtain the possible values for ϕ_i:

$$\underline{\phi}_i = \arcsin\left(\frac{\underline{w}_{03}^i \underline{w}_{11}^i - \underline{w}_{13}^i \underline{w}_{02}^i}{\underline{w}_{11}^{i\,2} + \underline{w}_{13}^{i\,2}}\right) \bigcap \arccos\left(\frac{\underline{w}_{02}^i \underline{w}_{11}^i - \underline{w}_{13}^i \underline{w}_{03}^i}{\underline{w}_{11}^{i\,2} + \underline{w}_{13}^{i\,2}}\right) \, . \tag{6.6}$$

As previously, we can cut the initial interval of ϕ_i by intersecting it with the range obtained in (6.6).

The division of intervals by an interval including the origin leads to disjoint intervals (Section 5.2.2). However, in equation (6.6), the denominators are greater or equal than 0 and the division leads to a single interval, possibly extending to infinity. This does not imply any difficulty, since we have to intersect this interval with $[-1, 1]$, the interval where the arcsin and arccos functions are defined.

Note also that the translation equation depends on where we place the first bar of the n-bar mechanism in the chain. This can be easily seen by observing that the translation equation (2.4) does not involve the last angle ϕ_n. In general, the resulting direct cut using Equation (6.6) will be different from the one obtained using a translational equation where the first angle is another one. Therefore, we have n different translation equations and each angle ϕ_i can be cut using any of them. This was not the case with the rotation equation, since it is the same wherever we take the reference.

In Section 6.4 we show how the interval evaluation of $\underline{\mathbf{w}}_0^i(\underline{\boldsymbol{\phi}}, \underline{\mathbf{d}})$ and $\underline{\mathbf{w}}_1^i(\underline{\boldsymbol{\phi}}, \underline{\mathbf{d}})$ can be computed much more accurately than using directly the definitions (6.3) and (6.4).

6.1.3 Cutting d_i through the Translation Equation

In order to isolate d_i, we express the translation equation (2.4) as follows:

$$\sum_{k=1}^{i-1} \mathbf{A}_1^{k-1}(\boldsymbol{\phi}) \begin{pmatrix} d_k \\ 0 \\ 0 \end{pmatrix} + \mathbf{A}_1^{i-1}(\boldsymbol{\phi}) \begin{pmatrix} d_i \\ 0 \\ 0 \end{pmatrix} + \sum_{k=i+1}^{n} \mathbf{A}_1^{k-1}(\boldsymbol{\phi}) \begin{pmatrix} d_k \\ 0 \\ 0 \end{pmatrix} = \mathbf{0} \, .$$

Multiplying this equation by $\left(\mathbf{A}_1^{i-1}\right)^t$, we get:

$$\begin{pmatrix} d_i \\ 0 \\ 0 \end{pmatrix} = - \left(\mathbf{A}_1^{i-1}(\boldsymbol{\phi})\right)^t \sum_{k=1}^{i-1} \mathbf{A}_1^{k-1}(\boldsymbol{\phi}) \begin{pmatrix} d_i \\ 0 \\ 0 \end{pmatrix} - \sum_{k=i+1}^{n} \mathbf{A}_i^{k-1}(\boldsymbol{\phi}) \begin{pmatrix} d_k \\ 0 \\ 0 \end{pmatrix} =$$
$$= \mathbf{w}_0^i(\boldsymbol{\phi}, \mathbf{d}) - \mathbf{A}_i^i(\boldsymbol{\phi}) \mathbf{w}_1^i(\boldsymbol{\phi}, \mathbf{d}) \, . \quad (6.7)$$

Defining

$$\underline{\mathbf{w}}_3^i(\underline{\boldsymbol{\phi}}, \underline{\mathbf{d}}) \overset{\triangle}{=} \underline{\mathbf{w}}_0^i(\underline{\boldsymbol{\phi}}, \underline{\mathbf{d}}) - \mathbf{A}_i^i(\underline{\boldsymbol{\phi}}) \underline{\mathbf{w}}_1^i(\underline{\boldsymbol{\phi}}, \underline{\mathbf{d}}) \, , \quad (6.8)$$

it can be seen from Equation (6.7), that the chain can be closed if

$$0 \in \underline{w}_{32}^i, \underline{w}_{33}^i \, .$$

The possible values for d_i are:

$$\underline{d}_i = \underline{w}_{31}^i \, . \quad (6.9)$$

We can cut the initial interval of d_i by intersecting it with the range obtained in (6.9).

Here n different translation equations can also be used to cut each $\underline{d}_i$, as in the previous subsection.

6.2 Interval Propagation in Spherical Mechanisms

Interval propagation in spherical mechanisms is an unsolved problem in its most general formulation. Sometimes a revolute pair is not able to perform complete rotations due to design constraints, collision between links, or even limitations coming from other loops sharing the same revolute pair. Thus, in general, the values for the variables are externally restricted to an interval. The most general problem is then finding all the values for a rotation ϕ_k that are compatible with those actually possible for the other variables ϕ_i $(i \neq k)$.

E. Celaya obtained an algorithm for interval propagation within a rotation equation based on geometric considerations, which could be extended to multiple loops [10]. Although this algorithm works for any number of variables, it only gives the range for a certain variable when another one is restricted to an interval and all other variables are free. This limitation makes the algorithm of little interest, specially for more than 5 variables. We have not devised any way to extend this algorithm to the general case.

We have developed an algorithm based on spherical geometry which is able to perform a nearly general interval propagation: it propagates the range of all except one variable over another one. Therefore, the resulting interval is usually not the minimal one, except for those cases in which at least one variable is not restricted.

Interval propagation can be used as a cut over the rotation equation. Consider a box of variables of rotations. We can propagate the intervals of $n-1$ variables over the remaining variable and reducing its range of possible values. Observe that these cuts use only the rotation equation. Therefore, they act similarly than direct cuts of Section 6.1.1. Nevertheless, the interval propagation described here and direct cuts are based on completely different basis, thus leading to different results. Usually, interval propagation gives tighter enclosures, but it is also computationally much more expensive. This trade-off between speed and tight enclosures should be carefully evaluated to obtain a good performance.

6.2.1 Background and Definitions

The closure equation of any spherical mechanism (not necessarily orthogonal) can always be expressed as the rotational equation of an n-bar mechanism (it is a particular case of Remark 2.2):

$$\mathbf{F}(\boldsymbol{\phi}) = \prod_{i=1}^{n} \mathbf{R}(\phi_i)\mathbf{Z} = \mathbf{I}\,. \tag{6.10}$$

Consider the geometric interpretation of this equation as an n-sided spherical polygon with arcs of length $\pi/2$ and exterior angles ϕ_i as described in Section 3.2.2. The rotation equation (6.10) will hold for a vector of rotations $\boldsymbol{\phi} = (\phi_1, \phi_2, \ldots, \phi_n)$ if, and only if, the corresponding spherical chain closes (Figure 6.1).

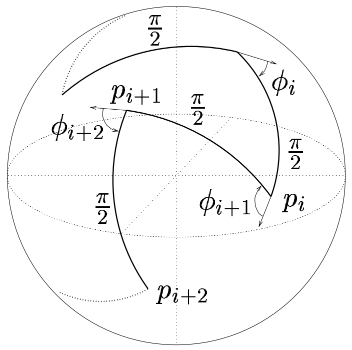


Figure 6.1: Spherical polygon generated by the x-axis with sides of length $\frac{\pi}{2}$ and exterior angles equal to ϕ_i.

We will name the vertices of this chain $p_0, p_1, \ldots, p_n$, where p_i is the vertex between arc i and arc $i+1$.

Definition 6.1 (Spherical Interval). We define a spherical interval $[\phi_a, \phi_b]$ as the set of values between ϕ_a and ϕ_b, considered as angles between 0 and 2π, in counter-clockwise sense (Figure 6.2).

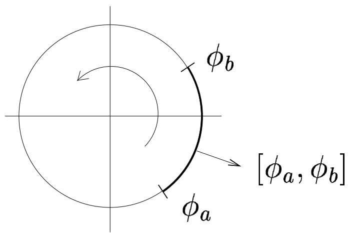


Figure 6.2: A spherical interval, $[\phi_a, \phi_b]$.

For instance, the spherical interval $[150, 120]$ includes all angles from 150 to 360 and from 0 to 120 degrees.

Definition 6.2 (Spherical Region). A spherical region $\mathcal{R}_s^t$ is the set of possible positions of the end of arc t, assuming that $\phi_i \in [\phi_{ia}, \phi_{ib}]$, for $i = s+1, \ldots, t$, and that the polygon is open and fixed at the beginning of arc s.

For example, R_1^3 is the region swept by point p_3 (end of arc 3) with respect to arc 1 when ϕ_2 and ϕ_3 vary in $[\phi_{2a}, \phi_{2b}]$ and $[\phi_{3a}, \phi_{3b}]$ respectively (Figure 6.3).

Let us point out some properties about spherical regions:

1. $\mathcal{R}_i^i = p_i$.

2. $\mathcal{R}_{i-1}^i$ is a maximal arc of center p_{i-1} and "length" $\phi_b - \phi_a$.

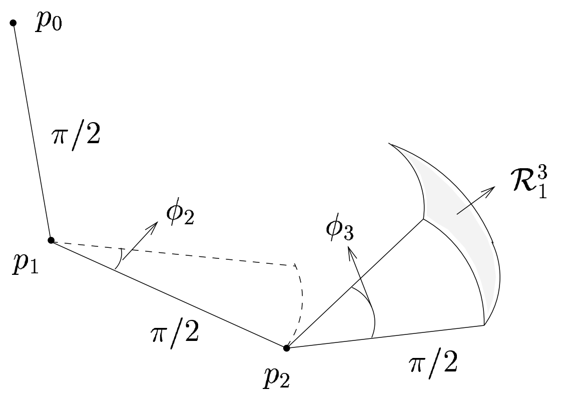


Figure 6.3: An example of a spherical region, $\mathcal{R}_1^3$.

3. $\mathcal{R}_i^t \subset \mathcal{R}_j^t$ if $j \leq i$.

4. R_s^t is always a region bounded by maximal arcs.

5. All points in $\mathcal{R}_s^t$ are possible centers for arcs $\mathcal{R}_t^{t+1}$.

6.2.2 Outlook of the Algorithm

We will consider an interval propagation of variables ϕ_i $(i = 2, \ldots, n)$ over variable ϕ_1. Propagations over other variables can be achieved by simply shifting indices of the variables.

The basic idea of the propagation algorithm is to characterize exactly region $\mathcal{R}_1^n$. This region includes all possible positions of p_n when $\phi_i \in [\phi_{ia}, \phi_{ib}]$, for $i = 2, \ldots, n$. If the starting point of the chain is not in this region ($p_0 \notin \mathcal{R}_1^n$), then the spherical chain cannot be effectively closed for those rotations. Otherwise, if $p_0 \in \mathcal{R}_1^n$, there are values for ϕ_1 that close the chain.

The problem is now to compute the range of possible values for ϕ_1. Region $\mathcal{R}_1^{n-1}$ includes all possible positions for p_{n-1}. But p_{n-1} has to be at $\pi/2$ radians from p_0. Thus, we can intersect a circle of radius $\pi/2$ radians with $\mathcal{R}_1^{n-1}$ and p_{n-1} will necessarily be in this arc (we will name it $\breve{\mathcal{R}}_1^{n-1}$). We can now compute the possible angles for ϕ_1 (see Figure 6.4 on next page). But this range of possible values is not the minimal range. In other words, that variable ϕ_1 has to be included in this range of possible values, is a necessary but not sufficient condition for the spherical chain to close for the given ranges of the other variables. The problem comes from the fact that all points in arc $\breve{\mathcal{R}}_1^{n-1}$ are obviously possible locations for point p_{n-1} (since $\breve{\mathcal{R}}_1^{n-1} \in \mathcal{R}_1^{n-1}$). But if we draw an arc from some of these points to p_0, it will probably require ϕ_{n-1} to be out of its range.

The interval propagation over ϕ_1 leads only to the exact range if variable ϕ_{n-1} is not restricted; otherwise, the resulting range for ϕ_1 is wider.

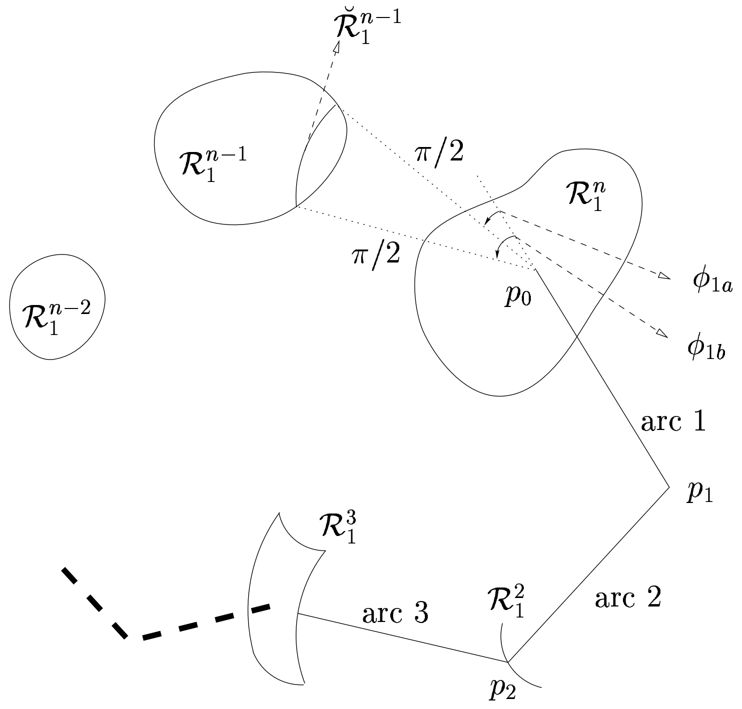


Figure 6.4: Possible angles for ϕ_1.

6.2.3 The Overall Propagation Algorithm

The interval propagation algorithm can be summarized in the following steps:

```
1. Compute $\mathcal{R}_1^n$

2. IF $p_0 \notin \mathcal{R}_1^n$
   THEN the chain cannot be closed
   ELSE ($p_0 \in \mathcal{R}_1^n$)
         a. Compute $\mathcal{R}_1^{n-1}$
         b. Compute $\breve{\mathcal{R}}_1^{n-1}$
         c. Compute the range of possible angles for $\phi_1$
```

Algorithm 6.1: Interval propagation in Spherical Mechanisms.

They key point of the algorithm is to efficiently compute the spherical region $\mathcal{R}_i^j$ (steps 1, 2.a. and 2.b.). We describe it in next section.

