UNIVERSITAT POLITÈCNICA DE VALÈNCIA

# Synthesis of the Complete Inverse Kinematic Model of Non-Redundant Open-Chain Robotic Systems Using Groebner Basis Theory


Author: José Guzmán-Giménez

Director: Dr. Ángel Valera Fernández



November 2021

# Abstract

One of the most important elements of a robot's control system is its Inverse Kinematic Model (IKM), which calculates the position and velocity references required by the robot's actuators to follow a trajectory. The methods that are commonly used to synthesize the IKM of open-chain robotic systems strongly depend on the geometry of the analyzed robot, so they are not systematic procedures that can be applied equally in all situations. This project presents the development of a systematic procedure to synthesize the complete IKM of non-redundant open-chain robotic systems using Groebner Basis theory, which does not depend on the robot's geometry. The inputs to the developed procedure are the robot's Denavit-Hartenberg parameters and the movement range of its actuators, while the output is the IKM, ready to be used in the robot's control system or in a simulation of its behavior. This procedure's performance was proved synthesizing the IKMs of a PUMA manipulator and a walking hexapod robot. The computation times of both IKMs are comparable to those required by the kinematic models calculated by traditional methods, while the errors of their computed references were absolutely negligible. The synthesized IKMs are complete in the sense that they not only supply the position reference for all the robot's actuators, but also the corresponding references for their velocities and accelerations, so the developed procedure can be used in a wide range of robotic systems.

# Resumen

Uno de los elementos más importantes en el sistema de control de un robot es su Modelo Cinemático Inverso (IKM, por sus siglas en inglés), el cual calcula las referencias de posición y velocidad requeridas para que dicho robot pueda seguir una trayectoria. Los métodos más comúnmente empleados para la síntesis del IKM de sistemas robotizados de cadena cinemática abierta dependen fuertemente de la geometría del robot, por lo que no son procedimientos sistemáticos que puedan ser aplicados uniformemente en todas las situaciones. Este proyecto presenta el desarrollo de un procedimiento sistemático para la síntesis del IKM completo de sistemas robotizados no redundantes de cadena cinemática abierta usando la teoría de Bases de Groebner, el cual no depende de la geometría del robot. Las entradas del procedimiento desarrollado son los parámetros de Denavit-Hartenberg del robot y el rango de movimiento de sus actuadores, mientras que la salida es el IKM sintetizado, listo para ser usado en el sistema de control del robot o en una simulación de su funcionamiento. El desempeño del procedimiento desarrollado fue demostrado sintetizando los IKMs de un manipulador PUMA y un hexápodo caminante. Los tiempos de ejecución de ambos IKMs son comparables con los requeridos por los modelos cinemáticos calculados por procedimientos tradicionales, y los errores de las referencias que ofrecen como salida son totalmente despreciables. Los IKMs sintetizados son completos, porque no sólo ofrecen las referencias de posición para todos los actuadores del robot, sino que también calculan las correspondientes referencias de velocidades y aceleraciones de dichos actuadores, por lo que el procedimiento desarrollado puede ser empleado en una amplia variedad de sistemas robotizados.

# Resum

Un dels elements més importants en el sistema de control d'un robot és el seu Model Cinemàtic Invers (IKM, per les seues sigles en anglés), el qual calcula les referències de posició i velocitat requerides perquè aquest robot puga seguir una trajectòria. Els mètodes més comunament emprats per a la síntesi del IKM de sistemes robotitzats de cadena cinemàtica oberta depenen fortament de la geometria del robot analitzat, per la qual cosa no són procediments sistemàtics que puguen ser aplicats uniformement en totes les situacions. Aquest projecte presenta el desenvolupament d'un procediment sistemàtic per a la síntesi del IKM complet de sistemes robotitzats no redundants de cadena cinemàtica oberta usant la teoria de Bases de Groebner, el qual no depén de la geometria del robot. Les entrades del procediment desenvolupat són els paràmetres de Denavit-Hartenberg del robot i el rang de moviment dels seus actuadors, mentre que l'eixida és el IKM sintetitzat, llest per a ser usat en el sistema de control del robot o en una simulació del seu funcionament. L'acompliment del procediment desenvolupat va ser demostrat sintetitzant els IKMs d'un manipulador PUMA i un robot caminante. Els temps d'execució de tots dos IKMs són comparables amb els requerits pels models cinemàtics calculats per procediments tradicionals, i els errors de les referències que ofereixen com a eixida són totalment menyspreables. Els IKMs sintetitzats són complets, perquè no sols ofereixen les referències de posició per a tots els actuadors del robot, sinó que també calculen les corresponents referències de velocitats i acceleracions d'aquests actuadors, per la qual cosa el procediment desenvolupat pot ser emprat en una àmplia varietat de sistemes robotitzats.

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

*The main contribution of this work is the development of a systematic procedure for the synthesis of the complete Inverse Kinematic Model (IKM) of non-redundant open-chain robotic systems. The developed procedure employs Groebner Basis theory to synthesize this IKM, which solves the Inverse Kinematic Problem of the analyzed robots, finding an analytical solution if certain conditions are met. But before we get to the developed procedure's description, first we have to understand the Kinematic Problem and study the state of the art regarding this Robotic's topic.*

## 1.1 Background: The Kinematic Problem

The modeling and design of a robot's control system begins with the resolution of its kinematic problem, which is divided into two parts: the Forward Kinematics Problem (FKP) and the Inverse Kinematic Problem (IKP).

The Forward Kinematics Problem (FKP) involves finding the pose of a specific point in the robot's structure given its current state. This point is normally an important one in the robot's body, such as the end effector of a robotic manipulator or the center of mass of a mobile robot. The solution of the FKP, commonly referred as the robot's Forward Kinematics, is the equation system that establishes a mapping from the robot's state to the pose of this relevant point in its structure. The Forward Kinematics is fundamental for modeling

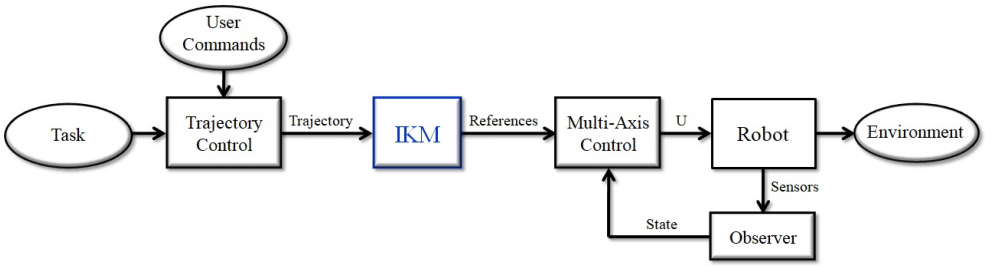


**Figure 1.1:** Inverse Kinematic Model (IKM) in the robot's control system. The IKM's function is to calculate the required references for the robot's multi-axis control.

the robot's movements, while it is also necessary to solve the IKP of open-chain robotic systems.

The Inverse Kinematics Problem (IKP), as its name clearly indicates, is the inverse of the FKP, so it involves finding the required robot's state when an specific point of its structure should reach a certain pose or, more generally, follow a path. By solving the IKP, the robot's Inverse Kinematic Model (IKM) is synthesized, whose function is to calculate the position and velocity references required by the robot's actuators to follow a trajectory. The IKM is a fundamental part of the robot's control system, as it supplies all the references required by the robot's multi-axis control. Figure 1.1 presents the relevance of the IKM in the robot's control system.

There are different procedures to compute the Forward Kinematics of an open-chain robotic system, which include Denavit-Hartenberg's algorithm (Atique, Sarker, and Ahad 2018; Flanders and Kavanagh 2015; Fu, Gonzalez, and Lee 1987), dual quaternions (Özgür and Mezouar 2016; X. Wang et al. 2012), and the modeling by Displacement Matrices (Barrientos et al. 2012). All these procedures calculate the Forward Kinematics through the execution of systematic algorithms, which are completely independent of the mechanical complexity of the robot's structure or its geometry.

In contrast, the techniques most commonly used to solve the IKP of open-chain robotic systems, the geometric method and the analytical procedure, strongly depend on the robot's structure (Guzmán-Giménez, Valera Fernández, et al. 2020). The geometric method, which will be properly explained in Section 4.1.2, requires an extensive analysis of the robot's geometry, separating the IKP into several plane geometry problems in order to find the relevant geometric equations (Y. Liu et al. 2015; Fu, Gonzalez, and Lee 1987). By its own

definition, it is obvious that this method depends heavily on the geometry of the robot's structure, so any sequence of steps used to synthesize the IKM of a robotic system may not be valid for any other different structure.

The analytical procedure, that will be deeply analyzed in Section 4.1.2, needs a detailed scrutiny of the robot's Forward Kinematics, to select the proper mathematical relations that solve the robot's state vector given a certain pose (Rodriguez et al. 2018; Petrescu et al. 2017; Chen et al. 2017; Bouzgou and Ahmed-Foitih 2014; Aydm and Kucuk 2006). The drawback of this analytical procedure is that the robot's Forward Kinematics is usually composed of non-linear geometric equations, which forbids the use of traditional matrix algebra procedures. Instead, individual relations inside the studied equation system must be found, which are specific for each case. This implies that the analytic procedure is also a non-systematic method, because the mathematical relations, found to solve the IKP of a robot with a certain structure, may not be valid or suitable for a different robotic system.

In summary, the two techniques most commonly used to solve the IKP heavily depend on the geometry of the robot's structure. Therefore they are not systematic procedures, because the steps executed to solve this problem would probably not be applicable to other robots.

To address this issue, various projects have opted to use different artificial intelligence techniques to solve the IKP of open-chain robotic systems. Within this group, the most recent and relevant are the works of Mahajan, Singh, and Sukavanam (2017) and Duka (2014), which employ Feed-Forward Neural Networks to solve this problem for robotic manipulators. The work of Toshani and Farrokhi (2014) use Radial Basis Function Neural Networks to solve the IKP of a redundant manipulator. Köker (2013) also uses Feed-Forward Neural Networks, but trains them with Genetic Algorithm techniques, in order to improve the precision of the solution given by the neural networks. The works of Deshmukh et al. (2020), Hussein, Gafer, and Fadhel (2020), Narayan and Singla (2017), Duka (2015) and Pérez-Rodríguez et al. (2012) solve the IKP of different types of open-chain robots with Adaptive Neuro-Fuzzy Inference Systems (ANFIS), which is an special kind of Neural Network based on the Takagi-Sugeno fuzzy inference system.

In all the previous works the IKP solver is a system that is first trained with a set of inputs and their respective solution targets. This training set is obtained using the robot's Forward Kinematics, and the trained IKP solver is validated with a separate validation set. The main problem with the artificial intelligence techniques arises when the target pose is far from the training set, because

the Neural Network's extrapolation capability may not be enough to obtain an acceptable solution. Also these systems are highly time consuming and normally cannot satisfy real time constraints (Rokbani, Casals, et al. 2015).

Other projects use evolutionary algorithms to solve the IKP of different types of robotic systems. In this category stands out the work of Rokbani and Alimi (2013), which employs Particle Swarm Optimization (PSO) to solve the IKP of a double link manipulator. Jiang et al. (2017) combine neural networks with evolutionary algorithms to solve the positioning problem of an open-chain puncturing robot using a PSO-optimized Neural Networks, while Rokbani, Casals, et al. (2015) implement a PSO variant, known as Firefly Algorithm (IK-FA).

While these procedures satisfactorily solve the IKP, they could suffer from the known training problems of evolutionary algorithms, such over-fitting or the convergence to local optima. It is also important to bear in mind that the solution offered by these techniques is not a properly defined IKM. This is because their solution is not the output of fully differentiable functions, therefore it will not be able to calculate the speed or acceleration references for the robot's actuators.

Nowadays several projects are being developed in the field of Robotics that use Groebner Basis theory (Buchberger 2001) to implement systematic methods that solve the Kinematic Problem, in a way that is completely independent of the geometry of the robot's structure.

These new type of works calculate a Groebner Basis from the analyzed robot's kinematic equations, to simplify the process of solving the Kinematic Problem and, if possible, find an analytical solution. The works of Kendricks (2013) and Y. Wang, Hang, and Yang (2006) present a systematic method to solve the IKP of robotic manipulators using Groebner Bases, while Rameau and Serré (2015) use this theory to calculate the mobility conditions of several mechanisms that are employed in robotic arms and parallel robots.

The Groebner Basis theory may also be used to solve the kinematic problem of closed-chain robotic systems, as is the case of the works of Gan et al. (2009) and Huang and He (2009), which employ it to solve the FKP of parallel robots. Abbasnejad and Carricato (2015) use Groebner Bases for the kinematic analysis of cable-driven parallel robots, while Uchida and McPhee (2012) utilize this theory to triangularize the kinematic constraint equations of this type of robots.

All the aforementioned works use Groebner Bases to solve their corresponding Kinematic Problems, proving in all cases that the full set of solutions can be obtained using this theory. However, they do not provide a systematic procedure to synthesize a Kinematic Model that can be used in the robot's control system (see Figure 1.1). Another issue that these works have is that the selection of the basis' monomial order, which is an important point when using Groebner Bases, is not specified. Some of those works state that they employ the same monomial order as the one in which the variables are solved by the traditional methods (Guzmán-Giménez, Valera Fernández, et al. 2020), while most just present the optimal order for their robotic system, without mentioning how it was selected (Rameau and Serré 2015; Kendricks 2013; Uchida and McPhee 2012; Gan et al. 2009; Huang and He 2009).

Chapter 2 presents all the necessary background to be able to use Groebner Basis theory to solve a wide variety of engineering problems, including the IKP of open-chain robotic systems. But first we need to establish the main objective of this work.

## 1.2 Objective and main contribution

The main objective of this work was to use the Groebner Basis theory to develop a systematic procedure for the synthesis of the complete Inverse Kinematic Model of non-redundant open-chain robotic systems.

The developed procedure only requires as inputs the robot's Denavit-Hartenberg parameters and the movement range of its actuators, while its output is the synthesized IKM, ready to be used in the robot's control system or to simulate its behavior. During its execution, the procedure does not require any further input from the user, as it is able to automatically select the optimal monomial order of the Groebner Basis, as well as prepare and configure all the elements that compose the complete IKM.

The synthesized complete IKM not only supplies the position reference for all the robot's actuators, as it is case of the IKP solvers obtained by artificial intelligence techniques or evolutionary algorithms, but also provides the corresponding references for the actuator's velocities and accelerations. This way, the IKMs that are synthesized by the developed procedure can be used in a wide range of robot's control systems, including those that have an acceleration feedback loop or an acceleration feed-forward strategy.

The main contribution of this project is the development of this systematic procedure that automatically synthesizes the complete Inverse Kinematic Model of non-redundant open-chain robotic systems. The developed procedure can be used to synthesize the IKM of a wide range of mobile and open-chain industrial robots, including cartesian robotic systems, SCARA robots, multi-legged walking and climbing robots and all non-redundant manipulators that satisfy the in-line wrist condition.

In addition to this main contribution, this work has other four contributions:

- The developed procedure automatically selects the optimal monomial order for the Groebner Basis used in the IKM synthesis. Therefore, the user is relieved from the burden of selecting this monomial order. The procedure's user only has to provide the D-H parameters of the analyzed robot and the movement range of its actuators.
- The synthesized IKM is ready to be used in the robot's microcontroller. The output of the developed procedure is the IKM, both in C++ and as a MATLAB® script, which can be used directly in the robot's control system or in a simulation of its behavior.
- The synthesized IKM does not request any kind of complex number operations from the robot's microcontroller. All the required operations from the microcontroller are: floating point additions, multiplications and divisions, atan2, cosines, square roots and, in some special cases, a cubic root. All these operations can be easily executed by modern-day microcontrollers, even when they do not have any capability of operating with complex numbers.
- The last contribution is the framework presented in this project, which can be used to apply Groebner Basis theory in the resolution of a wide variety of engineering problems. This last contribution will be fully explained in Section 5.1.

## 1.3 Test Benches

Two robotic systems were used in this project as test benches for the developed procedure:

1. Walking hexapod robot BH3-R, built and distributed by Lynxmotion Inc. (Swanton, Vermont, USA)

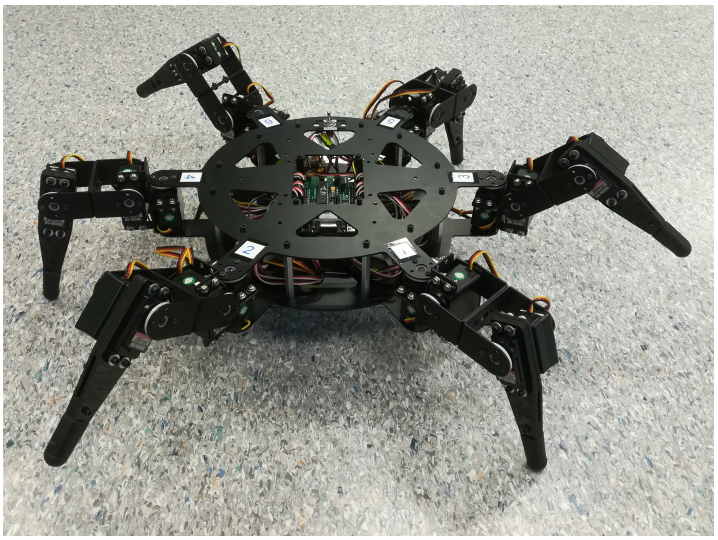

**Figure 1.2:** BH3-R hexapod walking robot by Lynxmotion Inc.

2. Unimate's PUMA 560 robotic arm (Danbury, CT, USA)

### *1.3.1 Test bench 1: Hexapod walking robot*

The hexapod walking robot used as a test bench is shown in Figure 1.2. This robot was selected as a test bench because each of its legs has three rotational degrees of freedom for positioning, like most industrial robotic arms and many multi-legged walking robots, and it only requires the resolution of the position's kinematic problem.

Because this hexapod has circular geometry, its kinematic problem can be simplified to the synthesis of the IKM of one of the robot's legs, to later apply the corresponding transformations between the hexapod's center and the origin of each of its extremities.

Applying the Denavit-Hartenberg's convention, the coordinate systems created for a leg of the BH3-R walking hexapod are shown in Figure 1.3. Based on the coordinate systems presented in Figure 1.3, the D-H parameters of one leg are shown in Table 1.1, while Table 1.2 lists the dimensions, in mm, of those parameters. To complete the kinematic description of the hexapod's leg, Table 1.3 presents the movement range of all its actuators.

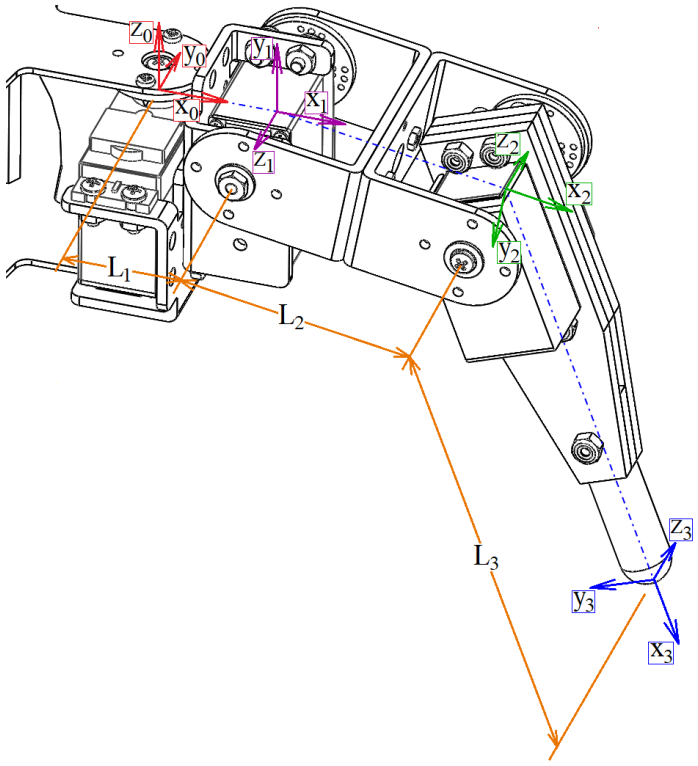


**Figure 1.3:** Coordinate systems for the hexapod's leg.

**Table 1.1:** Denavit-Hatenberg Parameters of the hexapod's leg.

| **Link** | $\boldsymbol{\theta}$ | $\boldsymbol{d}$ | $\boldsymbol{a}$ | $\boldsymbol{\alpha}$ |
|---|---|---|---|---|
| 1 | $q_1$ | 0 | $L_1$ | $\pi/2$ |
| 2 | $q_2$ | 0 | $L_2$ | $\pi$ |
| 3 | $q_3 + (\pi/2)$ | 0 | $L_3$ | 0 |

**Table 1.2:** Hexapod's leg parameter dimensions.

| Parameter | Dimension [mm] |
|---|---|
| $L_1$ | 28 |
| $L_2$ | 58 |
| $L_3$ | 110 |

**Table 1.3:** Movement range of the hexapod's actuators.

| Actuator | Minimum [rad] | Maximum [rad] |
|---|---|---|
| $q_1$ | -1.3990 | 1.3990 |
| $q_2$ | -0.6627 | 1.5217 |
| $q_3$ | -1.5585 | 0.6013 |

### *1.3.2 Test bench 2: PUMA robotic arm*

Figure 1.4 presents the PUMA robotic arm that was used as a test bench in this work.

Following the Denavit-Hartenberg's (D-H) convention (Atique, Sarker, and Ahad 2018; Fu, Gonzalez, and Lee 1987), the coordinate systems created for all the links of the PUMA 560 manipulator are shown in Figure 1.5. Based on those coordinate systems, the D-H parameters of every link of the robot are the ones contained in Table 1.4, while Table 1.5 lists the dimensions, in mm, of these PUMA's parameters.

The kinematic description of the PUMA manipulator is completed with the movement range of all its actuators, which is shown in Table 1.6.

The D-H parameters shown in Table 1.4 and the actuators' movement range of Table 1.6 are all the inputs required from the user to synthesize the PUMA's IKM with the procedure developed in this work. Chapter 3 will expand all this information regarding the developed procedure's inputs and the output that it offers.


**Figure 1.4:** Unimate's PUMA 560 robotic arm

**Table 1.4:** Denavit-Hatenberg Parameters of PUMA 560.

| Link | $\boldsymbol{\theta}$ | $\boldsymbol{d}$ | $\boldsymbol{a}$ | $\boldsymbol{\alpha}$ |
|---|---|---|---|---|
| 1 | $q_1 + (\pi/2)$ | $d_1$ | 0 | $-\pi/2$ |
| 2 | $q_2$ | $d_2$ | $a_2$ | 0 |
| 3 | $q_3 + (\pi/2)$ | 0 | $-a_3$ | $\pi/2$ |
| 4 | $q_4$ | $d_4$ | 0 | $-\pi/2$ |
| 5 | $q_5$ | 0 | 0 | $\pi/2$ |
| 6 | $q_6$ | $d_6$ | 0 | 0 |

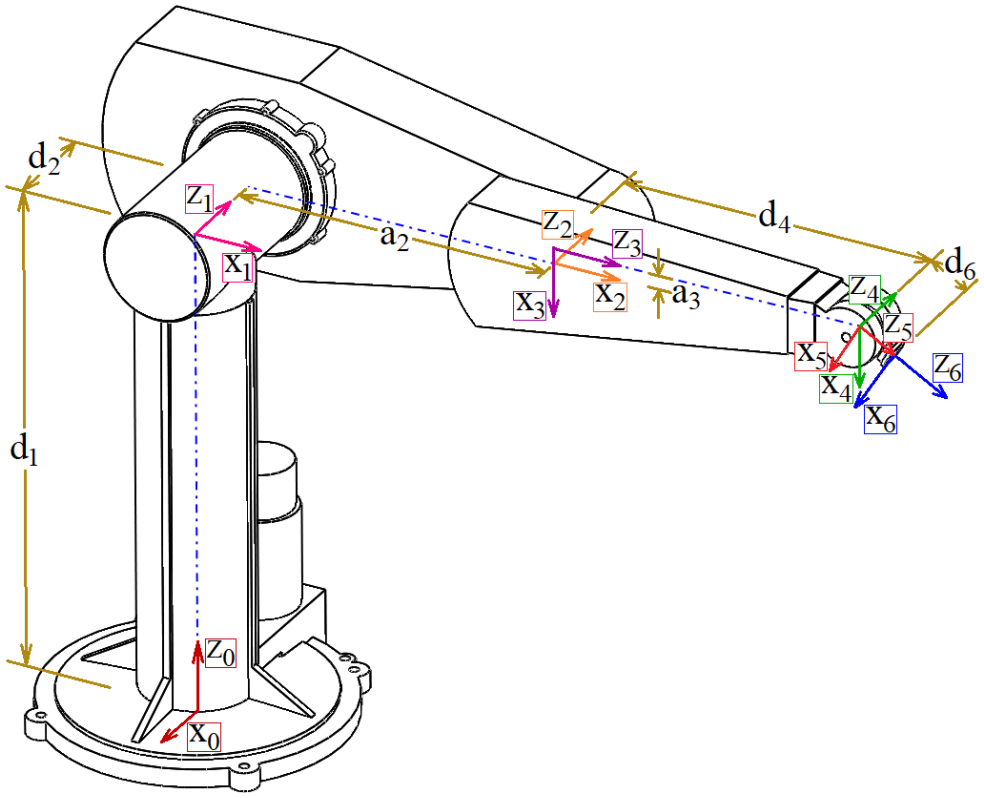


**Figure 1.5:** Coordinate systems for the PUMA 560 robotic arm.

**Table 1.5:** PUMA 560 parameters dimensions.

| Parameter | Dimension [mm] |
|---|---|
| $d_1$ | 660.4 |
| $d_2$ | 149.1 |
| $a_2$ | 431.8 |
| $a_3$ | 20.3 |
| $d_4$ | 433.1 |
| $d_6$ | 56.2 |

**Table 1.6:** Movement range of the PUMA's actuators.

| Actuator | Minimum [rad] | Maximum [rad] |
|---|---|---|
| $q_1$ | -2.7925 | 2.7925 |
| $q_2$ | -3.1415 | 0.7854 |
| $q_3$ | -2.3562 | 2.3562 |
| $q_4$ | -1.9199 | 2.9671 |
| $q_5$ | -1.7453 | 1.7453 |
| $q_6$ | -3.1415 | 3.1415 |

## 1.4 Structure of this work

The next chapter shows the applications of the Groebner Basis theory in a wide variety of engineering problems, and it also explains all the basic concepts required to understand these applications. The procedure developed in this work is presented in Chapter 3, and it was used to synthesize the IKMs of the two test benches: a PUMA manipulator and a walking hexapod. Chapter 4 shows the performance analysis of the two synthesized IKMs, comparing their outputs with those of the corresponding reference models. Finally Chapter 5 summarizes the developed procedure, explains the conclusions of this work, and also presents the possible future works for the project.

