Universidad de Extremadura
Escuela Politécnica de Cáceres

Algorithms for efficient computation of generalized Inverse Kinematics in humanoid robots

Pedro Yuste Gallego

2014

Grado de Ingeniería de Sonido e Imagen en Telecomunicación

Universidad de Extremadura
Escuela Politécnica de Cáceres

Grado en Ingeniería de Sonido e Imagen en Telecomunicación
Trabajo Fin de Grado

Algorithms for efficient computation of generalized Inverse Kinematics in humanoid robots

Author: Pedro Yuste Gallego

Director: Dr. Pedro M. Núñez Trujillo

Director: Dr. Carmen Ortiz Caraballo

- **Qualifier Committee**
  - **President:** Mr. Jesús Rubio Ruiz
  - **Secretary:** Mrs. Rosa María Navarro Olmo
  - **Chair:** Mr. Pablo Bustos García de Castro

Al extinto Colegio Mayor Francisco de Sande,
a mi abuelo Pepe.

"We must maintain that mathematical geometry
is not a science insofar as we understand
by space a visual structure that can be
filled with objects, it is a pure
theory of manifolds".

- Hans Reichenbach (1891-1953) -
- The Philosophy of Space and Time -

Abstract

The following document is intended to describe what is hidden behind the Inverse Kinematics problem and all the knowledge which is needed to explain it. In order to accomplish this purpose, the main mathematical concepts and their application to engineering will be developed in detail. Nevertheless, since the whole process required to characterize a spatial movement is really complex, some simplifications and middle steps must be treated before. Taking this reason into account, many examples have been included along this work to make the understanding easier when progressing.

Finally, the basic ideas named above will be extended by applying some improvements on the conventional systems. To achieve it, different numerical methods or mathematical possibilities will also be introduced and briefly discussed.

Contents

1 Introduction **25**

1.1 Motivation 29

1.2 Objectives in each chapter 30

1.3 Related projects 31

2 Fundamentals of important mathematical concepts **33**

2.1 Basic algebraic structures 34

2.2 Euclidean space 36

2.2.1 Properties 37

2.2.2 Important derivated notions 39

2.2.2.1 Euclidean norm 39

2.2.2.2 Distance 40

2.2.2.3 Angles 40

2.2.2.4 Orthogonality 41

2.3 Manifolds 42

2.3.1 Definitions 44

2.3.1.1 Charts 44

2.3.1.2 Atlases 45

2.3.1.3 Transition function 45

2.3.2 General example 46

2.4 Concepts about groups 47

2.4.1 Rigid transformations 47

2.4.2 Congruences and Symmetry group 48

2.4.3 Lie groups 49

2.4.3.1 Lie algebras 49
2.4.4 The special Euclidean Group in three dimensions, $SE(3)$, and relevant subgroups 51
2.4.4.1 Subgroups 52
2.5 The exponential map 53
2.5.1 The logarithm map 56

3 The group SO(2): Bi-dimensional rotations 57
3.1 Planar rotations and translations 58
3.1.1 2D translations 58
3.1.2 2D rotations 59
3.1.2.1 Rotation by complex numbers 62
3.2 Method for defining planar movements 65

4 The group SO(3): Three-dimensional rotations 69
4.1 Spatial rotations and traslations 70
4.1.1 3D translations 70
4.1.2 Rotation by Euler angles 71
4.1.3 The problem of gimbal lock 75
4.1.4 Rotation by Euler axis/angle 76
4.1.4.1 Euclidean vectors rotation by Rodrigues formula . . 77
4.1.5 Rotation by quaternions 79
4.2 Method for defining spatial movements 86

5 Forward Kinematics 91
5.1 Structure and components of the kinematic chain 91
5.1.1 Links 92
5.1.2 Joints 92
5.2 Forward kinematics resolution by homogeneous transformation matrices 94
5.3 Denavit-Hartenberg representation 95
5.3.1 Denavit-Hartenberg parameters 95
5.3.2 Denavit-Hartenberg algorithm for obtaining the forward kinematics model: 97

5.3.2.1 Some examples for better understanding: Three-link planar manipulator (see [46]) . . . 100
5.3.2.2 Some examples for better understanding: Anthropomorphic arm . . . 101

6 Inverse Kinematics 103
6.1 Important points when solving Inverse Kinematics . . . 104
6.2 Analytical methods . . . 106
6.2.1 Solution by geometric methods . . . 106
6.2.1.1 Example of a three-DOF-robot geometric resolution . 106
6.2.2 Solution by algebraic methods . . . 108
6.2.2.1 Example of a Three-DOF-spherical-arm robot algebraic resolution . . . 110
6.3 Iterative methods . . . 114
6.3.1 Solution by the Jacobian inversion method . . . 114
6.3.1.1 Iterative model . . . 115

7 Numerical methods and their applications to solve the Inverse Kinematics Problem 117
7.1 Taylor's Theorem . . . 118
7.1.1 Taylor's theorem in one variable . . . 118
7.1.2 Taylor's Theorem for multivariable functions . . . 120
7.2 Optimization methods to solve equations . . . 122
7.2.1 Gradient descent . . . 122
7.2.1.1 Descent gradient algorithm . . . 124
7.2.2 Newton-Raphson Method . . . 125
7.2.2.1 Newton-Raphson method to solve one-variable functions . . . 125
7.2.2.2 Newton-Raphson method to solve and minimize multivariable functions . . . 127
7.2.3 Gauss-Newton Method . . . 129
7.2.3.1 Deduction from the Newton-Raphson method . . . 130
7.2.4 Levenberg-Marquardt . . . 132

7.2.4.1 Levemberg-Marquardt's aim 132
7.2.4.2 Levenberg-Marquardt's formula 132

8 Manifolds in order to avoid non-Euclidean spaces 135
8.1 Manifolds as state representations 136
8.2 Encapsulation of manifolds 137
8.2.1 Summary of properties 139
8.2.1.1 Consequences 140
8.2.2 Generic definition 141
8.3 Examples 141
8.3.1 Manifold approach to vector space $\mathbb{R}^n$ 141
8.3.2 Manifold approach to $SO(2)$ 142

9 Conclusions 143
9.1 A world of rotations 143
9.2 The Inverse Kinematics problem 146
9.2.1 Numerical methods and other improvements 146
9.3 Further study fields 148

Acknowledgements 149

A Conversions 153
A.1 2D rotations 153
A.1.1 From 2 × 2 rotation matrices to complex numbers 153
A.1.2 From complex numbers to 2 × 2 rotation matrices 154
A.2 3D rotations 154
A.2.1 From rotation matrix to Euler angles 154
A.2.2 From Euler angles to rotation matrix 155
A.2.3 From rotation matrix to Euler axis/angle 155
A.2.4 From Euler axis/angle to rotation matrix 156
A.2.5 From rotation matrix to quaternions 156
A.2.6 From quaternions to rotation matrix 156
A.2.7 From Euler angles to quaternions 157
A.2.8 From quaternions to Euler angles 157

A.2.9 From Euler axis/angle to quaternions 157
A.2.10 From quaternions to Euler axis/angle 158

B Coordinate systems 159
B.1 Right-handed coordinate system 160
B.2 Left-handed coordinate system 160
B.3 Conversion from Right-Handedness to Left-Handedness 161
B.3.1 Conversion of points 161
B.3.2 Conversion of rotations 161

C MATLAB Code 163
C.1 Rotation matrices in 2D 163
C.2 Homogeneous transformation matrices in 2D 164
C.3 Rotation matrices in 3D 165
C.4 Homogeneous transformation matrices in 3D 168

Bibliography 171

List of Figures

1.1 Robots working in manufacturing, clear example of the use of inverse kinematics. 28
1.2 Humanoid robot Loki in Robolab, which was used in the attempt to calibrate the solution to the Inverse Kinematics problem by using GTSAM. 31

2.1 Graphic representation of the possible operation with vectors and their equivalences. 38
2.2 Cartesian coordinate system in the three-dimensional space, where the axes form an orthonormal system and each vector can be represented as combination of these basis vectors or axes (see equation (2.1)). . . 41
2.3 Example of one-dimensional manifolds. These curves can be evaluated as lines in local neighbourhoods. 43
2.4 Two-dimensional manifold M embedded in three-dimensional space, a point $p \in M$, the tangent space at p (T_xM) and the algebra m, which defines the vector operations. 43
2.5 Example of two-dimensional manifolds in three-dimensional space, like spheres, paraboloids and torus. 44
2.6 On the left side, example of an atlas from a sphere, where all the polygons are charts (local planar approximations). On the right, the map of topological relations between charts. 44
2.7 Example of a transition function between two charts, where M is the manifold, ϕ_α and ϕ_β are the charts and $T_{\alpha,\beta}$ and $T_{\beta,\alpha}$ are the transition functions for going from one to another. 45

2.8 Definition of a manifold M, where it is possible to see two different charts (ϕ_α and ϕ_β), and the transition function $T_{\alpha,\beta}$ between them. Some important concepts are also inside red boxes. 46

2.9 **Manifold** M composed by a set of **charts** (**atlas**). Two charts (in pink and yellow) have been highlighted to exemplify how they must share a common part (in green) to make the atlas coherent. 46

2.10 The set of symmetry transformations which defines the symmetry group of the equilateral triangle: the identity, rotations of order 3, and combinations of rotations and reflections. 48

2.11 The Lie algebra m is the tangent space of M at the identity. 49

2.12 Lie algebras $so(2)$ and $so(3)$ in relation to the groups $SO(2)$ and $S0(3)$. 50

2.13 Translation and rotation by rigid body movements. 51

2.14 Graphical interpretation of the exponential map. 54

2.15 Graphical interpretation of the logarithm map. 56

3.1 Plots of an arrow before and after the specified rotations ($2\pi/3$ and π). It has been inscribed inside a circle to visualize how the modulus is not altered. The code details of this plot on MATLAB are contained in the **Appendix C.1**. 61

3.2 Effect of the rotation in the plane by complex numbers. 64

3.3 Plots of an arrow which represents the position on the vector intending to specify the different transformations, the rotation by $-\pi/3$ and the displacement by $t_x = 2$ and $t_y = -1$. Now the circle can be used as reference to see how after the rotation $v_2 = (-0.866, 0.5)$ the vector has left the unit circumference to sum what indicated in each axis. . . 67

4.1 Single representation of the corresponding angles around the cited axes and its effect in each case. 71

4.2 Graphical representation of the successive partial rotations on the final result. 72

4.3 Plots of an arrow before and after the specified rotations. It has been inscribed inside two perpendicular circumferences to visualize how the modulus is not altered. The details of the script to code this plot on MATLAB are contained in **Appendix C.3.** 74
4.4 On the left, all the gimbals are totally free to rotate on their respective axes. On the right, the three axis are parallel and the locking out of the axis which depends on the rest (in this case the blue axis) is produced. The blue gimbal cannot rotate along the orange one. . . . 75
4.5 Schematic representation of a rotation by Euler axis/angle method. . 77
4.6 Decomposition of the red vector in two: one parallel to the rotation axis $\hat{r}$ (yellow) and the remainder in green. **R** is the rotation operation. 77
4.7 Representation of the new defined vector $\vec{t}$. The cross represent the vector $\hat{r}$ with outside direction. 78
4.8 Cayley graph a Cayley table for quaternions. The blue line is a multiplication by **i** and the red one by **j** in the the indicated direction. . . 80
4.9 Plots of an arrow which have been moved according to the specified transformations: a translation ($t_x = -1$, $t_y = 3$, $t_z = 2$) and two consecutive rotations ($-|pi/4$ around the x-axis and $3\pi/5$ around the z-axis). The two perpendicular circumferences can be used to visualize how the norm 1-limits are overpassed after a translation. 89

5.1 Most used kind of links. 92
5.2 Most known one DOF joints. 93
5.3 Most known joints with **more** than one DOF. 93
5.4 Graphical example to explain how to place axes in Denavit-Hartenberg representation (see [44]). 98
5.5 Three-link planar manipulator schematic. 100
5.6 Anthropomorphic arm schematic. 101

6.1 Forward Kinematics vs Inverse Kinematics. 103
6.2 Reachable workspace depending on different configurations. 104
6.3 Graph to exemplify the dependence between system and task degrees of freedom. 105

6.4 Robot to solve: Three-DOF-arm robot. 106

6.5 Possible two choices for the same purpose. 108

6.6 Robot to solve: Three-DOF-spherical-arm robot. 110

6.7 Iterative model for the Jacobian inversion method. 116

7.1 Different approaches to the function $sin(x)$ (in dark blue) by several Taylor polynomials (k=1 in red, k=3 in orange, k=5 in green, k=7 in light blue and k=9 in violet). 119

7.2 First order Taylor polynomial approximation for the sinusoidal function f (left) and second order Taylor polynomial approximation for the same function (right). One can observe that the first degree polynomial in 2D is a plane (in 1D is a line) and the second degree polynomial is corresponded with a curved plane inside a three-dimensional space. 121

7.3 Graph example to show the different aspects to consider when using **descent gradient** depending on the function shape and the chosen initial point. 123

7.4 Zig-zag movement when reaching the steepest direction by using decent gradient. 123

7.5 Newton-Raphson method for a one-variable funtion, where r is the root and x_i the successive approximations by using tangents. 126

7.6 Comparison between the methods of Gradient descent and Newton-Raphson. Convergence is much faster in this last one (and because of that the efficiency is higher), but it is not assured if the initial guess is not close to the root. 129

7.7 Example of the optimization process for one-variable functions by using the Gauss-Newton method. 131

7.8 The Levenber-Marquardt method applied to a bi-dimensional error function. 134

8.1 Local neighbourhood in the manifold M (here the unit sphere) mapped into $\mathbb{R}^n$ (here $\mathbb{R}^2$, the plane). 138

8.2 Graph to exemplify that the distance between $x \boxplus \delta_1$ and $x \boxplus \delta_2$, represented with the dashed line, is less or equal to the distance in the parametrization around x, which is represented with the dotted line. 140

9.1 Local neighbourhood in the manifold M (here the unit sphere) mapped into $\mathbb{R}^n$ (here $\mathbb{R}^2$, the plane). 147

B.1 Comparative between right-handed and left-handed coordinate systems: x-axis in red, y-axis in green and z-axis in blue. 160

List of Tables

5.1 Three-link planar manipulator D-H parameters. 100
5.2 Anthropomorphic arm D-H parameters. 101

6.1 D-H parameters to solve the given robot. 110

9.1 Comparative graph about the storage requirements. 145
9.2 Comparative graph detailing the comparison of rotation chaining operations. 145
9.3 Comparative graph detailing the vector rotating operations. 145

Chapter 1

Introduction

The relationship between mathematics and engineering is as old as mankind, both of them being bound together by the optics of the common sense. This union, in the shape of which both theory and practice develop and complement each other, has offered society the possibility of transforming the world and of inventing models out of thin air to explain the reality of things, and the fertility and attractiveness of this bond still has people flocking to exploit it. Writing the world anew is the most ambitious game there is – it knows no limits and no end. All the methods and the tricks used so far cannot begin to cover the entirety of creation itself. They are but insufficient patches. All the struggle, the attempts and the discoveries, fundamental in themselves, no doubt – yet still lacking. It is the grandeur of science that embodies the promise to answer all questions, conceivable or not by the human mind, but the certainty of the most impressive of them are an impossible dream.