$\breve{\mathcal{R}}_1^{n-1}$ is obtained by intersecting $\mathcal{R}_1^{n-1}$ with a circle of radius $\pi/2$ centered at p_0. Note that, depending on the shape of $\mathcal{R}_1^{n-1}$, $\breve{\mathcal{R}}_1^{n-1}$ may consist of one or more disjoint spherical intervals. The range of possible values for ϕ_1 (step 2.c.) is then straightforwardly computed as the exterior angle between the possible arcs from $\breve{\mathcal{R}}_1^{n-1}$ to p_0 and the first arc of the chain (Figure 6.4).

6.2.4 Computation of Spherical Regions

First, we need some definitions.

Definition 6.3 (Arc or Maximal Arc). We define a *maximal arc* (or simply an *arc*) as the connected subset of the intersection between the unit sphere and a plane through its center.

Definition 6.4 (Center of an Arc). The *center of an arc c* is a point on the unit sphere, which is equidistant to all points of the arc. There are two diametrically opposed centers for each arc. For an oriented arc (an arc with an initial point p_i and an end point p_e), we will consider the center the one that defines an angle in counter-clockwise sense from p_i to p_e (Figure 6.5), when the sphere is seen from outside.

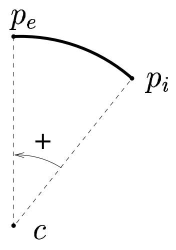


Figure 6.5: A spherical arc and its center.

Equivalently, the center c of an oriented arc is a point over the sphere which is equidistant to all points of the arc, such that

$$|\ \overline{oc}\ \overline{op_i}\ \overline{op_e}\ | > 0\ ,$$

where o is the center of the sphere.

Definition 6.5 (Spherical Quadrangle). The spherical region of the type $\mathcal{R}_{i-2}^{i}$, which is delimited by four arcs (Figure 6.6), is called a *spherical quadrangle.*

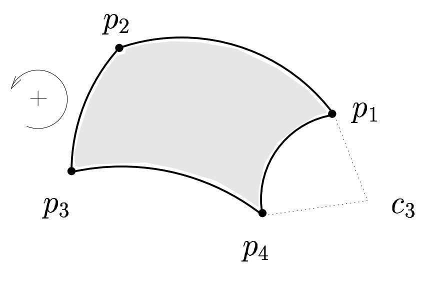


Figure 6.6: A spherical quadrangle.

We will define a spherical quadrangle by its four vertices (p_1, p_2, p_3 and p_4) and the four centers of the arcs (c_1, c_2, c_3 and c_4), where c_i is the center of arc $p_i\, p_{i+1}$. However, the contour is not enough to define a quadrangle; we also need to know which is the *interior* of the quadrangle. Note that a quadrangle can be seen from two sides as shown in Figure 6.7.

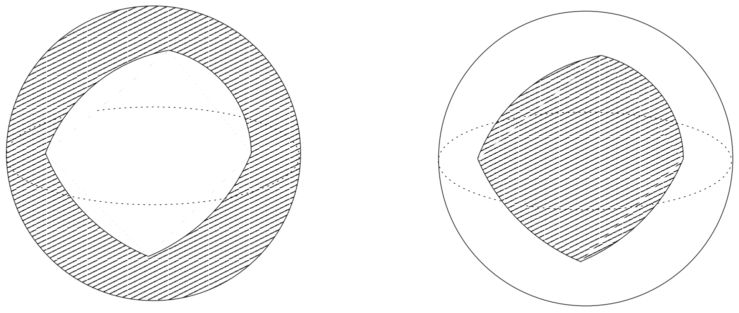


Figure 6.7: The same outline may define two spherical quadrangle.

We define the interior of a quadrangle as those points which see points p_1, p_2, p_3 and p_4 in counter-clockwise sense (Figure 6.7).

A spherical region can be a point, an arc, a spherical quadrangle or a connected set of points with a more complex shape. In the latter case, the region can always be described as the union of possibly overlapping spherical quadrangles. Now, we can represent a spherical region as a set of quadrangles that cover completely the region and by an outline, i.e. a set of arcs that cover completely the border. Some of the arcs of the outline, or parts of them, can also be inside the region (Figure 6.8). The order of the arcs also determine the region interior as in a single quadrangle.

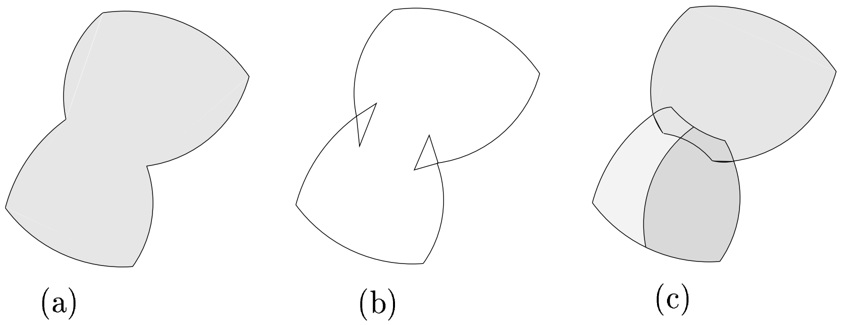


Figure 6.8: A region (a), its outline (b) and a possible set of quadrangles covering the region (c).

To check if a point is included in a region, we have only to verify that this point is contained at least in one of the quadrangles that compose that region.

We will use spherical coordinates (Figure 6.9) to express the position of points over the unit sphere.

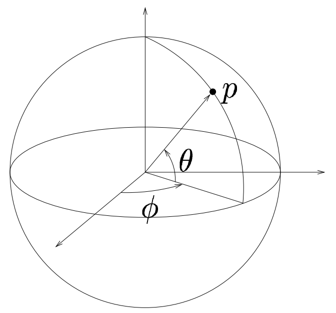


Figure 6.9: Spherical coordinates.

The algorithm for computing spherical regions generates them by progressively sweeping all rotation variables in their ranges. It begins with an initial chain with all the variables set to the lower limit of their interval: $\phi_i = \phi_{i_a}$. Then it generates the region $\mathcal{R}^n_{n-1}$, which will be an arc. Rotating ϕ_{n-1} to its higher limit (ϕ_{n-1_b}), it generates $\mathcal{R}^n_{n-2}$, which will be a spherical quadrangle. We can enlarge the region until we get $\mathcal{R}^n_1$ by successively sweeping all variables in their ranges.

Let's assume that we have a region $\mathcal{R}^n_i$ and we want to enlarge this region rotating ϕ_i to get $\mathcal{R}^n_{i-1}$. An easy way to do this is to rotate everything in the unit sphere (the quadrangles and the initial chain) in order to put p_{i-1} on the South pole, $(0, -\pi/2)$. Then, the rotation about p_{i-1} of any point will only affect its first coordinate. To generate the new region, we add to the sides facing left (the sense of rotation of ϕ_i) new quadrangles, enlarging also the outline (Figure 6.10).

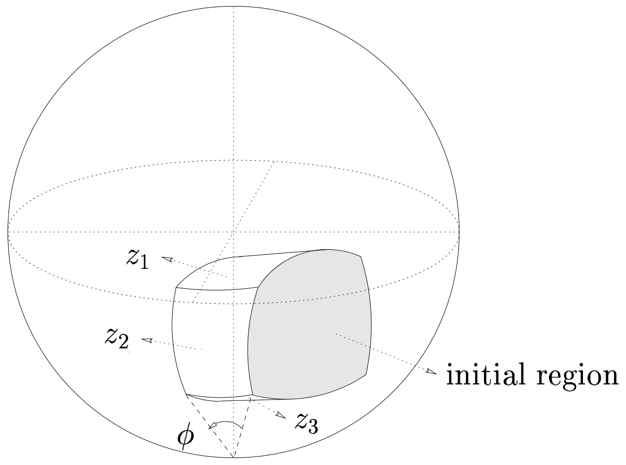


Figure 6.10: Generation of a new spherical region. Rotating the initial region around the South pole ϕ degrees, quadrangles z_1, z_2 and z_3 are generated.

Attention has to be paid to the arcs that have a maximum or a minimum with respect to coordinate θ. If this happens, the arcs are divided into two arcs and only those that face left lead to new quadrangles.

The algorithm for generating spherical regions can be summarized as follows:

```
1. Generate the initial chain with $\phi_i = \phi_{i_a}$

2. Change the coordinates in order to locate $p_{n-1}$ at the South pole
   Rotate $p_n$ $\phi_n$ degrees around the South pole and generate arc $\mathcal{R}^n_{n-1}$

3. Change the coordinates in order to locate $p_{n-2}$ at the South pole
   Rotate arc $\mathcal{R}^n_{n-1}$ $\phi_{n-1}$ degrees around the South pole and generate quadrangle $\mathcal{R}^n_{n-2}$
   Put $\mathcal{R}^n_{n-2}$ into an empty list of quadrangles $\mathcal{Z}$ and its arcs in a cyclic ordered
      list $\mathcal{O}$ (the outline)

4. FOR $i = 3$ TO $n$ DO
   a. Change the coordinates in order to locate $p_{n-i}$ at the South pole
   b. Divide those arcs of the outline that contain maximums or minimums
   c. FOR each arc of the outline that faces left ($\theta_i > \theta_e$) DO
         Generate a new quadrangle rotating the arc $\phi_{n-i}$ degrees around the
            South pole and include it in $\mathcal{Z}$
         Remove that arc from the outline $\mathcal{O}$ and add the three new arcs
            of the new quadrangle
```

Algorithm 6.2: Generating a spherical region.

Key Points and Implementation

Although the general ideas of the propagation algorithm are not complicated (Sections 6.2.2 and 6.2.4), the implementation requires some attention.

We enumerate here the main subtasks we need to solve and point out some of their key points:

1. *Changing coordinates.* Locating the rotation point at the South pole makes the generation of new quadrangles much easier. However, this implies a change of coordinates of all previously computed quadrangles and of the original chain. Minimizing here the number of computations is essential.

   If we want to change the coordinates of point p to put q at the South pole, we apply the following change of coordinates:

$$p_\phi = \operatorname{atan2}\bigl(\sin(p_\phi - q_\phi)\ ,\ \cos q_\theta * \tan p_\theta - \sin q_\theta * \cos(p_\phi - q_\phi)\bigr)$$
$$p_\theta = -\arcsin\bigl(\cos q_\theta * \cos(p_\phi - q_\phi) * \cos p_\theta + \sin q_\theta * \sin p_\theta\bigr)\ .$$

2. *Dividing arcs at maximums or minimums.* We have to consider the case of an arc with a minimum and a maximum, although it seldom happens in practice.

3. *Generating new quadrangles.* This is an inexpensive operation, since only the second coordinate of the new arcs is affected. The center of the new horizontal arcs will be the South or north pole.

4. *Enlarging the outline.* This is one of the trickiest points. Four different cases have to be considered in order to avoid arcs of the outline to be in the interior of the spherical region, which would generate useless quadrangles.

5. *Verifying whether a point is in the interior of a quadrangle.* We locate successively the centers of the arcs at the South pole and verify whether the point is under or above the parallel of the arc.

6. *Computing arc $\breve{\mathcal{R}}_1^{n-1}$.* We put p_0 at the South pole and look for those arcs of the outline that intersect the equator. Note that $\breve{\mathcal{R}}_1^{n-1}$ may consist of one, two or more disjoint spherical arcs.

7. *Computing the range of ϕ_1.* It is straightforward from $\breve{\mathcal{R}}_1^{n-1}$.

This interval propagation algorithm has been implemented in C and tested in several spherical mechanisms. The propagation does not result in the smallest possible interval, but by iterating it for all variables gives usually tight enclosures of the actual range.

Used as interval cuts for the rotation equation of an n-bar mechanism, this interval propagation does effectively cut the box of variables of rotation. Moreover, it gives tighter enclosures of the actual range of possible values for the variables than applying direct cuts over the rotation equation.

However, when solving an inverse kinematics problem of a spatial mechanism, the influence of cuts acting only over the rotation equation is small. Direct cuts are useful since they require few computations, but cuts derived from an interval propagation are computationally too expensive. Experiments show that it is not worth the time required by them for the small improvement achieved in front of direct cuts.

6.3 Accurate Interval Evaluation of Matrices $\mathbf{A}_i^j(\boldsymbol{\phi})$

It has already been mentioned that one of the major problems in interval methods is the overestimation of the actual range of possible values when evaluating a function with one or more variables appearing more than once. This is exactly the case in the interval evaluation of matrices $\mathbf{A}_i^j(\boldsymbol{\phi})$, which appear both in direct cuts and in the parametric equations. In this section we introduce an exact interval evaluation for the product

$$\mathbf{R}(\phi_i)\mathbf{Z}\begin{pmatrix} x \\ y \\ z \end{pmatrix} . \qquad (6.11)$$

This product appears repeatedly in the direct cuts derived from the translation equation (Sections 6.1.2 and 6.1.3) and also leads to a more accurate evaluation of matrices $\mathbf{A}_i^j(\underline{\boldsymbol{\phi}})$ without much extra computing.

A natural evaluation of (6.11) results in:

$$\begin{pmatrix}\underline{a}\\ \underline{b}\\ \underline{c}\end{pmatrix} = \begin{pmatrix}0 & -1 & 0\\ \cos\underline{\phi}_i & 0 & -\sin\underline{\phi}_i\\ \sin\underline{\phi}_i & 0 & \cos\underline{\phi}_i\end{pmatrix}\begin{pmatrix}\underline{x}\\ \underline{y}\\ \underline{z}\end{pmatrix} = \begin{pmatrix}-\underline{y}\\ \underline{x}\cos\underline{\phi}_i - \underline{z}\sin\underline{\phi}_i\\ \underline{x}\sin\underline{\phi}_i + \underline{z}\cos\underline{\phi}_i\end{pmatrix}.$$

The first component, $\underline{a}$, yields the exact value, but there is an overestimation in the evaluation of $\underline{b}$ and $\underline{c}$ since ϕ_i appears twice in each expression. We can regard both expressions as particular cases of

$$\underline{p} = \underline{m}\sin\underline{\phi} + \underline{n}\cos\underline{\phi}\ . \tag{6.12}$$

Our goal is to obtain the exact range of $\underline{p}$ evaluated over a box $<\underline{m}, \underline{n}, \underline{\phi}>$. Since $\underline{p}$ is linear with respect to m and n, the maximum and the minimum values of $\underline{p}$ must lie in one of the four edges of the box corresponding to the extremes of $\underline{m}$ and $\underline{n}$ (Figure 6.11).