# Chapter 2

# Groebner Bases theory: applications in engineering and basic concepts

*This chapter presents the applications of the Groebner Basis theory in engineering projects, with special emphasis in Robotics, and explains all the basic concepts needed to apply it.*

## 2.1 Applications of Groebner Bases theory in engineering projects

Groebner Basis theory is a useful methodology to solve any problem that can be expressed as a polynomial equation system. This theory allows to transform any polynomial equation system to a new form, a Groebner Basis, that is easier to solve. This transformation is analogous to the row reduction technique (also called Gaussian elimination) used to lower the complexity of linear equation systems (Cox, Little, and O'Shea 2015).

Groebner Basis theory has been applied in a wide range of engineering projects, like Digital Circuit Design, as is the case of the work of Farahmandi and Alizadeh (2015), which studies the verification of large integer arithmetic circuits through Symbolic Computer Algebra techniques and Groebner Basis theory. The circuits and their specifications are modeled as polynomial equation sys-

tems, in such a way that the verification is formulated as a membership problem that is solved using Groebner Basis theory. This method was improved by the work of Sabbagh and Alizadeh (2021), in which the output of the verification process is a remainder polynomial that indicates the correctness of the analyzed circuit, and can also be used to correct those circuits that are defective.

This theory can also be applied to solve problems in Automation and Control, like the Time Optimal Feedback Control of a controllable LTI system with bounded inputs, as shown by Patil et al. (2015). Their work employs Groebner basis theory to formulate the control switching surfaces as semi-algebraic sets, which are then used to synthesize the nested switching logic required for the time optimal feedback control.

Returning to the field of Robotics, the leader selection problem in Multi-Agent Robotics can also be solved using Groebner Basis theory, as is proven by Mulla et al. (2018). They formulate this problem as a set of polynomial equations, which can be easily solved once the corresponding Groebner Basis is calculated.

The work of Koukos-Papagiannis, Moulianitis, and Aspragathos (2019) uses Groebner Basis theory to classify the cuspidal anatomies of a 3R orthogonal metamorphic manipulator with two pseudo-joints. H. Liu and Han (2021) employ Groebner Basis theory to solve the position analysis problem of a 1CS-4SS spatial linkage.

Before we proceed to the explanation of the developed procedure, we need to establish the proper mathematical definitions for polynomials and Groebner Bases.

## 2.2 Polynomials, Ideals and Affine Varieties

The polynomials that will be studied and analyzed in this work are defined over a Ring, specifically a Commutative Ring (also known as Polynomial Ring), while their coefficients are defined over a Field. Therefore, as a prelude to the theory of Groebner Bases, first we have to establish the definitions of Fields, Commutative Rings and polynomials.

The basic intuitive idea of a Field is that it is a set where the operations of addition, subtraction, multiplication and division are defined, with their usual properties (Cox, Little, and O'Shea 2015). The formal definition of Field is presented in Definition 2.2.1.

**Definition 2.2.1** ***Field.*** *A Field consists of a set $k$ and two binary operations, addition (represented with the symbol "+") and multiplication (indicated by "·"), both defined on $k$, for which the following seven properties are satisfied:*

1. *Associative: $(a + b) + c = a + (b + c)$ and $(a \cdot b) \cdot c = a \cdot (b \cdot c)$ for all $a, b, c \in k$.*
2. *Commutative: $a + b = b + a$ and $a \cdot b = b \cdot a$ for all $a, b \in k$.*
3. *Distributive: $a \cdot (b + c) = a \cdot b + a \cdot c$ for all $a, b, c \in k$.*
4. *Additive identity: There is $0 \in k$ such that $a + 0 = a$, for all $a \in k$.*
5. *Multiplicative identity: There is $1 \in k$ such that $a \cdot 1 = a$, for all $a \in k$.*
6. *Additive inverse: Given $a \neq 0$, there is $b \in k$ such that $a + b = 0$, for all $a \in k$.*
7. *Multiplicative inverse: Given $a \neq 1$, there is $b \in k$ such that $a \cdot b = 1$, for all $a \in k$.*

The most common fields are $\mathbb{Z}$ and $\mathbb{C}$. The coefficients of a polynomial are defined over a Field, normally $\mathbb{Z}$, but the polynomial itself cannot be part of a Field. This is because no polynomial can satisfy the multiplicative inverse property, which will be obvious after the formal definition of polynomials is presented. This is the reason we have to bring forward the definition of Commutative Ring as a set that contains the analyzed polynomials.

**Definition 2.2.2** ***Commutative Ring.*** *A Commutative Ring is composed by a set $R$ and two binary operations, addition and multiplication, both defined on $R$, that satisfy the first six properties of the Field's operations: associative, commutative, additive and multiplicative identity, and additive inverse.*

As can be seen in Definition 2.2.2, the only difference between a Commutative Ring and a Field is that the operations defined for the Commutative Ring do not have to comply with the multiplicative inverse property. All the Commutative Rings used in this work are sets that contain polynomials, so we will also refer to them as Polynomial Rings (Cox, Little, and O'Shea 2015).

The best way to define a polynomial is as a linear combination of monomials over a Polynomial Ring. In this work we will be constantly analyzing multivariable polynomial equations, so it is important to establish a proper definition for a monomial that contains multiple variables, as presented in Definition 2.2.3

**Definition 2.2.3** ***Monomial.*** *A monomial over a set of variables $x = [x_1, x_2, ..., x_n]$ on the Commutative Ring $k$, normally represented as $k[x_1, x_2, ..., x_n]$, is a product over these variables:*

$$x^{\alpha} = x_1^{\alpha_1} \cdot x_2^{\alpha_2} \cdot ... \cdot x_n^{\alpha_n} \tag{2.1}$$

*In Equation 2.1, the exponent $\alpha$ is an n-tuple of the form $\alpha = (\alpha_1, \alpha_2, ..., \alpha_n)$, where all its elements are non-negative integers.*

With the previous definition for multivariable monomials, a polynomial can be defined as a linear combination of monomials, as presented in Definition 2.2.4.

**Definition 2.2.4** ***Polynomial.*** *A polynomial $f$ is a linear combination of a finite number of monomials in $k[x_1, x_2, ..., x_n]$, which is written in the form:*

$$f = \sum_{\alpha} c_{\alpha} \cdot x^{\alpha} \tag{2.2}$$

*The coefficients in Equation 2.2, $c_{\alpha}$, are defined over a Field, normally $c_{\alpha} \in \mathbb{Z}$.*

Every polynomial that follows Definition 2.2.4 is an element of the Commutative Ring $k[x_1, x_2, ..., x_n]$, which from now on will be referenced as the Polynomial Ring $k$, or simply $k$. It is important to always bear in mind that $k$ is a Commutative Ring and not a Field, because the multiplicative inverse for any monomial $x^{\alpha} \in k$, which should be $x^{-\alpha}$, is not a valid element of $k$. But all the properties that define a Commutative Ring (see Definition 2.2.2) are completely fulfilled in $k$.

After defining the basic structure of the polynomials that will be used in this work, now we have to bring up the concept of Ideal, which is fundamental in Groebner Basis Theory, and will be crucial when solving the polynomial equation systems that appear as part of the Kinematic Problem of robotic systems.

An Ideal is just a subset of a Polynomial Ring $k$ that satisfies the conditions presented in Definition 2.2.5 (Cox, Little, and O'Shea 2015).

**Definition 2.2.5** ***Ideal.*** *A subset $I \subset k[x_1, x_2, ..., x_n]$ is an Ideal if it satisfies the following three conditions:*

1. $0 \in I$.
2. *If* $f, g \in I$ *then* $f + g \in I$.
3. *If* $f \in I$ *and* $h \in k[x_1, x_2, ..., x_n]$ *then* $h \cdot f \in I$.

An important characteristic about the Ideals of Polynomial Rings is that they can be defined by a linear combination of polynomials. For example, if we have a set of $s$ polynomials, $\{b_1, b_2, ..., b_s\} \in k[x_1, x_2, ..., x_n]$, we can generate the linear combination expressed in Equation 2.3:

$$< b_1, b_2, ..., b_s >= \{\sum_{i=1}^{s} h_i \cdot b_i \ : \ h_1, h_2, ..., h_n \in k[x_1, x_2, ..., x_n]\} \tag{2.3}$$

The crucial fact is that $< b_1, b_2, ..., b_s >$ is an Ideal, known as the Ideal generated by $b_1$ to $b_s$. The generating set, $\{b_1, b_2, ..., b_s\} \in k$, is called a basis of $I$. And any Ideal can be generated by a finite basis, as it is established by the Hilbert Basis Theorem (Cox, Little, and O'Shea 2015):

**Theorem 2.2.1** ***Hilbert Basis Theorem.*** *Every ideal* $I \subset k[x_1, x_2, ..., x_n]$ *has a finite generating set that is also part of* $I$. *That is,* $I =< g_1, ..., g_s >$ *for some* $\{g_1, ..., g_s\} \in I$ .

Note that following the expression presented in Equation 2.3, any given Ideal may have many different bases, but they all generate the same subset $I \subset k[x_1, x_2, ..., x_n]$. The Groebner Bases that will be used in this work are based in this idea of a set of polynomials that generate an Ideal. But before we define this type of bases, we need to present the concept of Affine Varieties, which will be fundamental to prove that the solution of the calculated basis is useful for our work.

**Definition 2.2.6** ***Affine Variety.*** *Let* $k[x_1, x_2, ..., x_n]$ *be a Polynomial Ring and* $\{f_1, f_2, ..., f_s\}$ *be a set of* $s$ *polynomials in* $k$. *The Affine Variety of this set is the expression presented in Equation 2.4.*

$$V(f_1, f_2, ..., f_s) = \{(a_1, a_2, ..., a_n) \ : \ f_i(a_1, a_2, ..., a_n) = 0 \quad \forall i \in [1, s]\} \tag{2.4}$$

Thus, an Affine Variety, $V(f_1, f_2, ..., f_s)$, is the set of all the solutions of the polynomial equation system composed of $\{f_1, f_2, ..., f_s\} = 0$.

Since an Ideal generated by a basis is just a linear combination of the polynomials that conform its basis (see Equation 2.3), then the Affine Variety of the whole Ideal is equal to the Affine Variety of the polynomial set that compose the basis. That is, if we have an Ideal, $I \subset k[x_1, x_2, ..., x_n]$, where $I =< b_1, ...b_s >$, then:

$$V(I) = V(b_1, ..., b_n) \tag{2.5}$$

Using the definition of Affine Variety presented in Equation 2.4 and the Hilbert Basis Theorem, we can extend the definition of Affine Varieties to any Ideal, $I \subset k[x_1, x_2, ..., x_n]$, which will be equal to:

$$V(I) = \{(a_1, ..., a_n) \ : \ f(a_1, ..., a_n) = 0 \quad \forall f \in I\} \tag{2.6}$$

Although a non-zero Ideal contains infinitely many polynomials (check Equation 2.3), its Affine Variety, denoted as $V(I)$, is unique. Therefore the Affine Variety of the basis that generated the Ideal will be the same as the Affine Variety of any other basis that is calculated from said Ideal.

This means that the set of solutions of the original polynomial equation system (the one that generated the Ideal) will be the same of solutions of any Groebner Basis derived from this original equation system. So, as the calculated Groebner Basis normally is easier to solve than the original polynomial equation system, by solving this easier system we obtain the desired set of solutions.

## 2.3 Monomial Ordering in Groebner Bases

An important factor of the Groebner Basis obtained from any Ideal is the monomial ordering used in its calculation. This monomial ordering is defined in Definition 2.3.1 (Cox, Little, and O'Shea 2015).

**Definition 2.3.1** ***Monomial Ordering.*** *A monomial ordering over a set of variables $x = [x_1, x_2, ..., x_n]$ on the field $k$, normally represented as $k[x_1, x_2, ..., x_n]$, is any relation $>$ on the set of monomials $x^\alpha$, with $\alpha \in \mathbb{Z}^n_{\geq 0}$, that satisfies three conditions:*

1. *The relation $>$ is a total ordering on $\mathbb{Z}^n_{\geq 0}$. This condition establishes that for every pair of monomials, $x^\alpha$ and $x^\beta$, with $\alpha = (\alpha_1, \alpha_2, ..., \alpha_n)$ and*

$\beta = (\beta_1, \beta_2, ..., \beta_n) \in \mathbb{Z}^n_{\geq 0}$, *exactly one of the three following statements is true:* $x^\alpha > x^\beta$ , $x^\alpha = x^\beta$ *or* $x^\beta > x^\alpha$.

2. *If* $x^\alpha > x^\beta$ *and* $\gamma = (\gamma_1, \gamma_2, ..., \gamma_n) \in \mathbb{Z}^n_{\geq 0}$, *then* $x^{(\alpha+\gamma)} > x^{(\beta+\gamma)}$.

3. $>$ *is a well-ordering on* $\mathbb{Z}^n_{\geq 0}$. *This means that every non-empty subset of* $\mathbb{Z}^n_{\geq 0}$ *has a smallest element under this ordering.*

The main three types of monomial orderings used in Groebner Bases calculations are: Lexicographic Order (lex), Graded Lexicographic Order (grlex) and Graded Reverse Lexicographic Order (grevlex) (Cox, Little, and O'Shea 2015).

The lexicographic order (lex) arranges the monomials following a strict ordering in which a variable always precedes those of lesser value in the order, in an analogous way to the ordering of words in a dictionary. This type of monomial ordering is defined in definition 2.3.2 (Cox, Little, and O'Shea 2015).

**Definition 2.3.2** ***Lexicographic Order (lex).*** *The lex order establishes that* $x^\alpha >_{lex} x^\beta$ *if and only if, in the vector difference* $\alpha - \beta \in \mathbb{Z}^n_{\geq 0}$, *the leftmost non-zero entry is positive.*

With the lex order, a variable will always dominate any monomial involving only smaller variables, regardless of its total degree. For some purposes, it is desirable to take into account the total degree of monomials, ordering them by total degree first. This is done by using the graded lexicographic order (grlex), presented in definition 2.3.3 (Cox, Little, and O'Shea 2015).

**Definition 2.3.3** ***Graded Lexicographic Order (grlex).*** *The grlex order establishes that* $x^\alpha >_{grlex} x^\beta$ *if the condition of Equation (2.7) is met:*

$$|\alpha| = \sum_{i=1}^{n} \alpha_i > |\beta| = \sum_{i=1}^{n} \beta_i \tag{2.7}$$

*or, in the case that* $|\alpha| = |\beta|$, *if* $x^\alpha >_{lex} x^\beta$, *i.e. if the leftmost non-zero entry of* $\alpha - \beta$ *is positive.*

To reduce the computation time of Groebner Bases, a variation of the grlex, known as the graded reverse lexicographic order (grevlex), was developed, which is defined in definition 2.3.4 (Cox, Little, and O'Shea 2015).

**Definition 2.3.4** ***Graded Reverse Lexicographic Order (grevlex).*** *The grevlex order establishes that $x^\alpha >_{grevlex} x^\beta$ if the condition of Equation (2.7) is met or, in the case that $|\alpha| = |\beta|$, if the rightmost non-zero entry of the difference $\alpha - \beta$ is negative.*

This variation begins like grlex, ordering the monomials by total degree, but then breaks any possible ties by applying a reverse lexicographical order. This ordering is the most efficient way to calculate a Groebner Basis for a zero-order Ideal, i.e. an Ideal that has a finite amount of solutions (Cox, Little, and O'Shea 2015), which is exactly the type of Ideal that the developed procedure has to work with (Guzmán-Giménez, Valera Fernández, et al. 2020).

The main objective of our procedure, as presented in Guzmán-Giménez, Valera Fernández, et al. 2020, is to use the Groebner Basis theory to find an analytical solution to the Inverse Kinematic Problem of non-redundant open-chain robotic systems, as a means to synthesize their IKM. This objective is achieved by calculating a Groebner Basis in a two step process: First, an initial basis is obtained using a grevlex monomial order, using Faugère's F4 algorithm (Jean Charles Faugère 2010; Jean Charles Faugère 1999). Then this first basis is converted to a Groebner Basis with a lex order, employing the FGLM algorithm developed by Faugère et al (J. C. Faugère et al. 1993). This conversion is done because the lex monomial order is the only one that guarantees the existence of a simple analytical solution for the calculated basis.

# Chapter 3

# Description of the Developed Procedure

*The developed procedure is fully explained in this chapter, going from the initial input of the robot's information to the synthesized IKM. The chapter begins with the description of the structure that all the IKMs synthesized by the developed procedure will have. This structure allows the synthesized IKM not only to provide the position references for all the robot's actuators, but also the references of their corresponding velocities and accelerations. The next four sections of the chapter explain each of the modules that compose the IKM's structure and how the developed procedure generates and configures them.*

As stated in Section 1.2, the main objective of this work was the development of a procedure that synthesizes the IKM of non-redundant open-chain robotic systems. The developed procedure only requires as inputs the robot's D-H parameters and the range of its actuators. With these inputs, and without any further information from the user, the procedure's output is the full IKM, ready to be used in the robot's microcontroller or to simulate its behavior.

To fully understand the procedure that was developed in this work, first we have to analyze the structure of the IKM that this procedure has to synthesize. Section 3.1 presents this general structure, which will be the same for all the IKMs synthesized by the developed procedure.

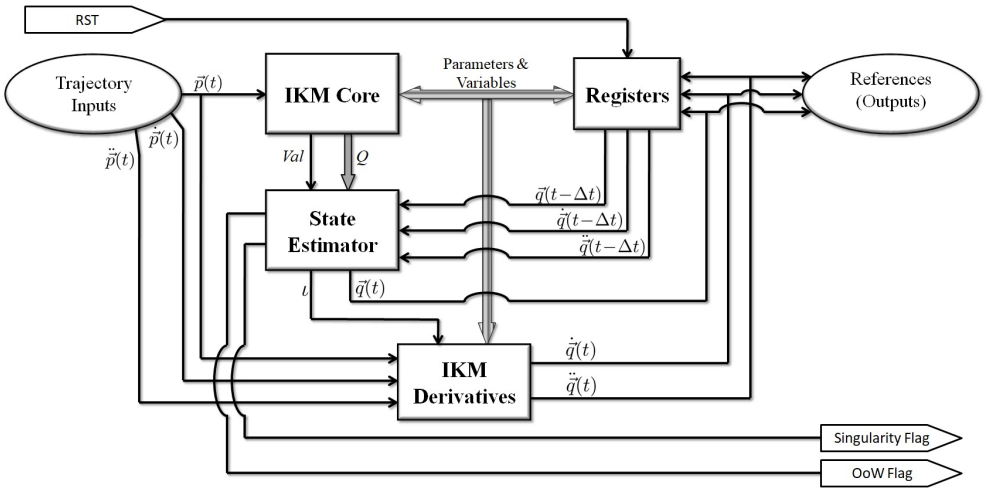


**Figure 3.1:** Structure of the synthesized IKM

## 3.1 Structure of the Synthesized IKM

The synthesized IKMs will all have the structure shown in Figure 3.1. This structure presents four modules:

1. IKM Core: Synthesized applying Groebner Basis theory. Its function is to solve the IKP of position for all the robot's actuators.
2. State estimator: Selects the best solution given by the IKM Core.
3. IKM Derivatives: Solves the IKP of velocities and accelerations for all the robot's actuators, to complete the robot's current state.
4. Registers: Stores the robot's previous and current states, as well as all the variables required by the other modules.

## 3.2 IKM Core Module

The main module of the synthesized IKM is the IKM Core, whose main function is to solve the Inverse Kinematic Problem (IKP) of the robot's position in the configuration space. The input of this module is the desired position for the robot in the cartesian space, and its outputs are the array $Q$, which

contains all the possible solutions for the IKP in configuration space, along with the vector $Val$, that signals the solutions that are valid, i.e. the ones that are inside the robot's workspace.

The developed procedure for the synthesis of the IKM Core module is composed of seven major steps, including a verification step that checks if the selected monomial order for the Groebner Basis is valid in all the robot's workspace. Figure 3.2 shows the complete flowchart of this procedure (Guzmán-Giménez, Valera Fernández, et al. 2021).

It is important to highlight that all these steps are only executed once, out of line, to synthesize the IKM Core before the robotic system is activated. Therefore, the execution time of these steps does not affect in any way the online computation time of the synthesized IKM (Guzmán-Giménez, Valera Fernández, et al. 2020).

### *3.2.1 First Step: Information Input*

The first step of the procedure presented in Figure 3.2 is the information input, in which the user has to supply only two data sets: the Denavit-Hartenberg (D-H) parameters of the robot and the movement range of its actuators. The first data, the D-H parameters, are necessary to solve the Forward Kinematics problem, which is a required step to calculate the Groebner Basis that will aid in the IKM's synthesis. These parameters represent the fundamental information of any robotic system, so they are also indispensable for all traditional methods that solve the Kinematic Problem.

The actuators' movement range is employed to eliminate all the solutions that are not reachable by the robotic system. This way, the IKM is able to discard those solutions that are out of the robot's workspace.

This is all the information that the user has to know about the analyzed robot, which constitutes the absolute minimum data that is required to describe a robotic system. From this point, the procedure automatically synthesizes the robot's IKM Core module, and subsequently the whole IKM, without any further input from the user (Guzmán-Giménez, Valera Fernández, et al. 2021).

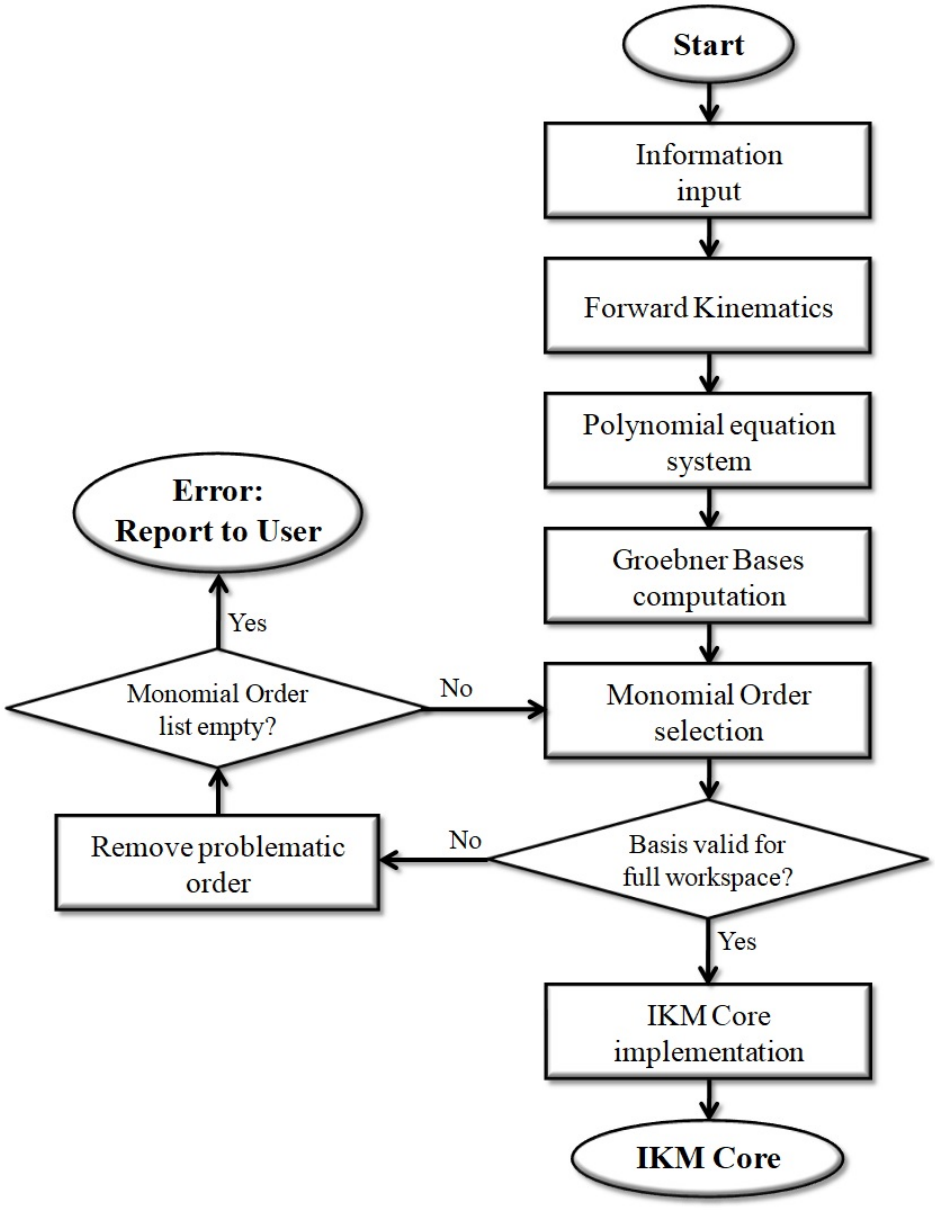


**Figure 3.2:** Flowchart of the developed procedure for the synthesis of the IKM Core module

### 3.2.2 Second Step: Forward Kinematics

The second step of the procedure, as is shown Figure 3.2, is the calculation of the robot's Forward Kinematics, using as inputs the D-H parameters that were provided in the first step. This calculation is done by applying the Denavit-Hartenberg method (Guzmán-Giménez, Valera Fernández, et al. 2020; Atique, Sarker, and Ahad 2018; Fu, Gonzalez, and Lee 1987). The result of this second step is the homogeneous transformation matrix ($^0A_n$) between the origin of the coordinate system in the robot's base and the one in its end effector.