The following work takes its inspiration from mathematics to describe and simulate something so purely tied to the field of physics such as directed movement. When interacting with our environment, we are habitually witnesses of the meaning of inverse kinematics, yet our eyes have gotten so used to it, that we do not stop to ask how or why anymore. There is no doubt that when engineering runs into such a question the search for an answer becomes an essential part of following this path. In our case, the study of inverse kinematics is fundamental for the development of disciplines such as robotics, artificial vision or videogames.

Inverse kinematics is a relatively new study field, but its usefulness goes beyond the immediate and unconscious relationship between robotics and science fiction that the Hollywood influence has conditioned us to see. Robots are present in many aspects of daily life and in industry and their design requires the intertwining of knowledge in the fields of mechanics, computer science, electronics, control, physics and mathematics. The world of robotics we see today still has a long way to go before it reaches the heights imagined by Isaac Asimov but the presence of these machines in manufacturing and even in the service of people grows more and more with each day.

As can be inferred, this project does not pretend to explain the entire development which, aside from being too complex, would also prove too wide for it to be possible to cover it, but it seeks to present the most important tools to begin to comprehend this world. The most pure generic concepts will be treated in every case, from what is necessary to look at the hexa-dimensional space where our problem is placed to the most important definitions in mechanics which allow us to catch a short glimpse of how robotic arms can be articulated. To analyse this aspect even more deeply, the end of this report will also include some of the most innovative techniques used in solving the inverse kinematics problem via the use of algorithms. However, this would not prove appropriate without previously having skilfully deciphered the fundamentals on which the pillars of our problem is built on.

As of now, an universe comprised of orientations, effects, consequences and unknown variables opens itself to the possibility of being solved. With the aim of making its interpretation easier for the future audience, this work represents the struggle of adapting (as much as possible) the mathematical language which although at times seems chaotic, has proved itself fundamental for the progress of technology.

Introducción

La relación entre la matemática y la ingeniería es tan antigua como el hombre, y ambas están ligadas desde la óptica del más estricto sentido común. Esta unión, en que la teoría y la práctica se desarrollan y complementan en conjunto, ha brindado al hombre la posibilidad de transformar el mundo, le ha permitido inventar patrones de la nada para explicar la realidad de las cosas. Y es tan fértil el matrimonio y su atractivo tan potente, que la curiosidad de acercarse a la gallina de los huevos de oro es irresistible. Jugar a escribir el mundo es el más ambicioso de los propósitos, pues es un propósito sin final. Todos los trucos, artimañas y ardides que han podido idearse hasta hoy han resultado insuficientes parches de la creación. Todos los esfuerzos, tentativas y descubrimientos, sin embargo, fundamentales. La grandeza de la ciencia es la promesa de respuesta a todos los interrogantes que nadie pueda plantearse, pero la certeza de que los más imponentes son inalcanzables en una vida.

El trabajo que sigue a continuación se nutre de las matemáticas para describir y simular algo tan físico como el movimiento dirigido. Al interactuar con nuestro entorno, asistimos habitualmente al significado de la cinemática inversa con unos ojos tan acostumbrados a lo común que no plantean la pregunta del cómo. Sin embargo, cuando los propósitos de la ingeniería se topan con tal interrogante, la búsqueda de una respuesta se hace imprescindible para continuar el camino. En el caso que nos ocupa, el estudio del problema de cinemática inversa es fundamental para el desarrollo de disciplinas como la robótica, la visión artificial o los videojuegos.

La cinemática inversa es un estudio relativamente reciente, pero su utilidad se escapa más allá de la relación casi inmediata e inconsciente que Hollywood nos condiciona entre la robótica y la ciencia ficción. Los robots están presentes en numerosos campos de la vida cotidiana y la industria, y para su diseño es necesario entrelazar conocimientos sobre mecánica, informática, electrónica, control, física y matemática. El mundo de la robótica no es todavía comparable al que Isaac Asimov se afanó en construir, pero día tras día se multiplica la presencia de estas máquinas en los procesos de fabricación o incluso al servicio más directo de las personas.

Como bien puede inferirse, la pretensión de este proyecto no consiste en explicar todo un desarrollo que, además de complejo, resultaría inabordable, sino que presentará las herramientas más importantes para iniciarse a la comprensión de este mundo. Tan sólo los conceptos más puramente genéricos serán tratados en cada caso, desde lo necesario para mirar al espacio hexadimensional donde se asienta nuestro problema hasta las definiciones más importantes sobre mecánica que permiten entrever someramente la manera en que los brazos robóticos pueden ser articulados. A modo de ampliación, se incluirán también al final de esta memoria algunas de las técnicas más innovadoras que se utilizan para resolver mediante algoritmos el problema cinemático inverso, pero esto no resultaría apropiado sin haber descifrado previamente con soltura los más básicos cimientos que conforman la armadura del asunto.

De aquí en adelante se descubre un universo de orientaciones, efectos, consecuencias e incógnitas que resolver. Con el ánimo de facilitar su interpretación al futuro público, este trabajo representa el esfuerzo por adaptar en la medida de lo posible un lenguaje matemático que, si bien a veces resulta caótico, se ha demostrado fundamental para progresar en el desarrollo de las tecnologías.

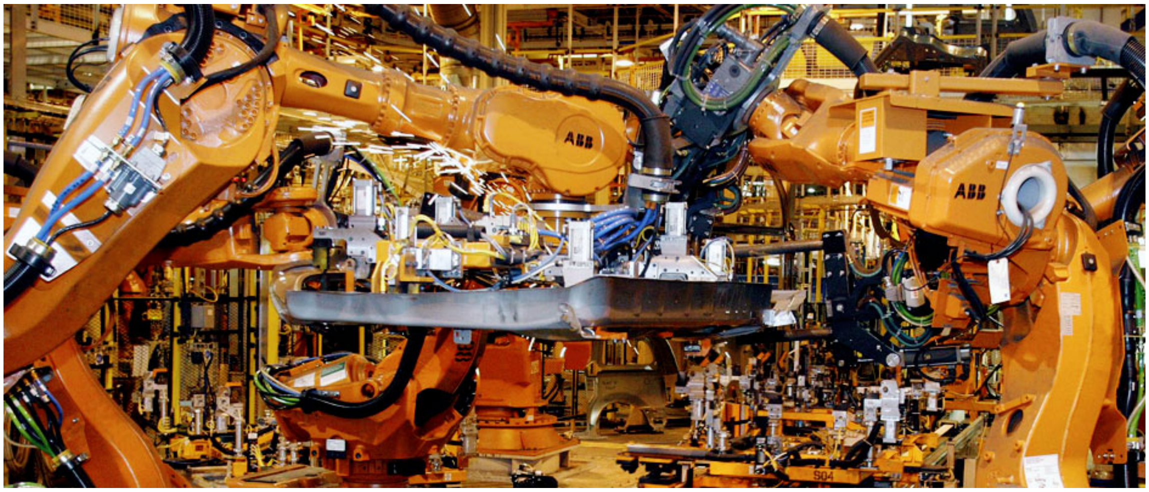


Figure 1.1: Robots working in manufacturing, clear example of the use of inverse kinematics.

1.1 Motivation

This work is mainly motivated by the attention in three particular issues: researching, mathematics and robotics.

Research is the motor of progress and the cause of most of the advances. A very important part of the discoveries and improvements is the result of the work of many university research staff, a benefit which should be appreciated and valued with investment. Research, the **R&D** itself, has an effect on many areas of society and also on the economy and competitiveness of countries. University means research, and its prestige can be measured by their researchers' standing and the quality of their works. Since research is a consequence of curiosity, people can simultaneously both help towards society's progress and prove their limits.

Mathematics is something usually loved or completely hated. There is no middle term in this discussion. As the rest of sciences, it must be built step by step but it becomes more interesting as time goes on, when one can find out the way in which all its subfields perfectly fit each other. The attraction of mathematics lies in the application of logic to solve problems, and analysing this structure in depth has been a truly interesting road along the entire degree.

Finally, robotics and the labour of the research group **Robolab** have also been very important in making the decision to start this project. Having had a look at the world of robotics in a more personal way and observed how people work inside it and the huge open fronts they must face at the same time, it is impossible not to feel interested in this study. Thanks to getting involved in this field, some of the secrets about how a robot can move or "think" have started to clear up. In addition, the chance to admire how difficult it is to simulate the simplest human behaviour has allowed the admiration of a work which people think is much easier than it really is.

1.2 Objectives in each chapter

The goals this work follows are summarized chapter by chapter below. In this way, the project is intended to do the following:

- **Chapter 2 (see 2):** To describe what the Euclidean space is and its properties, the idea of manifold, the concept of mathematical group and Lie groups and the exponential map.
- **Chapter 3 (see 3):** To explain the way of getting rotations and translations in the bi-dimensional space and the most known methods to do it ($SO(2)$).
- **Chapter 4 (see 4):** To explain the way of getting rotations and translations in the three-dimensional space and the most known methods to do it ($SO(3)$).
- **Chapter 5 (see 5):** To present the fundamentals of forward kinematics: the structure and components of the kinematic chain and the different methods to solve the forward kinematics problem.
- **Chapter 6 (see 6):** To present the fundamentals of inverse kinematics: the basis to consider and the difference between analytical and iterative methods when solving the inverse kinematics problem.
- **Chapter 7 (see 7):** To explain the application of some numerical methods in order to minimize the function of error and reaching the best solution: gradient descent, Newton-Raphson and Levenberg- Marquardt.
- **Chapter 8 (see 8):** To apply the encapsulation of manifolds in order to make easier algorithms to deal with state representations in $SO(2)$ and $SO(3)$.

1.3 Related projects

At the beginning this project was orientated to a different matter: the calibration of inverse kinematics systems using **graph factors** by **MATLAB** and, especially focusing on the robot **Loki** (an humanoid robot developed by **Robolab**). When working on this task, which in the ended proved to be unsuccessful, some results were shown to explain the reasons why the method failed. These analyses are contained in the papers **Calibración del Robot Loki mediante factores gráficos** (see [68]) and **Introducción a los factores gráficos de GTSAM** (see [69]), both written in Spanish with **Mercedes Paoletti Ávila** while working with a scholarship in collaboration with the Computer Technology department and the Mathematics department.

In the same line of study, the project titled **Cinemática inversa en robots sociales** (Inverse Kinematics in social robots, see) describes a way of looking at the inverse kinematics problem for robots which are focussed on social tasks. To do that, the general solution to the Inverse Kinematics problem by the method of Levenberg-Marquardt (see 7.33) is perfectly detailed and it represents a very good complement to the ideas which will be mentioned in this work.

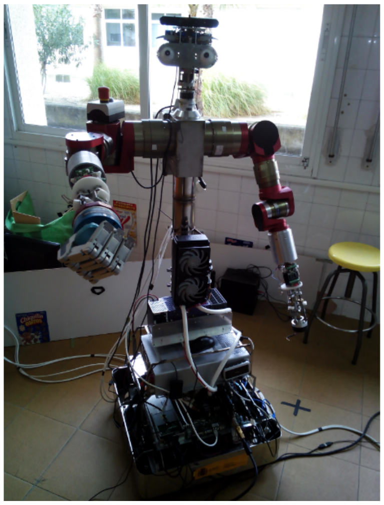


Figure 1.2: Humanoid robot Loki in Robolab, which was used in the attempt to calibrate the solution to the Inverse Kinematics problem by using GTSAM.

Chapter 2

Fundamentals of important mathematical concepts

Since the beginning, the task of describing reality has become a natural necessity to the human being. But how can one explain the physical world and its behaviour? How can one describe the way in which things interact? To manage this, an admirable tool is required. Fortunately, after much development and improvement throughout History, this tool is now real and can be used by society: it is Mathematics which must be congratulated for allowing us to reach a point in evolution where the possibility of explaining anything, with more or less complexity, is now so much in our grasp, that we can explain the way in which something works which we previously considered to be magic.

When facing an engineering project, the mathematical basis must be firstly studied and the required concepts introduced, especially the most concrete ones which can escape the common general knowledge. Hence, this work, whose goal is to describe and analyse the physical position and orientations, must mention the Mathematics which makes it possible before proceeding to the description related to how it makes it.

The purpose of this chapter is to present several mathematical concepts to define and parametrize dimensions or spaces. This is needed because it is otherwise impossible to specify an orientation without knowing where it is defined, which also applies to kinematics. Knowing the different ways of representing space becomes vital considering that this idea is going to work as a background in the following chapters.

2.1 Basic algebraic structures

Definition 1 *A* ***binary operation*** *on a set is a calculation that combines two operands (elements of the set) to produce another element of the set.*

Definition 2 *A* ***group*** *is a non-empty set G in which a binary operation*

$$(a, b) \to ab$$

is defined satisfying the following properties (see [1]):

1. ***Closure:*** *If two elements a and b belong to G, then ab is also in G.*
2. ***Associativity:*** *$a(bc) = (ab)c$ for all $a, b, c \in G$.*
3. ***Identity:*** *There is an element $1 \in G$ such that $a1 = a$, where a can be any element in G.*
4. ***Inverse:*** *If a is in G, then there is an element a^{-1} in G such that $aa^{-1} = a^{-1}a = 1$.*

A group G is ***abelian*** *if the binary operation is commutative ($ab = ba$) for all $a, b \in G$.*

- Example. The integers:

  The set of integers $\mathbb{Z}$ is a group because it satisfies the properties above with the addition operation:

  1. For any two integers a, b, the sum $a + b = c$ and c always belongs to $\mathbb{Z}$, never to any set such as fractions.
  2. For all the integers a, b and c, it is satisfied the associativity property $(a + b) + c = a + (b + c)$.
  3. There is an element (0 in this case) which acts like the identity element for any integer a: $a + 0 = 0 + a = a$.
  4. For every integer a, there is an integer b such that $a+b = b+a = 0$. Then the inverse element b is denoted $-a$.