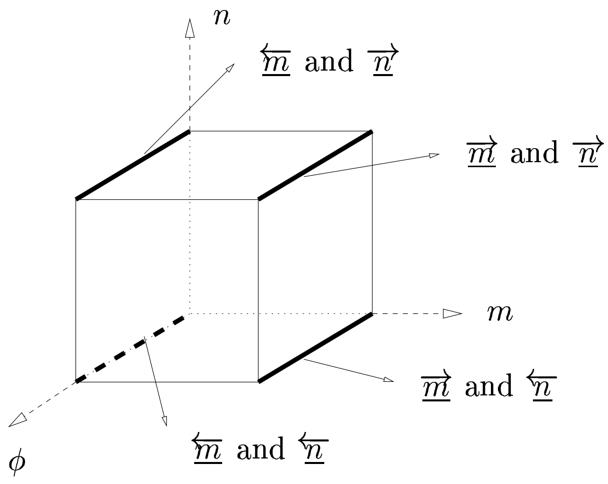

n
$\overleftarrow{m}$ and $\overrightarrow{n}$
$\overrightarrow{m}$ and $\overrightarrow{n}$
m
$\overrightarrow{m}$ and $\overleftarrow{n}$
$\overleftarrow{m}$ and $\overleftarrow{n}$
ϕ

Figure 6.11: Box $<\underline{m}, \underline{n}, \underline{\phi}>$ and the edges where the maximum and minimum values of $\underline{p}$ (6.12) have to be.

Let us analyze the upper right edge corresponding to $\overrightarrow{m}$ and $\overrightarrow{n}$. Equating the derivative of p with respect to ϕ to zero,

$$\frac{\partial p}{\partial \phi} = \overrightarrow{m}\cos\phi - \overrightarrow{n}\sin\phi = 0\ ,$$

gives us two extremes for ϕ:

$$\phi_{e1} = \arctan\frac{\overrightarrow{m}}{\overrightarrow{n}} \quad \text{and} \quad \phi_{e2} = \arctan\frac{\overrightarrow{m}}{\overrightarrow{n}} + \pi\ .$$

The exact range of p for that edge will be the convex hull of p evaluated at $\overrightarrow{\underline{\phi}}$, $\overleftarrow{\underline{\phi}}$, ϕ_{e1} (only if $\phi_{e1} \in \underline{\phi}$) and ϕ_{e2} (only if $\phi_{e2} \in \underline{\phi}$):

$$\begin{aligned}\underline{p}_{\overrightarrow{\underline{m}},\overrightarrow{\underline{n}}} =& p(\overrightarrow{\underline{m}}, \overrightarrow{\underline{n}}, \overrightarrow{\underline{\phi}}) \cup p(\overrightarrow{\underline{m}}, \overrightarrow{\underline{n}}, \overleftarrow{\underline{\phi}}) \\ & \cup p(\overrightarrow{\underline{m}}, \overrightarrow{\underline{n}}, \phi_{e1}) \quad \text{(only if } \phi_{e1} \in \underline{\phi}) \\ & \cup p(\overrightarrow{\underline{m}}, \overrightarrow{\underline{n}}, \phi_{e2}) \quad \text{(only if } \phi_{e2} \in \underline{\phi}) \ .\end{aligned}$$

Similarly, we obtain the range of p for the other edges. The exact range for p over the whole box $< \underline{m}, \underline{n}, \underline{\phi} >$ is then the convex hull of the ranges of p for the four edges:

$$\underline{p} = \underline{p}_{\overrightarrow{\underline{m}},\overrightarrow{\underline{n}}} \cup \underline{p}_{\overrightarrow{\underline{m}},\overleftarrow{\underline{n}}} \cup \underline{p}_{\overleftarrow{\underline{m}},\overrightarrow{\underline{n}}} \cup \underline{p}_{\overleftarrow{\underline{m}},\overleftarrow{\underline{n}}} \ .$$

The computation of the exact range of $\underline{p} = \underline{m} \sin \underline{\phi} + \underline{n} \cos \underline{\phi}$ can be summarized in the following algorithm:

```
$\underline{p} = \emptyset$
FOR $a = \overrightarrow{\underline{m}}$ and $a = \overleftarrow{\underline{m}}$ DO
    FOR $b = \overrightarrow{\underline{n}}$ and $b = \overleftarrow{\underline{n}}$ DO
        $\underline{p} = \underline{p} \cup p(a, b, \overrightarrow{\underline{\phi}}) \cup p(a, b, \overleftarrow{\underline{\phi}})$
        $\phi_e = \arctan \frac{a}{b}$
        IF $\phi_e \in \underline{\phi}$ THEN $\underline{p} = \underline{p} \cup p(a, b, \phi_e)$
        IF $\phi_e + \pi \in \underline{\phi}$ THEN $\underline{p} = \underline{p} \cup p(a, b, \phi_e + \pi)$
```

Algorithm 6.3: Exact range of $\underline{p} = \underline{m} \sin \underline{\phi} + \underline{n} \cos \underline{\phi}$.

Observe that this procedure gives the exact range of the product (6.11) only if the components of the vector (x, y and z) are independent.

When evaluating $\mathbf{A}_i^j(\boldsymbol{\phi})$, we have products of the type (6.11) with components which are dependent. Therefore, the evaluation will not give the exact range, but an enclosure for $\mathbf{A}_i^j(\boldsymbol{\phi})$. However, this enclosure is much tighter than the one obtained using its natural evaluation.

The interval evaluation of $\mathbf{A}_i^j(\boldsymbol{\phi})$ involves $j - i$ intermediate matrices $\mathbf{A}_i^k(\boldsymbol{\phi})$. Each of these matrices is a rotation matrix, with all its elements in the interval $[-1, 1]$. However, due to overestimation, we will obtain intervals including values outside $[-1, 1]$. Thus, we *normalize* each of the intermediate matrices $\mathbf{A}_i^k(\boldsymbol{\phi})$ by intersecting them with the unit matrix

$$\begin{pmatrix} [-1,1] & [-1,1] & [-1,1] \\ [-1,1] & [-1,1] & [-1,1] \\ [-1,1] & [-1,1] & [-1,1] \end{pmatrix} .$$

Using this accurate interval evaluation of matrices $\mathbf{A}_i^j(\boldsymbol{\phi})$ in the overall algorithm described in next chapter, we reduce the number of processed boxes between 20% and 50%. The reduction is higher in large boxes, where overestimation is more important.

6.4 Accurate Interval Evaluation of Vectors $\mathbf{w}_0^i(\boldsymbol{\phi}, \mathbf{d})$ and $\mathbf{w}_1^i(\boldsymbol{\phi}, \mathbf{d})$

The definitions of $\mathbf{w}_0^i(\boldsymbol{\phi}, \mathbf{d})$ and $\mathbf{w}_1^i(\boldsymbol{\phi}, \mathbf{d})$ given in Section 6.1.2 are not suitable for a natural interval evaluation. In the expressions given there, (6.3) and (6.4), many of the matrices $\mathbf{A}_i^j(\boldsymbol{\phi})$ use the same rotation matrices and the corresponding rotation variables appear several times. For instance, for $n = 5$,

$$\mathbf{w}_1^2(\boldsymbol{\phi}, \mathbf{d}) = \mathbf{A}_3^2(\boldsymbol{\phi}) \begin{pmatrix} d_3 \\ 0 \\ 0 \end{pmatrix} + \mathbf{A}_3^3(\boldsymbol{\phi}) \begin{pmatrix} d_4 \\ 0 \\ 0 \end{pmatrix} + \mathbf{A}_3^4(\boldsymbol{\phi}) \begin{pmatrix} d_5 \\ 0 \\ 0 \end{pmatrix} .$$

Here, ϕ_3 appears twice; in $\mathbf{A}_3^3(\boldsymbol{\phi})$ and in $\mathbf{A}_3^4(\boldsymbol{\phi})$. A natural interval evaluation of this expression will lead to an overestimation of $\mathbf{w}_1^2(\boldsymbol{\phi}, \mathbf{d})$.

We rewrite here the definitions of $\mathbf{w}_0^i(\boldsymbol{\phi}, \mathbf{d})$ and $\mathbf{w}_1^i(\boldsymbol{\phi}, \mathbf{d})$ in order to minimize the appearances of each variable:

$$\begin{aligned} \mathbf{w}_0^i(\boldsymbol{\phi}, \mathbf{d}) = & \begin{pmatrix} d_i \\ 0 \\ 0 \end{pmatrix} + \left(\mathbf{A}_{i-1}^{i-1}(\boldsymbol{\phi})\right)^t \left[\begin{pmatrix} d_{i-1} \\ 0 \\ 0 \end{pmatrix} + \left(\mathbf{A}_{i-2}^{i-2}(\boldsymbol{\phi})\right)^t \left[\begin{pmatrix} d_{i-2} \\ 0 \\ 0 \end{pmatrix} + \right.\right. \\ & \left.\left. + \left(\mathbf{A}_{i-3}^{i-3}(\boldsymbol{\phi})\right)^t \left[\begin{pmatrix} d_{i-3} \\ 0 \\ 0 \end{pmatrix} + \quad \dots \quad + \left(\mathbf{A}_2^2(\boldsymbol{\phi})\right)^t \left[\begin{pmatrix} d_2 \\ 0 \\ 0 \end{pmatrix} + \left(\mathbf{A}_1^1(\boldsymbol{\phi})\right)^t \begin{pmatrix} d_1 \\ 0 \\ 0 \end{pmatrix} \right] \dots \right] \right] \right] . \end{aligned}$$

$$\begin{aligned} \mathbf{w}_1^i(\boldsymbol{\phi}, \mathbf{d}) = & \begin{pmatrix} d_{i+1} \\ 0 \\ 0 \end{pmatrix} + \mathbf{A}_{i+1}^{i+1}(\boldsymbol{\phi}) \left[\begin{pmatrix} d_{i+2} \\ 0 \\ 0 \end{pmatrix} + \mathbf{A}_{i+2}^{i+2}(\boldsymbol{\phi}) \left[\begin{pmatrix} d_{i+3} \\ 0 \\ 0 \end{pmatrix} + \quad \dots \right.\right. \\ & \left.\left. \dots \quad + \mathbf{A}_{n-2}^{n-2}(\boldsymbol{\phi}) \left[\begin{pmatrix} d_{n-1} \\ 0 \\ 0 \end{pmatrix} + \mathbf{A}_{n-1}^{n-1}(\boldsymbol{\phi}) \begin{pmatrix} d_n \\ 0 \\ 0 \end{pmatrix} \right] \dots \right] \right] . \end{aligned}$$

Each of the groups

$$\mathbf{A}_j^j(\boldsymbol{\phi}) \begin{pmatrix} x \\ y \\ z \end{pmatrix} \quad \text{and} \quad \left(\mathbf{A}_j^j(\boldsymbol{\phi})\right)^t \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

can be evaluated with the algorithm described in the previous section in order to minimize the overestimation effect.

Chapter 7

Implementation and Experiments

Solving positional inverse kinematics problems using interval methods can be done in a wide variety of ways. First, we can choose as closure equation any of those derived in Chapters 2 and 3, or any other one that can be found in the literature. Second, the obtained system of equations can then be solved by any of the variants of general global interval methods summarized in Chapter 5, together with specific interval methods developed for the closure equations as those described in Chapter 6. Moreover, interval methods usually rely on a large number of heuristic decisions, which can be determinant in the efficiency of the algorithm.

A complete and exhaustive experimental study of interval methods for the solution of inverse kinematics problems is out of the scope of this thesis. As already mentioned, the objectives of this work are more on the theoretical side. However, this work would be incomplete without a chapter dedicated to an implementation of the theoretical results in both the kinematic analysis of spatial mechanisms and the specific interval methods for the derived closure equations.

This chapter should be seen as a proposal for a simple algorithm for solving inverse kinematics problems of single-loop, non-redundant spatial kinematic chains. It has been developed in order to be as effective and as fast as possible. However, still much experimental research can be done in the applied interval methods themselves, in the set of closure equations used and in the involved heuristics, which would probably lead to important improvements in the overall efficiency of the algorithm. In particular, many ideas from the general algorithm in [24] have been avoided here for the sake of simplicity. Their inclusion in our algorithm may improve its efficiency.

This chapter is divided in three sections: the first one describes the derived algorithm, the second section is devoted to its implementation, and the last one presents some experimental results.

7.1 The General Algorithm

The derived overall algorithm is summarized here. It combines an interval Newton method over the parametric rotation and translation equations (Table 3.1) with our direct cuts and a branch-and-bound strategy, in order to isolate all solutions into boxes as small as desired. Details concerning the interval Newton method and direct cuts appear in subsequent sections. The algorithm solves inverse kinematics problems for single-loop, non-redundant mechanisms, although it can be straightforwardly extended to more general cases.

```
INPUT:    An initial box $\underline{\mathbf{x}}_0$, the maximum allowable number of subboxes $M$ to be
          processed, and a domain tolerance $\epsilon$.

OUTPUT:   One of the following:
          1. If the search was successful, a list $\mathcal{R}$ of boxes with relative diameter
             smaller than $\epsilon$, such that all solutions are necessarily contained in
             these boxes.
          2. If the search did not complete with $M$ boxes processed, a list $\mathcal{R}$ as
             above and a list $\mathcal{L}$ of boxes that have not been fully analyzed.

$k = 1$
FOR $sol$=1 to 2
    1. Place the initial box $\underline{\mathbf{x}}_0$ onto an empty list $\mathcal{L}$.
    2. DO WHILE $\mathcal{L} \neq \emptyset$ AND $k < M$.
          a. $k = k + 1$.
          b. Remove a box from $\mathcal{L}$ and place it in the current box $\underline{\mathbf{x}}_c$.
          c. Reduce $\underline{\mathbf{x}}_c$ using direct cuts (Algorithm 7.2).
          d. Reduce $\underline{\mathbf{x}}_c$ using an interval Newton method (Algorithm 7.3) over the
               parametric equations for solution $sol$.
          e. IF c. or d. proved that $\underline{\mathbf{x}}_c$ could not contain any roots THEN
               CYCLE loop 2.
          f. IF $\|\mathrm{w}(\underline{\mathbf{x}}_c)\|_{rel} < \epsilon$ THEN
               Insert $\underline{\mathbf{x}}_c$ into $\mathcal{R}$.
             ELSE
               Bisect $\underline{\mathbf{x}}_c$ (Algorithm 7.4) and insert both boxes into $\mathcal{L}$.
    END DO
END FOR
```

Algorithm 7.1: Overall branch-and-bound algorithm for the solution of inverse kinematics problems.

The input of the algorithm is an initial box $\underline{\mathbf{x}}_0$, the desired accuracy ϵ and the maximum number of boxes to process M. The box $\underline{\mathbf{x}}_0$ is described by means of an interval vector of rotations and an interval vector of translations. Most of the elements of these vectors are degenerate intervals of zero width corresponding to the geometry of the mechanism. The other elements are those representing the configuration of the mechanism (rotations and translations of joints) and are intervals whose limits are the allowed range of joint motions.