When the developed procedure is applied to the hexapod's leg, this step's output is the homogeneous transformation matrix presented in Equation 3.1 (Guzmán-Giménez, Valera Fernández, et al. 2020). In this equation, the term $q_{23}$ is equal to $q_2 - q_3$.

$$^0A_3 = \begin{bmatrix} \cos(q_1)\sin(q_{23}) & -\cos(q_1)\cos(q_{23}) & -\sin(q_1) & \cos(q_1)[L_1 + L_2\cos(q_2) + L_3\sin(q_{23})] \\ \sin(q_1)\sin(q_{23}) & -\sin(q_1)\cos(q_{23}) & \cos(q_1) & \sin(q_1)[L_1 + L_2\cos(q_2) + L_3\sin(q_{23})] \\ -\cos(q_{23}) & -\sin(q_{23}) & 0 & L_2\sin(q_2) - L_3\cos(q_{23}) \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.1}$$

### 3.2.3 Third Step: Obtention of the Polynomial Equation System

Once the robot's Forward Kinematics is calculated, the procedure's third step, which is identified in Figure 3.2 as "Polynomial equation system", begins by extracting the robot's kinematic equations from the homogeneous transformation matrix obtained in the second step. For simplicity, in this work we are only interested in the position of the end effector. However, the procedure can be extended to also include the end effector's orientation, to complete its whole pose (Guzmán-Giménez, Valera Fernández, et al. 2021).

Continuing with the case of the hexapod's leg, the extracted position equations are the ones presented in Equation 3.2, which correspond to the first three elements of the fourth column of Equation 3.1.

$$\begin{cases} p_x = \cos(q_1)[L_1 + L_2\cos(q_2) + L_3\sin(q_2 - q_3)] \\ p_y = \sin(q_1)[L_1 + L_2\cos(q_2) + L_3\sin(q_2 - q_3)] \\ p_z = L_2\sin(q_2) - L_3\cos(q_2 - q_3) \end{cases} \tag{3.2}$$

Equation 3.2 conforms a trigonometric equation system that describes the end effector's position as a function of the current state of the robot's actuators. The previous equation system is completed with trigonometric identities of the form $\sin(q_i)^2 + \cos(q_i)^2 = 1$ for all the robot's rotational degrees of freedom (DoFs). Then, all the trigonometric expressions are expanded, and variable substitutions of the form $\sin(q_i) = s_i$ and $\cos(q_i) = c_i$ are applied.

This step's objective is to obtain a polynomial equation system where the variables are either a pair sine-cosine of a rotational DoF ($s_i$ and $c_i$), or directly the position value of a prismatic DoF ($q_j$).

For the hexapod's leg, this step's output is the polynomial equation system presented in Equation 3.3, where the input parameters are the three components of the position vector of the leg's end point ($\vec{\mathbf{p}}$), represented by $p_x$, $p_y$, and $p_z$. The six variables of Equation 3.3 that should be solved are $s_1$, $c_1$, $s_2$, $c_2$, $s_3$, and $c_3$.

$$\begin{cases} 28c_1 + 58c_1c_2 + 110c_1s_2c_3 - 110c_1c_2s_3 - p_x = 0 \\ 28s_1 + 58s_1c_2 + 110s_1s_2c_3 - 110s_1c_2s_3 - p_y = 0 \\ 58s_2 - 110c_2c_3 - 110s_2s_3 - p_z = 0 \\ s_1^2 + c_1^2 - 1 = 0 \\ s_2^2 + c_2^2 - 1 = 0 \\ s_3^2 + c_3^2 - 1 = 0 \end{cases} \tag{3.3}$$

The polynomial equation system obtained as the third step's output is the Ideal generator set that will be used for the IKM's synthesis (Cox, Little, and O'Shea 2015).

When the procedure is applied to the PUMA manipulator, this step's output is the polynomial equation system presented in Equation 3.4. As was the case of the hexapod's leg, the six variables that should be solved in this equation are $s_1$, $c_1$, $s_2$, $c_2$, $s_3$, and $c_3$.

$$\begin{cases} -a_2 s_1 c_2 - d_4 s_1 c_2 c_3 + d_4 s_1 s_2 s_3 - a_3 s_1 s_2 c_3 - a_3 s_1 c_2 s_3 - d_2 c_1 - p_x = 0 \\ a_2 c_1 c_2 + d_4 c_1 c_2 c_3 - d_4 c_1 s_2 s_3 + a_3 c_1 s_2 c_3 + a_3 c_1 c_2 s_3 - d_2 s_1 - p_y = 0 \\ d_1 - a_2 s_2 - d_4 s_2 c_3 - d_4 c_2 s_3 + a_3 c_2 c_3 - a_3 s_2 s_3 - p_z = 0 \\ s_1^2 + c_1^2 - 1 = 0 \\ s_2^2 + c_2^2 - 1 = 0 \\ s_3^2 + c_3^2 - 1 = 0 \end{cases} \tag{3.4}$$

### *3.2.4 Fourth Step: Groebner Bases computation*

The polynomial equation system shown in Equation 3.3 constitutes an Ideal generator basis, which will be the starting point for the Groebner Basis calculation (Buchberger 2001). As was explained in Section 2.2, the solution set of the calculated Groebner Basis is also the solution of the original basis, the one that generated the Ideal (Cox, Little, and O'Shea 2015). Therefore, the objective of this step is to obtain a Groebner Basis that is easier to solve than the original polynomial equation system (the output of the procedure's third step), in this way simplifying the process of solving the IKP.

Before proceeding to the calculation of the Groebner Basis, it is important to acknowledge one restriction of the implementation of Faugère's F4 algorithm that is used to compute this basis: all the constant parameters of the input equation system must be integer numbers. To circumvent this restriction, the developed procedure analyzes the parameters that conform the Ideal calculated in the third step, searching for a multiplying factor that will make all those parameters integer numbers, and then multiplies all the constant parameters by this computed factor. The multiplying factor will also be applied to the inputs of the synthesized IKM, in order to preserve all the measurement units.

Up to this point, the procedure has one of the two important elements for a Groebner Basis calculation, the Ideal generator basis, but needs the other one: the monomial order. As was presented in Section 2.3, the monomial order establishes the order in which the variables are solved in the calculated Groebner Basis. Different monomial orders will produce different bases, but it is important to remember that all the possible lex order combinations will produce an array of Groebner Bases, whose solutions are all the same solution set of the Ideal generator basis. Therefore, solving any of those Groebner Bases will also solve the polynomial equation system obtained in the procedure's third step (Cox, Little, and O'Shea 2015).

The only difference between obtaining the solution from the Groebner Basis generated by one specific lex order over another one, would be on the computation times required to solve each basis. So the optimal monomial order is the one that requires the least computation time to find its solution.

The main objective of the procedure's fourth step, identified in Figure 3.2 as "Groebner Bases computation", is to establish the lex orders that are relevant to the analyzed robot, and then calculate their associated Groebner Bases. The analysis of those bases' computations times will be done in the fifth step, along with the selection of the optimal monomial order.

The first task of this fourth step is to recognize all the possible lex order combinations that can be relevant for the analyzed system. The variables that should be solved in the polynomial equation system are the ones related with the rotational DoFs, that come in pairs of the form $s_i$ and $c_i$, and the variables of the prismatic DoFs. For the case of the hexapod's leg, shown in Equation 3.3, the monomials of the Ideal are the variables $s_1$, $c_1$, $s_2$, $c_2$, $s_3$, and $c_3$, so the optimal lex order must be a combination of those variables.

All the possible combinations of the Ideal monomials will be equal to $(n+r)!$, where $n$ is the total amount of DoFs of the analyzed robot, and $r$ is the quantity of those DoFs that are rotational. But all those combinations include the orders that separate the two trigonometric variables of a rotational DoF ($s_i$ and $c_i$). These combinations are not relevant lex orders, because the two variables of a rotational DoF are strongly related, and should always be adjacent in the lex order. This restriction greatly reduces the amount of possible lex orders, but it can be further reduced if a relative order is established between the two variables of each rotational DoF (either $s_i > c_i$ or $c_i > s_i$). This way the total amount of relevant lex orders gets reduced to $n!$, independently if the robot's DoFs are rotational or prismatic.

To impose the aforementioned restriction, the developed procedure analyzes which of the trigonometric variables of every rotational DoF is less likely to be zero. This is done because, when solving the Groebner Basis, if the lex order is of the form "$... > c_i > s_i > ...$", the value of $s_i$ ($sin(q_i)$) will be calculated first, while the computation of $c_i$ will surely depend on the calculated value of $s_i$. Therefore, if $s_i$ is equal to zero, it is very probable that a term of the basis' equation that defines $c_i$ will be canceled out, which will surely generate an indetermination in the computation of $c_i$.

So, by analyzing the probable values of $sin(q_i)$ and $cos(q_i)$ in the range of the rotational DoF ($q_i$), the procedure detects the trigonometric variable that is

less likely to be zero, and orders these two variables properly inside the lex order. This analysis is done by calculating the expected value of $|sin(q_i)|$ and $|cos(q_i)|$, following the expressions shown in Equation 3.5.

$$\begin{aligned} E[|sin(q_i)|] &= \int_{q_{i_{lo}}}^{q_{i_{up}}} |sin(q_i)| \cdot p(q_i) \cdot dq_i \\ E[|cos(q_i)|] &= \int_{q_{i_{lo}}}^{q_{i_{up}}} |cos(q_i)| \cdot p(q_i) \cdot dq_i \end{aligned} \tag{3.5}$$

In Equation 3.5, $q_{i_{lo}}$ and $q_{i_{up}}$ are the lower and upper limits of the range of the actuator $q_i$, and $p(q_i)$ is the probability density function (pdf) of $q_i$. This pdf may be supplied by the user, if the probable distribution of the positions of $q_i$ in all its range is known. Otherwise, the developed procedure assumes a Gaussian pdf, that covers all the rotational DoF's range, with its mean in the middle of this range. If $E[|sin(q_i)|] > E[|cos(q_i)|]$, then the relative order of the couple of variables related to $q_i$ is selected as $c_i > s_i$. Otherwise, this relative order is established as $s_i > c_i$. This way the variable that is less likely to be zero is always computed first.

Figure 3.3 shows the graphical representation of this order selection process when applied to the three actuators of the hexapod's leg. In each of the graphs presented in Figure 3.3, the black dots outline the Gaussian pdf chosen for the corresponding actuator. These Gaussians were automatically selected by the procedure, because in this case the user did not provide any information regarding the probable distribution of positions among these DoFs. The mean of each of those Gaussian pdfs is the middle point of the movement range of its corresponding DoF (see 1.3), while the variance of each Gaussian was selected in such a way that six of its standard deviations cover said movement range, three on each side of the middle point.

The blue areas shown in Figure 3.3 represent $E[|cos(q_i)|]$, while the red areas equate $E[|sin(q_i)|]$. In all cases $E[|cos(q_i)|] > E[|sin(q_i)|]$, so the selected orders for these couples are $s_1 > c_1$, $s_2 > c_2$ and $s_3 > c_3$, and they will always appear that way in all the possible lex orders for the hexapod's leg.

Table 3.1 summarizes all the information presented in Figure 3.3, showing also the relative order selected for the trigonometric variables related with the hexapod's DoFs. The relative orders shown in Table 3.1 are absolutely logical, because the movement ranges of these three DoFs have their centers close to the angular position of 0 radians. Therefore it is more probable that $sin(q_i) = 0$, rather than any of the variables related with the corresponding cosines.

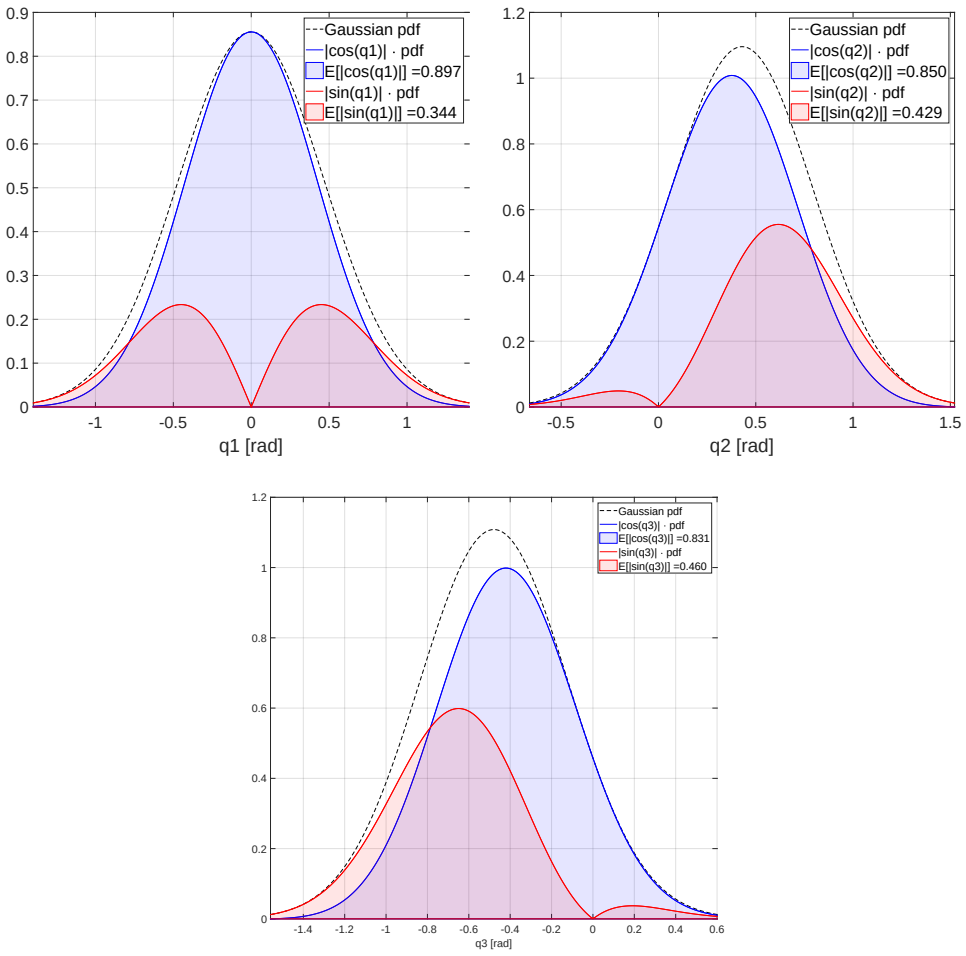


**Figure 3.3:** Graphical representation of the expected values for the variables related with all three DoFs of the hexapod's leg. The black dotted line show the chosen pdf for each DoF. The blue areas represent the $E[|cos(q_i)|]$, while the red areas are equal to $E[|sin(q_i)|]$.

**Table 3.1:** Expected values for the trigonometric variables related with the rotational DoFs of the hexapod's leg. The columns titled "$\mathbf{E}[|\mathbf{cos}(\mathbf{q_i})|]$" and "$\mathbf{E}[|\mathbf{sin}(\mathbf{q_i})|]$" contain the expected values for the corresponding trigonometric variables. The largest expected value of each DoF is marked in **bold**, and the selected order for each rotational DoF is shown in the last column.

| Rotational DoF | $\mathbf{E}[\lvert\mathbf{cos}(\mathbf{q_i})\rvert]$ | $\mathbf{E}[\lvert\mathbf{sin}(\mathbf{q_i})\rvert]$ | Relative order |
|---|---|---|---|
| $q_1$ | **0.897** | 0.344 | $s_1 > c_1$ |
| $q_2$ | **0.850** | 0.429 | $s_2 > c_2$ |
| $q_3$ | **0.831** | 0.460 | $s_3 > c_3$ |

**Table 3.2:** Relevant lex orders for the hexapod's leg.

| N° | Lexicographic Order |
|---|---|
| 1 | $s_1 > c_1 > s_2 > c_2 > s_3 > c_3$ |
| 2 | $s_1 > c_1 > s_3 > c_3 > s_2 > c_2$ |
| 3 | $s_2 > c_2 > s_1 > c_1 > s_3 > c_3$ |
| 4 | $s_2 > c_2 > s_3 > c_3 > s_1 > c_1$ |
| 5 | $s_3 > c_3 > s_1 > c_1 > s_2 > c_2$ |
| 6 | $s_3 > c_3 > s_2 > c_2 > s_1 > c_1$ |

It is important to keep in mind that the computation of the relative orders shown in Table 3.1 is only required for the rotational DoFs of the analyzed robot, to establish the order in which the pair of variables $s_i$ and $c_i$ will appear inside all the relevant lex orders. The three DoFs of the hexapod's leg are rotational, so the procedure establishes a relative order for all three. The same happens for the case of the PUMA robot (see Figure 1.5) and most of the industrial manipulators, whose DoFs are all rotational. In the case that the analyzed robot has a combination of rotational and prismatic DoFs, only the rotational ones require this relative order calculation.

Once all the relative orders of the rotational DoFs are established, the developed procedure calculates all the relevant lex orders for the analyzed system. To continue with the hexapod's leg example, Table 3.2 contains the relevant lex orders for this robotic system. The relevant amount of lex orders is six, because the hexapod's leg contains three DoFs ($3! = 6$).

After establishing all the relevant lex orders, the developed procedure proceeds to calculate the Groebner Bases related to those orders. These calculations are executed in a two step process: First, an initial Groebner Basis is obtained using Faugère's F4 algorithm (Jean Charles Faugère 2010; Jean Charles Faugère 1999), with a graded reverse lexicographic order (grevlex). After this first basis is calculated, a series of FGLM basis conversions, based on the algorithm developed by J. C. Faugère et al. (1993), are made to convert the original basis to all the lex orders previously identified as relevant. This way a set of $n!$ different bases is calculated, one for each of the relevant lex orders. The procedure's next step is the selection of the optimal monomial order.

### *3.2.5 Fifth Step: Automatic monomial order selection*

Once all the Groebner Bases are calculated, the procedure's fifth step, represented in Figure 3.2 as "Monomial Order selection", selects the basis related with the optimal monomial order. As was stated in Section 3.2.4, the optimal monomial order is the one that provides the Groebner Basis that requires the least computation time to obtain the solution. So, in order to find this optimal monomial order, the following three-step classification criterion is applied:

1. Lowest computational cost of the highest degree equation
2. Lowest accumulated computational cost of all the basis' equations
3. Lowest accumulated cost of all the equations' coefficients

The first step of the classification process consists on identifying the Groebner Bases whose highest degree equation presents the lesser computational time, and discard the rest.

Table 3.3 presents all the possible types of polynomial equations that can be found in the previously calculated Groebner Bases. The maximum possible degree of the polynomial equations shown in Table 3.3 is four, because the IKP of the positioning of non-redundant open-chain robotic systems has at most four solutions (Guzmán-Giménez, Valera Fernández, et al. 2020).

The linear and quadratic equations that our procedure encounters are solved by the well known algorithms for these types of polynomials. The bi-quadratic polynomial equations are solved as two concatenated quadratic equations. Finally, the quartic equations that the developed procedure may encounter are solved using Salzer's algorithm (Salzer 1960). It is important to highlight that it is not necessary to prepare for the appearance of cubic equations, because

**Table 3.3:** Possible types of polynomial equations found in the calculated Groebner Bases

| Type | Degree | Polynomial equation |
|---|---|---|
| Linear | 1 | $a_1x + a_0 = 0$ |
| Quadratic | 2 | $a_2x^2 + a_1x + a_0 = 0$ |
| Bi-Quadratic | 4 | $a_4x^4 + a_2x^2 + a_0 = 0$ |
| Quartic | 4 | $a_4x^4 + a_3x^3 + a_2x^2 + a_1x + a_0 = 0$ |

**Table 3.4:** Computational cost required to solve different types of polynomial equations on an ARM Cortex-M4 CPU (ARM Limited 2009).

| Type | | Operations | | | | | Cost [Cycles] |
|---|---|---|---|---|---|---|---|
| | | adml | div | sqrt | trig | atan | |
| Linear | | 1 | 1 | 0 | 0 | 0 | 15 |
| Quadratic | | 7 | 2 | 1 | 0 | 0 | 49 |
| Bi-Quadratic | | 9 | 2 | 3 | 0 | 0 | 79 |
| Quartic | min | 68 | 4 | 3 | 0 | 0 | 166 |
| | max | 80 | 5 | 5 | 1 | 1 | 282 |

the solutions of the IKP of open-chain robotic systems always come in pairs (Fu, Gonzalez, and Lee 1987).

A list of all the operations required to solve the polynomial equations previously shown is compiled in Table 3.4. The last column of Table 3.4 contains the computational cost, in clock cycles, of every type of polynomial equation, assuming that the microcontroller in charge of the robot's control has an ARM Cortex-M4 CPU (ARM Limited 2009). Table 3.5 breaks down the cost in clock cycles of each operation for the selected microcontroller. If the robot has a control system with a different type of CPU, the user can change the operations' cost in the initial settings of the procedure, in order to reflect the real computation times on the robot's CPU.

In Table 3.4, the quartic polynomial equation encompasses two rows, identified as "min" and "max", because Salzer's algorithm can take different paths, depending on the type of roots of the analyzed polynomial equation (Salzer 1960). In this case the computational cost is the average between the mini-

**Table 3.5:** Computational cost, in clock cycles, for a microcontroller with an ARM Cortex-M4 CPU (ARM Limited 2009).

| Operation | Identifier | Cost [Cycles] |
|---|---|---|
| Addition or Multiplication | adml | 1 |
| Division | div | 14 |
| Square root | sqrt | 14 |
| Trigonometric operation | trig | 29 |
| Arctangent (atan2) | atan | 33 |

mum cost for the quartic equation and its maximum, which is equal to 224 clock cycles for an ARM Cortex-M4 CPU.

Using the costs shown in the last column of Table 3.4, the procedure selects the Groebner Bases whose highest degree equation has the lower computational cost, thus quickly discarding all the bases that would require a larger amount of time and resources to be solved. It is possible that more than one basis passes this first selection criterion, so the second criterion is applied to all these remaining bases.

The second classification criterion consists on selecting the bases with the lower accumulated computational cost from all their equations. The accumulated cost of each basis is equal to the sum of the computational cost of all the polynomial equations that compose that basis, using again Table 3.4 to evaluate these costs. This second criterion is only passed by those bases with the lowest accumulated computational cost.

Once again, more than one basis can progress beyond the second criterion, in which case the third one is applied. This third classification criterion searches for the basis with the lowest accumulated cost from all its equation's coefficients. This last accumulated cost is quantified by adding all the clock cycles required to calculate all the coefficients in all the basis' equations. Therefore, after previously discarding all the bases whose equation types would require larger amounts of time and resources to be solved, the procedure narrows down the list of Groebner Bases to those which effectively have the least computation time.

If after these three classification criteria, there is still more than one candidate for the optimal lex order, then any of the orders related with these remaining

**Table 3.6:** Selection of the lex order for the Hexapod's IKM. After each step of the selection process, the discarded orders are marked in *italic*. A value of "-" indicates that the lex order was discarded in a previous step. The first classification criterion is to search for those lex orders whose basis' highest degree equation present the least computation time. The second one seeks for the bases with the lesser accumulated cost of all their equations, while the last one searches for the lowest amount of time required to compute all the basis' coefficients. All these costs, expressed in clock cycles, were calculated for a microcontroller with an ARM Cortex-M4 CPU (ARM Limited 2009). The selected lex order is marked in **bold**.

| | **Classification Criteria** | | |
|---|---|---|---|
| **Order N°** | **Highest Deg. [Cycles]** | **Acc. Cost [Cycles]** | **Coefficients [Cycles]** |
| 1 | *79* | - | - |
| 2 | *224* | - | - |
| 3 | *79* | - | - |
| **4** | **49** | **158** | **103** |
| 5 | *224* | - | - |
| 6 | 49 | 158 | *201* |

bases may be selected, because all the bases that pass the three criteria will surely present similar computation times.