Definition 3 ***Algebra*** *is a branch of mathematics which uses mathematical statements to describe relationships between things which can vary. When a mathematical statement describes a relationship, letters are used to represent the quantity that varies, since it is not a fixed amount. These letters and symbols are referred to as variables. The mathematical statements that describe relationships are expressed using algebraic terms, expressions, or equations (mathematical statements containing letters or symbols to represent numbers) (see [2]).*

Definition 4 *A* ***topology*** *T on a set S is a collection of subsets of S which verify the following rules (see [3]):*

1. *The empty set $\emptyset \in T$, where $\emptyset$ is the empty set. In addition, the set $S \in T$.*
2. *Any union of elements of T is an element of T. Let be two elements $O_1 \in T, O_2 \in T$, then $O_1 \cap O_2 \in T$.*
3. *Any intersection of finitely many elements of T is an element of T. Let be two elements $O_1 \in T, O_2 \in T$, then $O_1 \cup O_2 \in T$.*

The three restrictions before are called the axioms of topology. If they are satisfied, then T is called a topology on S. A ***topological space*** *is a set S with a topology T.*

2.2 Euclidean space

Definition 5 *A space* $\mathbb{E}$ *is an* ***Euclidean space*** *when it is a kind of geometric space where Euclid's postulates are satisfied. The extended real number line, the Euclidean plane or the three-dimensional space are special cases of Euclidean spaces with one, two or three dimensions. Moreover, the abstract concept of Euclidean space is also generalized to higher dimensions.*

It is possible to define n-dimensional Euclidean space, denoted $\mathbb{E}^n$*, as all the real vector space equipped with an inner product (or scalar product).*

The traditional approach to geometry defines Euclidean space to have the following properties:

1. A straight line may be drawn from any one point to any other point, so only two points are needed to define a line.
2. A finite straight line may be produced to any length in a straight line.
3. A circle may be described with any centre at any distance from that centre (radius).
4. If two angles are right angles $\left(\frac{\pi}{2}\right)$, then they are congruent (identical).
5. At an outer point to a line, it can be only defined a parallel line to the first one.

The enumeration above is known as Euclid's postulates.

As the inner product (or scalar product) is defined, then Euclidean space acts like a linear real vector space. Hence, the vector in the vector space corresponds to the points of the Euclidean space which can be defined by using terms of a linear combination of orthogonal basis vectors:

$$P = \alpha_1 \cdot v_1 + \alpha_2 \cdot v_2 + \ldots + \alpha_n \cdot v_n \tag{2.1}$$

where P is the vector representation expressed as a point in the Euclidean space, v_i are the different basis vector (one for each dimension) and α_i are the scalar multipliers, used in order to combine the different vector basis to define a concrete point.

On the other hand, **the scalar product of two vectors returns a single scalar** and it is algebraically defined in $\mathbb{R}^n$ as:

$$v_1 \cdot v_2 = \sum_{i=1}^{n} v_{1i} v_{2i} = v_{11}v_{21} + v_{12}v_{22} + \ldots + v_{1n}v_{2n} \tag{2.2}$$

It can also be seen as the product of a row matrix by a column matrix:

$$\begin{pmatrix} v_{11}, v_{12}, \ldots, v_{1n} \end{pmatrix} \begin{pmatrix} v_{21} \\ v_{22} \\ \vdots \\ v_{2n} \end{pmatrix} = v_{11}v_{21} + v_{12}v_{22} + \ldots + v_{1n}v_{2n} \tag{2.3}$$

2.2.1 Properties

Since the usual scalar product is defined, all its properties are satisfied in Euclidean spaces $\mathbb{E}$ (see [4]):

- Commutativity: $v_1 \cdot v_2 = v_2 \cdot v_1$
- Distributivity: $v_1 \cdot (v_2 + v_3) = v_1 \cdot v_2 + v_1 \cdot v_3$
- Outplacement of the scalar factor: $\alpha(v_1 \cdot v_2) = (\alpha v_1) \cdot v_2 = v_1 \cdot (\alpha v_2)$
- Definite positive: $v \cdot v \geq 0$, $\forall v \in \mathbb{E}$, and $v \cdot v = 0 \Leftrightarrow v = 0$.

In the same way, as it has been seen, the Euclidean space contains two kind of mathematical objects: scalars and vectors. Hence, the properties of this space can be defined in terms of vector addition and scalar multiplication (see [5]):

- Commutativity of addition: $v_1 + v_2 = v_2 + v_1$
- Associativity of addition: $v_1 + (v_2 + v_3) = (v_1 + v_2) + v_3$
- Identity element of addition: $v + 0 = v$
- Inverse element of addition: $-v$. Then $v + (-v) = 0$
- Distributivity of scalar multiplication with respect to vector addition:

$$(v_1 + v_2)\alpha = \alpha v_1 + \alpha v_2$$

- Compatibility of scalar multiplication with field multiplication:

$$\alpha_1(\alpha_2 v) = (\alpha_1\alpha_2)v$$

- Identity element of scalar multiplication: $1v = v$

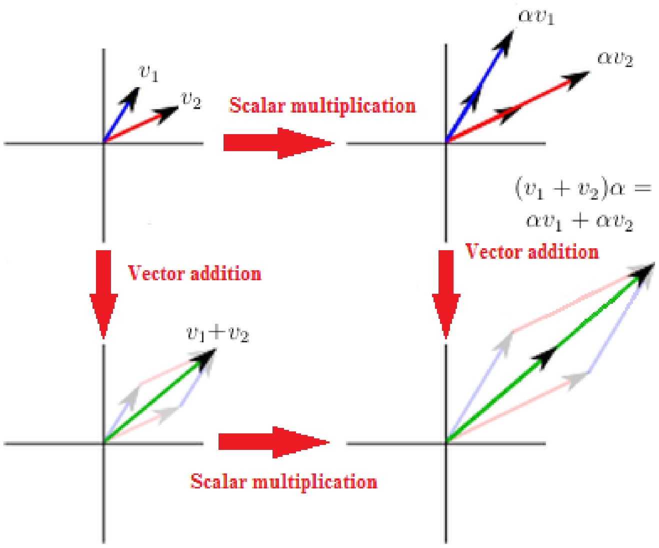

v_1
v_2
Scalar multiplication
αv_1
αv_2
$(v_1 + v_2)\alpha =$
$\alpha v_1 + \alpha v_2$
Vector addition
Vector addition
v_1+v_2
Scalar multiplication

Figure 2.1: Graphic representation of the possible operation with vectors and their equivalences.

2.2.2 Important derivated notions

Some definitions are important when working in Euclidean spaces, because they will be used later in deductions and explanations:

2.2.2.1 Euclidean norm

Definition 6 *The* ***norm*** *of the vector* $v \in \mathbb{E}^n$ *is defined by (see [6]):*

$$\|v\| = \sqrt{v \cdot v} \tag{2.4}$$

This expression can also be written as:

$$\|v\| = \left(\sum_{i=1}^{n} v_i^2 \right)^{1/2} \tag{2.5}$$

Definition 7 *A vector* $v \in \mathbb{E}^n$ *is an* ***unit vector or a normalized vector*** *if* $\|v\| = 1$.

- Example:

  If $v = \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix} \Rightarrow \|v\| = \sqrt{1^2 + 2^2 + 3^2} = \sqrt{14} \Rightarrow$ **It is not an unit vector**.

- Let be v_1,$v_2 \in \mathbb{E}^n$, and $\lambda \in \mathbb{R}^n$. Then it verifies (see [7]):

  1. $\|v_1\| > 0 \Leftrightarrow v_1 \neq 0$
  2. $\|\lambda v_1\| = |\lambda| \|v_1\|$
  3. Cauchy-Schwarz inequality: $|v_1 \cdot v_2| \leq \|v_1\| \|v_2\|$
  4. Minkowski (or triangle) inequality: $\|v_1 + v_2\| \leq \|v_1\| + \|v_2\|$
  5. Pythagoras's theorem: $\|v_1\|^2 + \|v_2\|^2 = \|v_1 + v_2\|^2$

2.2.2.2 Distance

Definition 8 *A* ***metric (or distance function)*** *on* $\mathbb{R}^n$ *between two vectors* $v_1, v_2 \in \mathbb{E}^n$ *is defined as follows:*

$$d(v_1, v_2) = \|v_1 - v_2\| = \sqrt{\sum_{i=1}^{n}(v_{1i} - v_{2i})^2} \tag{2.6}$$

The distance function is called Euclidean metric, and it expresses a special case of the Pythagoras' Theorem.

- Example:

  If $v_1 = \Big(1, 3, 5\Big)$ and $v_2 = \Big(3, 1, 4\Big)$, then:
  $d(v_1, v_2) = \|v_1 - v_2\| = 3$

2.2.2.3 Angles

Definition 9 *The* ***angle*** *formed by two vectors* $v_1, v_2 \in \mathbb{E}^n$, $\widehat{(v_1, v_2)}$ *is defined by its cosine:*

$$cos\widehat{(v_1, v_2)} = \frac{v_1 \cdot v_2}{\|v_1\|\|v_2\|}, \tag{2.7}$$

where $\widehat{v_1, v_2} = \theta + 2k\pi, k \in \mathbb{Z}, \theta \in [o, \pi]$.

- Example:

  If $v_1 = \Big(1, 3, 5\Big)$ and $v_2 = \Big(3, 1, 4\Big) \Rightarrow$
  $(v_1 \hat{,} v_2) = arccos\left(\frac{(1,3,5) \cdot (3,1,4)}{\|(1,3,5)\|\|(3,1,4)\|}\right) = arccos\left(\frac{26}{\sqrt{910}}\right) = 0.53118$ rad

- Properties:

  1. $v_1 \neq 0$ and $v_2 \neq 0$ are linearly dependent $\Leftrightarrow$ $\widehat{v_1, v_2}$ is 0 or π.
  2. $cos\widehat{(\lambda v_1, v_2)} = \frac{\lambda}{|\lambda|} cos\widehat{(v_1, v_2)}$

2.2.2.4 Orthogonality

The concept of orthogonality is defined in any space equipped with the inner product (see [8]):

Definition 10 *Two vectors* $v_1, v_2 \in \mathbb{E}^n$ *are* ***orthogonal*** *if* $v_1 \cdot v_2 = 0$. *In this case, it is written* $v_1 \perp v_2$.

Given a set vectors $\{v_1, v_2, \ldots, v_n\} \in \mathbb{E}^n$, *it can be checked that:*

- $v_i \cdot v_j = 0$ *and* $i \neq j \Rightarrow$ *They are an* ***orthogonal system****.*
- $v_i \cdot v_j = 0$ *if* $i \neq j$ *and* $v_i \cdot v_j = 1$ *if* $i = j \Rightarrow$ *They are an* ***orthonormal system****.*
- Example:

  Every coordinate system is an orthonormal system because each axis is perpendicular with each and every other of the rest and its norm is 1.

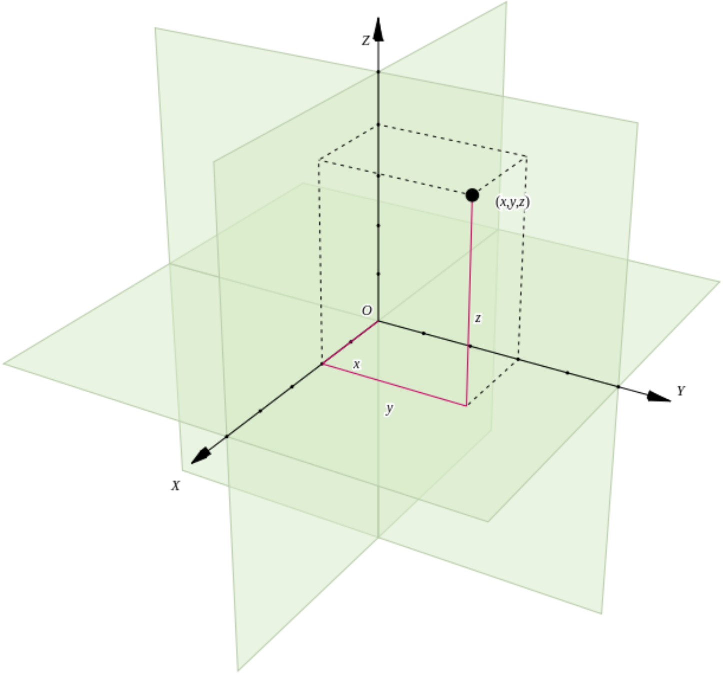


Figure 2.2: Cartesian coordinate system in the three-dimensional space, where the axes form an orthonormal system and each vector can be represented as combination of these basis vectors or axes (see equation (2.1)).

Definition 11 *Given a euclidean vector space with finite dimension $\mathbb{E}$, any orthogonal basis of $\mathbb{E}$ will be called **orthogonal basis** or **orthonormal basis** of $\mathbb{E}$ (depending on the case).*

- Properties for orthogonal systems:
  1. If a system is orthonormal, then is also orthogonal.
  2. Given $v_1, v_2 \neq 0 \in \mathbb{E}^n$, then they create a **right angle**.
  3. The null vector is perpendicular to any vector. It is the only orthogonal vector to all the vectors in $\mathbb{E}^n$.
  4. Given a orthonormal system $v_1, v_2, \ldots, v_n \neq 0$ then the system $\{\frac{v_1}{\|v_1\|}, \frac{v_2}{\|v_2\|}, \ldots, \frac{v_n}{\|v_n\|}\}$ is orthonormal.

To sum up, although Euclidean space is usually presented as a vector space, many others structures can also be defined around it. In this work, all the explained mathematical tools are intended to facilitate the understanding of kinematics. Therefore, the relationship between Euclidean space and the **Lie group** and **Lie Algebra** will be studied in the following sections.

2.3 Manifolds

Definition 12 *An N-dimensional **manifold** M is a topological space where every point $p \in M$ is endowed with local Euclidean structure. In other words, in an infinitely small neighbourhood of a point p the space looks smooth, so it resembles Euclidean space near each point.*

*A manifold has a well defined tangent space at every point. This fact allows us to apply the methods of **Calculus** or **Linear Algebra** to study them. Although near each point a manifold resembles Euclidean space, globally a manifold might not.*

The dimension of a manifold is the dimension of its tangent spaces, so a manifold in $\mathbb{R}^n$ cannot have a higher dimension than n.

- Example: One-dimensional manifolds

  A one-dimensional manifold is a curve without self intersections or peaks. Curves can be closed, unbounded (indicated by arrows) or they can have one or two endpoints, also called boundary points (see [9]).

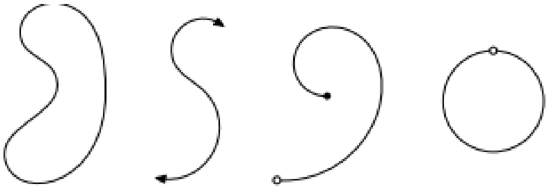


Figure 2.3: Example of one-dimensional manifolds. These curves can be evaluated as lines in local neighbourhoods.

- Example: Two-dimensional manifolds A two-dimensional manifold is a smooth surface without self interactions. It may has a boundary, which is always a one-dimensional manifold. In this kind of manifolds, each point in the surface can be locally analysed by a tangent plane in which the properties of a concrete **algebra** are defined.

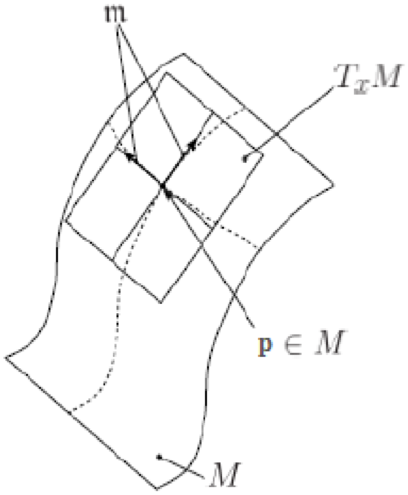


Figure 2.4: Two-dimensional manifold M embedded in three-dimensional space, a point $p \in M$, the tangent space at p (T_xM) and the algebra m, which defines the vector operations.