The algorithm is based on a simple branch-and-bound strategy. Bounding is done in two steps: first the box is reduced using direct cuts (Step 2.c) and then it is reduced with an interval Newton method (Step 2.d).

Actually, the inverse kinematic problem is solved twice: once for the first solution of the parametric closure equations and once for the second solution (Equation (3.7)). In Section 7.1.6 we suggest a possible improvement to avoid having to repeat the whole branch-and-bound process.

The reason for applying first direct cuts and then an interval Newton method comes from the efficiency of these methods for bounding boxes depending on their relative size. Newton methods are based on a Taylor interval extension, which gives usually tighter bounds in small boxes (Section 5.3.2). On the other hand, direct cuts are based on a natural evaluation, working better in larger boxes. Moreover, our direct cuts are much faster than an interval Newton method, making them useful for quickly eliminating boxes far from solutions, without requiring the computation of an interval Newton method.

This strategy is based on the one described in [67] for interval cuts (Section 5.9). Our direct cuts and interval Newton methods are analogous to Newton cuts and combined cuts, respectively. The former are more efficient at the beginning of the bounding phase, while the latter work better for small boxes.

We have also tested to apply again direct cuts after the interval Newton method. However, direct cuts seldom reduce a box after an interval Newton method did it. And if they are able to, they do not reduce it significantly.

Existence and uniqueness are not verified in this algorithm (in Section 7.1.4 we give some considerations on these properties). Therefore, although the resulting boxes will contain necessarily all solutions, some of them could include no solutions at all or even they could include two or more solutions. However, taking ϵ small enough, we never got *false* boxes containing no solutions in our experiments (refer to Section 7.3).

If the search exceeds the number of allowed boxes M, we simply stop. The output will consist of boxes in $\mathcal{R}$, whose width is smaller than ϵ, which will probably contain solutions, and larger boxes in $\mathcal{L}$ that have not been fully analyzed. Note that, if the algorithm finished with $sol = 1$, then only the solutions corresponding to the first solution of the parameterized closure equations are included in the boxes in $\mathcal{R}$ and $\mathcal{L}$. There may be some solutions, which are neither in boxes in $\mathcal{R}$ nor in boxes in $\mathcal{L}$, which would be obtained for $sol = 2$.

7.1.1 Direct Cuts

As explained above, direct cuts are first applied in the bounding phase because of its simplicity and its ability to cut *large* boxes.

The process of reducing a box using direct cuts (Step 2.c) is described in the following algorithm:

```
INPUT:     A box $\underline{\mathbf{x}}$.

OUTPUT:    One of the following:
           1. The same box $\underline{\mathbf{x}}$ if it could not be reduced.
           2. A reduced box $\underline{\mathbf{x}}$ if it could be reduced.
           3. The status "has no root" associated with box $\underline{\mathbf{x}}$.

FOR $var = 1$ TO $nvar$ (number of variables)
  1. (cut variable $var$ using the rotation equation)
     IF $var$ is a rotational variable
        Cut $\underline{\mathbf{x}}$ in the direction of variable $var$ using the rotation equation
           (Section 6.1.1, Equations (6.1) and (6.2)).
  2. (cut variable $var$ using the translation equation)
     FOR $first = 1$ TO $n$ (number of bars)
        IF $var$ is a rotational variable
           Cut $\underline{\mathbf{x}}$ in the direction of variable $var$ using the translation equation
              beginning with bar $first$ (Section 6.1.2, Equations (6.3) to (6.6)).
        ELSE ($var$ is a translational variable)
           Cut $\underline{\mathbf{x}}$ in the direction of variable $var$ using the translation equation
              beginning with bar $first$ (Section 6.1.3, Equations (6.8) and (6.9)).
     END FOR
END FOR
```

Algorithm 7.2: Reducing a box with direct cuts.

For the computation of direct cuts we will use the accurate interval evaluations of matrices $\mathbf{A}_i^j(\boldsymbol{\phi})$ and vectors $\mathbf{w}_0^i(\boldsymbol{\phi}, \mathbf{d})$ and $\mathbf{w}_1^i(\boldsymbol{\phi}, \mathbf{d})$ described in Sections 6.3 and 6.4, respectively. Refer to Section 6.1 for a detailed description on direct cuts.

Observe that this algorithm does not iterate direct cuts until no more reduction is achieved. Each variable is cut only once with the rotation equation (if the variable is a rotation) and once with each of the n possible cuts with the translation equation (n is the number of bars of the n-bar mechanism).

Boxes can be reduced much more iterating this algorithm. Sometimes, direct cuts are able by themselves to isolate boxes as small as desired with the solutions. However, experiments show that the reduction after the first or the second iteration decreases significantly and the convergence becomes extremely slow. In general, it is more efficient to apply an interval Newton method after the first reduction by direct cuts.

Table 7.1 shows the performance (number of bisections and CPU time) of the example in Section 7.3.1 when iterating direct cuts more than once. Observe that the number of bisections decreases and the CPU time increase with the number of iterations. In some cases, the time-optimal number of iterations is two, but in general one iteration is enough.

number of direct cuts	1	2	3	4	5	6	7	8	9	10
number of bisections	197	137	133	129	125	123	123	123	122	120
CPU time	75"	76"	81"	82"	82"	83"	84"	86"	88"	96"

Table 7.1: Number of bisections and computation time versus direct cuts iterations.

Actually, direct cuts are useful to reduce *asymmetric* boxes, i.e. they are usually able to significantly cut boxes where a variable has a much larger width than all the others. In these cases, direct cuts tend to reduce the box in the direction of the variable with larger width, and the box becomes *less asymmetric.*

7.1.2 Interval Newton Method

An interval Newton method is applied in Step 2.d. of Algorithm 7.1 after reducing the box using direct cuts.

An interval Newton method could work over any set of closure equations defining the mechanism. For instance, we could take as rotation equation the original one obtained from the factorization (Section 2.4, Equation (2.3)),

$$\mathbf{F}(\boldsymbol{\phi}) \triangleq \prod_{i=1}^{n} \mathbf{R}(\phi_i)\mathbf{Z} = \mathbf{I}\ , \tag{7.1}$$

or the equations derived from the parameterization of the SM^{n-3} (Equation (3.7)):

$$\begin{aligned} \phi_{n-2} &= \operatorname{atan2}(\pm a_{21}, \mp a_{31}) \\ \phi_{n-1} &= \mp\operatorname{acos}(-a_{11}) \\ \phi_n &= \operatorname{atan2}(\mp a_{12}, \mp a_{13})\ . \end{aligned}$$

Both equations are equivalent except for the singularities of the parameterization, where the second one is not valid.

As translation equations, we can use any of the variants described in Chapters 2 and 3; for instance:

- The original translation equation derived from the factorization (Equation (2.4)):

$$\mathbf{T}(\boldsymbol{\phi}, \mathbf{d}) \triangleq \boldsymbol{\mathcal{N}}(\boldsymbol{\phi})\mathbf{d} = \mathbf{0}\ .$$

- The translation equation of Section 3.3.2 (Equation (3.15)):

$$\nabla \mathbf{F}(\boldsymbol{\phi})\mathbf{d} = \mathbf{0}\ .$$

- The parameterized translation equation in terms of bar directions (Equation (3.20)):

$$d_a = -\sum_{\substack{i=1\\ i\neq a,b,c}}^{n} \frac{|\ \mathbf{n}_i\ \mathbf{n}_b\ \mathbf{n}_c\ |}{|\ \mathbf{n}_a\ \mathbf{n}_b\ \mathbf{n}_c\ |}\, d_i\ , \quad d_b = -\sum_{\substack{i=1\\ i\neq a,b,c}}^{n} \frac{|\ \mathbf{n}_a\ \mathbf{n}_i\ \mathbf{n}_c\ |}{|\ \mathbf{n}_a\ \mathbf{n}_b\ \mathbf{n}_c\ |}\, d_i\ , \quad \ldots \tag{7.2}$$

- The first set of parameterized translation equations in terms of rotations (Equation (3.35)):

$$\begin{pmatrix} d_{n-2} \\ d_{n-1} \\ d_n \end{pmatrix} = \mathbf{L} \sum_{i=1}^{n-3} \left(\prod_{k=1}^{n-i-1} \mathbf{M}_{n-k} \right) \begin{pmatrix} 0 \\ d_i \\ 0 \\ 0 \end{pmatrix}\ .$$

- The second set of parameterized translation equations in terms of rotations (Table 3.1):

$$\begin{pmatrix} d_{n-2} \\ d_{n-1} \\ d_n \end{pmatrix} = \mathbf{L}(\phi_{n-1}) \sum_{i=1}^{n-3} \left[\left(\mathbf{A}_i^{n-2}(\boldsymbol{\phi}) \right)^t \begin{pmatrix} d_i \\ 0 \\ 0 \end{pmatrix} \right]\ .$$

Several reasons have pushed us to choose the parametric rotation and translation equations of Table 3.1 in front of the other possible closure equations.

The main reason is that the rotation equation (7.1) is expressed as a matrix equation consisting of nine scalar equations. As explained in Section 3.2, we cannot take three independent equations from these nine ones. However, any two elements of the diagonal of $\mathbf{F}(\boldsymbol{\phi})$ equated to 1 are equivalent to considering the whole rotation matrix equation (Remark 3.1). Problem arise when the derivatives of these equations with respect to any variable are identically zero in points of the SS^{n-3}, which is obvious from Lemma 3.1. Thus, the zeros of the equations of the diagonal of $\mathbf{F}(\boldsymbol{\phi})$ correspond to maximums or minimums of the function. It is not suitable to use an interval Newton method over equations whose derivatives are zero in the solutions for two reasons:

- the Jacobian or slope matrix tends to zero in small boxes including a solution, which entails problems both in the computation of the preconditioner and in the convergence of the Gauss-Seidel method; and
- existence and uniqueness usually cannot be verified.

Moreover, we have explicit expressions for the derivatives of the parameterized closure equations (Table 3.1), which can be directly used to compute the Jacobian or slope matrix, without requiring automatic differentiation.

The fact that these equations are not valid for a singularity of the parameterization does not cause any problem, because we can always choose a trivial parameterization provided that the n-bar mechanism is not planar.

We do not use the translation equations in terms of bar directions (7.2) since our variables are rotations. Getting bar directions from rotations implies an overestimation of the interval evaluations due to repeated variables.

Finally, we have chosen the second set of parameterized translation equations in terms of rotations in order to have as simple expressions as possible and, again, to avoid overestimation due to multiple appearance of the variables.

The process of reducing a box with an interval Newton method (Step 2.d of Algorithm 7.1) is described in the following algorithm:

```
INPUT:    A box $\underline{\mathbf{x}}$ and the variable $sol$, taking the value 1 or 2, depending
          on whether we use the first or the second solution of the rotation
          equations (3.7).

OUTPUT:   One of the following:
          1. The same box $\underline{\mathbf{x}}$ if it could not be reduced.
          2. A reduced box $\underline{\mathbf{x}}$ if it could be reduced.
          3. The status "has no root" associated with box $\underline{\mathbf{x}}$.

$m = 0$
DO WHILE (*enough reduction*) AND $m < 10$
   1. $m = m + 1$
   2. Compute the Hansen slope matrix in box $\underline{\mathbf{x}}$ according to Section 5.5.2 .
         using the expressions of the derivatives of Table 3.1
   3. Evaluate the constraint equations in the center of the box.
   4. FOR $var = 1$ TO $nvar$ (number of variables)
         a. Compute the preconditioner row $Y_{var}$.
         b. Reduce the range of variable $var$ with a Gauss-Seidel iteration
               (Equation (5.14)).
      END FOR
END DO
```

Algorithm 7.3: Reducing a box with an interval Newton method.

The main loop is iterated until no *relevant reduction* is achieved. Experimental results have shown that the efficiency of the overall algorithm increases if boxes are split when the convergence of interval Newton method becomes too slow. Two criteria have been taken to measure the reduction of a box: the *maximum reduction* and the *average reduction.* The maximum reduction considers only the reduction of the variable that has been cut the most; the average reduction is the mean value of the reductions of all variables, considering as variable those which are not of zero width. It seems to be a good choice to impose a maximal reduction greater than 1% in order to iterate again the main loop.

However, if the number of iterations increases too much, it is worth splitting the box, although still a maximal reduction greater than 0.01 can be achieved. We have limited experimentally the number of iterations to 10.

Despite the apparent simplicity of the algorithm, some of the steps need a deeper insight. Let us analyze them step by step:

Step 2. The Hansen Jacobian matrix is computed as described in Section 5.5.2. We use the expressions for the derivatives given in Table 3.1. In order to minimize the overestimation effect, we evaluate matrices $\mathbf{A}_i^j(\boldsymbol{\phi})$ using Algorithm 6.3 in Section 6.3.

In our experiments, we always use the sequence of the variables as in Equation 5.12. However, particular orders may give better results than others. In [24, p. 34], Kearfott proposes two heuristics for choosing the sequence of the variables in order to have a Hansen slope matrix leading to sharper inclusions. The idea is to expand last with respect to the variables leading to wide intervals, that is, the variables for which the scaled partial derivative widths differ the most from the corresponding point approximation widths.

The Hansen slope matrix is computed for each Gauss-Seidel iteration. However, we could use the Hansen slope computed for the previous iteration, which is an enclosure of the Hansen slope that would be obtained for the present box. We loss in efficiency, but we do not have to recalculate the Hansen slope at each iteration. This strategy has been said to be effective in some instances, but in our experiments, it rather slowed down the whole process.

Step 3. The evaluation of the constraints are based on the rotation and translation equations of Table 3.1. Note that the evaluation of the constraint equations in the center of the box could actually be computed using floating point arithmetic. However, if we want to bound roundoff errors, we should use interval arithmetic. Here, we do not need an accurate evaluation of matrices $\mathbf{A}_i^j(\boldsymbol{\phi})$, since the variables are all of zero (or nearly zero) width.

Step 4.a. The preconditioner row Y_{var} is computed as a width-optimal C-preconditioner, using the structure given in Section 5.7.3. The resulting linear programming problem can be solved using the simplex method. We have adapted the algorithm described in [43] for our case. This algorithm does not take into account the cases when the simplex method does not converge (this happens less than once out of 1000 executions). In those cases, we use the inverse midpoint preconditioner described in Section 5.7.1. If the center of the Jacobian matrix were to be a singular matrix, we would not use any preconditioner; but it never happened in practice in our experiments.