Continuing with the case of the hexapod's leg, Table 3.6 compiles the results obtained when applying the presented three-step classification criterion to the set of six Groebner Bases calculated in the fourth step of the procedure. The lex orders 1, 2, 3 and 5 were discarded after the first step of the selection process, leaving only orders 4 and 6 active. Both remaining orders passed successfully the second step, but in the third one, lex order 6 was eliminated, setting lex order 4 as the chosen one.

The outcome of the selection process shown in Table 3.6 is that lex order 4 is the selected monomial order for the final Groebner Basis. This final basis will be employed for the synthesis of the hexapod's IKM Core.

The output of the procedure's fifth step is the Groebner Basis calculated with the selected lex order. For the case of the hexapod's leg, this Groebner Basis is the set of polynomial equations presented in Equations (3.6) to (3.11):

$$-p_x^2+[p_x^2+p_y^2]c_1^2=0 \tag{3.6}$$

$$-p_y c_1 + p_x s_1 = 0 \tag{3.7}$$

$$\begin{aligned} p_x^5 + p_x p_y^4 + p_x p_z^4 - 29360 p_x p_z^2 - 26224(p_x^3 + p_x p_y^2) + \\ + 52684800 p_x + 2(p_x^3 p_y^2 + p_x^3 p_z^2 + p_x p_y^2 p_z^2) + \\ + [1644160(p_x^2 + p_y^2) - 112(p_x^4 + p_y^4 + p_x^2 p_z^2 + p_y^2 p_z^2) - 224 p_x^2 p_y^2] c_1 + \\ + 162817600 p_x c_3^2 = 0 \end{aligned} \tag{3.8}$$

$$p_x^3 + p_x p_y^2 + p_x p_z^2 - 14680 p_x - 56[p_x^2 + p_y^2] c_1 + 12760 p_x s_3 = 0 \tag{3.9}$$

$$\begin{aligned} 28(p_x^5 + p_x p_y^4 + p_x p_z^4) + 56(p_x^3 p_y^2 + p_x^3 p_z^2 + p_x p_y^2 p_z^2) + \\ + 200704(p_x^3 + p_x p_y^2 - p_x p_z^2) - 174562304 p_x + \\ + [10304(p_x^4 + p_y^4) + 20608 p_x^2 p_y^2 - p_x^6 - p_y^6 - 4 p_x^2 p_y^2 p_z^2 - 3(p_x^4 p_y^2 + p_x^2 p_y^4) + \\ + 7168(p_x^2 p_z^2 + p_y^2 p_z^2) - p_x^2 p_z^4 - p_y^2 p_z^4 - 2(p_x^4 p_z^2 + p_y^4 p_z^2) - 7463680(p_x^2 + p_y^2)] c_1 + \\ + [12760(p_x^3 p_z + p_x p_z^3 + p_x p_y^2 p_z) + 10003840 p_x p_z] c_3 + 714560[p_x^2 p_z + p_y^2 p_z] c_1 c_3 + \\ + [116(p_x^5 + p_x p_y^4 + p_x p_z^4) + 232(p_x^3 p_y^2 + p_x^3 p_z^2 + p_x p_y^2 p_z^2) + \\ + 181888(p_x p_z^2 - p_x^3 - p_x p_y^2) + 71300096 p_x] c_2 = 0 \end{aligned} \tag{3.10}$$

$$\begin{aligned} 10304(p_x^3 p_z + p_x p_y^2 p_z) - p_x^5 p_z - p_x p_z^5 - p_x p_y^4 p_z + 6234368 p_x p_z + \\ + 7168 p_x p_z^3 - 2(p_x^3 p_z^3 + p_x^3 p_y^2 p_z + p_x p_y^2 p_z^3) + 489216[p_x^2 p_z + p_y^2 p_z] c_1 + \\ + [357280(p_x p_z^2 - p_x^3 - p_x p_y^2) + 280107520 p_x] c_3 + \\ + [10003840(p_x^2 + p_y^2) - 12760(p_x^4 + p_y^4 + p_x^2 p_z^2 + p_y^2 p_z^2) - 25520 p_x^2 p_y^2] c_1 c_3 + \\ + [116(p_x^5 + p_x p_y^4 + p_x p_z^4) + 232(p_x^3 p_y^2 + p_x^3 p_z^2 + p_x p_y^2 p_z^2) + \\ + 181888(p_x p_z^2 - p_x^3 - p_x p_y^2) + 71300096 p_x] s_2 = 0 \end{aligned} \tag{3.11}$$

As can be seen in the previous equations, the obtained Groebner Basis constitutes a triangular equation system that can be solved in a staggered way (Guzmán-Giménez, Valera Fernández, et al. 2020; Uchida and McPhee 2012). This is because the first equation of the system, shown in Equation 3.6, only depends on one variable, the lesser monomial in the selected lex order. After that equation is solved, the second one has at most two variables, the one that was previously calculated in the first equation and a new one, while the third

depends at most on three variables, two computed in the preceding equations and a new one, and so on. Solving this triangular equation system gives the solution for the all the variables of the original polynomial equation system (Guzmán-Giménez, Valera Fernández, et al. 2020).

The algorithm that solves this triangular equation system will be implemented in the seventh step of the developed procedure. But first it is important to check if the basis generated by the selected monomial order is valid for the full workspace of the robotic system, which constitutes the procedure's sixth step.

### 3.2.6 Sixth Step: Basis checkup

This checkup step is done to analyze if the selected basis has a possibility to fall into a false singularity. These false singularities happen when the leading coefficient of one the basis' polynomial equations is equal to zero, which should indicate that the robot's position is on a singular point of its mechanical structure, but the robotic system is not really on a singular point, nor near one. Section 4.2 shows how this problem with false singularities may affect one of the possible Groebner Basis calculated for the hexapod's leg.

To prevent this problem, the developed procedure checks the leading coefficient (LC), i.e. the highest degree coefficient, of all the polynomials that integrate the selected basis, analyzing the conditions in which these coefficients are equal to zero. Each of the conditions that cancel any of the basis' LC are verified with the determinant of the Jacobian matrix of the robot's mechanical structure.

The determinant of this Jacobian matrix is equal to zero if the robot's position is really on one of the singular points of its mechanical structure. Equation 3.12 presents the Jacobian matrix ($J$) that the procedure calculates for a robot with three Degrees of Freedom (DoFs), $p_x$, $p_y$ and $p_z$, and three actuators $q_1$, $q_2$ and $q_3$, just like the analyzed hexapod's leg.

$$J = \begin{bmatrix} \frac{\partial p_x}{\partial q_1} & \frac{\partial p_x}{\partial q_2} & \frac{\partial p_x}{\partial q_3} \\ \frac{\partial p_y}{\partial q_1} & \frac{\partial p_y}{\partial q_2} & \frac{\partial p_y}{\partial q_3} \\ \frac{\partial p_z}{\partial q_1} & \frac{\partial p_z}{\partial q_2} & \frac{\partial p_z}{\partial q_3} \end{bmatrix} \tag{3.12}$$

Therefore, the developed procedure uses the expressions of $p_x$, $p_y$ and $p_z$ calculated in the third step (Equation 3.2 presents the case of the hexapod's leg), computes the corresponding partial derivatives shown in Equation 3.12, evaluates the determinant of this Jacobian matrix ($|J|$), and finally checks if $|J|$ is

equal to zero in all the conditions that lead to a cancellation of a LC. If there is a situation in which any of the LC is canceled but $|J|$ is not equal to zero (or at least close to zero by a margin of $1 \times 10^{-3}$), then that basis is marked as problematic.

If this basis checkup does not find any problem with the selected basis, then the procedure advances to the seventh step: the IKM Core implementation. In the case that the basis is marked as problematic, then the procedure removes from the list of relevant monomial orders the order that generated this problematic basis, and checks if this list is empty after the removal (see Figure 3.2). If the list is still not empty, the procedure goes back to the fifth step, the one identified as "Monomial Order selection", where a new monomial order is selected from the remaining ones. This sequence of steps and checks is repeated until a basis successfully passes the checkup, or the list of relevant orders is emptied.

In the event that this list is emptied, an error flag is raised and the user is notified of this situation. But this extreme case has not been reached in any of the test benches used in this work, nor in all the other robotic systems on which this procedure has been tested.

### *3.2.7 Seventh Step: IKM Core implementation*

The objective of the procedure's last step, identified in Figure 3.2 as "IKM Core implementation", is to transform the triangular polynomial equation system of the selected lex order's basis into an algorithm that implements the IKM Core module.

The IKM Core's algorithm will always be the one presented in Figure 3.4. This algorithm has three main sections: Initialization, Main Loop and Wrap-up.

*Initialization section*

In this first section of the IKM Core's algorithm, enclosed by a red rectangle in Figure 3.4, all the variables are initialized and the required multiplying factor is applied to the position input. This multiplying factor might be required if the robot's kinematic parameters were previously adjusted, i.e. multiplied by a certain factor, to comply with the restrictions that the used implementation of the F4 algorithm impose to the basis' coefficients. This way all the measurement units are preserved, both in the robot's kinematic parameters and in the position inputs.

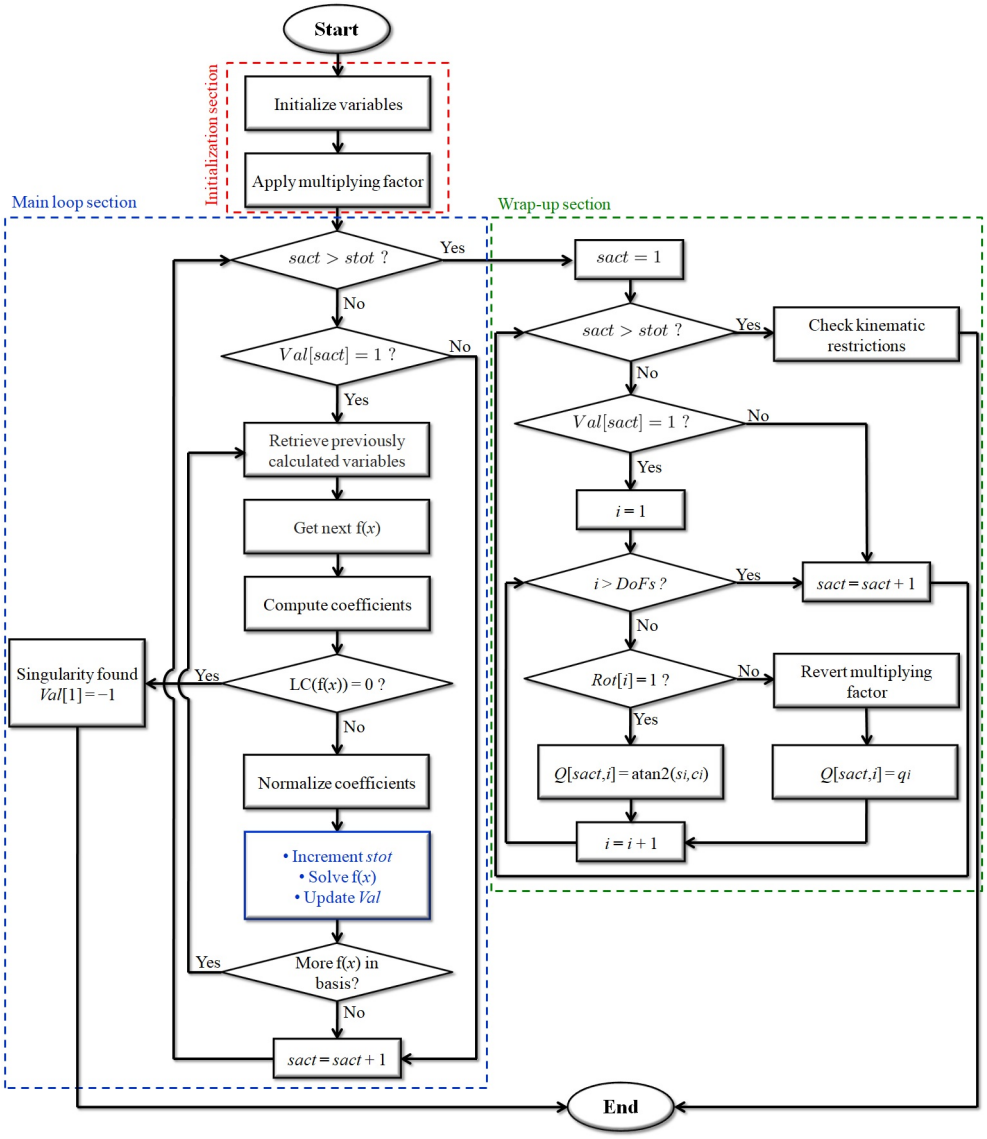


**Figure 3.4:** IKM Core's algorithm

At the end of the algorithm, the corresponding inverse factor will be applied to all the outputs related with linear positions, in order to preserve the proper measurement units of the outputs offered by the IKM Core. This operation is not necessary for those outputs that represent an actuators' angular position.

*Main Loop section*

After the initialization is completed, comes the algorithm's main loop, which is enclosed in Figure 3.4 by a blue rectangle. This section solves the basis calculated at the end of the procedure's fifth step, taking into account that this basis will surely have multiple solutions. It is important to always bear in mind that the main objective of this IKM Core's algorithm is to offer as output all the valid solutions of the robot's Inverse Kinematic Problem.

To achieve this objective, the Main Loop section is actually conformed by two nested loops: the inner loop runs through all the equations of the basis, while the outer one go through all the possible solutions, but without knowing beforehand the total amount of these possible solutions. This is achieved with the help of two inner variables, *sact* and *stot*. The variable *sact* counts the current solution that the algorithm is calculating, while *stot* indicates the total number of different solutions that have been found. Both variables are initialized at the value of "1", to signal that the algorithm is currently calculating the first possible solution, and that so far only one solution is known.

During the execution of this section's inner loop, the algorithm will probably encounter equations of a degree equal or greater than two, which are the cause of the existence of multiple solutions. In this case, the algorithm will increase the value of *stot* by the proper amount: +1 if the encountered equation has degree 2, by +3 if its degree is 4, and so on. The value of *sact* will only be increased when the inner loop ends running through all the equations of the basis. Given this setup, the exit condition of the outer loop is reached when $sact > stot$ (see Figure 3.4), signaling that the equation system has been thoroughly solved, i.e. all the possible solutions were found. This way the user does not need to know the total amount of solutions of the IKP, because this information comes out naturally during the execution of the algorithm.

The inner loop of the algorithm's Main Loop section starts at the step titled "Retrieve previously calculated variables" (see Figure 3.4), and begins by executing the following five steps:

1. Retrieve previously calculated variables: All the previously calculated variables are retrieved from the registers, because they will surely take part in the computation of the next equation's coefficients.

2. Get next equation: The next equation that has to be solved is fetched from the equation system.

3. Compute coefficients: The equation's coefficients, which depend entirely on the IKM inputs, the robot's kinematic parameters and the previously calculated variables (all known values), are computed.

4. Check if the LC is not equal to zero: If the LC is equal to zero, then the IKM has found a singular point. If this situation occurs, the first element of the vector that indicates which solutions are valid, "Val[1]", is overwritten with a value of -1 to flag this singularity, and the execution of the algorithm comes to an end (see Figure 3.4).

5. Normalize coefficients: All the coefficients of the polynomial equation are divided by the LC.

After these five steps are completed, the equation that was fetched from the equation system is ready to be solved. At this point the algorithm executes the step that is highlighted in blue in Figure 3.4, which depends on the degree of the fetched equation ($f(x)$):

Linear equation: After the coefficient normalization applied in this section's fifth step, all the linear equations will have the structure presented in Equation 3.13.

$$x + b_0 = 0 \qquad \because\ b_0 = \frac{a_0}{a_1} \tag{3.13}$$

In Equation 3.13, $x$ is the equation's variable that has to be solved, while $b_0 = a_0/a_1$, i.e. the division of the independent term of the original equation and its first degree coefficient. This simple equation is solved by a rearrangement of terms.

In this case there is no possibility of obtaining a solution that is out of the robot's workspace, so the $Val$ vector, which contains the information of the valid solutions, remains untouched.

Quadratic equation: The structure of the quadratic equations will be the one shown in Equation 3.14.

$$x^2 + b_1\, x + b_0 = 0 \qquad \because\ b_i = \frac{a_i}{a_2} \quad \forall i = \{0, 1\} \tag{3.14}$$

The quadratic equations have two solutions, so the IKM Core's algorithm increases by one the total amount of solutions that have been found so far ($stot = stot + 1$). This new solution is initially marked as valid in the $Val$ vector, but this may change during the execution of the algorithm.

The IKM Core solves the quadratic equations using the algorithm shown in Figure 3.5.

As shown in Figure 3.5, the IKM Core's algorithm computes first the discriminant of the fetched equation. If the discriminant is negative, then the solutions of this second degree equation will be a complex-conjugate pair, which signals that the desired solution is out of the robot's workspace. In this case, the current solution is marked as invalid in the $Val$ vector, as well as the new solution that was created.

On the other hand, if the discriminant is positive, then both solutions are real, therefore they are both possible solutions. This does not mean yet that the solutions are inside the robot's workspace. It only states that there exist possible configurations that are valid for the position input, but a restriction on the movement range of an actuator could still invalid any possible solution. All these movement restrictions are checked at the end of the "Wrap-up Section", as the final step of the IKM Core's algorithm.

After the discriminant is calculated, the quadratic equation is solved using the well-known algorithm for this type of the equations.

Bi-quadratic equation: After the coefficient normalization applied in this section's fifth step, the bi-quadratic equations have the structure presented in Equation 3.15.

$$x^4 + b_2\, x^2 + b_0 = 0 \qquad \because\ b_i = \frac{a_i}{a_4} \quad \forall i = \{0, 2\} \tag{3.15}$$

The bi-quadratic polynomial equations are solved as the concatenation of a quadratic equation followed by a squared variable, following the algorithm shown in Figure 3.6.

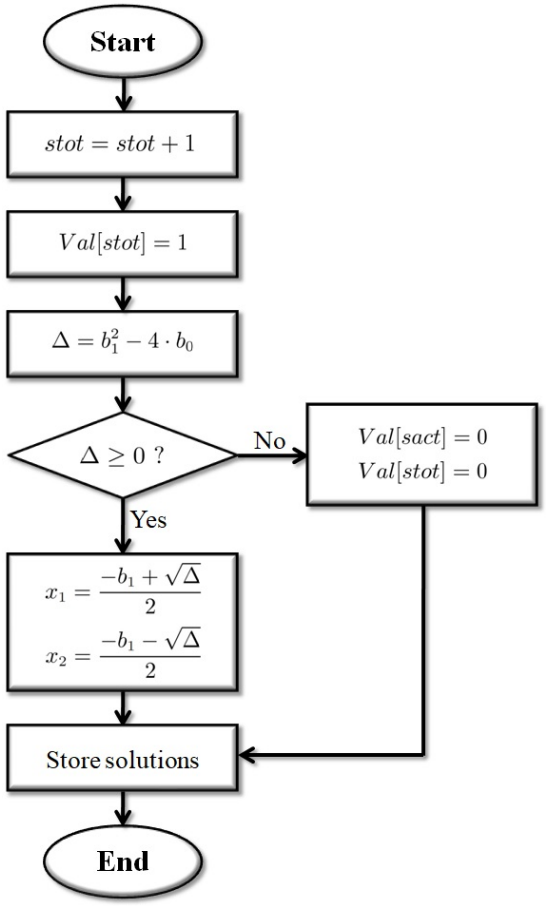


**Figure 3.5:** Algorithm used to solve quadratic equations

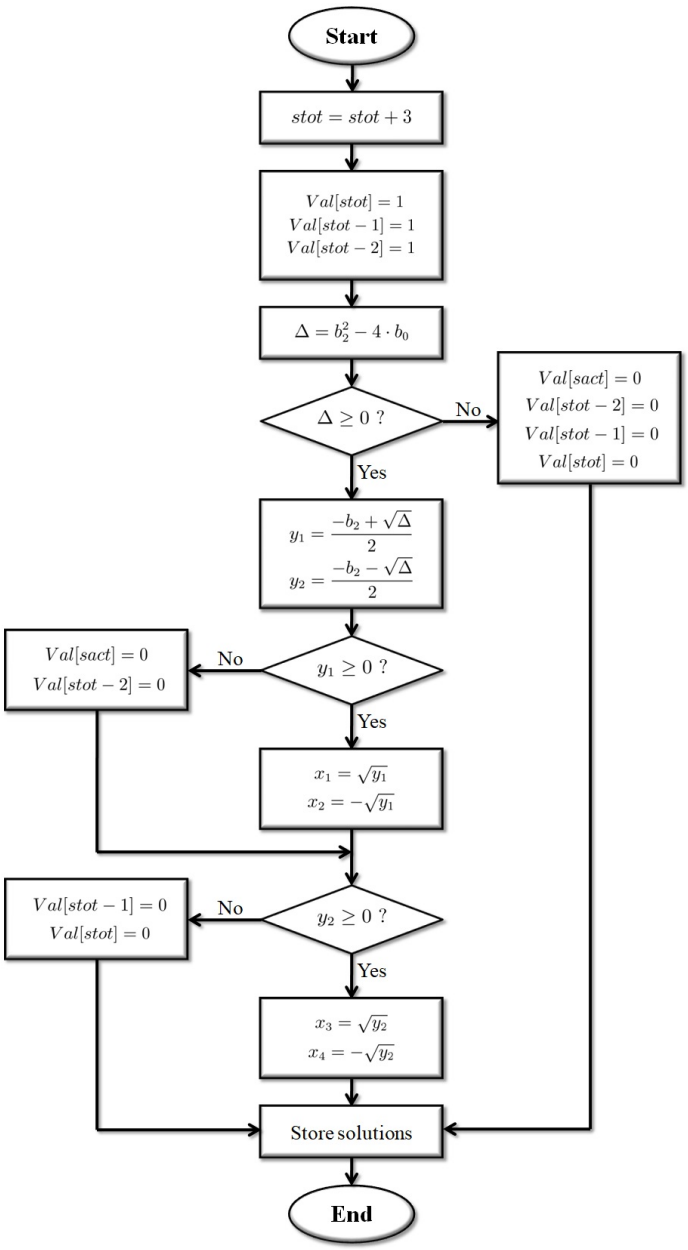


**Figure 3.6:** Algorithm used to solve bi-quadratic equations

The algorithm of Figure 3.6 begins by increasing by three the total amount of solutions that have been found so far ($stot = stot+3$), because all bi-quadratic equations have four solutions. As in the case of the quadratic equations, all the new solutions are initially marked as valid in the $Val$ vector.

The algorithm continues computing the discriminant of the quadratic polynomial shown in Figure 3.6. If the discriminant is negative, then all the four possible solutions of the bi-quadratic equation are invalid, which is reflected in the corresponding positions of the $Val$ vector.

If this discriminant is positive, then both quadratic equation's solutions are treated as squared variables, each of which is responsible for a pair of solutions of the original bi-quadratic polynomial. If one of the solutions of the quadratic equation is negative, then their corresponding pair of solutions will be marked as invalid, because the square root of this negative number will never give a real value. If it is positive, then a pair of solutions of the bi-quadratic polynomial equation are found.

Quartic equation: These equations have the structure shown in Equation 3.16.

$$x^4 + b_3\,x^3 + b_2\,x^2 + b_1\,x + b_0 = 0 \qquad \because\ b_i = \frac{a_i}{a_4} \quad \forall i = [0,3] \tag{3.16}$$

To solve quartic polynomial equations, the developed procedure uses the algorithm presented in Figure 3.7, which is based on the algorithm developed by Salzer 1960.

The algorithm shown in Figure 3.7 has a step, highlighted in green, in which a cubic equation must be solved. The algorithm used to solve this cubic is presented in Figure 3.8.

In Figure 3.8, the term $\Delta_3$ is the discriminant of the cubic equation, which is calculated using the expression shown in Equation 3.17.

$$\Delta_3 = (18{\cdot}c_2{\cdot}c_1{\cdot}c_0) - (4{\cdot}{c_2}^3{\cdot}c_1) + ({c_2}^2{\cdot}{c_1}^2) - (4{\cdot}{c_1}^3) - (27\cdot{c_0}^2) \tag{3.17}$$

The algorithm shown in 3.7 allows to solve all quartic polynomial equations without requiring any operation with complex numbers. This is one of the contributions of this work, because the IKMs synthesized by the developed procedure do not request any kind of complex number operations from the robot's microcontroller.