A very intuitive example of what is not a manifold is the closed square, because its corners are not smooth. However, there are a lot of surfaces which can be evaluated locally as planes along their whole structure.

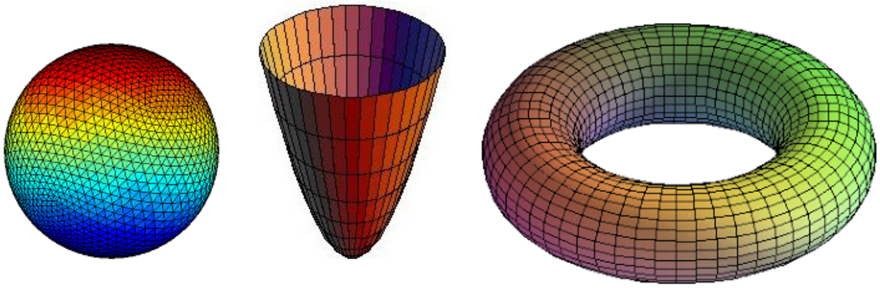


Figure 2.5: Example of two-dimensional manifolds in three-dimensional space, like spheres, paraboloids and torus.

2.3.1 Definitions

2.3.1.1 Charts

Definition 13 *A* ***chart*** *is an invertible map between a subset of the manifold and a simple space such that both the map and its inverse preserve the same structure. In a topological manifold, the simple space is some* ***Euclidean space*** $\mathbb{R}^n$*. In the case of a differentiable manifold, a set of charts is called an* ***atlas, which allows to calculate in manifolds.***

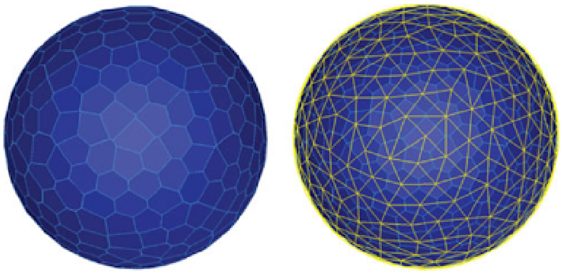


Figure 2.6: On the left side, example of an atlas from a sphere, where all the polygons are charts (local planar approximations). On the right, the map of topological relations between charts.

2.3.1.2 Atlases

Definition 14 *As the description of most manifolds requires more than only a chart, a collection of them is required which covers the whole manifold. This collection is an* ***atlas****. An atlas is not unique because all the manifolds can be covered in multiple ways using different combinations of charts.*

2.3.1.3 Transition function

Definition 15 *Charts in an atlas may overlap, so set of the manifold points may be represented in several charts. If two charts overlap, then different parts of them represent the same region.*

Given two overlapping charts, ϕ_α *and* ϕ_β*, a* ***transition function*** *goes from an open ball in* $\mathbb{R}^n$ *to the manifold and then back to another (or the same) open ball in* $\mathbb{R}^n$*, usually called* $T_{\alpha,\beta} = \phi_\beta \circ {\phi_\alpha}^{-1}$*. These functions,* $T_{\alpha,\beta}$ *and* $T_{\beta,\alpha}$ *are also called* ***transition maps.***

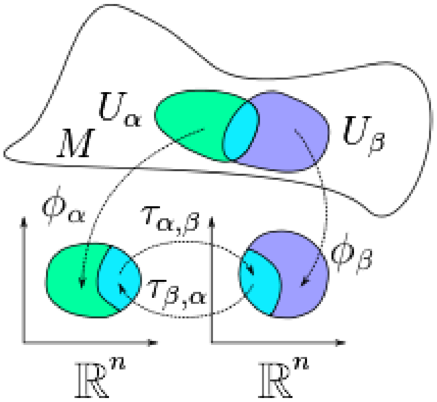


Figure 2.7: Example of a transition function between two charts, where M is the manifold, ϕ_α and ϕ_β are the charts and $T_{\alpha,\beta}$ and $T_{\beta,\alpha}$ are the transition functions for going from one to another.

2.3.2 General example

The idea of manifold and all the important concepts can be easily understood looking at the following pictures:

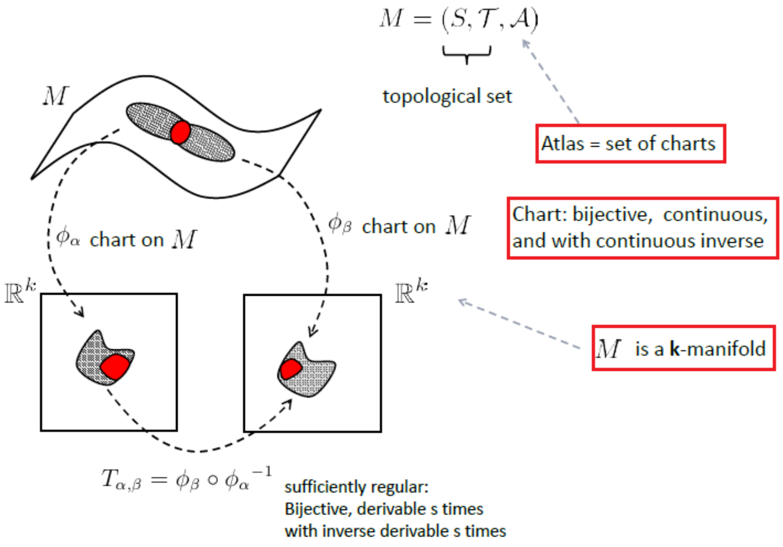

$M = (\underbrace{S, \mathcal{T}}_{\text{topological set}}, \mathcal{A})$
M
Atlas = set of charts
Chart: bijective, continuous, and with continuous inverse
ϕ_α chart on M
ϕ_β chart on M
$\mathbb{R}^k$
$\mathbb{R}^k$
M is a **k**-manifold
$T_{\alpha,\beta} = \phi_\beta \circ {\phi_\alpha}^{-1}$
sufficiently regular:
Bijective, derivable s times
with inverse derivable s times

Figure 2.8: Definition of a manifold M, where it is possible to see two different charts (ϕ_α and ϕ_β), and the transition function $T_{\alpha,\beta}$ between them. Some important concepts are also inside red boxes.

A more general aspect of the charts inside the manifold is bellow:

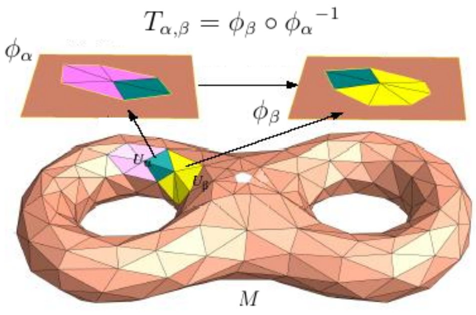

$T_{\alpha,\beta} = \phi_\beta \circ {\phi_\alpha}^{-1}$
ϕ_α
ϕ_β
U_α
U_β
M

Figure 2.9: **Manifold** M composed by a set of **charts** (**atlas**). Two charts (in pink and yellow) have been highlighted to exemplify how they must share a common part (in green) to make the atlas coherent.

2.4 Concepts about groups

2.4.1 Rigid transformations

Definition 16 *A* ***rigid transformation****, also called* ***isometry****, is a transformation in which the distance between points is always constant, in other words, a transformation where the output space is the same as the input one (see [10]).*

$$d(P, Q) = d\left(T(P), T(Q)\right) \tag{2.8}$$

Properties:

- Those transformations are injective (invertible) so the composition of two rigid transformations is another rigid transformation. Thus, a rigid transformation does not distort the size or shape of the transformed body.

$$d\left(T_2 \circ T_1(P), T_2 \circ T_1(Q)\right) = d\left((T_2(T_1(P)), T_2(T_1(Q))\right) \tag{2.9}$$

$$d(T_1(P), T_2(Q)) = d(P, Q) \tag{2.10}$$

- These equalities are due to the rigidity of T_1 and T_2, respectively. The rigid compositions in the Cartesian space form in this way a group under the composition.

As it will be seen next, these transformations are defined by matrices, so the group is not commutative. The group of rigid transformations is the special Euclidean group, $E(n)$. When the reflections are not included (**proper rigid transformations**), the group is called special Euclidean group, $SE(n)$.

2.4.2 Congruences and Symmetry group

Definition 17 *Two figures are **congruent** if ones can be changed into another using a combination of rotations, reflections and translations.*

*When a figure is congruent with itself in more than one way, these extra congruences are called **symmetries**. Hence, **the symmetric group** is the group of isometries under which an object is **invariant with composition** as operation.*

A symmetry group whose shape is not distorted can be represented as the subgroup of **orthogonal group** $O(n)$.

The **proper symmetry group** (the subgroup of orientation-preserving isometries) is a subgroup of the **special orthogonal group** $SO(n)$, and is also called rotation group of the figure. The difference between $O(n)$ and $SO(n)$ is that $SO(n)$ **does not allow reflections**.

- Example:

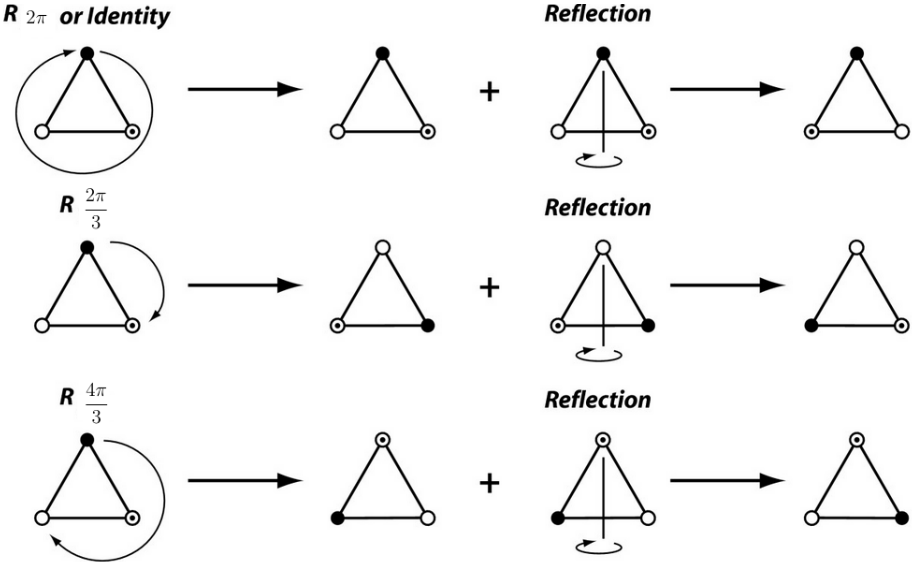


Figure 2.10: The set of symmetry transformations which defines the symmetry group of the equilateral triangle: the identity, rotations of order 3, and combinations of rotations and reflections.

2.4.3 Lie groups

Definition 18 *A **Lie group** is a smooth manifold M which has at the same time a group structure consistent with its manifold, so the group operations are compatible with the smooth structure of the manifold (multiplication, inversion and identity element). In consequence, this manifold is **differentiable** (see [11]).*

In other words, a Lie group is a group which locally has the topology of $\mathbb{R}^n$ around each one of its elements.

- Example. The Linear General Group (see [12]):

  Let $GL(n, \mathbb{R})$ be the group of square matrices $n \times n$ which are invertible. This is a non-commutative Lie group of dimension n^2. As $GL(n, \mathbb{R})$ is an **open set** in $\mathbb{R}^{n^2}$ (which contains an open ball around each of its points and finding a boundary is impossible), then it is also a differentiable manifold, because the product of matrices is a differentiable application which can be reduced to elementary operations.

$$(AB)_{ij} = \sum_{k=1}^{m} A_{ik} B_{kj} \tag{2.11}$$

2.4.3.1 Lie algebras

Definition 19 *The **Lie algebra** m associated to a Lie group is the tangent **vector space** to the manifold M at the identity element I. It has the same dimension as the manifold.*

$$m = T_M(I) \tag{2.12}$$

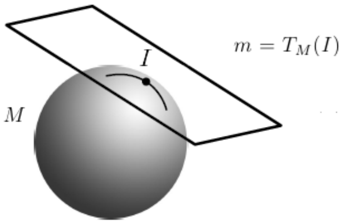


Figure 2.11: The Lie algebra m is the tangent space of M at the identity.

In a formal way, a Lie algebra is an algebra together with a Lie bracket operator $[\cdot,\cdot] : m \times m \to m$ *such as for any elements* a, b *and* $c \in m$ *it holds the following properties (see [13]):*

$$[\;] : m \times m \longrightarrow m \tag{2.13}$$

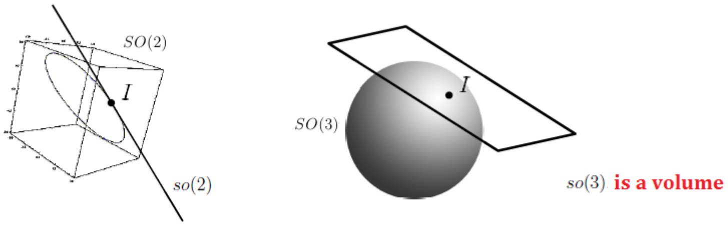
$$(a,b) \longrightarrow [a,b] = ab - ba \tag{2.14}$$

1. ***Bilinearity.***
2. ***Skew simmetry:*** $[a,a] = 0$
3. ***Anti-commutativity:*** $[a,b] = -[b,a]$
4. ***Jacobi's identity:*** $[a,[b,c] + [b,[c,a]] + [c,[a,b]] = 0$

- Example. In order to make the notation easier, as of now $T_{SO(2)} = so(2)$ and $T_{SO(3)} = so(3)$.

  $so(2)$ is the associated Lie Algebra to the Lie group $SO(2)$ and $so(3)$ is the associated Lie Algebra to the Lie group $SO(3)$. These groups are the set of rotation matrices in two and three dimensions, respectively. Both will be studied in the following chapters.

  Since lines and circles are one-dimensional manifolds (parametrized with curves), the Lie algebra of $SO(2)$ has dimension 1. In the case of $SO(3)$, which is a three-dimensional manifold (with three possible rotations), its Lie algebra defines a three-dimensional vector space.



An element in so(2) or so(3) represents an infinitesimal rotation from the identity matrix

Figure 2.12: Lie algebras $so(2)$ and $so(3)$ in relation to the groups $SO(2)$ and $S0(3)$.

2.4.4 The special Euclidean Group in three dimensions, $SE(3)$, and relevant subgroups

Definition 20 *The group $SE(n)$, **special Euclidean group**, is the group which includes all the rigid body transformations (isometries without reflections) in $\mathbb{R}^n$.*

In order to find utility in the physical world, one of the subgroups of $SE(n)$ will be focused: $SE(3)$. This group comprises of all the possible transformations in $\mathbb{R}^3$, in other words, any possible movement of rotation or translation. The members of this are the set of 4×4 matrices with the following structure (see [14]):

$$\left(\begin{array}{c|c} R(\theta) & t \\ \hline 0_{1\times 3} & 1 \end{array}\right), \tag{2.15}$$

where the rotation $R(\theta) \in S0(3)$ and the possible translations $t = [t_x, t_y, t_z]^T \in \mathbb{R}^3$. This way of representing transformations will be studied in detail in sections 3.2 and 4.2. Rotations will be define in the next chapter (see 4.1.1).