Step 4.b. Since we are using a C-preconditioner, we should never get a denominator including the origin and, thus, each Gauss-Seidel iteration will give a single interval. However, this would be true for a *real* C-preconditioner. Since we are introducing some heuristics in order to compute the preconditioner as a linear programming problem, we could get a denominator including the zero, at least theoretically, but we never got it in practice. Moreover, if the simplex method does not converge, we would use the inverse midpoint preconditioner, which could also lead to a denominator including the origin. Therefore, we have to consider this case and use extended interval arithmetic as explained in Section 5.4.3. In those cases, we could split the box, probably increasing the overall efficiency of the algorithm. However, since these cases are extremely rare, we take the convex hull of the resulting interval.

7.1.3 The Bisection Algorithm

The overall algorithm 7.1 comes to bisection when direct cuts and the interval Newton method have not been able to reduce the box below the specified accuracy. Bisection divides the box into two subboxes as described in Section 5.8.2. We use here the following simple algorithm:

```
INPUT:    A box $\underline{\mathbf{x}}$.

OUTPUT:   Two boxes $\underline{\mathbf{x}}^1$ and $\underline{\mathbf{x}}^2$, whose union is the input box $\underline{\mathbf{x}}$.

1. Choose the bisection coordinate $x_k$.

2. $\underline{\mathbf{x}}^1 = \underline{\mathbf{x}}^2 = \underline{\mathbf{x}}$ .

3. $\underline{x}_k^1 = [\overleftarrow{\underline{x}}_k, \frac{\overleftarrow{\underline{x}}_k + \overrightarrow{\underline{x}}_k}{2}]$ ,  $\underline{x}_k^2 = [\frac{\overleftarrow{\underline{x}}_k + \overrightarrow{\underline{x}}_k}{2}, \overrightarrow{\underline{x}}_k]$.
```

Algorithm 7.4: The bisection algorithm.

The box is divided across the bisection coordinate in two subboxes of the same size. We choose as bisection coordinate that of maximum width. However, this is not always the most effective at reducing overestimation or producing boxes in which the Gauss-Seidel method will converge. In Section 5.8.2 we describe an heuristic for choosing the bisection coordinate: the *maximum smear heuristic* [27].

7.1.4 Existence and Uniqueness

We have already said that existence and uniqueness are not verified in algorithm (7.1). Existence could easily be verified without extra computing according to the slope-based existence theorem (Theorem 5.2). However, in our algorithm, we would only need to check for existence in the final boxes. If we perform a Gauss-Seidel sweep in a final box and the resulting box is completely included in it, we know that there is at least a solution in that box (there may be more). Otherwise, there may or may not be a solution.

It is a nonsense checking for existence in the middle of the process. If we knew that there is a solution in an intermediate box, we would have to keep on reducing it in order to isolate it. And if we could not proof existence, we would have to keep on reducing too. The existence test in an intermediate box is useful if we make use of approximate roots and ϵ-inflation (Section 5.8.4), isolating the single roots using another –not global– method for solving systems of nonlinear equations (i.g. a gradient based method).

Uniqueness verification with slope matrices (as is our case) is only possible in certain contexts, such as with the ϵ-inflation technique, and cannot be used in our algorithm. Nevertheless, we can always perform a last interval Gauss-Seidel iteration with a Jacobian matrix over the final boxes an try to verify existence and uniqueness according to Theorem 5.1. However, in most practical applications, it does not matter if in the resulting box (of width smaller than the required accuracy) there is a single solution or if it is double.

Therefore, although the resulting boxes will contain necessarily all solutions, some of them could include no solutions or could include two or more solutions. However, with a small enough tolerance, we never got *false* boxes containing no solutions in our experiments (refer to Section 7.3).

7.1.5 Tolerance and Precision

An interesting property of the proposed algorithm is that high precision does usually not increase significantly the computation time. For instance, let us consider the second example in Section 7.3.1. The difference in CPU time if we require boxes of width smaller than 10^{-3} or smaller than 10^{-12} is less than 0.5 seconds, while the whole process takes 1'15".

This fact is due to the convergence of the interval Newton method. Once a box containing a solution has been reduced to a certain size, the interval Newton method converges without more bisections. This size depends on the problem, but is usually much larger than the specified tolerance. Then, when asking for higher precision, the number of processed boxes does not increase and the computation time is nearly the same.

7.1.6 Avoiding the Second Solution

One of the main drawbacks of our algorithm is that it has to be repeated twice, once for each of the two solutions of the parametric rotation equation. However, the parametric equations are only used in the interval Newton method. Thus, when reducing or eliminating an intermediate box with an interval Newton method, we may have eliminated roots, which would be obtained with the other solution of the parametric equations. That is the reason for having to solve the problem twice.

On the other hand, when reducing or eliminating a box with direct cuts, we never miss any solution, since we are using the original closure equations. It could be interesting to take advantage of the eliminated parts by direct cuts so that we could not have them for the second solution of the parametric equations.

A possible algorithm could try to reduce each box first with direct cuts. The resulting box would then be reduced with an interval Newton method using the first solution of the parametric equations. If it eliminates the box, we could label it as a candidate to only have roots corresponding to the second solution of the parametric equations. From then on, we would only need to reduce that box with the second solution and we could do similarly for the other solution. If the box were not eliminated by any of the two solutions, we could duplicate the box, so that each box is reduced only by one of the solutions, or we may take the convex hull of the two reduced boxes and keep on reducing the box with both solutions. This choice could be taken depending on the relative reductions achieved by both solutions.

At first glance, this could improve our algorithm, but this idea should be more extensively developed and tested to fully prove it. Actually, we would avoid reducing the same box (or part of it) with direct cuts twice. However, direct cuts are computationally simple and not much time consuming, and it is not clear whether the strategy proposed in this section would lead to any important improvement.

7.1.7 Singularities and Redundant Mechanisms

It is worth noting that Algorithm 7.1 also works for singular points. For instance, in a singular position of a PUMA 560 manipulator, there are infinite solutions for its inverse kinematics problem: the solution is actually a 1-dimensional self-motion manifold. Our algorithm would give as resulting boxes in $\mathcal{R}$ a discretization of this self-motion manifold. In practice, if we require a high or even a medium precision, we would require a large number of boxes and the algorithm becomes extremely slow.

This is also the case for redundant mechanisms. The solution of the inverse kinematics problem is also a manifold, whose dimension depends on the redundancy of the mechanism. Our algorithm could, theoretically, give a discretization of the self-motion manifold. But again, the number of boxes may be huge, mainly if the degree of redundancy is high.

7.2 Implementation using BIAS/PROFIL Interval Libraries

Algorithm 7.1 has been implemented in C++ using BIAS/PROFIL interval libraries. BIAS/PROFIL are portable C++ class libraries developed by Olaf Knüppel from the Technische Universität Hamburg-Harburg (Germany). BIAS/PROFIL introduce new data types as vectors, matrices, intervals, interval vectors and matrices, integer vectors and matrices, and lots of operations between them. Additionally, several commonly used routines as a basic linear interval system solver, etc., are also included, as well as some basic list utilities.

The philosophy of these libraries is similar to that of INTLIB, a `FORTRAN-77` package developed by Kearfott [23]. The idea is to offer totally portable routines upon which more efficient machine-specific versions can be based.

In Knüppel's libraries, the set of basic routines is named BIAS, while the set containing the higher-level functions is called PROFIL. All interval operations of PROFIL are based on BIAS with the advantage that PROFIL is independent of the internal representation and the implementation of the interval types.

The package is portable, except for a single small assembler language routine that is called to change rounding modes; versions of this routine are available for a large variety of machine, including Sun, PC, HP workstations, etc.

The package is available via anonymous ftp at

`ti3sun.ti3.tu-harburg.de/pub/profil/` .

An extensive documentation can be found in

`http://www.ti3.tu-harburg.de/Software/PROFIL/Profil.texinfo_toc.html`.

Reports 93.3 [28], 93.4, 93.5 and 95.4 of the Technische Universität Hamburg-Harburg also deal on the subject and are available via ftp at

`ti3sun.ti3.tu-harburg.de/pub/reports/` .

The general structure and main features of the BIAS/PROFIL package are described in Appendix C, together with some tips on its installation, which cannot be found elsewhere.

The implementation has been done in order to validate the theoretical results of this work. However, the code has not been optimized, but rather organized (or maybe disorganized) in order to allow to perform different experiments in a straightforward way. Thus, there is plenty of room for improvements.

We have been running the program in a SUN Ultra 2 2300 Creator with a 296MHz processor. The CPU times given for the experiments in next section correspond to this machine.

7.3 Experimental Results

We have tested the implementation of Algorithm 7.1 in kinematic chains with 6 degrees-of-freedom (rotational and translational d.o.f.). Some examples of the inverse kinematics of five different mechanisms are shown next. They include:

- an industrial manipulator (a PUMA 560) for three different poses of the end-effector;
- a particular 6R mechanism, also in three different configurations;
- a Stanford manipulator in the same configuration as in the example in [48];
- an RPRRPR manipulator in the same configuration as in [48] (originally from [46]); and
- the inverse kinematics problem for a 7R closed mechanism (the same example of Duffy [14] and Lee [30]).

For each of these examples, we give the number of branchings, the number of boxes eliminated by direct cuts and by an interval Newton iteration, the average sweeps of the interval Newton methods and the required CPU time. For all examples we require the resulting boxes be of width smaller than 1×10^{-6}. In all cases, the resulting boxes include all solutions (8, 6, 4 or 2 depending on the case).

7.3.1 A PUMA 560 Manipulator

We use 14 bars to describe a PUMA 560 and 3 more bars to close the chain. The vectors of rotations and translations we have used for representing a PUMA 560 are:

$$\begin{aligned}\boldsymbol{\phi} &= (90, \theta_1, -90, \theta_2, 180, \theta_3, 90, \theta_4, 90, \theta_5, 90, \theta_6, 90, 0, \phi_{15}, \phi_{16}, \phi_{17})\\ \mathbf{d} &= (0, 0, 0, 0, 432, 149.098, -20.5, 433, 0, 0, 0, 56.5, 0, 0, d_{15}, d_{16}, d_{17})\end{aligned}$$

Note that 12 bars would be enough for a 6R mechanism according to Remark 2.1. However, we have used here 14 bars in order to maintain the same reference frames used by the Denavit-Hartenberg parameters.

The last three bars close the kinematic chain and represent the end effector's pose with respect to the base. The values for these three bars can be obtained from the end effector's position as explained in Appendix A.

The results for three different end-effector's poses are shown in Table 7.2.

configuration	branching	direct cuts	Newton	avg. New. sweeps	CPU time
$\boldsymbol{\phi} = (\ldots, 195, 90, 335)$ $\mathbf{d} = (\ldots, -780, -15, 430)$	236	159	71	1.88	1'24"
$\boldsymbol{\phi} = (\ldots, 200, 56, 305)$ $\mathbf{d} = (\ldots, -370, 600, -550)$	197	154	37	2.40	1'15"
$\boldsymbol{\phi} = (\ldots, 178, 52, 279)$ $\mathbf{d} = (\ldots, 340, -200, 615)$	328	282	40	1.78	1'42"

Table 7.2: A PUMA 560 manipulator.

7.3.2 A Particular 6R Mechanism

Most industrial manipulators are of special geometries. In order to validate the algorithm for any general mechanism, we have tested it with many randomly generated mechanisms. This example corresponds to one of them, in particular it is a 6R open mechanism represented with 12 bars. Three more bars are used to close it. The randomly chosen vectors of rotations and translations are:

$$\boldsymbol{\phi} = (90, \theta_1, 90, \theta_2, 90, \theta_3, 90, \theta_4, 90, \theta_5, 90, \theta_6, \phi_{13}, \phi_{14}, \phi_{15})$$
$$\mathbf{d} = (5, 2, 7, 4, 8, 0, 2, 0, 12, 15, 6, 3, d_{13}, d_{14}, d_{15})$$

configuration	branching	direct cuts	Newton	avg. New. sweeps	CPU time
$\boldsymbol{\phi} = (\ldots, 200, 15, 85)$ $\mathbf{d} = (\ldots, 9, 0, 7)$	1 636	1 484	152	1.68	6'14"
$\boldsymbol{\phi} = (\ldots, 300, 340, 5)$ $\mathbf{d} = (\ldots, 1, 7, 5)$	1 983	1 804	179	1.79	7'52"
$\boldsymbol{\phi} = (\ldots, 20, 30, 340)$ $\mathbf{d} = (\ldots, 6, 8, 10)$	1 874	1 771	103	1.24	6'19"

Table 7.3: A particular 6R mechanism.

7.3.3 A Stanford Manipulator

In Rao's paper [48], an example is given for a Stanford manipulator. We reproduce here the same example solved with our algorithm in order to compare the results. We use a 12-bar mechanism to represent a Stanford manipulator, with three more bars to close it. These three bars have been calculated to get the same end-effector's pose as in Rao's paper. The vectors of rotations and translations are:

$$\boldsymbol{\phi} = (\theta_1, 90, \theta_2, 270, 180, 180, \theta_4, 90, \theta_5, 270, \theta_6, 180, 269.554, 126.53, 170.919)$$
$$\mathbf{d} = (0, 0, 2, 0, d_3, 0, 0, 0, 0, 0, 4, 0, -7.36145, -2.19932, 1.06964)$$

We have considered two cases. The first one searches for solutions in the same box as in Rao's paper and finds the same two solutions. In the second case, we search the whole space (the rotations ranging from 0 to 360 degrees) and get the four possible solutions.

box	branching	direct cuts	Newton	avg. New. sweeps	CPU time
Rao's box	16	13	3	2.19	6"
whole space	58	50	6	1.79	19"

Table 7.4: A Stanford manipulator.

In Rao's paper [48], 19,440 bisections are needed to isolate the two solutions using Krawczyk's method [37]. With our algorithm, we reduce the number of bisections to 16. This dramatic reduction is due to several factors:

- we use the interval Gauss-Seidel method, which usually converges much faster than the Krawczyk's method,
- we use optimal preconditioners, and
- we first reduce the box using direct cuts.

7.3.4 An RPRRPR Manipulator

We consider here an RPRRPR manipulator, which appeared originally in [46]. This example is also solved in [48] using Krawczyk's method. We reproduce here the same example solved with our algorithm, using a 12-bar mechanism with three more bars to close it. These three bars have been calculated to get the same end-effector's pose as in Rao's paper. The vectors of rotations and translations are:

$$\boldsymbol{\phi} = (\theta_1, 314.989, 244.973, 257.922, \theta_3, 202.976, \theta_4, 226.009, 233.973, 215.008, \theta_6, 226.983, 324.475, 130.632, 216.495)$$

$$\mathbf{d} = (0.21, 1.46, d_2, 0.56, 0.29, 0.38, -0.54, 0.56, d_5, 1.08, 0.48, 0.67, -2.11931, 1.37138, 2.81638)$$

We have considered two cases. The first one searches for solutions in the same boxes as in Rao's paper and finds the same two solutions. In the second case, we search the whole space (the rotations ranging from 0 to 360 degrees).

box	branching	direct cuts	Newton	avg. New. sweeps	CPU time
Rao's first box	106	54	53	3.56	51"
Rao's second box	40	3	38	3.87	24"
whole space	5 809	4 879	930	1.70	28'42"

Table 7.5: An RPRRPR manipulator.