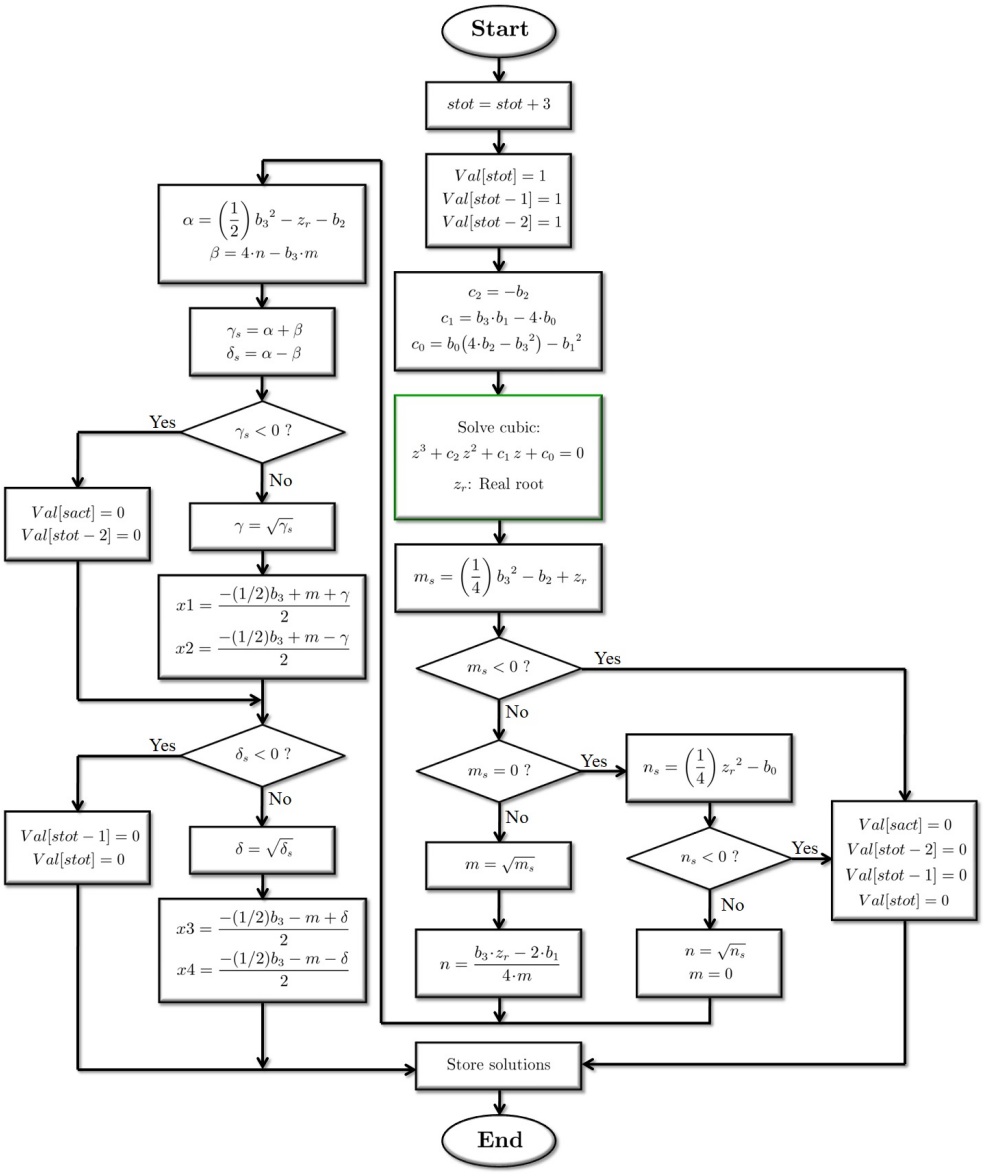


**Figure 3.7:** Algorithm used to solve quartic polynomial equations

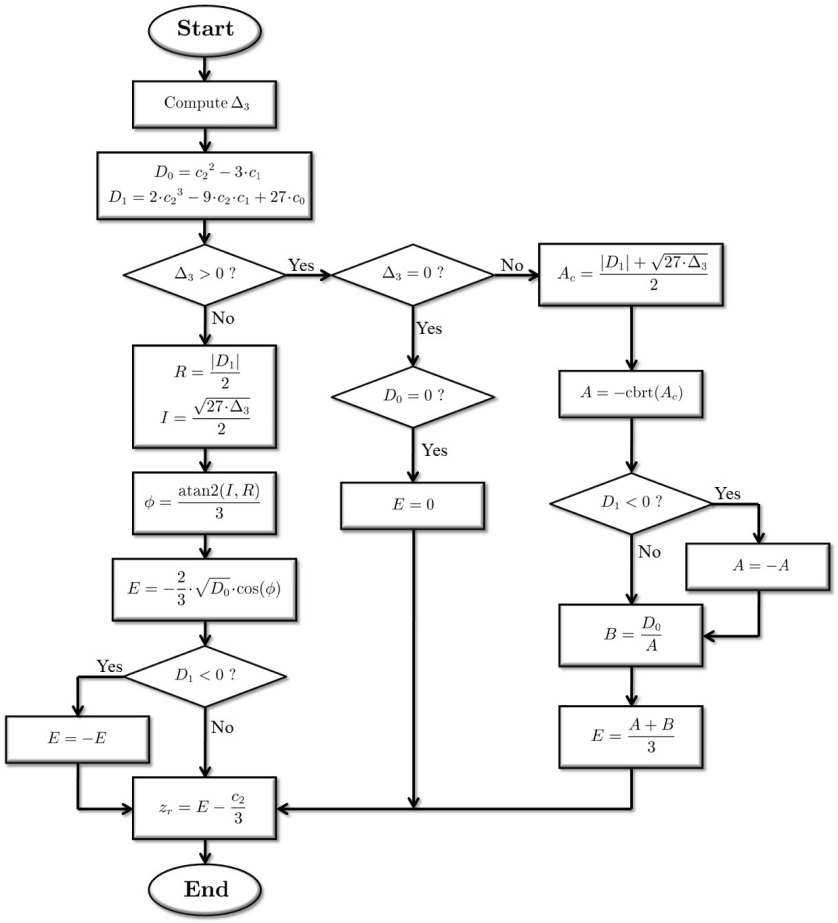


**Figure 3.8:** Algorithm used to calculate one real root of the cubic obtained during a quartic's equation resolution

After the fetched equation is solved, all the valid solutions are stored in their corresponding positions in the registers, and the main loop checks if all the equations of the basis have been solved. If there is any remaining equation, then the IKM Core's algorithm returns to the first step of this Main Loop's inner loop, the one titled "Retrieve previously calculated variables". Otherwise, *sact* is increased by one, to signal that the current set of solutions has been finished, and the algorithm returns to the step in which it is checked if $sact > stot$, i.e. if all the possible solutions of the basis have been found.

If the variable *sact* is still not greater than *stot*, then a new round of the inner loop begins. Otherwise, the IKM Core's algorithm passes to the last section: the Wrap-up Section.

*Wrap-up section*

The final section of the IKM Core's algorithm is the Wrap-up, enclosed in Figure 3.4 by a green rectangle. This last section organizes the set of valid solutions calculated in the main loop, and prepares the two outputs of the IKM Core module: $Q$, which is an array that contains all the found solutions for the robot's IKP, and $Val$, the vector that indicates which of those solutions are valid.

As was stated before, the solution of the Groebner Basis will be a set composed either by the variables of prismatic DoFs ($q_j$), pairs sine-cosine of rotational DoFs ($s_i$ and $c_i$), or a combination of both. Therefore, in order to prepare the IKM Core's outputs, the algorithm runs through all the set of found solutions, identifying each type of solution. If a solution is a pair sine-cosine of a rotational DoF, then the final value of this DoF is computed using the expression shown in Equation 3.18.

$$q_i = \text{atan2}\left(s_i, c_i\right) \tag{3.18}$$

On the other hand, if the solution is the position of a prismatic DoF ($q_j$), then the algorithm checks if a multiplying factor was applied in the Initialization section and reverts its effects, dividing by the same factor. This way we preserve the relationship between the measurement units of the position inputs and the units of the calculated prismatic DoFs. It is important to bear in mind that this compensation of the multiplying factor is not necessary for the rotational DoFs.

Finally, the Wrap-up section of the IKM Core's algorithm ends by checking all the robot's kinematic restrictions: if one of the found solutions, that is still valid at this point, does not meet any of these kinematic restrictions, then that solution is marked as invalid.

After the algorithm of the IKM Core is prepared, the developed procedure proceeds to configure the State Estimator (see Figure 3.1).

## 3.3 State Estimator

The function of the State Estimator is to select the solution offered by the IKM Core that better fits the projection of the previous state. It's important to keep in mind that the IKM Core module offers all the possible solutions for the robot's position in the configuration space, but only one of these solutions keeps the motion of the robot in a smooth way. Therefore, the selection made by this module is essential for the smoothness of all the robot's movements.

The inputs of the State Estimator, as it can be seen in Figure 3.1, are all the variables that conform the robot's previous state, i.e. the positions, velocities and accelerations of all the robot's actuators that were calculated in the previous execution cycle, as well as the two arrays that the IKM Core offers as outputs: One of these arrays contains all the possible solutions ($Q$), while the other indicates which solutions are valid ($Val$). The task of this module is to select the valid solution that better guarantees the smoothness of the robot's movements.

To achieve its task, the State Estimator first estimates the possible current state for each of the robot's actuators. This estimation is done by calculating the relation shown in Equation 3.19 for every one of the robot's actuators, $q_i$.

$$\hat{q}_i(t) = q_i(t-\Delta t) + \dot{q}_i(t-\Delta t)\cdot\Delta t + \ddot{q}_i(t-\Delta t)\cdot\frac{(\Delta t)^2}{2} \quad \forall i \in [1,n] \tag{3.19}$$

The relation presented in Equation 3.19 corresponds to the second-degree Taylor polynomial expansion of the position of actuator $q_i$. The previous polynomial, which also models an accelerated movement in the configuration space, contains a known interval of time, $\Delta t$, between the previous state, $q_i(t-\Delta t)$, and the current one that is being estimated, $\hat{q}_i(t)$.

After the state estimation is done, the State Estimator computes the euclidean distance between the estimated state and all the valid solutions offered by

the IKM Core. These distances are calculated in the configuration space by applying Equation 3.20 to all the solutions marked as valid.

$$d_j(\vec{q}^j(t), \hat{\vec{q}}(t)) = \sqrt{\sum_{i=1}^{n}(q_i^j(t) - \hat{q}_i(t))^2} \quad \Longleftarrow \text{ Val}[j] = 1 \tag{3.20}$$

In Equation 3.20, $q^j$ corresponds to the j-th solution offered by the IKM Core, while $Val[j]$ indicates if that j-th solution is valid or not ($Val[j] = 1$ means that this j-th solution is valid).

Finally, the State Estimator selects the valid solution that is closest to the estimated state, $\hat{q}$, i.e. the one that presents the smallest $d_j$. This solution with the smallest distance is the one that is closest to the expected state. Therefore, that must be the position's solution that better guarantees the smoothness of the robot's movement.

This selected solution becomes the position's solution for the IKP, which is transferred as input to the IKM Derivatives module, and also to the registers, for storage.

The full algorithm of the State Estimator is summarized in Figure 3.9.

The State Estimator produces four outputs:

1. $\vec{q}(t)$: The position solution selected by this module.
2. $\iota$: The identifier of the selected solution. This is required by the "IKM Derivatives" module.
3. Singularity Flag: This is a flag that gets activated if the desired position is on one of the robot's singular points.
4. OoW flag: Out of workspace flag. This flag is activated is the desired position is out of the robot's workspace.

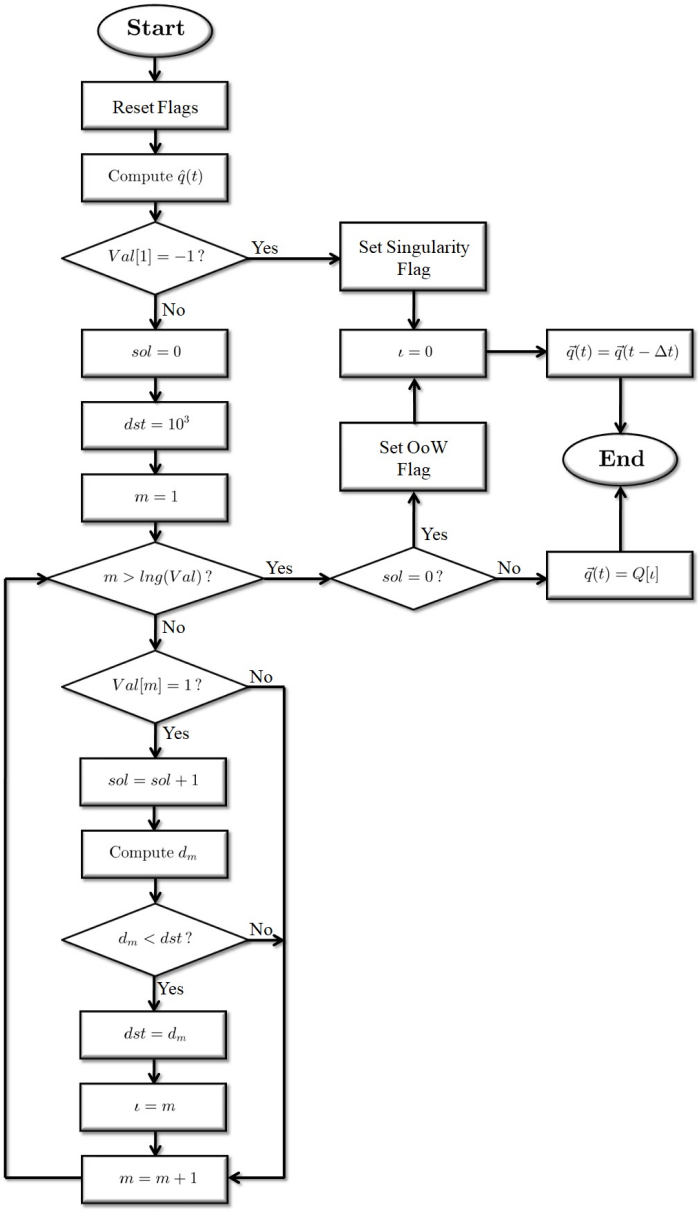


**Figure 3.9:** State estimator's algorithm

## 3.4 IKM Derivatives

The function of the last module that conforms the IKM structure, identified in Figure 3.1 as "IKM Derivatives", is to calculate the velocities and accelerations of all the robot's actuators, that is, the derivatives of the position's solution selected by the State Estimator.

The inputs of this module are the desired values, in cartesian space, of the robot's position, velocity and acceleration, as well as the identifier of the solution selected by the State Estimator ($\iota$). The outputs are the solutions for the velocities and accelerations of the all the robot's actuators that, together with the position's solution, complete the robot's new calculated state.

The required velocities and accelerations can be obtained through the calculation of the time derivatives of the basis' equations. Continuing with the case of the hexapod's leg, Equations 3.21 to 3.24 show the first order time derivatives of the first four equations of its corresponding basis.

$$\{2c_1(p_y^2+p_x^2)\}\dot{c}_1+[2{c_1}^2(p_y\dot{p}_y+p_x\dot{p}_x)-2p_x\dot{p}_x]=0 \tag{3.21}$$

$$\{p_x\}\dot{s}_1+[s_1\dot{p}_x-c_1\dot{p}_y-p_y\dot{c}_1]=0 \tag{3.22}$$

$$\begin{aligned}&\{325635200p_xc_3\}\dot{c}_3+\\&+[(-112(p_x^4+p_y^4-p_x^2p_z^2-p_y^2p_z^2-2p_x^2p_y^2)+1644160(p_x^2+p_y^2))\dot{c}_1+\\&+(5p_x^4+4p_x^3p_z+6p_x^2p_z^2+6p_x^2p_y^2+2p_y^2p_z^2+p_y^4+p_z^4+1600(101761c_3^2+32928)+\\&-16((4917p_x^2+1639p_y^2+1835p_z^2)+14p_xc_1(2p_y^2+p_z^2+2p_x^2-14680)))\dot{p}_x+\\&+(4p_xp_y(p_x^2+p_y^2+p_z^2)-52448p_xp_y-224c_1p_y(2p_x^2+p_z^2+2p_y^2-14680))\dot{p}_y+\\&+(4p_xp_z(p_y^2+p_z^2)-224c_1p_z(p_x^2+p_y^2)-58720p_xp_z)\dot{p}_z]=0\end{aligned} \tag{3.23}$$

$$\begin{aligned}\{12760p_x\}\dot{s}_3+[(12760s_3-112p_xc_1+3p_x^2+p_y^2+p_z^2-14680)\dot{p}_x+\\+2p_y(p_x-56c_1)\dot{p}_y+2p_xp_z\dot{p}_z-56\dot{c}_1(p_x^2+p_y^2)]=0\end{aligned} \tag{3.24}$$

In Equations 3.21 to 3.24, the variables that should be solved are $\dot{c}_1$, $\dot{s}_1$, $\dot{c}_3$ and $\dot{s}_3$, i.e. the first order time derivatives of the variables calculated by the IKM Core. Those variables previously computed by the IKM Core, as well as

the desired position ($p_x$, $p_y$ and $p_z$) and its time derivatives ($\dot{p}_x$, $\dot{p}_y$ and $\dot{p}_z$), are now inputs to the algorithm executed in this IKM Derivatives module. All these first order term derivatives are linear polynomial equations, regardless of the degree of the original polynomial equation, in which the leading term coefficient is surrounded by curly brackets ( $\{\bullet\}$ ), while the constant term is marked with square brackets ( $[\bullet]$ ). These equations are easily solved by applying the same method presented in 3.2.7.

After all these first order time derivatives are computed, the velocities of the robot's actuators can be calculated with Equation 3.25, which is the first order time derivative of the atan2 equation used at the end of the IKM Core's algorithm to compute the position of each one of these actuators (see Equation 3.18).

$$\dot{q}_m(t) = \frac{d}{dt}\big(q_m(t)\big) = \frac{d}{dt}\big(\text{atan2}(s_m(t), c_m(t))\big) = \frac{c_m(t)\dot{s}_m(t) - s_m(t)\dot{c}_m(t)}{c_m(t)^2 + s_m(t)^2} \tag{3.25}$$

In the same way, the accelerations of all the robot's actuators can be calculated by solving the second order time derivatives of the basis's equations, and finally applying the second order time derivative of atan2 shown in 3.26.

$$\ddot{q}_m(t) = \frac{d^2}{dt^2}\big(q_m(t)\big) = \\ \frac{c_m(t)\ddot{s}_m(t) - s_m(t)\ddot{c}_m(t)}{c_m(t)^2 + s_m(t)^2} - \frac{(c_m(t)\dot{s}_m(t) - s_m(t)\dot{c}_m(t))(2c_m(t)\dot{c}_m(t) + 2s_m(t)\dot{s}_m(t))}{(c_m(t)^2 + s_m(t)^2)^2} \tag{3.26}$$

The final algorithm of the IKM Derivatives is shown in Figure 3.10. This algorithm is very similar to the one used by the IKM Core, because all the computations of the derivatives are done starting from the calculated Groebner Basis.

In Figure 3.10, the block named "Compute $\dot{q}_m(t)$" refers to time derivative of $q_m(t)$, presented in Equation 3.25. In the same way, the block named "Compute $\ddot{q}_m(t)$" calculates the second order time derivative of $q_m(t)$, whose expression was shown in Equation 3.26.

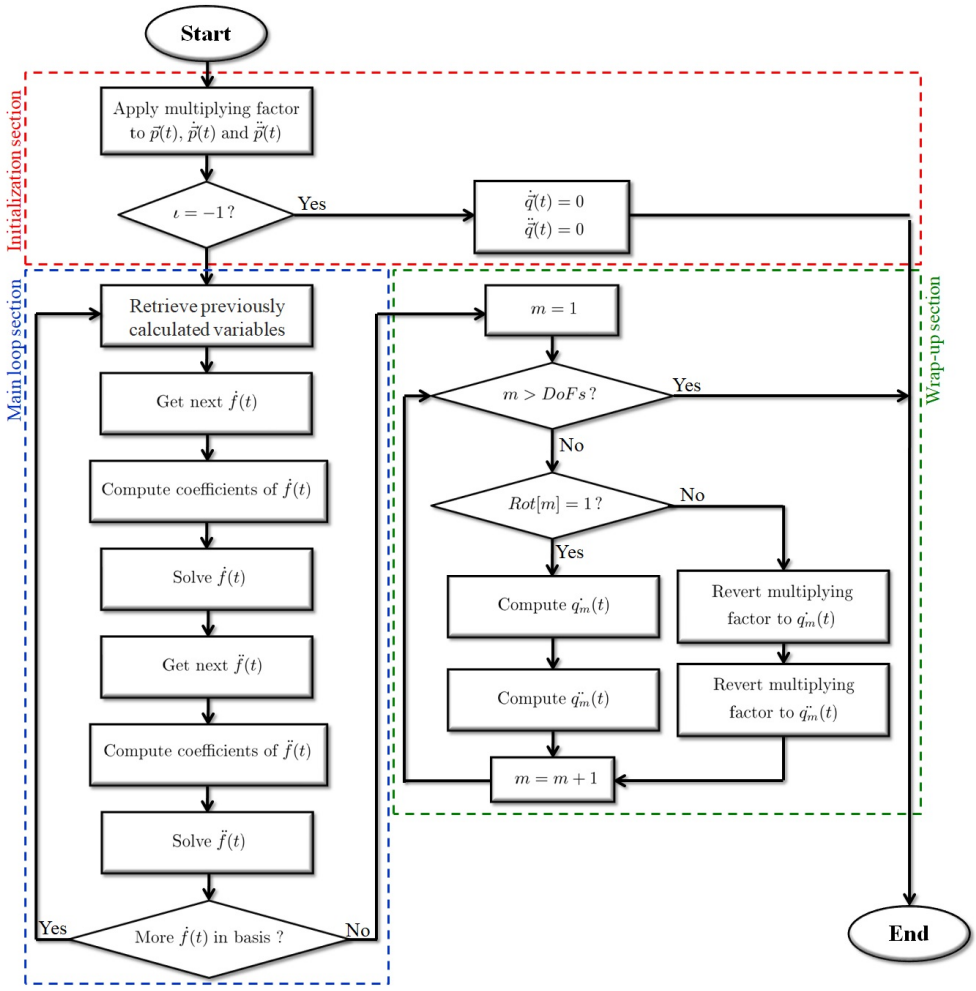


**Figure 3.10:** IKM Derivative's algorithm

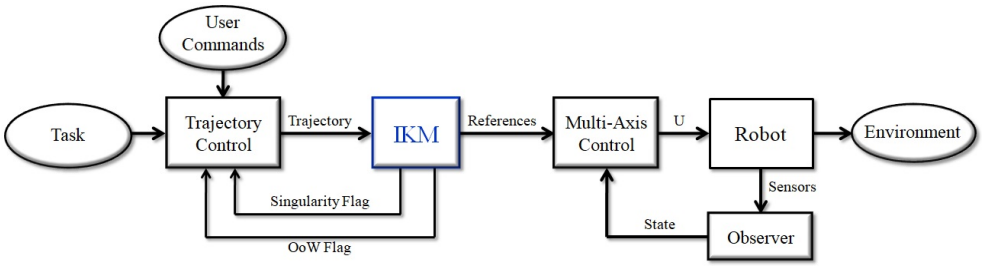


**Figure 3.11:** Synthesized IKM in the robot's control system.

## 3.5 Registers

The registers presented in Figure 3.1 store the previous state of the robot, as well as all the ongoing calculations for the current state. The previous state, which that is stored in the registers, is composed of the positions, velocities and accelerations of the robot's actuators in the configuration space. These values are used by the State Estimator, presented in Section 3.3, to decide which of the solutions offered by the IKM Core is the appropriate one for the current state. The reset signal, identified in Figure 3.1 with the label "RST", loads the initial state values in the section of the registers that is in charge of storing the robot's previous state.

The registers shown in Figure 3.1 also store all the variables related with the calculations of the robot's current state. These variables are shared between two modules: the "IKM Core" and the "IKM Derivatives" module.

After all the four modules shown in Figure 3.1 are prepared and configured, the final step of the developed procedure is to synthesize the final IKM, which is compiled as C++ code or as a MATLAB® script, depending on the initial selections made by the procedure's user.

The synthesized IKM can be used directly in the robot's control system, by connecting it as presented in Figure 3.11. The control scheme shown in this last figure is similar to the one presented in Figure 1.1, but with the addition of the two warning flags, the "Singularity Flag" and the "OoW Flag", which are used by the synthesized IKM to signal any possible problems to the Trajectory Control. The "Singularity Flag" is activated if the desired position falls on a singular point of the mechanical structure, while the "OoW Flag" indicates if this desired position is out of the robot's workspace.

The developed procedure was used to synthesize the IKMs of two open-chain robots: the previously mentioned BH3-R walking hexapod and a PUMA 560 manipulator (see Section 1.3). The performance of these two IKMs is presented in the next chapter.

# Chapter 4

# Performance Analysis of the Developed Procedure

*The developed procedure was used to synthesize the IKMs of two non-redundant open-chain robotic systems: a PUMA manipulator and a walking hexapod robot. This chapter begins presenting the traditional methods used to solve the Kinematic Problem of open-chain robotic systems, then proceeds to analyze the performance of the IKMs synthesized by the developed procedure. The performance of both IKMs was compared with the kinematics models calculated by traditional methods, finding in all cases that they are totally comparable, both in precision and computation time.*

The procedure presented in Section 3.2 was used to synthesize the IKMs of a leg of the BH3-R walking hexapod (Figure 1.2) and the PUMA 560 manipulator (Figure 1.4). As was previously explained, the inputs of the developed procedure are the D-H parameters of the analyzed robot and the movement range of its actuators, while the output is the synthesized IKM, both in C++ and in MATLAB® script. This procedure also selects automatically the optimal monomial order of the final Groebner Basis that will constitute the IKM, so the user does not need to have any further information about the robot.

But before proceeding to the performance analysis of the IKMs that were synthesized by the developed procedure, it is important to study the traditional methods normally employed to solve the IKP of open-chain robotic systems.

These traditional methods were used to synthesize the reference models, with which the performance of the synthesized IKMs was compared.

## 4.1 Resolution of the Kinematic Problem by Traditional Methods

The first step for the resolution of the Kinematic Problem by any of the traditional methods is to calculate the Forward Kinematics of the analyzed robot.