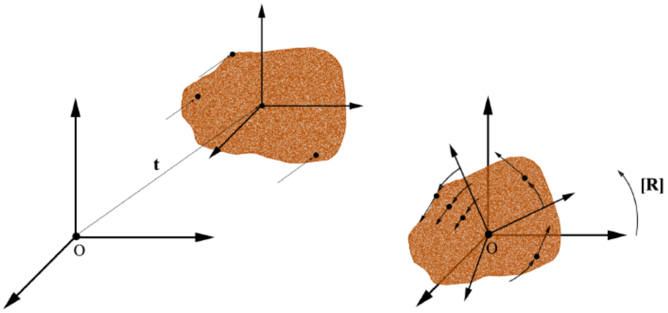


Figure 2.13: Translation and rotation by rigid body movements.

- Properties:
    - $SE(3)$ is a six-dimensional manifold, since it has six degrees of freedom (this concept will be explained in more detail later on): three for 3D translations and three for 3D rotations.
    - As $SE(3)$ is embedded in the more general $GL(4,\mathbb{R})$, then it also a **Lie group**.

- In addition, since $SE(3)$ is a group, it satisfies that:
  - The set is closed under the binary operation of multiplication: if $A, B \in SE(3) \Rightarrow AB \in SE(3)$.
  - This binary operation is associative: $(AB)C = A(BC)$
  - For every element $A \in SE(3)$, there is an identity element given by the 4×4 identity matrix, $I \in SE(3)$, such that $AI = IA = A$ (rotation of $k\pi$ with a null translation).
  - For every element $A \in SE(3)$, there is an identity inverse, $A^{-1} \in SE(3)$, such that $AA^{-1} = A^{-1}A = I$. All the matrices are square, so all of them have inverse.

2.4.4.1 Subgroups

There are many other groups which are interesting in **rigid body kinematics**, and they are subgroups of $SE(3)$. A **subgroup** consists of a collection of the group elements which themselves form a group with the same binary operation. Of course, all these groups satisfy the **group properties** simultaneously in each case (see [15]).

- The group of rotations in three dimensions: $SO(3)$ (chapter 4)

  Definition: It is the set of all the proper orthogonal 3×3 matrices:
  $SO(3) = R : R \in \mathbb{R}^{3\times 3}, \mathbb{R}^T\mathbb{R} = \mathbb{R}\mathbb{R}^T = I$

  Interpretation: All the spherical displacements. The set of rotations which can be generated by a spherical joint.

- The special Euclidean group in two dimensions: $SE(2)$

  Definition: The set of all 3×3 matrices with the structure:

  $$\begin{pmatrix} cos\theta & -sin\theta & t_x \\ sin\theta & cos\theta & t_y \\ 0 & 0 & 1 \end{pmatrix}$$

  , where θ is the rotation angle and t_x and t_y the defined displacements,

  Interpretation: All the planar displacements and rotations possible.

- The group of rotations in two dimensions: $SE(2)$ (chapter 3)

  Definition: It is the set of all the proper orthogonal 2×2 matrices:

$$\begin{pmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{pmatrix}$$

  Interpretation: All the rotations in the plane. The set of all the displacements which a single revolute joint can do.

- The groups of translations in $3D$, $2D$ and $1D$: $T(i)$ with $i = \{1, 2, 3\}$

  Definition: The set of all 3×1, 2×1 vectors or real numbers with vector addition (or simply addition in the last case) as binary operation.

  Interpretation: All the possible direct displacements conserving orientations in the the three-dimensional space ($n = 3$) (section 4.1.1), the Cartesian plane ($n = 2$) (section 3.1.1) or parallel to an axis ($n = 1$). In this last circumstance, it is the set of displacements which a single prismatic joint can do.

2.5 The exponential map

Definition 21 *The exponential map is a function which maps elements from the* ***Lie algebra*** *to the correspondent* ***Lie group*** *(manifold) (see [16]).*

$$exp : m \to M \tag{2.16}$$

For any square matrix M, the exponential map e^M is well defined and coincides with the matrix exponentiation, which can be written:

$$e^M = \sum_{k=0}^{\infty} \frac{1}{k!} M^k = I + M + \frac{M^2}{2!} + \frac{M^3}{3!} + \frac{M^4}{4!} + \dots \tag{2.17}$$

- **Properties [17]:**

1. It is a smooth map.
2. It is surjective (it covers the Lie group entirely).
3. It is not injective (there are many points to map one).
4. Identity: $e^{M=0} = e^0 = I$
5. Inverse: $e^{-M} = (e^M)^{-1}$
6. In general, it is non-linear: $e^{M_1+M_2} \neq e^{M_1}e^{M_2}$ and $e^{M_1}e^{M_2} \neq e^{M_2}e^{M_1}$
7. Exponential properties: $e^{\alpha M+\beta M} = e^{\alpha M}e^{\beta M}$, with $\alpha, \beta \in \mathbb{R}$.
8. Derivative: $\partial e^M = \partial M e^M = e^M \partial M$
9. In $SO(3)$, the exponential map coincides with the rotation's Rodrigues formula (see 4.1.4.1).

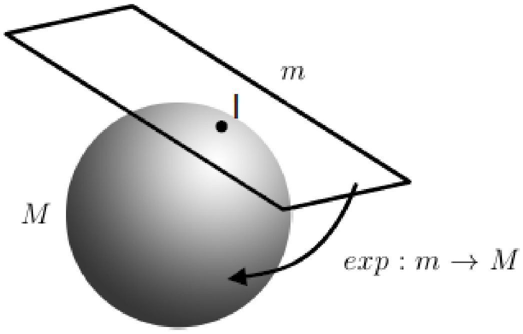


Figure 2.14: Graphical interpretation of the exponential map.

- Example. $SO(2)$ (see [18]):

$so(2)$ is one-dimensional, and it is possible to take as basis:

$$M(\theta) = \begin{pmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{pmatrix} \tag{2.18}$$

Near the identity, the tangent vector is calculated taking into account that $\theta \approx 0$. In $\theta = 0$:

$$\frac{dM(\theta)}{d\theta} = \begin{pmatrix} -sin0 & cos0 \\ sin0 & cos0 \end{pmatrix} = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} \tag{2.19}$$

A generic element of $so(2)$ is θM, with $\theta \in \mathbf{R}$. Then a generic element of $SO(2)$ is $m(\theta) = e^{\theta M}$. Considering that $M^2 = -1$:

$$m(\theta) = 1 + \theta M + \frac{\theta^2 M^2}{2!} + \frac{\theta^3 M^3}{3!} + \frac{\theta^4 M^4}{4!} + \ldots \tag{2.20}$$

$$m(\theta) = (1 - \frac{1}{2}\theta^2 + \ldots + \frac{(-1)^n}{(2n)!}\theta^{2n} + \ldots)1 + (\theta - \frac{1}{3!} + \ldots + \frac{(-1)^{n+1}}{(2n+1)!}\theta^{2n+1} + \ldots)M \tag{2.21}$$

Finally:

$$m(\theta) = cos\theta 1 + sin\theta L = \begin{pmatrix} cos\theta & sin\theta \\ -sin\theta & cos\theta \end{pmatrix} \tag{2.22}$$

2.5.1 The logarithm map

Definition 22 *Since the **exponential map** is surjective, at least an inverse exists. This inverse is the **logarithm map**, which map the elements from the manifold to the algebra.*

$$ln : M \to m \tag{2.23}$$

- **Properties [19]:**

  1. It is a smooth map.
  2. It is surjective (it covers the Lie algebra entirely).
  3. It is not injective (there are many points to map one).
  4. Identity: $ln(I) = 0 = (M = 0)$
  5. Inverse: $ln(M^{-1}) = -log(M)$
  6. In general, it is non-linear: $ln(M_1 M_2) \neq ln(M_1) + ln(M_2)$
  7. Exponential properties: $e^{ln(M)} = M$
  8. Derivative: $\partial ln(M) = M^{-1} \partial M$

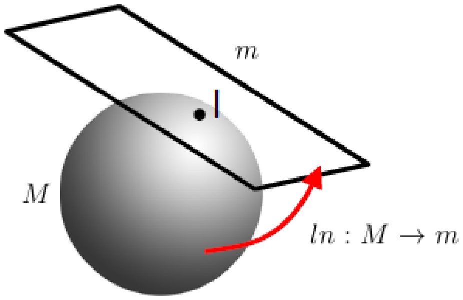


Figure 2.15: Graphical interpretation of the logarithm map.

Chapter 3

The group SO(2): Bi-dimensional rotations

Definition 23 *The* ***special orthogonal group in two dimensions****, SO(2), is defined by the set of uni-modular, real and orthogonal* 2×2 *matrices (see [20]).*

It is possible to construct a matrix of the form:

$$M = \begin{pmatrix} a & c \\ b & d \end{pmatrix} \tag{3.1}$$

where **a, b, d** and **d** are real numbers which verify $|M| = ad - cb = 1$
(as mentioned in section 3.1.2, last property).

Assuming **M** is orthogonal then $M^{-1} = M^T$. To accomplish that, **a** must be equal to **d** and **c** to **-b**, so the new matrix is:

$$M_2 = \begin{pmatrix} a & -b \\ b & a \end{pmatrix} \tag{3.2}$$

Given M_2, the condition for the determinant requires that $a^2 + b^2 = 1$, so the values for **a** and **b** are $a = cos\theta$ (or $sin\theta$) and $b = sin\theta$ (or $cos\theta$, respectively).

When $\theta \in [-\pi, +\pi]$, then it all the uni-modular, real and orthogonal 2x2 matrices can be obtained. This can be demonstrate by proving that $M^{-1} = M^T$, in other words, $M \cdot M^T = I$ (see [21]):

$$M \cdot M^T = \begin{pmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{pmatrix} \cdot \begin{pmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{pmatrix} = \tag{3.3}$$

$$\begin{pmatrix} cos^2\theta + sen^2\theta & cos\theta sin\theta - cos\theta sin\theta \\ sin\theta cos\theta - cos\theta sin\theta & sin^2\theta + cos^2\theta \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \tag{3.4}$$

So all the possible rotations in the plane are indeed in **SO(2)**.

3.1 Planar rotations and translations

3.1.1 2D translations

Definition 24 *Given a position in the plane $v = (x, y) \in \mathbb{R}^2$ in such way that $v = (x, y)^T$, a **translation** is only a displacement denoted by $t = (t_x, t_y)^T$ which preserve the distance between points only adding the values t_x and t_y to the original coordinates (see [22]).*

Mathematically,

$$v_1 = v + t = \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} t_x \\ t_y \end{pmatrix} = \begin{pmatrix} x + t_x \\ y + t_y \end{pmatrix}, \tag{3.5}$$

where v_1 is new vector composed by the original one plus the displacement.

- The main property of translations is the effect of two or more successive translations which is given by the sum of the translations implicated.

$$v_2 = v_1 + t_2 = v + t_1 + t_2 \;\mid\; v_3 = v_2 + t_3 = v + t_1 + t_2 + t_3 \tag{3.6}$$

NOTE: An example of translations will be given in combination with rotations at the end of the chapter (see 3.2).

3.1.2 2D rotations

A rotation around the origin in a Cartesian space is represented by a 2 × 2 matrix with the following form:

$$R\theta = \begin{pmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{pmatrix} \tag{3.7}$$

If there is a vector $v = (x, y)^T$ with a magnitude r and an angle from the x-axis ϕ, the multiplication of this matrix $R(\theta)$ by the vector will give another vector with the same magnitude but displaced $\theta + \phi$ from the x-axis in an anticlockwise rotation. It is easy to see using the polar form of the vector v:

$$v_1 = R\theta v = \begin{pmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{pmatrix} \begin{pmatrix} rcos\phi \\ rsin\phi \end{pmatrix} =$$
$$\begin{pmatrix} rcos\theta cos\phi - rsin\theta sin\phi \\ rsin\theta cos\phi + rcos\theta sin\phi \end{pmatrix} = \begin{pmatrix} rcos(\theta + \phi) \\ rsin(\theta + \phi) \end{pmatrix}, \tag{3.8}$$

where v_1 is the new vector resulted by the rotation.

Thus, a couple of properties are deduced (see [23]):

- A rotation of 0 radians has no effect because the resulting matrix is the identity matrix and any multiplication will keep constant the original one:

$$R(0) = \begin{pmatrix} cos0 & -sin0 \\ sin0 & cos0 \end{pmatrix} = R(0) = \begin{pmatrix} cos1 & -sin0 \\ sin0 & cos1 \end{pmatrix} \tag{3.9}$$

- The effect of successive rotations can be written as matrix multiplication:

  $$v_2 = R(\theta_1)v_1 = R(\theta_2)R(\theta_1)v \quad | \quad v_3 = R(\theta_3)v_2 = R(\theta_3)R(\theta_2)R(\theta_1)v \tag{3.10}$$

  That is true because $R(\theta_2)R(\theta_1) = R(\theta_2 + \theta_1)$ and it can be checked using standard trigonometric identities.

- The inverse of a rotation matrix can be understood by a rotation in the opposite direction by the properties of sine and cosine functions, so

  $$R(\theta)^{-1} = R(-\theta) \tag{3.11}$$

  Therefore, the main conclusion derived is

  $$R(\theta)^{-1}R(\theta) = R(-\theta)R(\theta) = R(-\theta + \theta) = R(0) = I$$

  Hence, this matrix multiplication has no effect.

- The determinant of a rotation matrix is always equal to one.

  $$det(R(\theta)) = \begin{vmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{vmatrix} = cos^2\theta + sin^2\theta = 1 \tag{3.12}$$

  This last property sustains that a rotation does not deform the shape of the rotated body by increasing or reducing its size, only spins it around. In addition, the determinant is positive because there is not reflection in the transformation neither. These two features coincide with the postulates in section 2.4.1.