In Rao's paper [48], 3,350 and 9,584 bisections are needed to isolate the two solutions using Krawczyk's method [37]. With our algorithm, we reduce the number of bisections to 106 and 40 from the same boxes.

7.3.5 Duffy's 7R Example

In [14], Duffy solved the inverse kinematics for a 7R closed mechanism as example of its results. 8 years later, Lee used the same example to corroborate his results [30]. We have chosen the same example here, which can be represented by a 14-bar mechanism with the following vectors of rotations and translations:

$$\begin{aligned}\boldsymbol{\phi} &= (\theta_1, 260, \theta_2, 273, \theta_3, 300, \theta_4, 300, \theta_5, 273, \theta_6, 260, \theta_7, 215)\\ \mathbf{d} &= (1.0, 0.9, 1.2, 1.1, 0.8, 1.5, 2.0, 1.5, 0.8, 1.1, 1.2, 0.9, 1.0, 0.5)\end{aligned}$$

We give here the results for a single value of θ_7:

configuration	branching	direct cuts	Newton	avg. New. sweeps	CPU time
$\theta_7 = 30$	11 264	9 977	1 284	1.53	38'25"

Table 7.6: Duffy's 7R example.

Although there are no apparent differences with the other examples, in these last two cases the program takes a long time to obtain the solutions. The reason for this behavior has not been explained yet.

7.3.6 Discussion

With a tolerance of $\epsilon = 1 \times 10^{-6}$ in both the rotation (in radians) and the translation variables, we never got boxes containing no solutions. This does not mean that it can not happen. It is theoretically possible indeed to have a box with relative width smaller than ϵ where existence could not be verified. Nevertheless, the fact that we have never come across such a case indicates, at least, that it is extremely unusual.

We also never came across a solution box containing more than one solution. However, this could be easily managed if we choose a configuration with two solutions separated less than ϵ. Nevertheless, in practical applications, probably it does not matter if the box contains one or more solutions because of mechanical tolerances.

Note that the number of bisections is much lower than the one obtained using an exhaustive search of the box. While the number of branchings in our algorithm ranges from some few bisections up to 12.000, an exhaustive search would require about 10^{30} bisections for the specified tolerance [48]; the reduction in the number of analyzed boxes is dramatic.

Chapter 8

Conclusions

The first part of this work expands on the structure of the self-motion set of the orthogonal spherical mechanism showing how a thorough understanding of it is fundamental in the study of spatial mechanisms. To this end, the following two facts have been exploited: (a) any kinematic loop equation can be modeled as the loop equation derived from the so-called n-bar mechanism by taking as many bars as needed and constraining some of the resulting degrees of freedom and (b) the solution of the translation equation resulting from the factorization of the loop equation of the n-bar mechanism is provided by the tangent bundle of the self-motion manifold of its spherical indicatrix. As a consequence, a new unified approach for the analysis of any single kinematic loop containing independent revolute, prismatic and cylindrical pairs has been devised.

In the second part of this thesis, it is proved how interval methods are suitable for solving inverse kinematics problems. They can already be seen as a competitive alternative to elimination and continuation methods. The main advantages of interval methods for solving the problem at hand are listed below.

1. The proposed interval method for solving inverse kinematics problems is *general* in the sense that it can be applied to any single loop kinematic chain, regardless of its geometry, its number of links or its degrees of freedom. The input of the algorithm is always a couple of interval vectors, the vectors of rotation and translation, which define completely the mechanism and its joint limits. No symbolic computations have to be performed and the closure equations are always treated in the same way.

2. The algorithm finds *all solutions* contained in the initial box. Also for singular configurations or for mechanisms with special geometries, all solutions are included in the resulting boxes.

3. Interval methods include in a natural way *joint limits*. Moreover, when searching for solutions in a tight range of the joint variable, the algorithm searches only in that small box and not in the whole joint space.

4. Interval methods used with outward rounding (Section 5.2.4) *avoid numerical problems* in the context of floating point arithmetic, which can be the case when dealing

with high-degree polynomials in elimination methods. We may get large boxes enclosing the solutions, but we would never lose solutions.

5. The algorithm works for redundant mechanisms too, returning a discretized version of the self-motion manifold. It can also be straightforwardly extended to multiple loops by adding additional constraint equations (closure equations) sharing some variables.

6. A final advantage of interval methods is that they are *simple to implement.* The main problem are the basic interval subroutines, but using an existing interval library saves us much of the work.

The main disadvantages of the proposed interval method can be summarized in the following points:

1. *Slowness* is obviously one of the biggest problems in our implementations. However, we should take two important aspects into account. Firstly, the computation times can be improved with an accurate code optimization and with an experimental study of the many heuristics that are involved in the method. Secondly, which is much more important, we cannot compare our method at the same level as continuation or elimination methods. While these methods are specific for some class of mechanisms, our interval method works for any single loop mechanism, regardless of its number and type of links or its geometry. But for the moment, the algorithm is still too slow for most applications.

2. Another potential problem is the appearance of *boxes containing no solutions.* We have already mentioned that we have never encountered such boxes in our experiments with small enough accuracies. However, it could theoretically happen. This problem is more a mathematical problem than an actual mechanical one: with the tolerances of current mechanical systems, it does not matter if a small box of width 10^{-6} does contain a solution or not, as long as the end-effector is close enough to the required pose.

3. Our interval method is not suitable to evaluate the maximum *number of solutions* of a certain class of mechanisms. And it also gives little or no information about how some parameters may influence the solutions.

4. A final problem is related to the *complexity* of the method. Although the actual complexity is much lower in order than the worst case complexity, we do not have any theoretical way to bound it, or, at least, to know its average value. We can relay on a large number of experiments, but we could always be surprised by an unexpected long computation.

This work has shown the potential of interval methods for inverse kinematics problems. Hopefully, this thesis will encourage other people to investigate in this direction, opening a new field of research in kinematics.

8.1 Future Research

The work presented here may be further developed in two main directions corresponding to the two parts in which we have divided the thesis:

1. **Kinematic analysis of spatial mechanisms**. Although the theory of this part has been widely explored, there are still some points where important results could be obtained. In particular, it would be interesting to characterize more deeply the SS^{n-3} and its singularities in terms of Morse theory and bifurcations [15]. A better knowledge of the SS^{n-3} could be useful in the understanding of spatial mechanisms.

   Another interesting idea that could be developed is to get a lower bound on how close two solutions can be in the SS^{n-3}. This bound will depend on the vector of translations. This would allow us to verify uniqueness in boxes whose width is smaller than that lower bound and we could apply a gradient method with much faster convergence.

2. **Interval methods**. This second part is the most challenging one, leaving plenty of directions for future work. They can all be grouped under the idea of realizing an exhaustive experimental study of interval methods for the solution of inverse kinematics problems. Many different mechanisms should be tried, as well as many different configurations for them, to be able to validate the results.

   (a) As stated above, the proposed algorithm can be improved from a wide variety of ways. In Section 7.1 (specially in Subsection 7.1.2), we have indicated many possibilities in order to improve the algorithm. Some of the improvements are related to heuristics and some to the variants of the interval Newton method.

   (b) Our algorithm is performed twice, once for each of the two solutions of the rotation equation. Following the ideas given in Section 7.1.6 a more efficient algorithm could probably be designed.

   (c) The interval Newton method could be extended to use not only first derivatives, but also second derivatives. The results of Section 3.3.4 could be used.

   (d) The interval propagation in spherical mechanisms is limited because one variable is not restricted. The algorithm can be probably generalized to all variables being restricted and could be simplified in order to make it useful as interval cuts.

   (e) Our algorithm could also use an interval Newton method working over another set of closure equations, as those summarized in Section 7.1.2. Different interval Newton methods could also be combined.

   (f) Some of the concepts from [24] could be introduced, as ϵ-inflation, approximate roots, box complementation or even the automatic differentiation. Some of these techniques allow to perform not only existence, but also uniqueness tests. Boxes that have been verified to contain a unique solution, could then be reduced with a much faster gradient method.

(g) Another interesting point is to apply the algorithm to redundant or to multiple loop mechanisms.

(h) It would also be interesting to develop an interval method for a specific class of mechanisms; for instance for a 6R manipulator, and optimize the algorithm for that particular case. The computation times would drastically drop down, so that they could be compared with continuation or elimination methods.

(i) Finally, some actual applications could be solved with an interval method. For example, calibration problems, CAD problems or any other that can be expressed in terms of kinematic restrictions.

Appendix A

Closing an Open Mechanism with Three Orthogonal Bars

Three orthogonal bars are enough to reach any point in 3D space with arbitrary orientation. Then, any open n-bar mechanism can be closed adding three more bars. For instance, an n-bar mechanism describing a manipulator can always be closed with three more bars, which represent the end-effector's position and orientation with respect to the base. Note that there are two possible solutions, depending on whether we take d_{n-1} to be positive or negative.

The rotations and translations of these last three *fictitious* bars can be computed from the position and orientation of the last bar (or end-effector). If $\mathbf{C}$ is the orientation matrix of the last bar with respect to the base and $\mathbf{p}$ is its position (Figure A.1), we can write the rotation and translation equations, (2.3) and (2.4), as:

$$\mathbf{C}\mathbf{R}(\phi_{n-2})\mathbf{Z}\mathbf{R}(\phi_{n-1})\mathbf{Z}\mathbf{R}(\phi_n)\mathbf{Z} = \mathbf{I} \qquad \text{and} \tag{A.1}$$

$$\mathbf{p} + \mathbf{C}\begin{pmatrix} d_{n-2} \\ 0 \\ 0 \end{pmatrix} + \mathbf{C}\mathbf{R}(\phi_{n-2})\mathbf{Z}\begin{pmatrix} d_{n-1} \\ 0 \\ 0 \end{pmatrix} + \mathbf{C}\mathbf{R}(\phi_{n-2})\mathbf{Z}\mathbf{R}(\phi_{n-1})\mathbf{Z}\begin{pmatrix} d_n \\ 0 \\ 0 \end{pmatrix} = \mathbf{0}\,. \tag{A.2}$$

From the rotation equation (A.1) we can isolate $\mathbf{C}$,

$$\mathbf{C} = \begin{pmatrix} c_{11} & c_{12} & c_{13} \\ c_{21} & c_{22} & c_{23} \\ c_{31} & c_{32} & c_{33} \end{pmatrix} = (\mathbf{R}(\phi_{n-2})\mathbf{Z}\mathbf{R}(\phi_{n-1})\mathbf{Z}\mathbf{R}(\phi_n)\mathbf{Z})^t =$$

$$= \begin{pmatrix} \mathrm{s}_{n-1}\mathrm{s}_n & \mathrm{c}_{n-1} & \mathrm{s}_{n-1}\mathrm{c}_n \\ -\mathrm{c}_{n-2}\mathrm{c}_n - \mathrm{s}_{n-2}\mathrm{c}_{n-1}\mathrm{s}_n & \mathrm{s}_{n-2}\mathrm{s}_{n-1} & \mathrm{c}_{n-2}\mathrm{s}_n - \mathrm{s}_{n-2}\mathrm{c}_{n-1}\mathrm{c}_n \\ -\mathrm{s}_{n-2}\mathrm{c}_n + \mathrm{c}_{n-2}\mathrm{c}_{n-1}\mathrm{s}_n & \mathrm{c}_{n-2}\mathrm{s}_{n-1} & \mathrm{s}_{n-2}\mathrm{s}_n - \mathrm{c}_{n-2}\mathrm{c}_{n-1}\mathrm{c}_n \end{pmatrix}^t ,$$

where $\mathrm{s}_i = \sin\phi_i$ and $\mathrm{c}_i = \cos\phi_i$.

Taking the first row and the second column we can obtain two possible solutions for

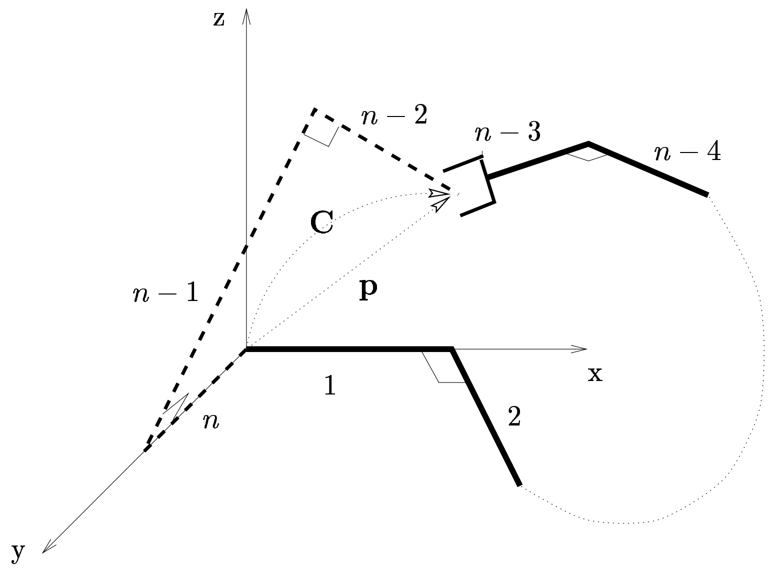


Figure A.1: Closing an open n-bar mechanism with three orthogonal bars.

the rotations when $c_{21} \neq \pm 1$:

$$\begin{aligned} \phi_{n-2} &= \operatorname{atan2}(\pm c_{22}, \mp c_{23}) \\ \phi_{n-1} &= \pm \operatorname{acos}(c_{21}) & \text{(if } c_{21} \neq \pm 1) \\ \phi_n &= \operatorname{atan2}(\pm c_{11}, \pm c_{31}) \end{aligned}$$

If $c_{21} = \pm 1$, ϕ_{n-1} is 0 or π, and ϕ_{n-2} and ϕ_n are dependent. Developing the other elements of matrix $\mathbf{C}$ we get

$$\begin{aligned} \phi_{n-2} - \phi_n &= \operatorname{atan2}(-c_{13}, -c_{12}) & \text{for } c_{21} = 1 \quad \text{and} \\ \phi_{n-2} + \phi_n &= \operatorname{atan2}(-c_{13}, -c_{12}) & \text{for } c_{21} = -1 \ , \end{aligned}$$

and ϕ_{n-1} is obviously 0 if $c_{21} = 1$ and π if $c_{21} = -1$.