### *4.1.1 Forward Kinematics*

The Forward Kinematics of any robotic system can be easily solved by applying the Denavit-Hartenberg's (D-H) method (Atique, Sarker, and Ahad 2018; Fu, Gonzalez, and Lee 1987). The Forward Kinematics of the hexapod's leg was already presented in Equation 3.1, which shows the homogeneous transformation between the origin of the hexapod's leg and its final point.

In the case of the PUMA 560, for simplicity's sake we will focus only on the first three links of the robotic arm, just before the robot's wrist. This simplification is valid because the last three links of the arm comply with the conditions of Pieper's Theorem (Rodriguez et al. 2018; Fu, Gonzalez, and Lee 1987), effectively conforming an in-line wrist. Therefore, the end effector of the PUMA 560 will be considered to be at the anchor point of this in-line wrist, just at coordinate system number 4, shown in Figure 1.5.

Applying the Denavit-Hartenberg's method, with the parameters presented in Table 1.4, the solution to the Forward Kinematics problem of the PUMA 560 is the homogeneous transformation shown in Equation 4.1, which establishes the transformation between the base of the robot and the anchor point of its in-line wrist. In Equation 4.1, $c_1$ and $s_1$ are equal to $\cos(q_1)$ and $\sin(q_1)$, respectively, while $c_2$ and $s_2$ correspond to $\cos(q_2)$ and $\sin(q_2)$. The term $c_{32}$ is equal to $\cos(q_3+q_2)$, while $s_{32}$ is $\sin(q_3+q_2)$.

$$
{}^{0}A_4 = \begin{bmatrix} s_1 s_{32} & s_1 c_{32} & -c_1 & -s_1[a_2 c_2 + d_4 c_{32} + a_3 s_{32}] - d_2 c_1 \\ -c_1 s_{32} & -c_1 c_{32} & -s_1 & c_1[a_2 c_2 + d_4 c_{32} + a_3 s_{32}] - d_2 s_1 \\ -c_{32} & s_{32} & 0 & d_1 - a_2 s_2 - d_4 s_{32} + a_3 c_{32} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.1}
$$

### 4.1.2 Inverse Kinematics Using Traditional Methods

As was mentioned in Section 1.1, the techniques most commonly used to solve the IKP of open-chain robotic systems are two: the geometric method and the analytical procedure. The geometric method was used to solve the IKP of the PUMA manipulator, while the analytical procedure was employed for the case of the walking hexapod.

*Inverse Kinematics of the PUMA 560 by the Geometric Method*

Based on the link coordinate systems presented in Figure 1.5, and inspired in the geometry of a human arm, various arm configurations can be identified for a PUMA robot with the assistance of three configuration indicators: ARM, ELBOW, and WRIST (Fu, Gonzalez, and Lee 1987). The first two, ARM and ELBOW, are associated with the solution of the first three joints, while WRIST is related to the solution of the last three. For a six-axis manipulator, like the PUMA 560, there are four possible solutions for the first three joints ($q_1$, $q_2$ and $q_3$), commonly referred as configurations, and for each one of these configurations, there are two possible solutions for the last three joints ($q_4$, $q_5$ and $q_6$) (Fu, Gonzalez, and Lee 1987).

As was stated in Section 4.1.1, we will focus exclusively in the first three joints of the PUMA robotic arm. Thus, we only need to find the four possible solutions for $q_1$ to $q_3$, which are related to the first two configuration indicators: ARM and ELBOW.

The ARM indicator can have two possible values, LEFT and RIGHT. LEFT indicates that positive values of $q_2$ will move the robot's wrist in the negative $\mathbf{z_0}$ direction when the third joint, $q_3$, is not activated, while RIGHT will move it in the positive $\mathbf{z_0}$ direction. ELBOW also has two possible values, ABOVE and BELOW, which indicate whether the third joint of the robot is above the imaginary line described between the top part of the robot's body and its wrist, or below this line, respectively. The combination of the two possible values of the ARM indicator together with the two of ELBOW gives the four possible configurations of the PUMA's arm (up to its wrist), which constitute the four solutions of its first three joints ($q_1$, $q_2$ and $q_3$).

If the position vector of the PUMA's wrist, $\vec{p}(p_x, p_y, p_z)$, is projected onto the $\mathbf{x_0} - \mathbf{y_0}$ plane, as shown in Figure 4.1, $q_1$ could be obtained by solving the equations shown in Equation 4.2. This projection of the PUMA's wrist is the

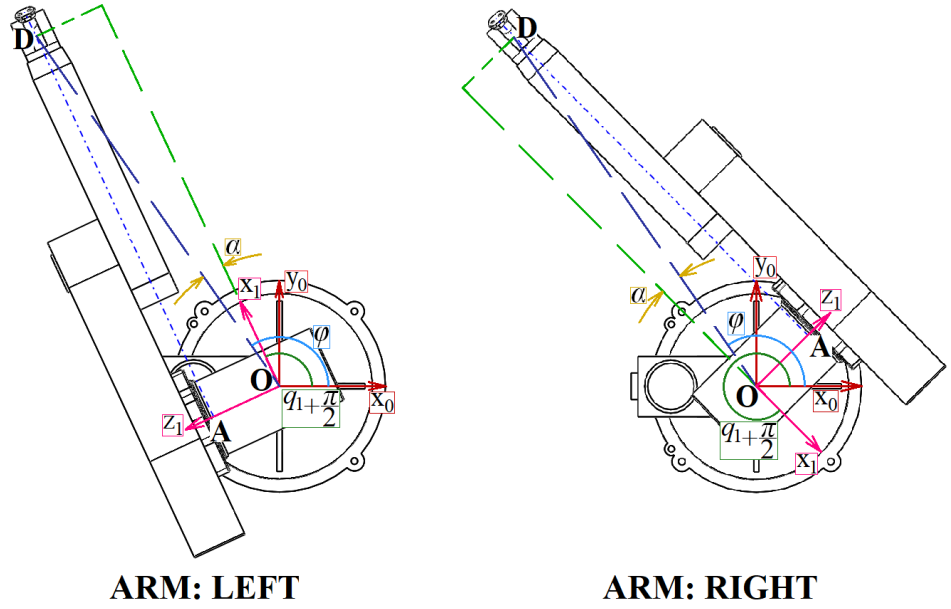


**Figure 4.1:** Geometric solution for the PUMA's first joint ($q_1$), showing both possible configurations: "LEFT Arm" (left) and "RIGHT Arm" (right)

point marked as "**D**" in Figure 4.1. The "$L$" and "$R$" superscripts added to $q_1$ differentiate between the LEFT and RIGHT arm configurations.

$$\begin{gathered}
\vec{\mathbf{OA}} = d_2 \\
\vec{\mathbf{OD}} = R_2 = \sqrt{(p_x)^2 + (p_y)^2} \\
\vec{\mathbf{AD}} = r_a = \sqrt{(R_2)^2 - (d_2)^2} = \sqrt{(p_x)^2 + (p_y)^2 - (d_2)^2} \\
\alpha = \operatorname{atan2}(d_2, r_a) \\
\varphi = \operatorname{atan2}(p_y, p_x) \\
q_1^L + \frac{\pi}{2} = \varphi - \alpha \implies q_1^L = \varphi - \alpha - \frac{\pi}{2} \\
q_1^R + \frac{\pi}{2} - \varphi - \alpha = \pi \implies q_1^R = \varphi + \alpha + \frac{\pi}{2}
\end{gathered} \tag{4.2}$$

In order to calculate $q_2$ and $q_3$, the position vector of the PUMA's wrist, $\vec{p}$, is projected onto the plane $\mathbf{x_2} - \mathbf{y_2}$, as shown in Figure 4.2. In this figure,

the plane $\mathbf{x_1} - \mathbf{y_1}$ is parallel to $\mathbf{x_2} - \mathbf{y_2}$, and the point "**D**" represents the position of the robot's wrist from coordinate system number 1. This position can be computed using the expression shown in Equation 4.3, where ${}^0A_1(q_1^i)$ is the transformation matrix between coordinate system number 1 and the one located at the robot's base, while the terms $p_{x_1^i}$, $p_{y_1^i}$, and $p_{z_1^i}$ represent the projection of the wrist's position vector over the $\mathbf{x_1}$, $\mathbf{y_1}$, and $\mathbf{z_1}$ axes, respectively. The transformation matrix shown in Equation 4.3 depends on the ARM configuration, so the superscript "$i$" indicates if ARM is on the LEFT configuration ("L") or the RIGHT one ("R").

$$\begin{bmatrix} p_{x_1^i} \\ p_{y_1^i} \\ p_{z_1^i} \\ 1 \end{bmatrix} = \left({}^0A_1(q_1^i)\right)^{-1} \cdot \begin{bmatrix} p_x \\ p_y \\ p_z \\ 1 \end{bmatrix} \tag{4.3}$$

The geometric equations presented in Equation 4.4 can be derived from the diagram shown in Figure 4.2.

$$\begin{gathered}
\vec{\mathbf{AB}} = a_2 \qquad \vec{\mathbf{BC}} = a_3 \qquad \vec{\mathbf{CD}} = d_4 \\
\vec{\mathbf{BD}} = r_3 = \sqrt{(a_3)^2 + (d_4)^2} \\
\vec{\mathbf{AD}} = r_1^i = \sqrt{(p_{x_1^i})^2 + (p_{y_1^i})^2} \\
\gamma = \operatorname{atan2}\left(|a_3|, d_4\right) \\
\cos(\beta^i) = \frac{(a_2)^2 + (r_3)^2 - (r_1^i)^2}{2 \cdot a_2 \cdot r_3} \\
\sin(\beta^i) = \sqrt{1 - (\cos(\beta^i))^2} \\
\beta^i = \operatorname{atan2}\left(\sin(\beta^i), \cos(\beta^i)\right) \\
\varepsilon^i = \operatorname{atan2}\left(p_{y_1^i}, p_{x_1^i}\right) \\
\cos(\delta^i) = \frac{(a_2)^2 + (r_1^i)^2 - (r_3)^2}{2 \cdot a_2 \cdot r_1^i} \\
\sin(\delta^i) = \sqrt{1 - (\cos(\delta^i))^2} \\
\delta^i = \operatorname{atan2}\left(\sin(\delta^i), \cos(\delta^i)\right)
\end{gathered} \tag{4.4}$$

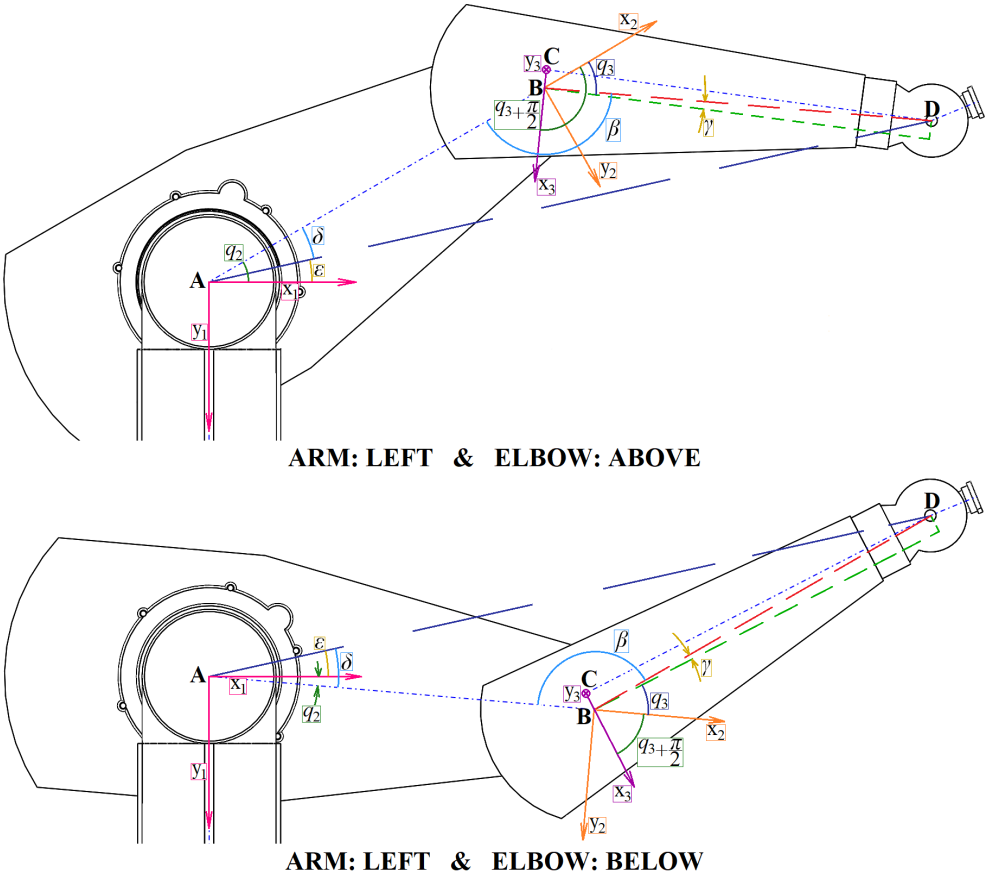


**Figure 4.2:** Geometric solution for the PUMA's second ($q_2$) and third ($q_3$) joints, showing two of the possible four configurations: "LEFT and ABOVE" (top) and "LEFT and BELOW" (bottom).

From the geometric relations computed previously, $q_3$ can be calculated using the expressions shown in Equation 4.5, where the superscripts "$LA$", "$LB$", "$RA$" and "$RB$" indicate the value of the ARM and ELBOW indicators.

$$\begin{aligned} q_3^{LA} &= \gamma + (\pi - \beta^L) \\ q_3^{LB} &= \gamma - (\pi - \beta^L) \\ q_3^{RA} &= \gamma - (\pi - \beta^R) \\ q_3^{RB} &= \gamma + (\pi - \beta^R) \end{aligned} \tag{4.5}$$

And the four possible solutions for $q_2$ are presented in Equation 4.6.

$$\begin{aligned} q_2^{LA} &= \varepsilon^L - \delta^L \\ q_2^{LB} &= \varepsilon^L + \delta^L \\ q_2^{RA} &= \varepsilon^R + \delta^R \\ q_2^{RB} &= \varepsilon^R - \delta^R \end{aligned} \tag{4.6}$$

Finally, Equation 4.7 summarizes the four configurations that conform the whole solution of the Inverse Kinematic Problem of the PUMA 560 manipulator.

$$\begin{aligned} q^{LA} &= \begin{bmatrix} q_1^L \\ q_2^{LA} \\ q_3^{LA} \end{bmatrix} \qquad q^{RA} = \begin{bmatrix} q_1^R \\ q_2^{RA} \\ q_3^{RA} \end{bmatrix} \\ q^{LB} &= \begin{bmatrix} q_1^L \\ q_2^{LB} \\ q_3^{LB} \end{bmatrix} \qquad q^{RB} = \begin{bmatrix} q_1^R \\ q_2^{RB} \\ q_3^{RB} \end{bmatrix} \end{aligned} \tag{4.7}$$

*Inverse Kinematics of the BH3-R Hexapod's Leg by the Analytical Procedure*

The analytical procedure begins with the Forward Kinematics equation system, presented in Equation 4.8, that calculates the position vector of the end point, $\vec{p}$, from the configuration of all actuators of the hexapod's leg ($q_1$, $q_2$ and $q_3$). In this equation, $p_{x_0}$ represents the projection of the vector $\vec{p}$ over the $\mathbf{x_0}$ axis, $p_{y_0}$ is the projection over $\mathbf{y_0}$, while $p_{z_0}$ constitutes the projection over $\mathbf{z_0}$.

$$\begin{bmatrix} p_{x_0} \\ p_{y_0} \\ p_{z_0} \\ 1 \end{bmatrix} = {}^0A_3 \cdot \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} \cos(q_1)\,[L_1 + L_2\cos(q_2) + L_3\sin(q_2 - q_3)] \\ \sin(q_1)\,[L_1 + L_2\cos(q_2) + L_3\sin(q_2 - q_3)] \\ L_2\sin(q_2) - L_3\cos(q_2 - q_3) \\ 1 \end{bmatrix} \tag{4.8}$$

The solution for the first actuator of the robot, $q_1$, can be obtained by dividing the second equation of the system shown in Equation 4.8 by the first one, as presented in Equation 4.9.

$$\begin{gathered} \frac{p_{y_0}}{p_{x_0}} = \frac{\sin(q_1)\,[L_1 + L_2\cos(q_2) + L_3\sin(q_2 - q_3)]}{\cos(q_1)\,[L_1 + L_2\cos(q_2) + L_3\sin(q_2 - q_3)]} = \frac{\sin(q_1)}{\cos(q_1)} \\ \Downarrow \\ q_1^F = \arctan\left(\frac{p_{y_0}}{p_{x_0}}\right) \end{gathered} \tag{4.9}$$

The right hand of Equation 4.9 shows one of the possible solutions for $q_1$, the one normally labeled as "FRONT" ($F$) for this type of mechanical structure. The second possible solution is the one presented in Equation 4.10, which also solves Equation 4.9. This second solution is usually identified as "BACK" ($B$).

$$\frac{\sin(q_1)}{\cos(q_1)} = \frac{p_{y_0}}{p_{x_0}} = \frac{-p_{y_0}}{-p_{x_0}} \implies q_1^B = \arctan\left(\frac{-p_{y_0}}{-p_{x_0}}\right) = q_1^F + \pi \tag{4.10}$$

The two possible solutions for $q_1$ are presented in Equation 4.11, where the arctan function has been substituted by the more general atan2, which is fully defined for all the range $[-\pi, \pi]$.

$$q_1^F = \text{atan2}(p_{y_0}, p_{x_0}); \qquad q_1^B = q_1^F + \pi; \tag{4.11}$$

To obtain the solution for $q_3$, we have to return to the definition of the Forward Kinematics equation system (see Equation 4.8), and do the matrix operations shown in Equation 4.12.

$$
\begin{bmatrix} p_{x_0} \\ p_{y_0} \\ p_{z_0} \\ 1 \end{bmatrix} = {}^0A_3 \cdot \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} = {}^0A_1(q_1^i) \cdot {}^1A_2 \cdot {}^2A_3 \cdot \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}
$$

$$
\Downarrow
$$

$$
\left({}^0A_1(q_1^i)\right)^{-1} \cdot \begin{bmatrix} p_{x_0} \\ p_{y_0} \\ p_{z_0} \\ 1 \end{bmatrix} = \left({}^0A_1(q_1^i)\right)^{-1} \cdot {}^0A_1(q_1^i) \cdot {}^1A_2 \cdot {}^2A_3 \cdot \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} = {}^1A_3 \cdot \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \tag{4.12}
$$

$$
\Downarrow
$$

$$
\begin{bmatrix} p_{x_1^i} \\ p_{y_1^i} \\ p_{z_1^i} \\ 1 \end{bmatrix} = \begin{bmatrix} L_2\cos(q_2) + L_3\sin(q_2 - q_3) \\ L_2\sin(q_2) - L_3\cos(q_2 - q_3) \\ 0 \\ 1 \end{bmatrix}
$$

The superscript $i$ that appears in some terms of Equation 4.12 may be either $F$ ("FRONT") or $B$ ("BACK"), depending on the related configuration of $q_1$. The terms $p_{x_1^i}$, $p_{y_1^i}$, and $p_{z_1^i}$ represent the projection of the position vector over the $\mathbf{x_1}$, $\mathbf{y_1}$, and $\mathbf{z_1}$ axes, respectively, taking into account that these axes depend on the aforementioned configuration of $q_1$.

From Equation 4.12, we can add the square of the first two equations of that equation system, obtaining the solution shown in Equation 4.13.

$$
\begin{gathered}
(p_{x_1^i})^2 + (p_{y_1^i})^2 = [L_2\cos(q_2) + L_3\sin(q_2 - q_3)]^2 + [L_2\sin(q_2) - L_3\cos(q_2 - q_3)]^2 \\
\Downarrow \\
(p_{x_1^i})^2 + (p_{y_1^i})^2 = (L_2)^2 + (L_3)^2 - 2L_2L_3\sin(q_3) \\
\Downarrow \\
\sin(q_3^i) = \frac{(L_2)^2 + (L_3)^2 - (p_{x_1^i})^2 - (p_{y_1^i})^2}{2L_2L_3}
\end{gathered} \tag{4.13}
$$

The term $\sin(q_3^i)$ in Equation 4.13 can have two possible solutions, depending on whether the configuration of $q_1$ is $F$ or $B$. Each of these solutions for $\sin(q_3^i)$ yield two possible values for $\cos(q_3)$ when the quadratic equation shown in Equation 4.14 is solved.

$$\cos(q_3^k) = \pm\sqrt{1 - [\sin(q_3^i)]^2} \tag{4.14}$$

The four possible values for $\cos(q_3^k)$ in Equation 4.14 lead to the expression of $q_3$ in Equation 4.15, where the superscript $k$ may be $FA$ ("FRONT ABOVE"), $FB$ ("FRONT BELOW"), $BA$ ("BACK ABOVE") or $BB$ ("BACK BELOW"). The names of these configuration identifiers were selected in a similar way as the ones of the the PUMA 560 (see Section 4.1.2).

$$q_3^k = \text{atan2}\big(\sin(q_3^i), \cos(q_3^k)\big) \tag{4.15}$$

Once $q_3$ has been calculated, $q_2$ may be obtained solving the equation system shown in the final part of Equation 4.12, as it is presented in Equation 4.16.

$$\begin{aligned} \sin(q_2^k) &= \frac{p_{y_1^i}[L_2 - L_3\sin(q_3^i)] + p_{x_1^i} L_3 \cos(q_3^k)}{[L_2 - L_3\sin(q_3^i)]^2 + [L_3\cos(q_3^k)]^2} \\ \cos(q_2^k) &= \frac{p_{x_1^i} - L_3\cos(q_3^k)\sin(q_2^k)}{L_2 - L_3\sin(q_3^i)} \end{aligned} \tag{4.16}$$

Finally, Equation 4.17 presents the four configurations structure that conform the solution of the Inverse Kinematic Problem of the analyzed walking hexapod.

$$\begin{aligned} q^{FA} &= \begin{bmatrix} q_1^F \\ q_2^{FA} \\ q_3^{FA} \end{bmatrix} \qquad q^{BA} = \begin{bmatrix} q_1^B \\ q_2^{BA} \\ q_3^{BA} \end{bmatrix} \\ q^{FB} &= \begin{bmatrix} q_1^F \\ q_2^{FB} \\ q_3^{FB} \end{bmatrix} \qquad q^{BB} = \begin{bmatrix} q_1^B \\ q_2^{BB} \\ q_3^{BB} \end{bmatrix} \end{aligned} \tag{4.17}$$

## 4.2 Hexapod's IKM

Throughout this work we have been using the hexapod's leg as an example for the application of the developed procedure, giving special attention to the automatic selection of the basis' monomial order. In this section we will prove that the selected monomial order is effectively the optimal one. This will be done by analyzing the performance and the computation times of all the

Groebner Bases calculated in 3.2.4, in order to demonstrate that the IKM synthesized from this selected order presents the lesser error and execution time.

Figure 4.3 presents the comparison between the outputs of the hexapod's leg reference model and the ones obtained by all six IKMs synthesized for this robot, one for each of the relevant lex orders presented in Table 3.2. This figure has marked in green ("Correctly calculated") all those positions inside the hexapod leg's workspace in which the corresponding IKM obtains the same amount of solutions as the reference model and, for all those solutions, the root mean square error (RMS) in the configuration space is less than $1\times10^{-6}$. Marked by red circles are the singularities found by the IKMs which, for the case of the hexapod's leg, are located in the axis defined by the intersection of the planes $X = 0 \cap Y = 0$.

Figure 4.3 shows that lex orders 1 to 5 correctly calculate all the positions of the robot's workspace, including its singularities. Only lex order 6 has some problems, because it incorrectly computes some false singularities, marked by black circles ("False Singularities"), along the axis defined by the intersection of the planes $Y = 0 \cap Z = 0$. This happens because the basis generated by lex order 6 has several leading terms that heavily depend on the values of $p_y$ and $p_z$, the position of the leg's tip along the axis $Y_0$ and $Z_0$, respectively. Therefore, when $p_y$ and $p_z$ are both equal to zero, those leading terms are cancelled out, generating an indetermination when the system is solved. Figure 4.4 presents a detailed view of the IKM generated by this problematic lex order.

These false singularities from the IKM generated by lex order 6 are not a problem, because the selected monomial order is lex order 4, which synthesizes an IKM that correctly calculates all the positions of the robot's workspace. But in the case that the selected monomial order has a problem similar to the one of lex order 6, the procedure's user can indicate that the problematic order is not a valid one. In this case the developed procedure will offer a new IKM, generated by the best monomial order from a list of lex orders that does not contain the problematic one.

The selected monomial order was used to synthesize the final IKM for the hexapod's leg. Figure 4.5 compares the results obtained with this IKM and those of the corresponding reference model. This figure shows that the calculated IKM successfully computes all the possible solutions inside the workspace of the hexapod's leg, while also finding all the singular points of the leg's mechanism, marked by the red circles ("Singularities"). These singular points are located in the axis defined by the intersection of the planes $X = 0 \cap Y = 0$.