- **Example. 2D rotation matrices**

  At this point, an example will be shown using the software **MATLAB** in order to exemplify what has been described before. The aim of this example is to rotate a vector $v = (1,0)^T$ twice. First by an angle of $2\pi/3$ and after that by an angle of π counterclockwise.

```
>> v1=[1 0]';
>> Rot_matrix=[cos(2*pi/3) -sin(2*pi/3); sin(2*pi/3) cos(2*pi/3)];
>> v2=Rot_matrix*v1

v2 =
   -0.5000
    0.8660

>> Rot_matrix=[cos(pi) -sin(pi); sin(pi) cos(pi)];
>> v3=Rot_matrix*v2

v3 =
    0.5000
   -0.8660
```

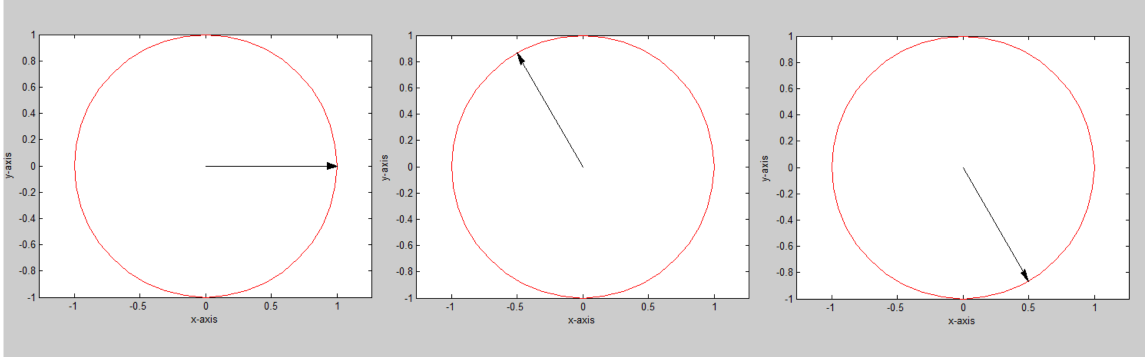


Figure 3.1: Plots of an arrow before and after the specified rotations ($2\pi/3$ and π). It has been inscribed inside a circle to visualize how the modulus is not altered. The code details of this plot on MATLAB are contained in the **Appendix C.1**.

3.1.2.1 Rotation by complex numbers

A rotation in the Cartesian space can be also implemented by complex numbers. It must be remembered that the position in the plane can be specified with a complex number, where the axis **x** and **y** reference the real axis and the imaginary one. A complex number has two main ways of being expressed (see [24]):

- Cartesian form:

  $$C = a + ib, \tag{3.13}$$

  where a and $b \epsilon \mathbb{R}$ and i, $C \in \mathbb{C}$.

  The module of a complex number is the square root of its squared elements. By dividing a complex number by its module, one obtains the normalized complex.

- Polar form (or Euler form): It indicates the modulus and argument (angle) of the position vector in the plane:

  $$C = re^{i\theta}, \tag{3.14}$$

  where **r** is the modulus (calculated as indicated before) and θ is the argument (which can be calculated as $tan^{-1} = \frac{b}{a}$).

  If the complex is normalized, the modulus is equal to 1 and the factor **r** disappears.

The rotation of a vector in the plane is the result of a multiplication between two or more exponentials. If the vector has unitary norm:

$$e^{i\phi} \cdot e^{i\theta} = e^{i(\phi+\theta)} = e^{i(\theta+\phi)} \tag{3.15}$$

- **Example. 2D rotation by complex numbers using MATLAB**

  In a similar way to the previous example, the same vector will be rotated first by and angle of $\frac{2\pi}{3}$ and later π. To do that with complex numbers, the first step consists in expressing the vector in Euler form:

  The vector $v = (1,0)^T$ can be written in the complex plane as $v = 0 + i$:

```
>> v1=1+0*i;
>> abs(v1)

ans =
     1

>> angle(v1)

ans =
     0
```

  Then, $v = (1,0)^T = 1e^{i0}$. Rotating this vector:

```
>> v1=1*exp(i*0);
>> v2=v1*exp(i*2*pi/3)

v2 =
  -0.5000 + 0.8660i

>> v3=v2*exp(i*pi)

v3 =
   0.5000 - 0.8660i
```

These complex numbers in Cartesian form are equivalent to the vectors $v2 = (-0.5, 0.8660)^T$ and $v3 = (0.5, 0.8660)^T$, which are coincident with the results in the previous example and with the figure 3.2. Anyway, the final rotation can be also calculated with a single operation:

```
>> v1=1*exp(i*0);
>> v3=v1*exp(i*2*pi/3)*exp(i*pi)

v3 =
   0.5000 - 0.8660i
```

To sum up, the way in which vectors change its argument inside the unitary circumference as the following figure illustrates:

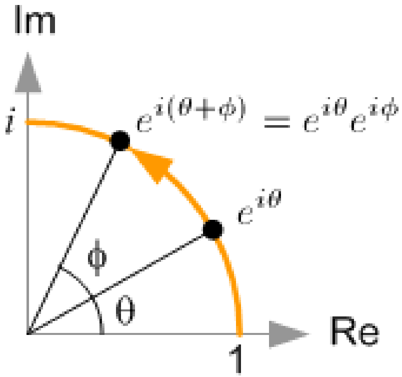


Figure 3.2: Effect of the rotation in the plane by complex numbers.

3.2 Method for defining planar movements

The way of specifying transformations on the plane is by 3 × 3 matrices which store this information. The two first columns are relative to a rotation angle and the last one to a translation on the plane. Since matrices must be square, the third row is independent and formed by zeros in relation with the rotation columns and a one in the last position (the relative to translation) to keep the property of not to be distorted. Hence, the matrix which denotes a transformation on the plane has the following aspect (see [25]):

$$T = \begin{pmatrix} cos\theta & -sin\theta & t_x \\ sin\theta & cos\theta & t_y \\ 0 & 0 & 1 \end{pmatrix} = \left(\begin{array}{c|c} R(\theta) & t \\ \hline 0 & 1 \end{array}\right) \tag{3.16}$$

It is important to clarify that these two kinds of transformation cannot take place simultaneously, because the result is different. Anyway, there is no robotic joint able to implement these two movements at the same time.

Developing different transformations in cascade, it is possible to see how:

$$\left(\begin{array}{c|c} R(\theta_1) & t_1 \\ \hline 0 & 1 \end{array}\right) \left(\begin{array}{c|c} R(\theta_2) & t_2 \\ \hline 0 & 1 \end{array}\right) = \left(\begin{array}{c|c} R(\theta_2+\theta_1) & R(\theta_2)t_1 + t_2 \\ \hline 0 & 1 \end{array}\right) \tag{3.17}$$

This fact can be shown making the corresponding operations:

$$\begin{pmatrix} cos\theta_2 & -sin\theta_2 & t_{2x} \\ sin\theta_2 & cos\theta_2 & t_{2y} \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} cos\theta_1 & -sin\theta_1 & t_{1x} \\ sin\theta_1 & cos\theta_1 & t_{1y} \\ 0 & 0 & 1 \end{pmatrix} = \tag{3.18}$$

$$\begin{pmatrix} cos(\theta_2+\theta_1) & -sin(\theta_2+\theta_1) & t_{1x}cos\theta_2 - t_{1y}sin\theta_2 + t_{2x} \\ sin(\theta_2+\theta_1) & cos(\theta_2+\theta_1) & t_{1x}sin\theta_2 + t_{1y}cos\theta_2 + t_{2y} \\ 0 & 0 & 1 \end{pmatrix} \tag{3.19}$$

- **Example. 2D transformations by using homogeneous transformation matrices in MATLAB**

  The following example consists in the implementation of a rotation followed by a translation on the plane. The initial vector is $v = (1, 0)^T$ and will be rotated by an angle of $-\pi/3$. After that, and using a matrix with same structure, the resulting vector will be displaced in 2 units in the x-axis and -1 in the y-axis. In order to be able to apply the product of matrices, the initial vector will be also completed with a 1 in an additional row.

```
>> v=[1,0,1]'

v =
     1
     0
     1

>> T_mat=[cos(-pi/3) -sin(-pi/3) 0; sin(-pi/3) cos(-pi/3) 0; 0 0 1];
>> v2=T_mat*v

v2 =
    0.5000
   -0.8660
    1.0000

>> T_mat=[1 0 2; 0 1 -1; 0 0 1];
>> v3=T_mat*v2

v3 =
    2.5000
   -1.8660
    1.0000
```

The previous transformations can also be observed graphically in the figure below (3.3). As always, the details of the script to code this plot on MATLAB are contained in the **Appendix C.2.**:

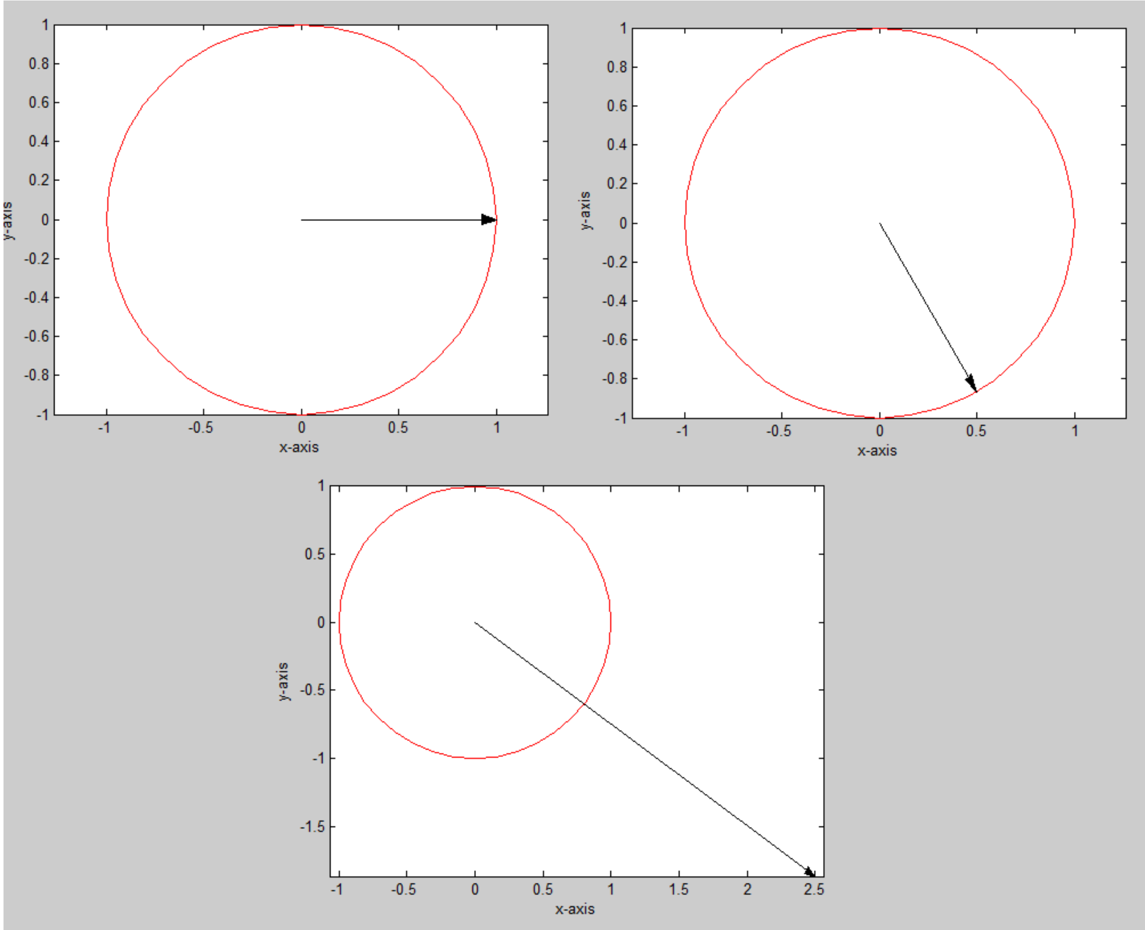


Figure 3.3: Plots of an arrow which represents the position on the vector intending to specify the different transformations, the rotation by $-\pi/3$ and the displacement by $t_x = 2$ and $t_y = -1$. Now the circle can be used as reference to see how after the rotation $v_2 = (-0.866, 0.5)$ the vector has left the unit circumference to sum what indicated in each axis.

Chapter 4

The group SO(3): Three-dimensional rotations

Definition 25 *In an analogue way to SO(2), the* ***special orthogonal group in three dimensions, SO(3)****, is defined by the set of uni-modular, real and orthogonal* 3×3 *matrices. In contrast with this last one, the rotation group SO(3) is non-Abelian because the multiplication of rotations in three-dimensional space becomes non-commutative.*

The matrices which represent a rotation in each spatial axis are these ones:

$$R(\Psi, x) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & cos\Psi & -sin\Psi \\ 0 & sin\Psi & cos\Psi \end{pmatrix} \tag{4.1}$$

$$R(\Theta, y) = \begin{pmatrix} cos\Theta & 0 & sin\Theta \\ 0 & 1 & 0 \\ -sin\Theta & 0 & cos\Theta \end{pmatrix} \tag{4.2}$$

$$R(\Phi, z) = \begin{pmatrix} cos\Phi & -sin\Phi & 0 \\ sin\Phi & cos\Phi & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{4.3}$$

where the three matrices represent, in this order, a rotation of Ψ along the x-axis, a rotation of Θ along the y-axis and a rotation of Φ along the z-axis.

Trivially it can be tested that these matrices are **uni-modular** (its determinant is equal to 1). In addition, all these matrices are orthogonal.

$$R(\Psi, x)R(\Psi, x)^T = I \qquad R(\Theta, y)R(\Theta, y)^T = I \qquad R(\Phi, z)R(\Phi, z)^T = I \quad (4.4)$$

Hence, all the possible spatial rotations are also (like in the case of the possible rotations in **SO(2)**) indeed in **SO(3)**. The lengths of position vectors are preserved and do not contain reflections.

4.1 Spatial rotations and traslations

4.1.1 3D translations

Definition 26 *A* ***translation in space*** *is completely analogue to a translation in plane. Consequently, if there is a position in space v given by three elements (x, y, z) and expressed as $v = (x, y, z)^T$, a translation can be defined as a displacement denoted by $t = (t_x, t_y, t_z)^T$ which preserve the distance between points only adding the values t_x, t_y and t_z to the original coordinates (see [26]).*

$$v_1 = v + t = \begin{pmatrix} x \\ y \\ z \end{pmatrix} + \begin{pmatrix} t_x \\ t_y \\ t_z \end{pmatrix} = \begin{pmatrix} x + t_x \\ y + t_y \\ z + t_z \end{pmatrix} \quad (4.5)$$

- In 3D, the property of translations which sustains that the effect of two or more successive translations is defined by the sum of the translations involved is still working (see section 3.1.1).

$$v_2 = v_1 + t_2 = v + t_1 + t_2 \;\mid\; v_3 = v_2 + t_3 = v + t_1 + t_2 + t_3 \quad (4.6)$$

where v_1, v_2 and v_1 are different vectors obtained from the displacement along a distance t as indicated in each axis.
NOTE: An example of translations will be given in combination with rotations at the end of the chapter (see 4.2).

4.1.2 Rotation by Euler angles

Definition 27 ***Euler angles*** *are probably the best common way of representing rotations in three-dimensional space. This method is based on the fact that any rotation may be described using three angles, as the Euler's rotation theorem sustains (see [27]).*

There are several configurations depending on the axes about which the rotations are carried out, but the most used one is the ***x-convention(also called 3-1-3 or z-x-z)****. This convention consists in a starting rotation by an angle Φ around the z-axis followed by Θ around the x-axis (with $\Theta \in [0, \pi]$) and finally a third one by Ψ around the z-axis again (see [28]).*

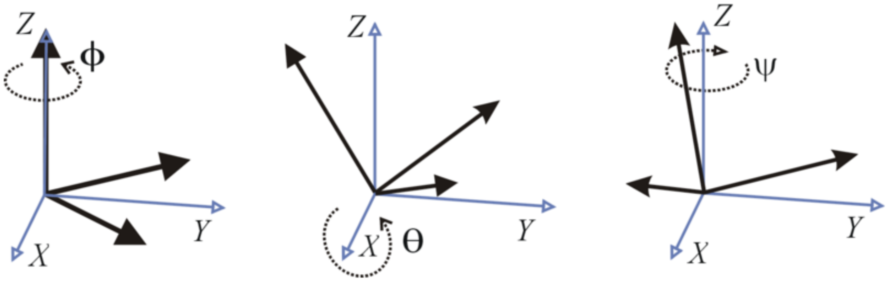


Figure 4.1: Single representation of the corresponding angles around the cited axes and its effect in each case.