From the translation equation (A.2) we can isolate $-\mathbf{C}^t\mathbf{p}$, which we will define as vector $\mathbf{b}$:

$$\mathbf{b} = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix} \triangleq -\mathbf{C}^t\mathbf{p} = \begin{pmatrix} d_{n-2} - \mathrm{c}_{n-1} d_n \\ \mathrm{c}_{n-2} d_{n-1} - \mathrm{s}_{n-2}\mathrm{s}_{n-1} d_n \\ \mathrm{s}_{n-2} d_{n-1} + \mathrm{c}_{n-2}\mathrm{s}_{n-1} d_n \end{pmatrix}$$

Solving this linear system we get the translations:

$$\begin{aligned} d_n &= \frac{b_3 \cos\phi_{n-2} - b_2 \sin\phi_{n-2}}{\sin\phi_{n-1}} \\ d_{n-1} &= b_2 \cos\phi_{n-2} + b_3 \sin\phi_{n-2} & \text{(if } c_{21} \neq \pm 1) \\ d_{n-2} &= b_1 + d_n \cos\phi_{n-1} \end{aligned}$$

When $c_{21} = \pm 1$, d_{n-2} and d_n are parallel. Then,

$$\begin{aligned}
\phi_{n-2} &= \text{atan2}\left(\tfrac{b_3}{\sqrt{b_2^2+b_3^3}}, \tfrac{b_2}{\sqrt{b_2^2+b_3^3}}\right) , \\
\phi_n &= \phi_{n-2} \mp \text{atan2}(-c_{13}, -c_{12}) , \\
\phi_{n-1} &= 0 \ (\text{if } c_{21} = 1) \text{ or } \pi \ (\text{if } c_{21} = -1) , \\
d_{n-1} &= \frac{b_3}{\sqrt{b_2^2 + b_3^3}} \qquad \text{and} \\
d_{n-2} &= b_1 \pm d_n \ .
\end{aligned}$$

We can summarize the computation of three orthogonal bars to close an open mechanism with the algorithm described below. It takes as input the orientation matrix of the last bar, $\mathbf{C}$, and its position vector, $\mathbf{p}$.

```
IF $c_{21} \neq \pm 1$
   $\phi_{n-2}$= atan2($c_{22}, -c_{23}$)
   $\phi_{n-1}$= acos($c_{21}$)
   $\phi_n$   = atan2($c_{11}, c_{31}$)
   $d_n$   = $(b_3 \cos\phi_{n-2} - b_2 \sin\phi_{n-2})/(\sin\phi_{n-1})$
   $d_{n-1}$= $b_2 \cos\phi_{n-2} + b_3 \sin\phi_{n-2}$
   $d_{n-2}$= $b_1 + d_n \cos\phi_{n-1}$
ELSE
   $b_{23}$  = $+\sqrt{b_2^2 + b_3^3}$
   $d_{n-1}$= $b_3/b_{23}$
   $\phi_{n-2}$= atan2($d_{n-1}, b_2/b_{23}$)
   IF $c_{21} = 1$
        $\phi_n = \phi_{n-2}$ − atan2($-c_{13}, -c_{12}$)
        $\phi_n = 0$
        $d_{n-2} = b_1 + d_n$
   ELSE ($c_{21} = -1$)
        $\phi_n = \phi_{n-2}$ + atan2($-c_{13}, -c_{12}$)
        $\phi_n = \pi$
        $d_{n-2} = b_1 - d_n$
   END IF
END IF
```

Algorithm A.1: Computation of the last three bars for closing open mechanisms.

Note that, in this algorithm, $\mathbf{C}$ is assumed to be a proper orthogonal matrix.

Appendix B

Exact-range Functions

Here is a list of the exact range functions we have used, together with its formulas. When implementing them in a computer, the lower bounds of the intervals shall be rounded down and the upper bounds, rounded up.

Integer Powers

$\triangleright$ $\underline{x} \in \mathbb{IR}$ and $n \in \mathbb{N}_0$

$$\underline{x}^n = \begin{cases} [<\underline{x}>^n, |\underline{x}|^n] & \text{if } n \text{ is even} \\ [\overleftarrow{\underline{x}}^n, \overrightarrow{\underline{x}}^n] & \text{if } n \text{ is odd} \\ [1, 1] & \text{if } n{=}0 \end{cases}$$

Sin

$\triangleright$ $\underline{x} \in [0, 2\pi]$

$$\sin \underline{x} = \begin{cases} [\sin \overleftarrow{\underline{x}}, \sin \overrightarrow{\underline{x}}] & \text{if } \overrightarrow{\underline{x}} \leq \pi/2 \text{ or } 3\pi/2 < \overleftarrow{\underline{x}} \\ [\sin(\min\{\overleftarrow{\underline{x}}, \pi - \overrightarrow{\underline{x}}\}), 1] & \text{if } \overleftarrow{\underline{x}} \leq \pi/2 < \overrightarrow{\underline{x}} \leq \pi \\ [\sin \overrightarrow{\underline{x}}, 1] & \text{if } \overleftarrow{\underline{x}} \leq \pi/2 \text{ and } \pi < \overrightarrow{\underline{x}} \leq 3\pi/2 \\ [-1, 1] & \text{if } \overleftarrow{\underline{x}} \leq \pi/2 \text{ and } 3\pi/2 < \overrightarrow{\underline{x}} \\ [\sin \overrightarrow{\underline{x}}, \sin \overleftarrow{\underline{x}}] & \text{if } \underline{x} \in [\pi/2, 3\pi/2] \\ [-1. \sin \overleftarrow{\underline{x}}] & \text{if } \pi/2 < \overleftarrow{\underline{x}} \leq \pi \text{ and } 3\pi/2 < \overrightarrow{\underline{x}} \\ [-1, \sin(\max\{\overrightarrow{\underline{x}}, \pi + \overleftarrow{\underline{x}}\})] & \text{if } \pi < \overleftarrow{\underline{x}} \leq 3\pi/2 < \overrightarrow{\underline{x}} \end{cases}$$

These formulas can only be used for $\underline{x} \in [0, 2\pi]$ and are therefore not valid for intervals including the origin (e.g. $[-0.1, 0.2]$) or for intervals whose width is greater than 2π. General formulas for any interval $\underline{x} \in \mathbb{IR}$, can be found in the code of BIAS (function `BiasSin` in file `BiasF.c`, see Appendix C).

Cos

$\triangleright\ \underline{x} \in [0, 2\pi]$

$$\cos \underline{x} = \begin{cases} [\cos \overrightarrow{\underline{x}}, \cos \overleftarrow{\underline{x}}] & \text{if } \overrightarrow{\underline{x}} \leq \pi \\ [-1, \cos \overleftarrow{\underline{x}}] & \text{if } \overleftarrow{\underline{x}} \leq \pi/2 \text{ and } \pi < \overrightarrow{\underline{x}} \leq 3\pi/2 \\ [-1, \cos(\max\{\overleftarrow{\underline{x}}, 2\pi - \overrightarrow{\underline{x}}\})] & \text{if } \overleftarrow{\underline{x}} \leq \pi/2 \text{ and } 3\pi/2 < \overrightarrow{\underline{x}} \\ [-1, \cos(\max\{\overleftarrow{\underline{x}}, 2\pi - \overrightarrow{\underline{x}}\})] & \text{if } \pi/2 < \overleftarrow{\underline{x}} \leq \pi < \overrightarrow{\underline{x}} \leq 3\pi/2 \\ [-1, \cos \overrightarrow{\underline{x}}] & \text{if } \pi/2 < \overleftarrow{\underline{x}} \leq \pi \text{ and } 3\pi/2 < \overrightarrow{\underline{x}} \\ [\cos \overleftarrow{\underline{x}}, \cos \overrightarrow{\underline{x}}] & \text{if } \overleftarrow{\underline{x}} > \pi \end{cases}$$

As for the sinus function, these formulas can only be used for $\underline{x} \in [0, 2\pi]$. General formulas for any interval $\underline{x} \in \mathbb{IR}$, can be found in the code of BIAS (function `BiasCos` in file `BiasF.c`, see Appendix C).

Arcssin

In some applications, inverse trigonometric functions can be viewed as an infinite sequence of intervals. However, for our purposes, we require that the results are in the interval $[0, 2\pi]$. The following formulas give the result in this interval, which can be composed by up to three disjoint intervals.

$\triangleright\ \underline{x} \in [-1, 1]$

$$\arcsin \underline{x} =$$

$$= \begin{cases} [0, 2\pi] & \text{if } \underline{x} = [-1, 1] \\ [\arcsin \overleftarrow{\underline{x}}, \pi - \arcsin \overleftarrow{\underline{x}}] & \text{if } 0 \leq \overleftarrow{\underline{x}} \text{ and } \overrightarrow{\underline{x}} = 1 \\ [0, \pi - \arcsin \overleftarrow{\underline{x}}] \cup [2\pi + \arcsin \overleftarrow{\underline{x}}, 2\pi] & \text{if } -1 < \overleftarrow{\underline{x}} < 0 \text{ and } \overrightarrow{\underline{x}} = 1 \\ [\arcsin \overleftarrow{\underline{x}}, \arcsin \overrightarrow{\underline{x}}] \cup [\pi - \arcsin \overrightarrow{\underline{x}}, \pi - \arcsin \overleftarrow{\underline{x}}] & \text{if } 0 \leq \overleftarrow{\underline{x}} \text{ and } 0 \leq \overrightarrow{\underline{x}} < 1 \\ [0, \arcsin \overrightarrow{\underline{x}}] \cup [\pi - \arcsin \overrightarrow{\underline{x}}, \pi - \arcsin \overleftarrow{\underline{x}}] \cup \ldots & \text{if } -1 < \overleftarrow{\underline{x}} < 0 \ldots \\ \qquad \ldots \cup [2\pi + \arcsin \overleftarrow{\underline{x}}, 2\pi] & \quad \ldots \text{ and } 0 \leq \overrightarrow{\underline{x}} < 1 \\ [0, \arcsin \overrightarrow{\underline{x}}] \cup [\pi - \arcsin \overrightarrow{\underline{x}}, 2\pi] & \text{if } \overleftarrow{\underline{x}} = -1 \text{ and } 0 \leq \overrightarrow{\underline{x}} < 1 \\ [\pi - \arcsin \overrightarrow{\underline{x}}, \pi - \arcsin \overleftarrow{\underline{x}}] \cup \ldots & \text{if } -1 < \overleftarrow{\underline{x}} < 0 \text{ and } \overrightarrow{\underline{x}} < 0 \\ \qquad \ldots \cup [2\pi + \arcsin \overleftarrow{\underline{x}}, 2\pi + \arcsin \overrightarrow{\underline{x}}] & \\ [\pi - \arcsin \overrightarrow{\underline{x}}, 2\pi + \arcsin \overrightarrow{\underline{x}}] & \text{if } \overleftarrow{\underline{x}} = -1 \text{ and } \overrightarrow{\underline{x}} < 0 \end{cases}$$

Arcscos

These formulas give the results in the interval $[0, 2\pi]$, which can be composed by one or two disjoint intervals.

▷ $\underline{x} \in [-1, 1]$

$$\arccos \underline{x} = \begin{cases} [\arccos \overrightarrow{\underline{x}}, 2\pi - \arccos \overrightarrow{\underline{x}}] & \text{if } \overleftarrow{\underline{x}} = -1 \\ [\arccos \overrightarrow{\underline{x}}, \arccos \overleftarrow{\underline{x}}] \cup [2\pi - \arccos \overleftarrow{\underline{x}}, 2\pi - \arccos \overrightarrow{\underline{x}}] & \text{if } -1 < \overleftarrow{\underline{x}} \end{cases}$$

Appendix C

BIAS/PROFIL Interval Libraries

BIAS/PROFIL Interval Libraries are actually two libraries: one containing the basic routines, named BIAS, and a set of routines containing the higher-level functions, called PROFIL. All interval operations of PROFIL are based on BIAS with the advantage that PROFIL is independent from the internal representation and the implementation of the interval types.

The package is portable, except for a single small assembler language routine that is called to change rounding modes; versions of this routine are available for a large variety of machines, including Sun, PC (running either DOS or Linux), HP workstations, etc (but not yet for Silicon Graphics workstations).

Documentation can be found either in html format at

`http://www.ti3.tu-harburg.de/Software/PROFIL/Profil.texinfo_toc.html`,

or in the technical reports 93.3 [28], 93.4, 93.5 and 95.4 of the Technische Universität Hamburg-Harburg (available via ftp at `ti3sun.ti3.tu-harburg.de/pub/reports/`).

C.1 Installation Tips

We will need the `dmake` utility, a free software package, for the installation of PROFIL/BIAS.

C.1.1 Dmake Installation

Dmake is a Make like tool written by Dennis Vadura (University of Waterloo), differing from other versions of Make in that it supports significant enhancements.

Dmake is available through anonymous ftp from `plg.uwaterloo.ca` in the `pub/dmake` directory.

Once we have downloaded and unpacked the source distribution, you can find the installation instructions in the files contained in the `readme` sub-directory.

Two steps should be pointed out:

1. Before compiling dmake, we shall edit the `startup.h` file in the corresponding directory (`unix` if we are in a UNIX machine) and indicate the path where we will set the file `startup.mk`; for instance:

   ```
   MAKESTARTUP := $(ROOTDIR)/usr/local/lib/dmake/startup.mk
   ```

2. In order to compile BIAS/PROFIL with dmake, we require a larger maximal name length. If we are not sure about the maximal name length defined by the compiler, we can edit the `posix.h` file in the `dmake` directory, replacing the three lines

   ```
   #ifndef _POSIX_NAME_MAX
   #define _POSIX_NAME_MAX 14
   #endif
   ```

   by the following two lines

   ```
   #undef _POSIX_NAME_MAX
   #define _POSIX_NAME_MAX 64
   ```
   .

C.2 BIAS/PROFIL Basic Installation

The package is available via anonymous ftp at `ti3sun.ti3.tu-harburg.de`. We will refer to the installation of the unix version throughout, which is contained in the directory `/pub/profil/unix`. We use only the basic package (`profil.tar.Z`) and the extension package (`profext.tar.Z`). We neither use the package related with long reals (`profillr.tar.Z`), nor the package dealing with global optimization (`profopt.tar.Z`).

The installation procedure is well described in the documentation in html format

`http://www.ti3.tu-harburg.de/Software/PROFIL/Profil.texinfo_toc.html`.

However, there are some details that can be confusing and are worth to be described here.