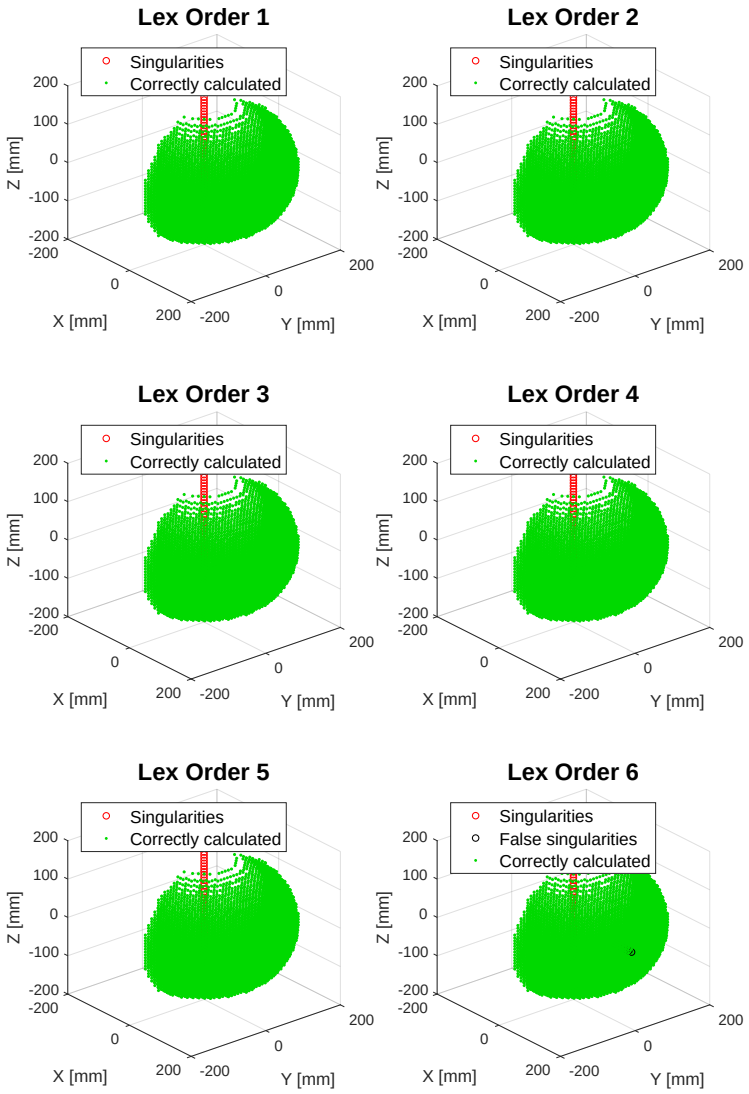


**Figure 4.3:** Performance analysis of the six synthesized IKMs for the hexapod's leg, one for each relevant lex order. Marked in green ("Correctly calculated") are all the the positions in which the corresponding IKM obtains the same amount of solutions as the reference model, with an RMS error lesser than $1 \times 10^{-6}$. The red circles correspond to the singularities of the leg's mechanism that are correctly identified by each IKM.

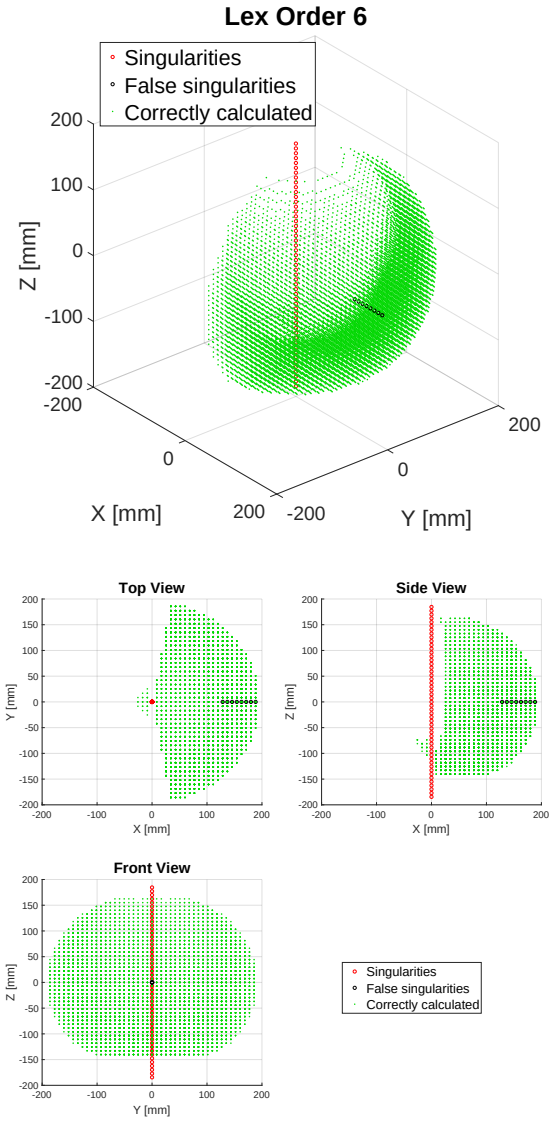


**Figure 4.4:** Detailed view of the performance of the hexapod's IKM generated by lex order 6. The top figure shows the isometric view of the leg's workspace, while the bottom one presents the top, side, and front views of the same workspace. The difference between this IKM and the ones generated by lex orders 1 to 5, is that the other IKMs do not present those false singularities along the axis defined by the intersection of the planes $Y = 0 \cap Z = 0$.

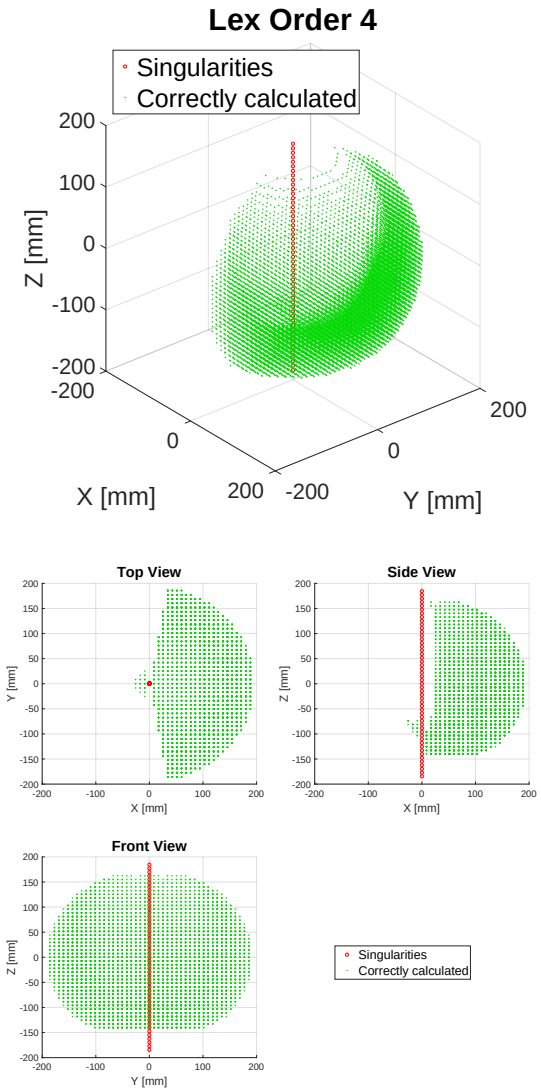


**Figure 4.5:** Performance analysis of the synthesized IKM for the hexapod's leg. The top figure shows the isometric view of the leg's workspace, while the bottom one presents the top, side, and front views of the same workspace. Marked in green ("Correctly calculated") are all the the positions in which the IKM obtains the same amount of solutions as the reference model, with an RMS error lesser than $1 \times 10^{-10}$. The red circles correspond to the singularities of the leg's mechanism that are correctly identified by the synthesized IKM.

**Table 4.1:** Maximum and average RMS errors obtained when all the points of the hexapod's workspace are processed by each of the six synthesized IKMs. The selected lex order is marked in **bold**.

| N° | Avg. RMS Error | Max. RMS Error |
|---|---|---|
| 1 | $5.566\times10^{-12}$ | $2.981\times10^{-8}$ |
| 2 | $2.577\times10^{-13}$ | $5.345\times10^{-10}$ |
| 3 | $5.754\times10^{-12}$ | $2.981\times10^{-8}$ |
| **4** | $\mathbf{4.542\times10^{-16}}$ | $\mathbf{1.243\times10^{-14}}$ |
| 5 | $2.561\times10^{-13}$ | $5.325\times10^{-10}$ |
| 6 | $1.177\times10^{-14}$ | $1.175\times10^{-12}$ |

In Figures 4.3 and 4.4, all the positions marked as "Correctly calculated" have an RMS error lesser than $1\times10^{-6}$ when compared to the ones calculated by the hexapod's reference model. In Figure 4.5, the one that shows the data comparison for the IKM synthesized using the selected monomial order, the RMS error margin was reduced to $1\times10^{-10}$, and still all the positions that are not in any of the singular points of the mechanical system are marked as "Correctly calculated". This further proves that the IKM synthesized by the developed procedure successfully solves the Inverse Kinematic Problem in all the robot's workspace.

Table 4.1 contains the maximum and average RMS errors obtained when all the points of the hexapod's workspace are processed by each of the six IKMs synthesized for this robot. As it can be seen in this table, the IKM that presents the least RMS error, both in its maximum value and its average, is the one generated by the selected lex order: number 4.

Regarding the computation cost of the six synthesized IKMs for the hexapod's leg, Table 4.2 contains a summary of the computation times when those IKMs are presented with all points in the robot's workspace, and compares those times with the ones of the reference model (row named "Ref"). This table shows the maximum ("Max [ms]"), minimum ("Min [ms]") and average ("Avg [ms]") times required to compute all the different positions inside the workspace. In this table it can be seen that, as expected, the smallest computation time among all the synthesized IKMs is the one achieved by the IKM related to the selected lex order. This proves that the three-step classification criterion presented in Section 3.2.5 effectively identifies the optimal monomial order.

**Table 4.2:** Computation times of the six IKMs generated for the hexapod's leg and its reference model ("Ref." row), when they are presented with all points in the robot's workspace. Column "Avg [ms]" shows the average computation times obtained for all the workspace's points (in milliseconds), while "Min [ms]" and "Max [ms]" present the minimum and maximum registered times, respectively. The selected lex order is marked in **bold**.

| **N°** | **Min [ms]** | **Avg [ms]** | **Max [ms]** |
|---|---|---|---|
| 1 | 0.226 | 0.265 | 0.726 |
| 2 | 0.236 | 0.285 | 1.031 |
| 3 | 0.210 | 0.251 | 0.748 |
| **4** | **0.199** | **0.238** | **0.670** |
| 5 | 0.231 | 0.281 | 1.049 |
| 6 | 0.203 | 0.247 | 0.787 |
| Ref. | 0.033 | 0.126 | 0.347 |

It is important to highlight that it is impossible to achieve computation times that are less than the ones of the reference model calculated by traditional methods, because this model is already composed of equations specially crafted for the analyzed robot. The computation times shown for the selected IKM are close enough to the ones of the reference model, while achieving a negligible positioning error, so it can be concluded that the IKM synthesized by our procedure is equivalent to this reference model.

The performance of the final IKM, synthesized with the selected lex order, was also tested following a trajectory that covers three consecutive full swings of the hexapod's leg, each one with a duration of 2.6s. The first swing has the hexapod walking in a diagonal direction that is $-\pi/6$ rad measured from the forward direction. The second swing is walking in the forward direction, while the third one has the robot moving in a $\pi/6$ rad diagonal. Figure 4.6 presents these tests results, where it can be seen that the outputs given by our IKM allow the hexapod's leg to follow any trajectory, with high precision and a completely negligible positioning error.

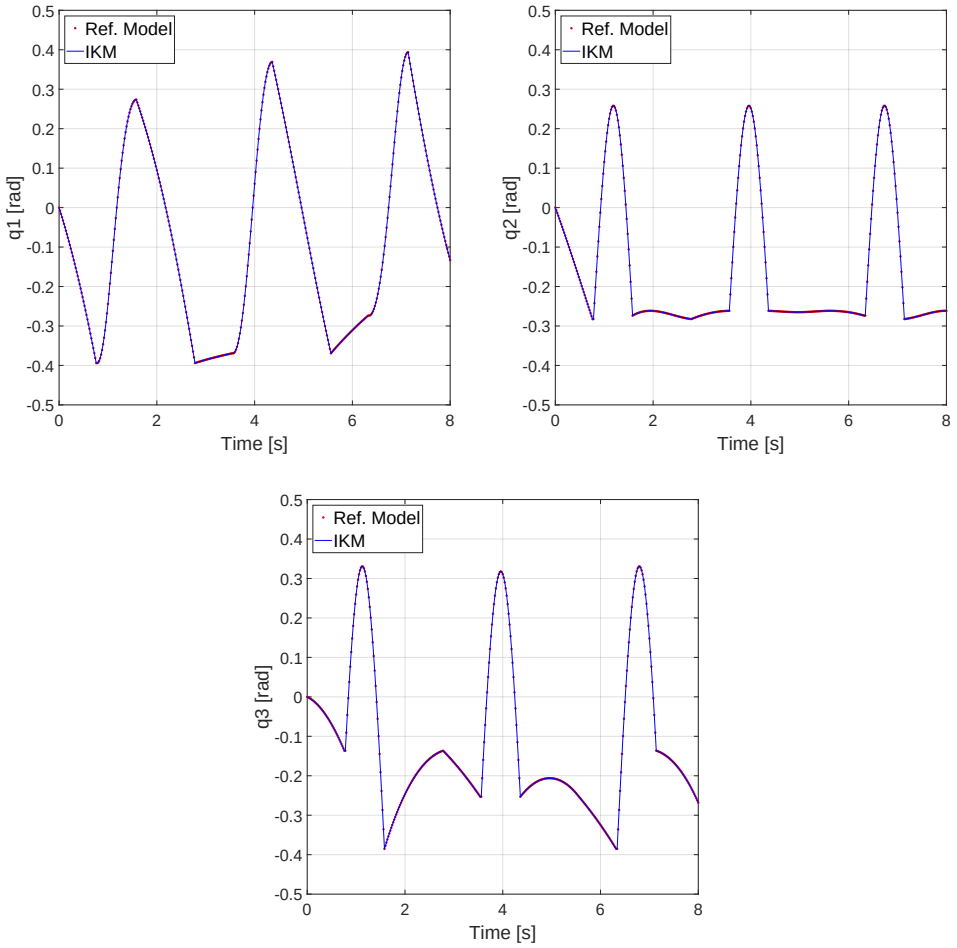


**Figure 4.6:** Trajectory tracking analysis for the hexapod's final IKM. The red dots represent the outputs of the reference model, for each of the leg's DoFs, when it is supplied with the predetermined trajectory, while the continuous blue lines are the IKM's outputs for that same trajectory. The top left graph contains the computed outputs for the first DoF ($q_1$). The data of the second DoF ($q_2$) is presented in the top right graph, while the bottom one displays the third DoF's outputs ($q_3$). The data show that the IKM's outputs follow those of the reference model for all three DoFs, with high accuracy and a negligible error along all the desired trajectory

**Table 4.3:** Expected values for the trigonometric variables related with the PUMA's rotational DoFs. The columns titled "$\mathbf{E}[|\mathbf{cos}(\mathbf{q_i})|]$" and "$\mathbf{E}[|\mathbf{sin}(\mathbf{q_i})|]$" are the expected values of the trigonometric variables related with a rotational DoF. The relative order of the trigonometric variables related with the rotational DoF is established depending on those expected values. The largest expected value of each DoF is marked in **bold**.

| **Rotational DoF** | $\mathbf{E}[\|\mathbf{cos}(\mathbf{q_i})\|]$ | $\mathbf{E}[\|\mathbf{sin}(\mathbf{q_i})\|]$ | **Relative order** |
|---|---|---|---|
| $q_1$ | **0.709** | 0.561 | $s_1 > c_1$ |
| $q_2$ | 0.507 | **0.763** | $c_2 > s_2$ |
| $q_3$ | **0.757** | 0.511 | $s_3 > c_3$ |

**Table 4.4:** Relevant lex orders for the PUMA manipulator.

| **N°** | **Lexicographic Order** |
|---|---|
| 1 | $s_1 > c_1 > c_2 > s_2 > s_3 > c_3$ |
| 2 | $s_1 > c_1 > s_3 > c_3 > c_2 > s_2$ |
| 3 | $c_2 > s_2 > s_1 > c_1 > s_3 > c_3$ |
| 4 | $c_2 > s_2 > s_3 > c_3 > s_1 > c_1$ |
| 5 | $s_3 > c_3 > s_1 > c_1 > c_2 > s_2$ |
| 6 | $s_3 > c_3 > c_2 > s_2 > s_1 > c_1$ |

## 4.3 PUMA's IKM

To demonstrate the application of the developed procedure to robotic manipulators, it was also used to synthesize the IKM of the PUMA 560 shown in Figure 1.4. As it was said before, the user just has to give as inputs the D–H parameters of the PUMA, presented in Table 1.4, and the movement range of its actuators.

Table 4.3 presents the expected values for each of the three PUMA's DoFs ($E[|cos(q_i)|]$ and $E[|sin(q_i)|]$), while Table 4.4 shows the relevant lex orders for the PUMA manipulator. It is important to highlight that the PUMA has three rotational DoFs, just like the hexapod's leg, but the lex orders presented in Table 4.4 are not the same that the ones used for the hexapod (see Table 3.2). This is because in the PUMA's case: $E[|sin(q_2)|] > E[|cos(q_2)|]$, so the two trigonometic variables related with $q_2$ are ordered as: $c_2 > s_2$.

**Table 4.5:** Selection of the lex order for the PUMA's IKM. After each step of the selection process, the discarded orders are marked in *italic*. The first classification criterion is to search for those lex orders whose basis' highest degree equation present the least computation time. The second one seeks for the bases with the lowest accumulated cost between all their equations, while the last one searches for the lesser amount of time required to compute all the coefficients of the basis. All these costs, expressed in clock cycles, were calculated for a microcontroller with an ARM Cortex-M4 CPU (ARM Limited 2009). The selected lex order is marked in **bold**.

| | Classification Criteria | | |
|---|---|---|---|
| **Order N°** | **Highest Deg. [Cycles]** | **Acc. Cost [Cycles]** | **Coefficients [Cycles]** |
| 1 | 37 | 110 | *360* |
| 2 | 37 | 110 | *405* |
| 3 | 37 | 110 | 47 |
| **4** | **37** | **110** | **47** |
| 5 | 37 | 110 | *405* |
| 6 | 37 | 110 | *65* |

The results of the three-step selection criterion applied to the relevant lex orders of the PUMA manipulator are presented in Table 4.5. After applying this selection criterion, the selected lex order for the PUMA is the fourth one, which is the same lex order in which the variables were solved while applying the geometric method used in Guzmán-Giménez, Valera Fernández, et al. 2020.

All of the six possible IKMs for the PUMA correctly calculate all the positions inside the robot's workspace, with an RMS error below $4 \times 10^{-9}$. The computation times required by the all the IKMs synthesized for the PUMA manipulator are compiled in Table 4.6. It can be seen that the IKM that has the least average computation time is the one automatically selected by our three-step criterion: lex order 4.

This selected order was used to synthesize the final IKM for the PUMA manipulator. Figure 4.7 presents the comparison, in all the robot's workspace, of the results obtained by this IKM and the corresponding reference model. This figure has marked in green ("Correctly calculated") all those positions in which the IKM obtains the same amount of solutions as the reference model and, for all those solutions, the root mean square error (RMS) in the configuration space is less than $1 \times 10^{-10}$. This figure proves that the IKM synthesized by

**Table 4.6:** Computation times of the six IKMs generated for the PUMA 560 and its reference model ("Ref." row), when they are presented with all points in the robot's workspace. Column "Avg [ms]" shows the average computation times obtained for all the workspace's points (in milliseconds), while "Min [ms]" and "Max [ms]" present the minimum and maximum registered times, respectively. The selected lex order is marked in **bold**.

| **N°** | **Min [ms]** | **Avg [ms]** | **Max [ms]** |
|---|---|---|---|
| 1 | 0.228 | 0.251 | 0.597 |
| 2 | 0.213 | 0.235 | 0.657 |
| 3 | 0.205 | 0.231 | 0.633 |
| **4** | **0.204** | **0.230** | **0.621** |
| 5 | 0.210 | 0.233 | 0.707 |
| 6 | 0.212 | 0.239 | 0.678 |
| Ref. | 0.108 | 0.117 | 0.275 |

the developed procedure successfully computes all the possible solutions inside the PUMA's workspace.

The synthesized IKM performance was also tested following a trajectory that covers the majority of the PUMA's workspace. Figure 4.8 presents these tests results, where it can be seen that the outputs given by our IKM allow the PUMA's wrist to follow any trajectory, with high precision and a completely negligible positioning error.

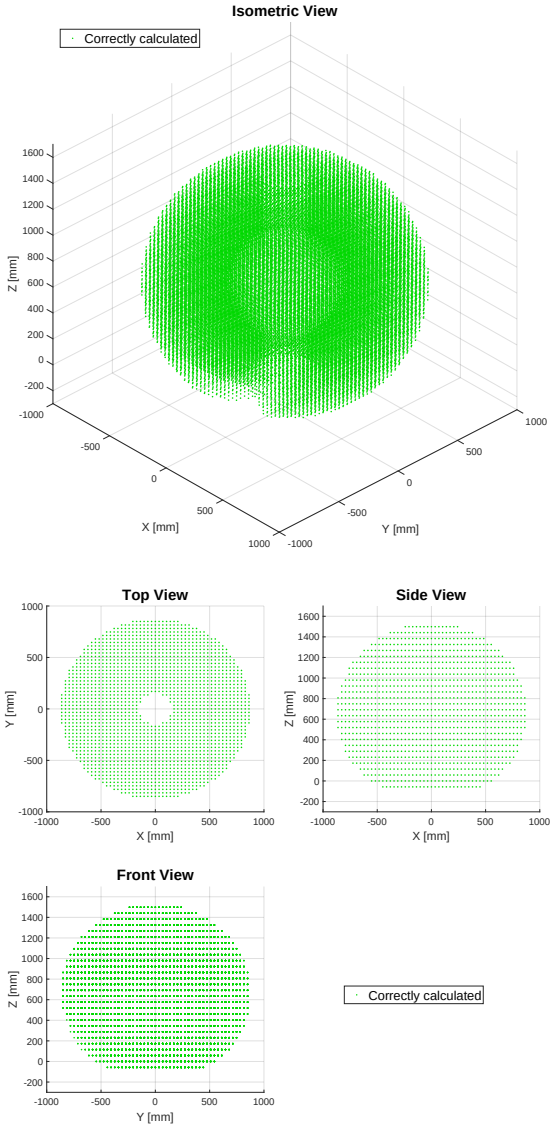


**Figure 4.7:** Performance analysis of the synthesized IKM for the PUMA 560. The top figure shows the isometric view of the PUMA's workspace, while the bottom one presents the top, side, and front views of the same workspace. Marked in green ("Correctly calculated") are all the the positions in which the IKM obtains the same amount of solutions as the reference model, with an RMS error lesser than $1 \times 10^{-10}$.

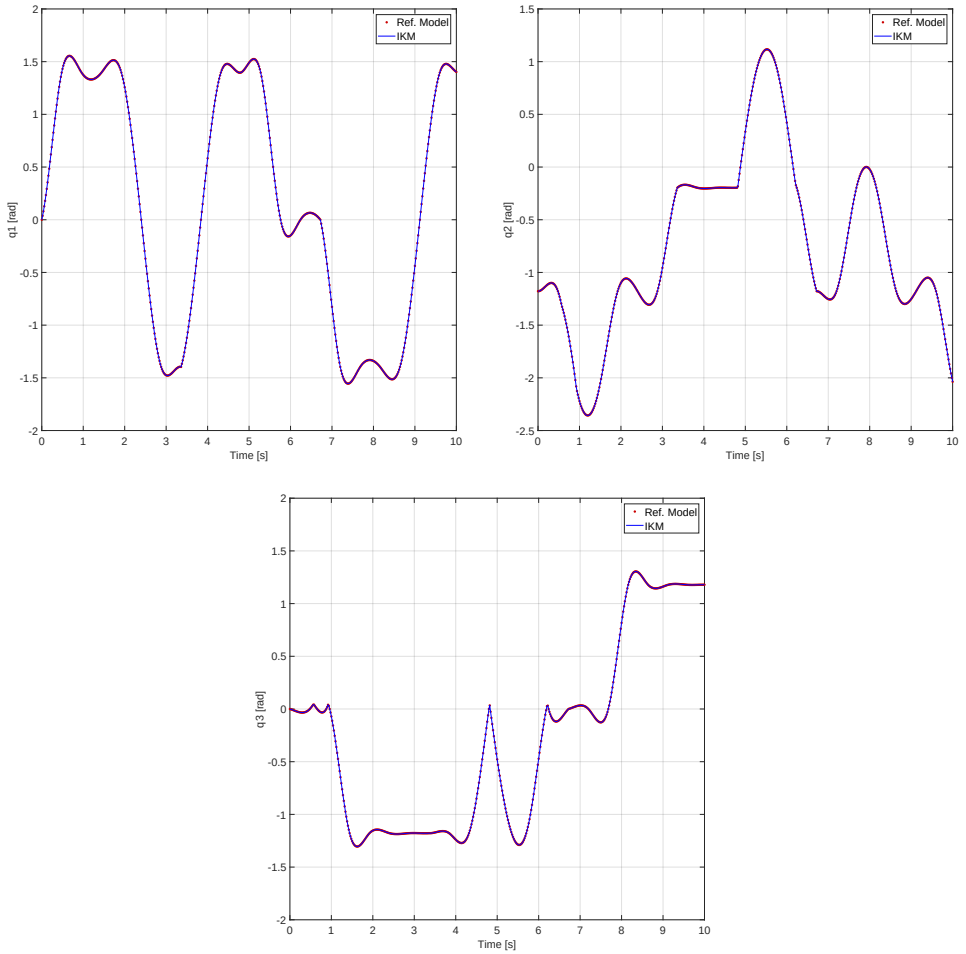


**Figure 4.8:** Trajectory tracking analysis for the PUMA's IKM. The red dots represent the outputs of the reference model, for each of the PUMA's DoFs, when it is supplied with the predetermined trajectory, while the continuous blue lines are the IKM's outputs for that same trajectory. The upper left figure presents the computed outputs for the first DoF ($q_1$), the upper right one corresponds to the second ($q_2$), while the bottom figure are the outputs for the third DoF ($q_3$). The data show that the IKM's outputs follow the ones of the reference model, with high accuracy and a negligible error along all the desired trajectory, in the robot's three DoFs.