These rotations are represented by the following matrices:

$$R(\Psi, z) = \begin{pmatrix} cos\Psi & -sin\Psi & 0 \\ sin\Psi & cos\Psi & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{4.7}$$

$$R(\Theta, x) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & cos\Theta & -sin\Theta \\ 0 & -sin\Theta & cos\Theta \end{pmatrix} \tag{4.8}$$

$$R(\Phi, z) = \begin{pmatrix} cos\Phi & -sin\Phi & 0 \\ sin\Phi & cos\Phi & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{4.9}$$

The effect of the three-dimensional rotation around the body will be given by the successive multiplication of the three partial rotations, with the following result:

$$R(\Phi,\Theta,\Psi) = R(\Psi,z)\cdot R(\Theta,x)\cdot R(\Phi,z) \tag{4.10}$$

$$R(\Phi,\Theta,\Psi) = \begin{pmatrix} cos\Psi cos\Phi - cos\Theta sin\Phi sin\Psi & cos\Psi sin\Phi + cos\Theta cos\Phi cos\Psi & sin\Psi sin\Theta \\ -sin\Psi cos\Phi - cos\Theta sin\Phi cos\Psi & -sin\Psi sin\Phi + cos\Theta cos\Phi cos\Psi & cos\Psi sin\Theta \\ sin\Theta sin\Phi & -sin\Theta cos\Phi & cos\Theta \end{pmatrix} \tag{4.11}$$

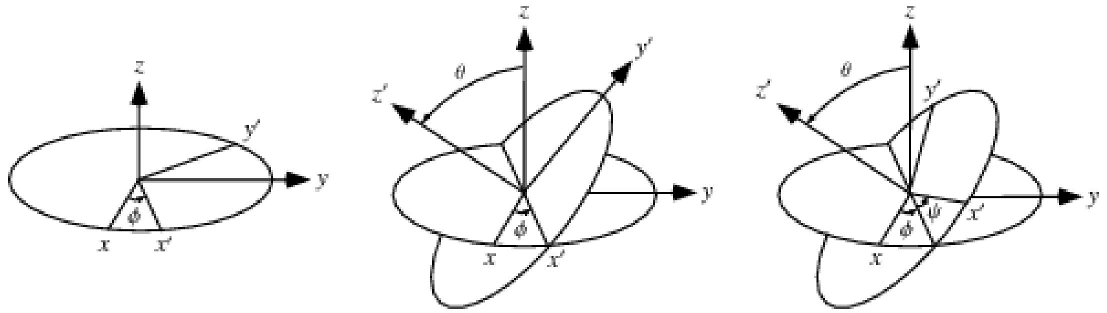


Figure 4.2: Graphical representation of the successive partial rotations on the final result.

- **Example. 3D rotation by Euler angles**

  Using **MATLAB**, an initial vector will be rotated along three times following the **x-convention**. This initial vector will be defined as $v = (1, 0, 0)$ and the rotations angles as $\Psi = \pi$, $\Theta = 2\pi/3$ and $\Phi = -\pi/2$:

```
>> v=[1,0,0]'
v =
     1
     0
     0

>> Rot_psi=[cos(pi) -sin(pi) 0; sin(pi) cos(pi) 0; 0 0 1];
>> v2=Rot_psi*v
v2 =
   -1.0000
    0.0000
         0

>> Rot_theta=[cos(2*pi/3) 0 sin(2*pi/3);0 1 0;-sin(2*pi/3) 0 cos(2*pi/3)];
>> v3=Rot_theta*v2
v3 =
    0.5000
    0.0000
    0.8660

>> Rot_phi=[cos(-pi/2) -sin(-pi/2) 0; sin(-pi/2) cos(-pi/2) 0; 0 0 1];
>> v4=Rot_phi*v3
v4 =
    0.0000
   -0.5000
    0.8660
```

The successive effects in the previous vector are exemplified in figure 4.3, so it is possible to see how any orientation in three-dimensional space can be reached by using the rotations defined before.

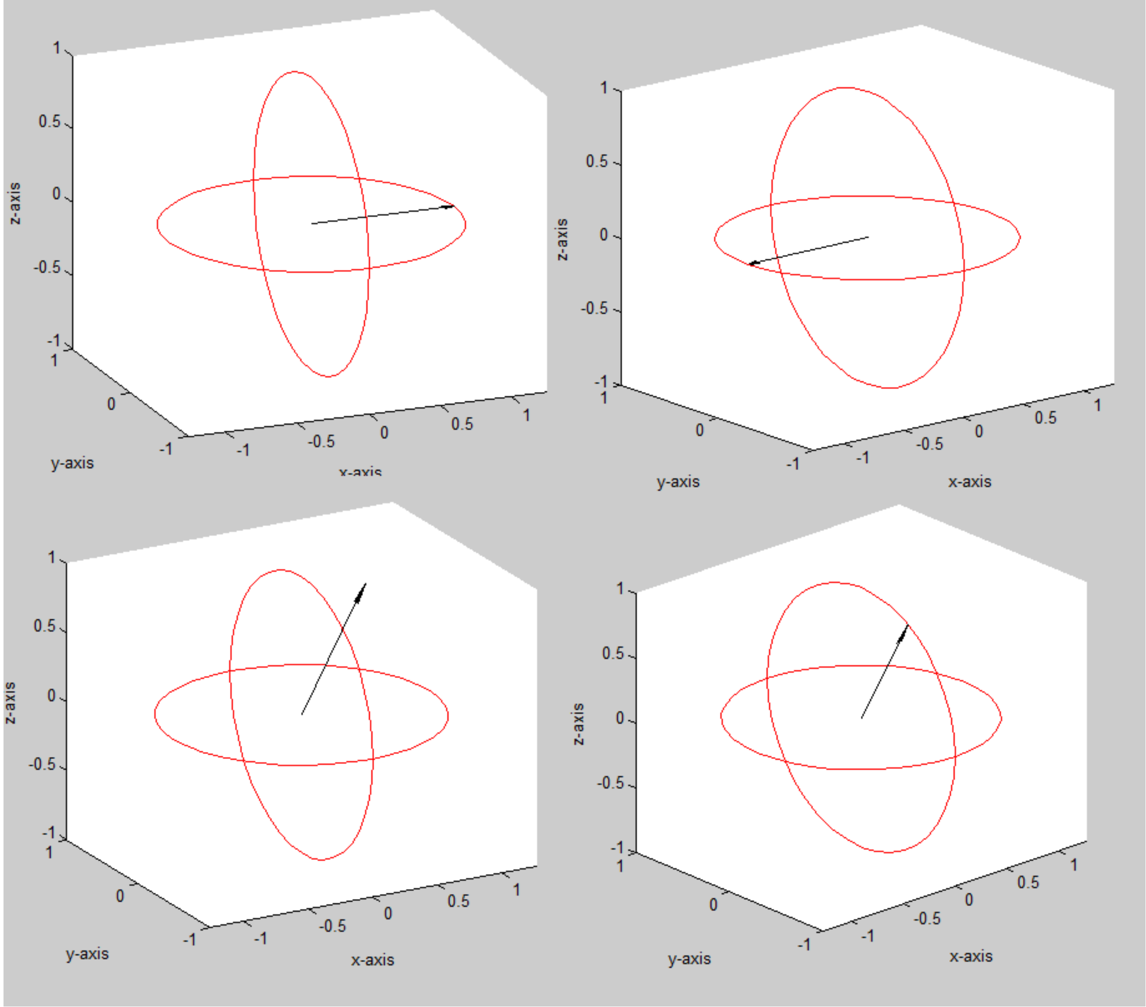


Figure 4.3: Plots of an arrow before and after the specified rotations. It has been inscribed inside two perpendicular circumferences to visualize how the modulus is not altered. The details of the script to code this plot on MATLAB are contained in **Appendix C.3.**

4.1.3 The problem of gimbal lock

Definition 28 ***Gimbal lock***, *or rotation singularity, is a phenomenon consisting of the alignment of two of the three axes of rotation (gimbals) causing the loss of one degree of freedom (or more) and impeding the system to rotate along the affected directions (see [29]). Any system that uses Euler angles will face this problem because the three axes are independently evaluated.*

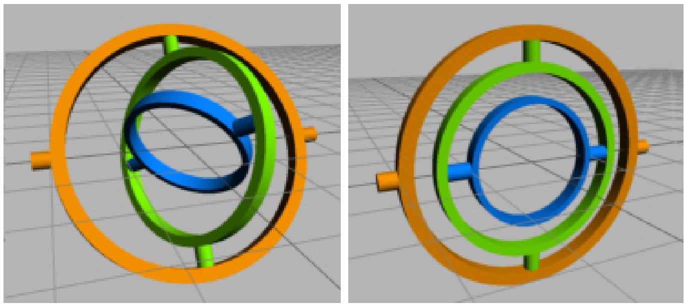


Figure 4.4: On the left, all the gimbals are totally free to rotate on their respective axes. On the right, the three axis are parallel and the locking out of the axis which depends on the rest (in this case the blue axis) is produced. The blue gimbal cannot rotate along the orange one.

Looking at the previous system of Euler angles, duplication is avoid by the restriction of $0 \leq \Theta \leq 2\pi$. In contrast, it is impossible to avoid duplication when $\Theta = 0$, which returns a matrix whenever $\Phi + \Psi$ has a constant value. Mathematically the second matrix (the identity matrix) has no effect on the product:

$$R(\Phi, 0, \Psi) = \begin{pmatrix} cos\Psi cos\Phi - sin\Psi sin\Phi & -cos\Psi sin\Phi - sin\Psi cos\Phi & 0 \\ sin\Psi cos\Phi + cos\Psi sin\Phi & -sin\Psi sin\Phi + cos\Psi cos\Phi & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{4.12}$$

Applying trigonometric formulas:

$$R(\Phi, 0, \Psi) = \begin{pmatrix} cos(\Psi + \Phi) & -sin(\Psi + \Phi) & 0 \\ sin(\Psi + \Phi) & cos(\Psi + \Phi) & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{4.13}$$

Changing the values of Ψ and Φ in the above matrix has the same effects: the rotation angle $\Psi + \Phi$ changes, but the rotation axis remains in the Z direction: the last column and the last row in the matrix will not change. Only one degree of freedom (corresponding to $\Psi + \Phi$) remains. The only solution for Ψ and Φ to recover different roles is to change Θ to some value other than 0. A similar problem appears when $\Theta = \pi$.

It can also be understood considering that the matrix has adopted the 2D rotation shape, where $(\Psi + \Phi)$ is a constant (for example α) and the rotation is only on the Cartesian plane (section (3.1.2)). Then, a spatial rotation has become impossible (see [30]).

$$R(\Phi, 0, \Psi) = \begin{pmatrix} cos\alpha & -sin\alpha & 0 \\ sin\alpha & cos\alpha & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{4.14}$$

This is a planar rotation inside a three-dimensional space.

Gimbal lock can be avoid using a fourth gimbal, for example by using quaternions (see 4.1.5).

4.1.4 Rotation by Euler axis/angle

Definition 29 *The* ***axis/angle representation*** *also derives from Euler's rotation theorem, and holds that a rotation in three-dimensional space can be parametrized by only a unit vector* $\hat{r}$ *and an angle* θ*, which describes the magnitude of the rotation about this axis.*

Hence, the whole rotation can be expressed as:

$$r = \theta \hat{r}, \tag{4.15}$$

where $\hat{r}$ is the unit vector which indicates the resulting direction which will be rotated. Then, this rotational axis has three spatial components: $\hat{r} = [r_x, r_y, r_z]^T$.

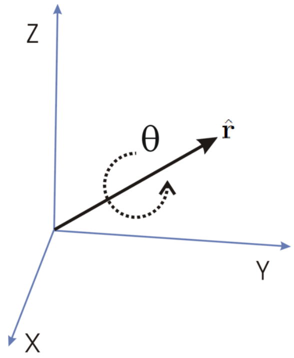


Figure 4.5: Schematic representation of a rotation by Euler axis/angle method.

4.1.4.1 Euclidean vectors rotation by Rodrigues formula

In a similar way to section 4.1.4, Euler-Rodrigues Formula is also based in one of the Euler's theorems for rotations. In this case, it is based on the fact that the general displacement of a rigid body with one point fixed is a rotation about some axis that passes through that fixed point. In order to understand it better, we could imagine there is an original vector and later it is decomposed in two: one parallel to the rotation axis and the remainder orthogonal to it (see [33] [34]). Once this is done, it will be possible to rotate the orthogonal vector in its plane using a 2D rotation.

The following figure tries to exemplify it:

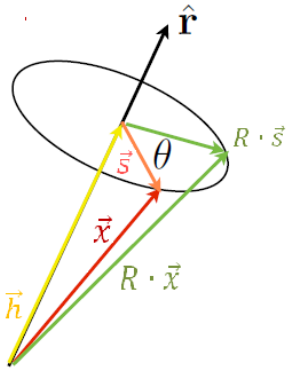


Figure 4.6: Decomposition of the red vector in two: one parallel to the rotation axis $\hat{r}$ (yellow) and the remainder in green. **R** is the rotation operation.

Now, some equalities will be defined [35]:

- The vector $\vec{h}$ is the part of $\vec{x}$ in the direction of $\hat{r}$, so:

$$\vec{h} = (\vec{x} \cdot \hat{r}) \cdot \hat{r} \tag{4.16}$$

- The vector $\vec{s}$ is defined to be the part of $\vec{x}$ which is perpendicular to $\hat{r}$, so:

$$\vec{s} = (\vec{x} - \vec{h}) = \vec{x} - (\vec{x} \cdot \hat{r}) \cdot \hat{r} \tag{4.17}$$

- If a new vector $\vec{t}$ is defined to be perpendicular to both $\hat{r}$ and $\vec{s}$, as shown in the next figure, one can assume that:

$$\vec{t} = \vec{r} \cdot \vec{s} = \hat{r} \cdot (\vec{x} - \vec{h}) = \hat{r} \cdot \vec{x} \tag{4.18}$$

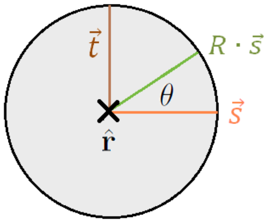


Figure 4.7: Representation of the new defined vector $\vec{t}$. The cross represent the vector $\hat{r}$ with outside direction.