1. Some compilers added an underscore before C symbols, but since the new format ELF, this is no longer true. BIAS was written before that and since it contains some few assembler statements, compilation may fail. Once you have downloaded and unpacked the files, we shall edit file `Bias0.c` in directory `/Profil/BIAS` and add a line in the section corresponding to *Implementation of global functions* (lines 82 and 83). It should look like:

   ```
   #if defined (__MSC__) || defined (__BORLANDC__) \
   || (defined (__I386__) && defined (__GNUC__)) \
   || (defined (sparc) && defined (__GNUC__))
   ```

```
VOID BiasRoundUp (VOID) { _BiasRoundUp (); }
VOID BiasRoundDown (VOID) { _BiasRoundDown (); }
VOID BiasRoundNear (VOID) { _BiasRoundNear (); }

#endif
```

In Linux with ELF format, we should remove one of the underscores in lines 80 to 83 in file `biasint.h`.

2. There is a mistake in the function `BiasArcCos`. We should edit file `BiasF.c` in directory `/Profil/BIAS` and change for that function all `asin` for `acos`. The first line after the variable declarations should be substituted by

   ```
   x_inf = BiasSup (pX); x_sup = BiasInf (pX);
   ```

   and in the next line, we shall interchange `x_inf` and `x_sup`.

3. It is convenient to add to the `Makefile` in the `Profil` directory the compiling option `-w` in line 64 to hide the warning messages.

Bibliography

[1] Albala, H., and Angeles, J. Numerical solution to the input output displacement equation of the general 7R spatial mechanism. In *Proc. Fifth World Congress Theory of Mach. and Mech.* (1979), vol. 1, pp. 1008–1011.

[2] Alefeld, G., and Herzberger, J. ALOG-60 Algorithmen zur Auflösung linear Gleichungs-Systeme mit Fehlerfassung. *Computing 6* (1970), 28–34.

[3] Baker, D. R. Some topological problems in robotics. *The Mathematical Intelligencer 12*, 1 (1990), 66–76.

[4] Baker, D. R., and Wampler II, C. W. On the inverse kinematics of redundant manipulators. *The International Journal of Robotics Research 7*, 2 (1988), 3–21.

[5] Burdick, J. W. On the inverse kinematics of redundant manipulators: Characterization of the self–motion manifolds. In *Proc. IEEE Int. Conf. Robotics Automat.* (Aug. 1989), vol. 1, pp. 264–270.

[6] Castellet, A., and Thomas, F. Towards and efficient interval method for solving inverse kinematic problems. In *Proc. IEEE Int. Conf. Robotics Automat.* (Apr. 1997), vol. 4, pp. 3615–3620.

[7] Castellet, A., and Thomas, F. Using interval methods for solving inverse kinematic problems. In *Computational Methods in Mechanisms* (June 1997), vol. 2, NATO Advanced Study Institute, Varna, Bulgaria, pp. 135–144.

[8] Castellet, A., and Thomas, F. An algorithm for the solution of inverse kinematics problems based on an interval method. In *Advances in Robot Kinematics: Analysis and Control*, J. Lenarčič and M. L. Husty, Eds. Kluwer Academic Publishers, Dordrecht, Boston, London, 1998, pp. 393–402.

[9] Castellet, A., and Thomas, F. Characterization of the self-motion set of the orthogonal spherical mechanism. *Mechanism and Machine Theory* (1998). In press.

[10] Celaya, E. *Geometric Reasoning for the Determination of the Position of Objects Linked by Spatial Relationships.* PhD thesis, Universitat Politècnica de Catalunya, Dept. de Llenguatges i Sistemes Informàtics, Sept. 1992.

[11] Chiang, C. H. *Kinematics of Spherical Mechanisms.* Cambridge University Press, Cambridge, England, 1988.

[12] Craig, J. J. *Introduction to Robotics: Mechanics and Control*, 2nd ed. Addison-Wesley, Reading, Massachusetts, 1989.

[13] Denavit, J., and Hartenberg, R. S. A kinematic notation for lower-pair mechanisms based on matrices. *ASME, Journal of Applied Mechanics 22* (1955), 215–221.

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

[15] Goresky, M., and MacPherson, R. *Stratified Morse Theory.* Modern Surveys in Mathematics. Springer–Verlag, 1988.

[16] Gottlieb, D. H. Topology and the robot arm. *Acta Applicandae Mathematicae 11* (1988), 117–121.

[17] Hansen, E. *Global optimization using interval analyisis.* No. 165 in Monographs and textbooks in pure and applied mathematics. Marcel Dekker, New York, 1992.

[18] Hanson, R. J. Interval arithmetic as a closed arithmetic system on a computer. Tech. Rep. 197, Jet Propulsion Laboratory, Pasadena, CA, 1968.

[19] Hu, C. *Optimal Preconditioners for the Interval Newton Method.* PhD thesis, University of Southwestern Lousiana, 1990.

[20] Hunt, K. H. *Kinematic Geometry of Mechanisms.* Clarendon Press, Oxford, 1978.

[21] Hyvönen, E., and Pascale, S. D. LIA InC++: A local interval arithmetic library for discontinuous intervals, Version 1.0. Tech. rep., VTT Information Technology, Finland, Nov. 1994-1995. Also available at URL http://www.vtt.fi/tte/projects/interval/lia.ps.gz.

[22] Kahan, W. A more complete interval arithmetic. Lecture notes for a Summer course at the University of Michigan, 1968.

[23] Kearfott, R. B. INTLIB: A portable FORTRAN 77 interval standard function library. *ACM Trans. Math Software 20*, 4 (Dec. 1994), 447–459. Preprint available at URL http://interval.usl.edu/preprints/intlib_toms_algorithm.dvi or .ps.

[24] Kearfott, R. B. *Rigorous Global Search: Continuous Problems*, vol. 13 of *Nonconvex Optimization and Its Aplications.* Kluwer Academic Publishers, Dordrecht, Boston, London, 1996.

[25] Kearfott, R. B. Empirical evaluation of innovations in interval branch and bound algorithms for nonlinear systems. *SIAM J. Sci. Comput. 18*, 2 (Jan. 1997), 574–594.

[26] Kearfott, R. B., Hu, C., and Novoa III, M. A review of preconditioners for the interval Gauss-Seidel method. *Interval Computations 1*, 1 (1991), 59–85. Preprint available at URL http://interval.usl.edu/preprints/owlfpaper.dvi or .ps.

[27] Kearfott, R. B., and Novoa III, M. Algorithm 681l INTBIS, a portable interval Newton/bisection package. *ACM Trans. Math. Software 16*, 2 (1990), 152–157.

[28] Knüppel, O. PROFIL - Programmer's runtime optimized fast interval libraries. Tech. Rep. 93.4, Technische Universität Hamburg-Harburg, 1993. Also available through ftp://ti3sun.ti3.tu-harburg.de/pub/reports/report93.4.ps.Z.

[29] Latombe, J.-C. *Robot Motion Planning.* Kluwer Academic Publishers, Dordrecht, Boston, London, 1993.

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

[31] Lee, H.-Y., and Liang, C.-G. A new vector theory for the analysis of spatial mechanisms. *Mechanism and Machine Theory 23*, 3 (1988), 209–217.

[32] Li, Z. Geometrical considerations of robot kinematics. *International Journal of Robotics and Automation 5*, 3 (1990), 139–145.

[33] Manocha, D., and Canny, J. F. Efficient kinematics for general 6R manipulators. *IEEE Transactions on Robotics and Automation 10*, 5 (Oct. 1994), 648–657.

[34] Manocha, D., and Zhu, Y. A fast algorithm and system for the inverse kinematics of general serial manipulators. In *Proc. IEEE Int. Conf. Robotics Automat.* (1994), vol. 4, pp. 3348–3353.

[35] McAllester, D., van Hentenryck, P., and Kapur, D. Three cuts for accelerated interval propagation. A.I. Memo 1542, M.I.T., Artificial Intelligence Laboratory, May 1995. Available at URL http://www.ai.mit.edu/people/dam/newton-short.ps.

[36] Moore, R. E. *Interval Analysis.* Prentice–Hall, Englewood Cliffs, New Jersey, 1966.

[37] Moore, R. E. A test for existence of solutions to nonlinear systems. *SIAM J. Numer. Anal, 14*, 4 (Sept. 1977), 611–615.

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

[39] Nielsen, J., and Roth, B. Formulation and solution of the direct and inverse kinematics problem for mechanisms and mechatronic systems. In *Computational Methods in Mechanisms* (June 1997), vol. 1, NATO Advanced Study Institute, Varna, Bulgaria, pp. 233–252.

[40] Novoa III, M. Theory of preconditioners for the interval Gauss-Seidel method, existence/uniqueness theory with interval Newton methods, and formulas for slopes of powers. Tech. rep., Department of Mathematics, University of Southwestern Lousiana, 1993. Available through ftp://interval.usl.edu/interval_math/papers/Novoa's-unpublished-work/1993_work/ndiss.ps.

[41] Paul, R. P. *Robot Manipulators: Mathematics, Programming, and Control.* MIT Press, Cambridge, MA, 1981.

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

[43] Press, W. H., Teukolsky, S. A., Vetterling, W. T., and Flannery, B. P. *Numerical Recipes in C*, second ed. Cambridge University Press, Cambridge, England, 1992.

[44] Raghavan, M., and Roth, B. Kinematic analysis of the 6R manipulator of general geometry. In *Proc. 5th Int. Symp. on Robotics Research* (Cambridge, MA, 1989), H. Miuara and S. Arimoto, Eds., MIT Press, pp. 263–269.

[45] Raghavan, M., and Roth, B. A general solution for the inverse kinematics of all series chains. In *Proc. 8th CISM-IFTOMM Symp. on Theory and Practice of Robots and Manipulators* (1990).

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

[47] Rall, L. B. A comparison of the existence theorems of Kantorovich and Moore. *SIAM J. Numer. Anal. 17*, 1 (1980), 148–161.

[48] Rao, R. S., Asaithambi, A., and Agrawal, S. K. Inverse kinematic solution of robot manipulators using interval analysis. *Transactions of the ASME, Journal of Mechanical Design 120* (Mar. 1998), 147–150. Preprint available at URL: http://www.me.udel.edu/˜rao/modify.dvi.

[49] Ratz, D. On extended interval arithmetic and inclusion isotonicity. Tech. Rep. 5, Institut für Angew. Mathematik, Universität Karlsruhe, 1996. Available at URL http://www.uni-karlsruhe.de/˜iam/html/reports/rep9605.ps.gz.

[50] Ratz, D. An optimized interval slope arithmetic and its application. Tech. Rep. 4, Institut für Angew. Mathematik, Universität Karlsruhe, 1996. Available at URL http://www.uni-karlsruhe.de/˜iam/html/reports/rep9604.ps.gz.

[51] Rohn, J. NP-hardness results for linear algebraic problems with interval data. Preprint, Faculty of Math. and Physics, Charles University, Czech Republic, 1994.

[52] Roth, B. Computations in kinematics. In *Computational Kinematics*, J. Angeles, Ed. Kluwer Academic Publishers, Dordrecht, Boston, London, 1993, pp. 3–14.

[53] Roth, B. Computational advances in robot kinematics. In *Advances in Robot Kinematics and Computational Geometry*, A. J. Lenarčič, Ed. Kluwer Academic Publishers, Dordrecht, Boston, London, 1994, pp. 7–16.

[54] Roth, B., and Freudenstein, F. Synthesis of path-generating mechanisms by numerical methods. *J. of Engineering for Industry, Trans. ASME 85* (1963).

[55] Roth, B., Rastegar, J., and Scheinman, V. On the design of computer controlled manipulators. In *First CISM-IFToMM Symp.* (Sept. 1973), vol. 1.

[56] Rump, S. M. Verification methods for dense and sparse systems of equations. In *Topics in Validated Computations*, J. Herzberger, Ed. Elsevier Science Publisher, 1994, pp. 63–135.

[57] Stevenson, D. ANSI/IEEE STD 754-1985 (IEEE Standard for Binary Floating-Point Arithmetic). Tech. rep., IEEE, 1985.

[58] Thomas, F. Graphs of kinematic constraints. In *Computer-Aided Mechanical Assembly Planning*, S. Lee and H. de Mello, Eds. Kluwer Academic Publishers, Dordrecht, Boston, London, 1991, ch. 4, pp. 81–111.

[59] Thomas, F. On the n-bar mechanism, or how to find global solutions to redundant single loop kinematic chains. In *Proc. IEEE Int. Conf. Robotics Automat.* (July 1992), vol. 1, pp. 403–408.

[60] Thomas, F. An approach to the mover's problem that combines oriented matroid and algebraic geometry. In *Proc. IEEE Int. Conf. Robotics Automat.* (1995), pp. 2285–2292.

[61] Thomas, F., and Torras, C. A group-theoretic approach to the computation of symbolic part relations. *IEEE Journal of Robotics and Automation 4*, 6 (Dec. 1988), 622–634.

[62] Thomas, F., and Torras, C. Inferring feasible assemblies from spatial constraints. *IEEE Transactions on Robotics and Automation 8*, 2 (Apr. 1992), 228–239.

[63] Thomas, F., and Torras, C. The self-motion manifold of the n-bar mechanism. In *Computational Kinematics*, J. Ángeles, Ed. Kluwer Academic Publishers, Dordrecht, Boston, London, 1993, pp. 95–107.

[64] Thomas, F., and Torras, C. Positional inverse kinematic problems in $T^n \times \mathcal{R}^n$ solved in $T^{2(n+m)}$. In *Advances in Robot Kinematics and Computational Geometry*, A. J. Lenarčič, Ed. Kluwer Academic Publishers, Dordrecht, Boston, London, 1994, pp. 291–300.

[65] Tsai, L. W., and Morgan, A. P. Solving the kinematics of the most general six and five-degree-of-freedom manipulators by continuation methods. *Trans. ASME J. Mechanism, Transmissions and Autom. in Design 107* (1985), 189–200.

[66] Uicker, J. J., Denavit, J., and Hartenberg, R. S. An iterative method for the displacement analysis of spatial mechanisms. *Journal of Applied Mechanics, Transactions of the ASME* (June 1964), 309–314.

[67] van Hentenryck, P., McAllester, D., and Kapur, D. Solving polynomial systems using a branch and prune approach. *SIAM J. Numer. Anal. 34*, 2 (Apr. 1997), 797–827. Preprint available via ftp at ftp://ftp.cs.albany.edu/pub/ipl/papers/siam.97.ps.gz.

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

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

[70] Warner, F. W. *Foundations of Differentiable Manifolds and Lie Groups*, vol. 94 of *Graduate Texts in Mathematics.* Springer–Verlag, 1983. 2nd printing.