# Chapter 5

# Discussion, Conclusions and Future Work

*This last chapter presents a summary of the developed procedure and the performance tests done over the IKMs synthesized by this procedure, the publications made from the developed procedure, and the conclusions and possible future works for this project.*

## 5.1 Summary of the developed procedure and discussion

This work presented the procedure developed to employ Groebner Basis theory in the synthesis of the IKM of non-redundant open-chain robotic systems. The inputs requested by this procedure to begin the IKM synthesis process are the D-H parameters of the robot and the movement range of its actuators, which corresponds to the least information required to properly describe a robotic system. With that information the developed process provides the synthesized IKM, ready to be used in the robot's control system or in a simulation of its behavior.

The synthesized IKMs always have the structure shown in Figure 3.1. The developed procedure automatically configures and prepares the four modules that conform said structure, without requiring any further information from the user.

All the performance tests presented in Chapter 4 prove that the developed procedure successfully synthesized the IKMs for both test benches: a PUMA manipulator and a walking hexapod. The execution times shown in Tables 4.2 and 4.6 prove that the developed procedure successfully selected the optimal Groebner Basis' monomial order for each of the synthesized IKMs. Figures 4.5 and 4.7 show that the synthesized IKMs correctly solve the Inverse Kinematic Problem in all the workspace of their corresponding robotic systems, with an RMS error lesser than $1 \times 10^{-10}$. Finally, Figures 4.6 and 4.8 prove that the outputs of the synthesized IKMs can follow any trajectory inside their corresponding robot's workspace.

The main contribution of this work is the development of the systematic procedure that automatically synthesizes the complete Inverse Kinematic Model of non-redundant open-chain robotic systems. This procedure can be used to synthesize the IKM of a wide variety of open-chain industrial robotic systems and mobile robots, including:

- Cartesian robotic systems.
- SCARA robots.
- Multi-legged walking and climbing robots.
- Non-redundant robotic manipulators that satisfy the in-line wrist condition.

The second contribution is that the developed procedure automatically selects the optimal monomial order for the Groebner Basis used in the IKM synthesis. As indicated above, this was proven with the execution times shown in Tables 4.2 and 4.6, where it can be seen that the developed procedure always selects the monomial order that ensures that the synthesized IKM has the shortest possible execution time. This way the user does not have to worry about the Grobner Basis' monomial order, nor know any additional information about the robotic system, besides the aforementioned inputs: the robot's D-H parameters and the movement range of its actuators.

This work's third contribution is that the synthesized IKM is ready to be used in the robot's microcontroller. As stated in Chapter 3, the output of the developed procedure is the IKM, compiled as C++ code or as a MATLAB® script, depending on the initial selections made by the procedure's user. Any of these IKM implementations can be used directly in the robot's control system.

This synthesized IKM does not request any kind of complex number operations from the robot's microcontroller, which corresponds to this work's fourth contribution. As was shown in Table 3.4, all the required operations from the microcontroller are: floating point additions, multiplications and divisions, atan2, square roots, cosines and a cubic root, although the cosine and the cubic root are only necessary if the Groebner Basis used in the IKM synthesis has a quartic polynomial equation. All the previously stated operations can be easily executed by modern-day microcontrollers, without requiring the ability to operate with complex numbers.

The last contribution is the framework presented in this work, which can be used to apply Groebner Basis theory in the resolution of a wide variety of engineering problems. The structure shown in Figure 3.1, as well as the procedure developed to synthesize it, can be applied to other solutions based on Groebner Basis theory, with some adjustments:

- IKM Core: This core module can be configured to solve other types of problems. It is only necessary to make three changes to this module to adapt it to any new system:

  1. Update the first and second steps of the synthesis process shown in Figure 3.2. These are the steps that take the system's inputs and prepare them for the polynomial equation system. The rest of the synthesis process may stay unaltered.

  2. Update the meaning of $Val[1] = -1$ and when all the values of the $Val$ vector equal to zero. These situations will still happen, but they will have different meanings, depending on the studied system.

  3. Update the "Wrap-Up section" of the core's algorithm (see Figure 3.4). The output of the core module may need a different preparation, which will depend on the requirements of the studied system.

- State Estimator: This module is responsible for selecting the best solution from the output given by the core module, based on an estimation of the current state. Any kind of estimation can be programmed in this module, as long as all the necessary information of the previous state (or states) is stored in the registers.

- IKM Derivatives: This module is optional, although it can be used to calculate any function that is directly derived from the computed Groebner Basis.

- Registers: The registers just have to be adjusted to store all the data required by the other modules.

With the six changes explained above, the presented framework can be easily configured to implement the solutions of other engineering problems that are solved using Groebner Basis theory.

## 5.2 Publications

The first concept of the developed procedure was presented in (Guzmán-Giménez, Valera, et al. 2019). Here we explain the necessary steps to synthesize the IKM of a non-redundant open-chain robotic system using Groebner Basis theory, applying them in the synthesis of the IKM of a walking hexapod.

The first version of the IKM Core's synthesis process was published in (Guzmán-Giménez, Valera Fernández, et al. 2020). This first version had almost all the steps presented in Section 3.2 to synthesize the IKM Core, except for the automatic selection of the basis' monomial order.

That automatic selection of the basis' monomial order was added in the second version of the IKM Core's synthesis process, which was published in (Guzmán-Giménez, Valera Fernández, et al. 2021), thus completing the description of the developed procedure to synthesize the IKM Core module.

## 5.3 Conclusions

This work presents the development of a systematic procedure that employs the Groebner Basis theory to synthesize the IKM of non-redundant open-chain robotic systems. The developed procedure does not require any special knowledge of the robot's mechanical structure, besides its Denavit-Hartenberg parameters and the movement range of its actuators. The procedure selects automatically the optimal monomial order for the Groebner Basis, and synthesizes a complete IKM from this basis. The procedure's output is the synthesized IKM, both in C++ and as a MATLAB® script, which is ready to be used directly in the robot's control system or to simulate its behavior.

The performance analysis presented in Chapter 4 shows that the developed procedure successfully synthesized the IKMs of a multi-legged robot, a Lynxmotion's BH3-R hexapod, and a non-redundant manipulator, a PUMA 560. The synthesized IKMs are totally comparable, both in precision and compu-

tation time, with their respective kinematic models calculated by traditional methods. This implies that the developed procedure represents a systematic solution to the Inverse Kinematic Problem of non-redundant open-chain robotic systems, one that is completely independent of the robot's mechanical structure.

The synthesized IKMs are ready to be used in the robot's microcontroller, and they do not only supply the position reference for all the robot's actuators, as it is case of most IKP solvers, but also provide the corresponding references for the actuators' velocities and accelerations. Therefore, the IKMs that are synthesized by the developed procedure can be used in a wide range of control systems, including those that have an acceleration feedback loop or an acceleration feed-forward strategy.

The developed procedure can be used to synthesize the IKM of a wide range of mobile and open-chain industrial robots, including cartesian robotic systems, SCARA robots, multi-legged walking and climbing robots and all non-redundant manipulators that satisfy the in-line wrist condition.

The last contribution of this work is the framework of the procedure, which can be used to apply Groebner Basis theory in the resolution of a wide variety of engineering problems. With some minor adjustments, the developed procedure, as well as all the structures and algorithms presented in this work, can be used to implement other solutions based on Groebner Basis theory.

## 5.4 Future Work

There are three lines of research that continue the work presented in this PhD Thesis:

The first one is to extend the procedure to also cover the end effector's orientation, completing this way the full pose computation for all types of non-redundant open-chain robots.

The second proposed line of research is the extension to redundant robots, in order to fully cover all the spectrum of open-chain robotic systems. The application of Groebner Basis theory to a redundant robot will surely produce an underdetermined equation system with infinite solutions. Properly solving these types of equation systems will require the application of kinematic restrictions, as is common for redundant robotic systems. It is possible that the degree of these equations systems would be greater than four, so the analytical

methods presented in Section 3.2.7 cannot be applied, and the final solution would surely require the use of numerical methods. Nevertheless, the structure presented in Figure 3.1 can still be employed, updating the IKM Core module with the implementation of the required numerical method, which would only change the "Main loop" section of the IKM Core's algorithm (see Figure 3.4).

Therefore, the developed procedure presented in this work is easily extendable to redundant open-chain robotic systems, only requiring some adjustments in the second step, where the kinematic equations that should be solved are calculated, and also in the seventh step, which corresponds to the IKM Core's implementation.

As a third line of research, we propose to expand the developed procedure to closed-chain robotic systems. The solution of the kinematic problem of this type of robots contains polynomial equations whose degree is greater than four, but, as in the case of the redundant robots, this only implies a change in the "Main loop" section of the Core's algorithm.

So, in a similar way to the case of redundant open-chain robots, the developed procedure can also be extended to closed-chain robotic systems, making only some adjustments in its second step, to properly prepare the kinematic equations of the closed-chain robot, and in the seventh step, in order to update Core's algorithm with the required numerical method.

# Bibliography

Abbasnejad, Ghasem and Marco Carricato (2015). "Direct Geometrico-static Problem of Underconstrained Cable-Driven Parallel Robots With n Cables". In: *IEEE Transactions on Robotics* 31.2, pp. 468–478. ISSN: 15523098. DOI: 10.1109/TRO.2015.2393173 (cit. on p. 4).

ARM Limited (2009). *Cortex-M4 Technical Reference Manual*. Tech. rep. version r0p0. Available at https://developer.arm.com/documentation/ddi0439/b/ (cit. on pp. 33–35, 75).

Atique, Moin Uddin, Rafiqul Islam Sarker, and Atiqur Rahman Ahad (2018). "Development of an 8DOF quadruped robot and implementation of Inverse Kinematics using Denavit-Hartenberg convention". In: *Heliyon* 4.12, e01053. ISSN: 24058440. DOI: 10.1016/j.heliyon.2018.e01053 (cit. on pp. 2, 9, 25, 58).

Aydm, Yavuz and Serdar Kucuk (2006). "Quaternion Based Inverse Kinematics for Industrial Robot Manipulators with Euler Wrist". In: *2006 IEEE International Conference on Mechatronics, ICM*, pp. 581–586. DOI: 10.1109/ICMECH.2006.252591 (cit. on p. 3).

Barrientos, Antonio et al. (2012). "Modelado de Cadenas Cinemáticas mediante Matrices de Desplazamiento. Una alternativa al método de Denavit-Hartenberg". In: *RIAI - Revista Iberoamericana de Automatica e Infor-*

*matica Industrial* 9.4, pp. 371–382. ISSN: 16977920. DOI: 10.1016/j.riai.2012.09.004 (cit. on p. 2).

Bouzgou, Kamel and Zoubir Ahmed-Foitih (2014). "Geometric modeling and singularity of 6 DOF Fanuc 200IC robot". In: *4th International Conference on Innovative Computing Technology, INTECH 2014*. Institute of Electrical and Electronics Engineers Inc., pp. 208–214. ISBN: 9781479942336. DOI: 10.1109/INTECH.2014.6927745 (cit. on p. 3).

Buchberger, Bruno (2001). "Gröbner Bases and Systems Theory". In: *Multidimensional Systems and Signal Processing* 12.3, pp. 223–251. ISSN: 09236082. DOI: 10.1023/A:1011949421611 (cit. on pp. 4, 27).

Chen, Saixuan et al. (2017). "A general analytical algorithm for collaborative robot (cobot) with 6 degree of freedom (DOF)". In: *2017 IEEE International Conference on Applied System Innovation (ICASI)*, pp. 698–701. ISBN: 9781509048977. DOI: 10.1109/ICASI.2017.7988522 (cit. on p. 3).

Cox, David A., John Little, and Donal O'Shea (2015). *Ideals, Varieties, and Algorithms. An Introduction to Computational Algebraic Geometry and Commutative Algebra*. Ed. by Sheldon Axler and Kenneth Ribet. Fourth Edition. Undergraduate Texts in Mathematics. Cham, Switzerland: Springer International Publishing. ISBN: 978-3-319-16720-6. DOI: 10.1007/978-3-319-16721-3 (cit. on pp. 13–20, 26, 27).

Deshmukh, Deepak et al. (Dec. 2020). "ANFIS-Based Inverse Kinematics and Forward Dynamics of 3 DOF Serial Manipulator". In: *Advances in Intelligent Systems and Computing* 1375 AIST, pp. 144–156. DOI: 10.1007/978-3-030-73050-5_15 (cit. on p. 3).

Duka, Adrian-Vasile (2014). "Neural Network based Inverse Kinematics Solution for Trajectory Tracking of a Robotic Arm". In: *Procedia Technology* 12, pp. 20–27. ISSN: 2212-0173. DOI: 10.1016/j.protcy.2013.12.451 (cit. on p. 3).

— (Jan. 2015). "ANFIS Based Solution to the Inverse Kinematics of a 3DOF Planar Manipulator". In: *Procedia Technology* 19, pp. 526–533. ISSN: 2212-0173. DOI: 10.1016/J.PROTCY.2015.02.075 (cit. on p. 3).

Farahmandi, Farimah and Bijan Alizadeh (2015). "Groebner basis based formal verification of large arithmetic circuits using Gaussian elimination and cone-based polynomial extraction". In: *Microprocessors and Microsystems* 39.2, pp. 83–96. ISSN: 01419331. DOI: 10.1016/j.micpro.2015.01.007 (cit. on p. 13).

Faugère, J. C. et al. (1993). "Efficient Computation of Zero-dimensional Gröbner Bases by Change of Ordering". In: *Journal of Symbolic Computation* 16.4, pp. 329–344. ISSN: 0747-7171. DOI: 10.1006/jsco.1993.1051 (cit. on pp. 20, 32).

Faugère, Jean Charles (1999). "A new efficient algorithm for computing Gröbner bases (F4)". In: *Journal of Pure and Applied Algebra* 139.1-3, pp. 61–88. ISSN: 0022-4049. DOI: 10.1016/S0022-4049(99)00005-5 (cit. on pp. 20, 32).

— (2010). "FGb: A Library for Computing Gröbner bases". In: *ICMS 2010. Lecture Notes in Computer Science*. Ed. by K. Fukuda et al. Vol. 6327. Springer, Berlin, Heidelberg, pp. 84–87. ISBN: 978-3-642-15582-6. DOI: 10.1007/978-3-642-15582-6_17 (cit. on pp. 20, 32).

Flanders, Megan and Richard C. Kavanagh (2015). "Build-A-Robot: Using virtual reality to visualize the Denavit-Hartenberg parameters". In: *Computer Applications in Engineering Education* 23.6, pp. 846–853. DOI: 10.1002/cae.21656 (cit. on p. 2).

Fu, K. S., R. C. Gonzalez, and C. S. G. Lee (1987). *Robotics: Control, Sensing, Vision and Intelligence*. Ed. by R. Sanjeev. CAD/CAM, robotics, and computer vision. McGraw-Hill Book Company. ISBN: 0-07-022625-3 (cit. on pp. 2, 9, 25, 33, 58, 59).

Gan, Dongming et al. (2009). "Forward displacement analysis of the general 6-6 Stewart mechanism using Gröbner bases". In: *Mechanism and Machine Theory* 44.9, pp. 1640–1647. ISSN: 0094114X. DOI: 10.1016/j.mechmachtheory.2009.01.008 (cit. on pp. 4, 5).

Guzmán-Giménez, José, Ángel Valera, et al. (Aug. 2019). "Obtención del modelo cinemático inverso de sistemas robotizados de cadena cinemática abierta empleando bases de Groebner: aplicación a un robot hexápodo". In: *XL*

*Jornadas de Automática*. Universidade da Coruna, pp. 726–734. DOI: 10.17979/spudc.9788497497169.726 (cit. on p. 82).

Guzmán-Giménez, José, Ángel Valera Fernández, et al. (2020). "Synthesis of the Inverse Kinematic Model of Non-Redundant Open-Chain Robotic Systems Using Groebner Basis Theory". In: *Applied Sciences* 10.8, p. 2781. ISSN: 2076-3417. DOI: 10.3390/app10082781 (cit. on pp. 2, 5, 20, 23, 25, 32, 36, 37, 75, 82).

— (2021). "Automatic selection of the Groebner Basis' monomial order employed for the synthesis of the inverse kinematic model of non-redundant open-chain robotic systems". In: *Mechanics Based Design of Structures and Machines*. ISSN: 15397742. DOI: 10.1080/15397734.2021.1899829 (cit. on pp. 23, 25, 82).

Huang, Xiguang and Guangpin He (2009). "Forward kinematics of the general Stewart-Gough platform using Gröbner basis". In: *2009 IEEE International Conference on Mechatronics and Automation, ICMA 2009*, pp. 3557–3561. ISBN: 9781424426935. DOI: 10.1109/ICMA.2009.5246088 (cit. on pp. 4, 5).

Hussein, Mustafa T, Ali S Gafer, and Essam Z Fadhel (2020). "Robot manipulator Inverse Kinematics using Adaptive Neuro-Fuzzy Inference System". In: *Journal of Engineering Science and Technology* 15.3 (cit. on p. 3).

Jiang, Guanwu et al. (2017). "A Precise Positioning Method for a Puncture Robot Based on a PSO-Optimized BP Neural Network Algorithm". In: *Applied Sciences* 7.10, p. 969. ISSN: 2076-3417. DOI: 10.3390/app7100969 (cit. on p. 4).

Kendricks, Kimberly D. (2013). "A kinematic analysis of the gmf a-510 robot: An introduction and application of groebner basis theory". In: *Journal of Interdisciplinary Mathematics* 16.2-3, pp. 147–169. ISSN: 09720502. DOI: 10.1080/09720502.2013.800304 (cit. on pp. 4, 5).

Köker, Raşit (Feb. 2013). "A genetic algorithm approach to a neural-network-based inverse kinematics solution of robotic manipulators based on error minimization". In: *Information Sciences* 222, pp. 528–543. ISSN: 00200255. DOI: 10.1016/j.ins.2012.07.051 (cit. on p. 3).

Koukos-Papagiannis, C. K., V. C. Moulianitis, and N. A. Aspragathos (July 2019). "Cuspidality Investigation of a Metamorphic Serial Manipulator." In: *Mechanisms and Machine Science* 73, pp. 2491–2500. DOI: 10.1007/978-3-030-20131-9_246 (cit. on p. 14).

Liu, Hualiang and Jianyou Han (Aug. 2021). "Solution region synthesis methodology of spatial 1CS-4SS linkages for six given positions". In: *Mechanism and Machine Theory* 162, p. 104369. ISSN: 0094-114X. DOI: 10.1016/J.MECHMACHTHEORY.2021.104369 (cit. on p. 14).

Liu, Yuan et al. (Sept. 2015). "Geometric approach for inverse kinematics analysis of 6-Dof serial robot". In: *2015 IEEE International Conference on Information and Automation, ICIA 2015*. Institute of Electrical and Electronics Engineers Inc., pp. 852–855. ISBN: 9781467391047. DOI: 10.1109/ICInfA.2015.7279404 (cit. on p. 2).

Mahajan, Akanshu, H. P. Singh, and N. Sukavanam (Mar. 2017). "An unsupervised learning based neural network approach for a robotic manipulator". In: *International Journal of Information Technology* 9.1, pp. 1–6. ISSN: 2511-2104. DOI: 10.1007/s41870-017-0002-2 (cit. on p. 3).

Mulla, Ameer K. et al. (Jan. 2018). "Leader selection for minimum-time consensus in multi-agent networks". In: *2017 IEEE 56th Annual Conference on Decision and Control, CDC 2017* 2018-January, pp. 1036–1041. DOI: 10.1109/CDC.2017.8263793 (cit. on p. 14).

Narayan, Jyotindra and Ashish Singla (Sept. 2017). "ANFIS based kinematic analysis of a 4-DOFs SCARA robot". In: *4th IEEE International Conference on Signal Processing, Computing and Control, ISPCC 2017* 2017-January, pp. 205–211. DOI: 10.1109/ISPCC.2017.8269676 (cit. on p. 3).

Özgür, Erol and Youcef Mezouar (Mar. 2016). "Kinematic modeling and control of a robot arm using unit dual quaternions". In: *Robotics and Autonomous Systems* 77, pp. 66–73. ISSN: 09218890. DOI: 10.1016/j.robot.2015.12.005 (cit. on p. 2).

Patil, Deepak et al. (May 2015). "Computation of feedback control for time optimal state transfer using Groebner basis". In: *Systems & Control Letters* 79, pp. 1–7. ISSN: 0167-6911. DOI: 10.1016/J.SYSCONLE.2015.02.003 (cit. on p. 14).

Pérez-Rodríguez, Rodrigo et al. (2012). "Inverse kinematics of a 6 DoF human upper limb using ANFIS and ANN for anticipatory actuation in ADL-based physical Neurorehabilitation". In: *Expert Systems with Applications* 39.10, pp. 9612–9622. ISSN: 09574174. DOI: 10.1016/j.eswa.2012.02.143 (cit. on p. 3).

Petrescu, Relly Victoria et al. (2017). "Anthropomorphic Solid Structures n-R Kinematics". In: *American Journal of Engineering and Applied Sciences* 10.1, pp. 279–291. ISSN: 1941-7020. DOI: 10.3844/ajeassp.2017.279.291 (cit. on p. 3).

Rameau, Jean François and Philippe Serré (Sept. 2015). "Computing mobility condition using Groebner basis". In: *Mechanism and Machine Theory* 91, pp. 21–38. ISSN: 0094114X. DOI: 10.1016/j.mechmachtheory.2015.04.003 (cit. on pp. 4, 5).

Rodriguez, Ruthber et al. (2018). "A consistent methodology for the development of inverse and direct kinematics of robust industrial robots". In: *ARPN Journal of Engineering and Applied Sciences* 13.1, pp. 293–301. ISSN: 18196608 (cit. on pp. 3, 58).

Rokbani, Nizar and Adel M. Alimi (Jan. 2013). "Inverse kinematics using particle swarm optimization, a statistical analysis". In: *Procedia Engineering*. Vol. 64. Elsevier, pp. 1602–1611. DOI: 10.1016/j.proeng.2013.09.242 (cit. on p. 4).

Rokbani, Nizar, Alicia Casals, et al. (2015). "IK-FA, a new heuristic inverse kinematics solver using firefly algorithm". In: *Studies in Computational Intelligence* 575, pp. 369–385. ISSN: 1860949X. DOI: 10.1007/978-3-319-11017-2_15 (cit. on p. 4).

Sabbagh, Negar Aghapour and Bijan Alizadeh (June 2021). "Arithmetic Circuit Correction by Adding Optimized Correctors Based on Groebner Basis Computation". In: *2021 IEEE European Test Symposium (ETS)*. Leuven, Belgium: Institute of Electrical and Electronics Engineers (IEEE), pp. 1–6. ISBN: 9781665418492. DOI: 10.1109/ETS50041.2021.9465454 (cit. on p. 14).

Salzer, Herbert E (Feb. 1960). "A Note on the Solution of Quartic Equations". In: *Mathematics of Computation* 14.71, pp. 279–281. ISSN: 00255718, 10886842. DOI: 10.2307/2003172 (cit. on pp. 32, 33, 45).

Toshani, Hamid and Mohammad Farrokhi (2014). "Real-time inverse kinematics of redundant manipulators using neural networks and quadratic programming: A Lyapunov-based approach". In: *Robotics and Autonomous Systems* 62.6, pp. 766–781. ISSN: 09218890. DOI: 10.1016/j.robot.2014.02.005 (cit. on p. 3).

Uchida, Thomas and John McPhee (June 2012). "Using Gröbner bases to generate efficient kinematic solutions for the dynamic simulation of multi-loop mechanisms". In: *Mechanism and Machine Theory* 52, pp. 144–157. ISSN: 0094114X. DOI: 10.1016/j.mechmachtheory.2012.01.015 (cit. on pp. 4, 5, 36).

Wang, Xiangke et al. (2012). "The geometric structure of unit dual quaternion with application in kinematic control". In: *Journal of Mathematical Analysis and Applications* 389, pp. 1352–1364. DOI: 10.1016/j.jmaa.2012.01.016 (cit. on p. 2).

Wang, Yan, Lu Bin Hang, and Ting Li Yang (Jan. 2006). "Inverse kinematics analysis of general 6R serial robot mechanism based on groebner base". In: *Frontiers of Mechanical Engineering in China* 1.1, pp. 115–124. ISSN: 16733479. DOI: 10.1007/s11465-005-0022-7 (cit. on p. 4).