- Since $\hat{r} \times \vec{h} = 0$ because $\vec{h}$ is parallel to $\hat{r}$, is possible to deduce that strictly in the plane of rotation (figure 4.7) the transformed location of $\vec{s}$ is given by:

$$R \cdot \vec{s} = cos\theta\vec{s} + sin\theta\vec{t} = cos\theta[\vec{x} - (\vec{x} - (\vec{x} \cdot \hat{r})\hat{r}] + sin(\theta)[\hat{r} \times \vec{x}] \tag{4.19}$$

- One can deduced from the figure 4.6 that the rotated vector $R\cdot\vec{x}$ can be broken into two parts as the vector sum of h and $R\cdot\vec{s}$:

$$R\cdot\vec{x} = h + R\cdot\vec{s} = (\vec{x}\cdot\hat{r})\hat{r} + (cos\theta[\vec{x} - \hat{r}(\hat{r}\cdot\vec{x})] + sin\theta[\hat{r}\times\vec{x}] \tag{4.20}$$

- This last expression can be written as follows, resulting in the **Euler-Rodrigues formula**:

$$R(\hat{r},\theta,\vec{x}) = \vec{x}cos\theta + (\hat{r}\times\vec{x})sin\theta + \hat{r}(\hat{r}\cdot\vec{x})(1 - cos\theta) \tag{4.21}$$

4.1.5 Rotation by quaternions

Definition 30 ***Quaternions****, represented by* ***Q****, are part of a kind of numbers called hypercomplex numbers. Compared to Euler angles they are easier to compose and avoid the problem of* ***gimbal lock****. Compared to rotation matrices they are more numerically stable and may be more efficient. Quaternions have found their way into applications in computer graphics, computer vision, robotics, navigation, molecular dynamics, flight dynamics, and orbital mechanics of satellites.*

Analogously to the complex numbers, they can be defined by composition of a real part and an imaginary one (but in this case the imaginary part is three-dimensional):

$$Q = a + ib + jc + kd, \tag{4.22}$$

where $a, b, c, d \in \mathbb{R}$, i, j, k are square roots of -1 and $Q \in \mathbb{R}^4$.

All the possible products and equivalences between the different imaginary units are collected below:

$$i^2 = j^2 = k^2 = -1 \tag{4.23}$$

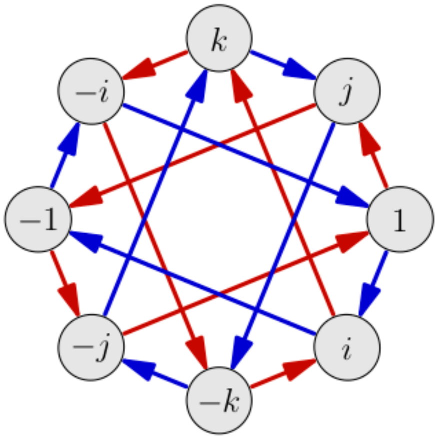
$$ij = -ji = k \tag{4.24}$$

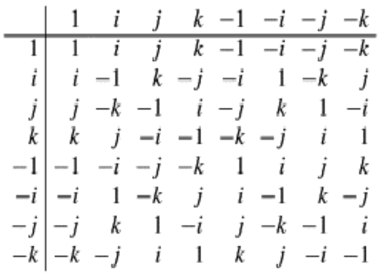
$$jk = -kj = i \tag{4.25}$$

$$ki = -ik = j \tag{4.26}$$

$$\tag{4.27}$$

A different way of thinking about these products is on the Cayley graph, which is resumed on the Cayley table, as can be seen next:





	1	i	j	k	-1	$-i$	$-j$	$-k$
1	1	i	j	k	-1	$-i$	$-j$	$-k$
i	i	-1	k	$-j$	$-i$	1	$-k$	j
j	j	$-k$	-1	i	$-j$	k	1	$-i$
k	k	j	$-i$	-1	$-k$	$-j$	i	1
-1	-1	$-i$	$-j$	$-k$	1	i	j	k
$-i$	$-i$	1	$-k$	j	i	-1	k	$-j$
$-j$	$-j$	k	1	$-i$	j	$-k$	-1	i
$-k$	$-k$	$-j$	i	1	k	j	$-i$	-1

Figure 4.8: Cayley graph a Cayley table for quaternions. The blue line is a multiplication by **i** and the red one by **j** in the the indicated direction.

In addition to the representation before, quaternions can also be expressed in other ways:

- Complex matrix representation:

$$\begin{pmatrix} a+ib & c+id \\ -c+id & a-ib \end{pmatrix} \tag{4.28}$$

One can deduce two conclusions:

  - As one can observe, for complex numbers this matrix becomes diagonal.
  - The norm of the quaternion is the square root of the determinant of this matrix ($\sqrt{a^2+b^2+c^2+d^2}$).

- Compacted representation (see [31]):

$$Q = [q_0, q]^T, \tag{4.29}$$

where the imaginary part is grouped in **q**. It is possible to see an analogy between quaternions and real numbers ($Q = [q_0, 0]$) and with the vectors in $\mathbb{R}^3$ ($Q = [0, q]$).

This last notation will be used as of now. In any case $a = q_0$, $b = q_1$, $c = q_2$ and $d = q_4$.

The main properties of quaternions are:

- The conjugated quaternion $Q*$ is:

$$Q^* = q_0 - iq_1 - jq_2 - kq_3 \tag{4.30}$$

- The sum of two quaternions Q_1 and Q_2 is:

$$Q_1 + Q_2 = (q_{01} + q_{02}) + i(q_{11} + q_{12}) + j(q_{21} + q_{22}) + k(q_{31} + q_{32}) \tag{4.31}$$

- The product of two quaternions Q_1 and Q_2 is:

$$\begin{align} Q_1 \cdot Q_2 = (q_{01}q_{02} - q_{11}q_{12} - q_{21}q_{22} - q_{31}q_{32}) \tag{4.32}\\ +i(q_{01}q_{12} + q_{11}q_{02} + q_{21}q_{32} - q_{31}q_{22}) \tag{4.33}\\ +j(q_{01}q_{22} - q_{11}q_{32} + q_{21}q_{02} + q_{31}q_{12}) \tag{4.34}\\ +k(q_{01}q_{32} + q_{11}q_{22} - q_{21}q_{12} + q_{31}q_{02}) \tag{4.35}\end{align}$$

- The norm of a quaternion Q is:

$$\|Q\| = \sqrt{Q \cdot Q^*} = \sqrt{Q^* \cdot Q} = \sqrt{q_0^2 + q_1^2 + q_2^2 + q_3^2} \tag{4.36}$$

In relation to rotations, quaternions can be written in terms of angle and axis in the following way:

$$q_0 = cos\left(\frac{\theta}{2}\right) \tag{4.37}$$

$$q = sin\left(\frac{\theta}{2}\right) n \tag{4.38}$$

where $n = \begin{pmatrix} i \\ j \\ k \end{pmatrix}$ and $\theta = \begin{pmatrix} \Psi(i) \\ \Theta(j) \\ \Phi(k) \end{pmatrix}$

NOTE: In this case the angle Ψ represents a rotation along the x-axis, instead of the z-axis as in the last convention for Euler angles. It is convenient to remember the convention used before in the section (4.7) is completely subjective and any other could be used in a similar way.

Hence, the unit quaternion (with a norm equal to 1) can be generalized as:

$$Q_r = cos\left(\frac{\alpha}{2}\right) + isin\left(\frac{\Psi}{2}\right) + jsin\left(\frac{\Theta}{2}\right) + ksin\left(\frac{\Phi}{2}\right) \tag{4.39}$$

And its conjugated as:

$$Q_r^* = cos\left(\frac{\alpha}{2}\right) - i \cdot sin\left(\frac{\Psi}{2}\right) - j \cdot sin\left(\frac{\Theta}{2}\right) - k \cdot sin\left(\frac{\Phi}{2}\right) \tag{4.40}$$

The rotation quaternion for a specified axis is obtained by imposing a spin of 0 rad in those axis which are not involved and replacing β by the corresponding angle in relation with the selected axis.

On the other hand, when mixing several axes it is necessary to use the norms of multiplication already studied:

$$v_2 = Q_r \cdot v \cdot Q_r^*, \tag{4.41}$$

where v is the initial vector, v_2 the rotated one, Q_r a quaternion vector with four elements and Q_r^* its conjugated.

Moreover, the same result can be obtained by the matrix conversion of the unit quaternion and the subsequent multiplication:

$$v_2 = R^Q \cdot v, \tag{4.42}$$

where R^Q is the conversion of the unit quaternion into matrix as it is written in the following equation:

$$R^Q = \begin{pmatrix} q_0^2 + q_1^2 - q_2^2 - q_3^2 & 2q_1q_2 - 2q_0q_3 & 2q_1q_3 + 2q_0q_2 \\ 2q_0q_3 + 2q_1q_2 & q_0^2 - q_1^2 + q_2^2 - q_3^2 & -2q_0q_1 + 2q_2q_3 \\ -2q_0q_2 + 2q_1q_3 & 2q_0q_1 + 2q_2q_3 & q_0^2 - q_1^2 - q_2^2 + q_3^2 \end{pmatrix} \tag{4.43}$$

This process must be follow several times for using this method to rotate again, after a previous conversion from the pertinent representation to quaternions.

- **Example. 3D rotation by using quaternions**

  At this point, all the operations needed to operate using quaternions will be developed in order to rotate the same vector used in the previous example for Euler angles (see 4.1.2) ($v = (1, 0, 0)$) and also by the same angles of rotation ($\Psi = \pi$, $\Theta = 2\pi/3$ and $\Phi = -\pi/2$). Each step will be explained in detail to make the understanding easier:

First, the initial vector must be written as quaternion. This process is done by replacing each one of the axes by the corresponding imaginary unit (the x-axis by i, the y-axis by j and z-axis by k). The real part is always 0.

$$v = (1, 0, 0) \longrightarrow Q = 0 + 1i + 0j + 0k = i \tag{4.44}$$

After that, the quaternion which represents the rotation is:

$$Q_r = cos\left(\frac{\theta}{2}\right) + (n_1 i + n_2 j + n_3 k)\, sin\left(\frac{\theta}{2}\right) \tag{4.45}$$

where n_1, n_2, n_3 are used to the define the rotation axis (which can be a combination of three spatial ones) and θ is the rotation angle.

In this case, the rotation will be only around the z-axis, so:

$$Q_r = cos\left(\frac{\pi}{2}\right) + (0i + 0j + 1k)\, sin\left(\frac{\pi}{2}\right) = 0 + 0i + 0j + k = k \tag{4.46}$$

Hence, the result vector, expressed as quaternion, is:

$$v_2 = Q_r \cdot v \cdot Q_r^* = (k)(i)(-k) = ki(-k) = -ik(-k) = -1 \longrightarrow v_2 = (-1, 0, 0) \tag{4.47}$$

The following rotation will be developed by an angle of $\frac{2\pi}{3}$ around the y-axis:

$$Q_r = cos\left(\frac{2\pi}{6}\right) + (0i + 1j + 0k)\, sin\left(\frac{2\pi}{6}\right) = 0.5 + 0.866j \tag{4.48}$$

$$v_3 = Q_r \cdot v_2 \cdot Q_r^* = (0.5 + 0.866j)(-i)(0.5 - 0.866j) = (-0.5i + 0.866k)(0.5 - 0.866j) = \tag{4.49}$$

$$v_3 = 0.5i + 0.866j \longrightarrow v_3 = (-0.5, 0, 0.866) \tag{4.50}$$

Finally, the last step is another rotation by $-\pi/2$ around the z-axis (according to the **x-convention**):

$$Q_r = cos\left(\frac{-\pi}{4}\right) + (0i + 0j + 1k)\, sin\left(\frac{-\pi}{4}\right) = 0.7071 + 0.7071k \qquad (4.51)$$

$$v_4 = Q_r \cdot v_3 \cdot Q_r^* = (0.7071 + 0.7071k)(0.5i + 0.866k)(0.7071 - 0.7071k) = \qquad (4.52)$$

$$v_4 = (0.3535i + 0.621k + 0.335j - 0.621)(0.7071 - 0.7071k) \qquad (4.53)$$

$$v_4 = 0.5j + 0.866k \longrightarrow v_3 = (0, -0.5, 0.866) \qquad (4.54)$$

As can be checked, **these results perfectly match with the ones obtained by using Euler angles** (see 4.1.2), so these rotations can be identified with the figure 4.3.

NOTE: All the operations shown above have been made manually, so they do not appear in any **Appendix**. The rules for getting results have also been explained at the beginning of this chapter.

4.2 Method for defining spatial movements

The process for defining a general motion in a three-dimensional space is very similar to the used in planar motion. The rotation matrices include translations in their last column, so the fundamentals already seen are also valid here (section 3.2):

$$\left(\begin{array}{c|c} R(\theta) & t \\ \hline 0 & 1 \end{array}\right) \tag{4.55}$$

- One must remember that the main advantage of this representation is that it makes the successive transformations in cascade easier:

$$\left(\begin{array}{c|c} R(\theta_1) & t_1 \\ \hline 0 & 1 \end{array}\right)\left(\begin{array}{c|c} R(\theta_2) & t_2 \\ \hline 0 & 1 \end{array}\right) = \left(\begin{array}{c|c} R(\theta_2+\theta_1) & R(\theta_2)t_1+t_2 \\ \hline 0 & 1 \end{array}\right) \tag{4.56}$$

The only difference with 2D is the size and composition of these matrices, which must be adapted for the kind of rotation depending on the axis where they are carried out in each case:

$$T(\Psi, x) = \begin{pmatrix} 1 & 0 & 0 & t_x \\ 0 & cos\Psi & -sin\Psi & t_y \\ 0 & sin\Psi & cos\Psi & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{4.57}$$

$$T(\Theta, y) = \begin{pmatrix} cos\Theta & 0 & sin\Theta & t_x \\ 0 & 1 & 0 & t_y \\ -sin\Theta & 0 & cos\Theta & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{4.58}$$

$$T(\Phi, z) = \begin{pmatrix} cos\Phi & -sin\Phi & 0 & t_x \\ sin\Phi & cos\Phi & 0 & t_y \\ 0 & 0 & 0 & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{4.59}$$

The way of working and the properties of this method are completely equivalent to what was told in 3.2. The only difference between 2D and 3D is that the product of matrices is not commutative. Therefore, when developing several transformations in a consecutive way this aspect must be considered carefully.

- **3D transformations by using homogeneous transformations matrices in MATLAB**

  In order to exemplify what has been told, some transformations will be developed on a predefined vector $v = (1, 0, 0)^T$. First, a translation of $t_x = -1$, $t_y = 3$ and $t_z = 2$ will be implemented. At that point, the resulting vector will be rotated by $-pi/4$ around the x-axis and by $3\pi/5$ around the z-axis. As in the case of 2D transformations, rotations and translations must be carried into effect separately and the initial vector must be completed with 1 in the final row.

```
>> v=[1 0 0 1]'

v =
       1
       0
       0
       1

>> T_mat=[1 0 0 -1; 0 1 0 3; 0 0 1 2; 0 0 0 1];
>> v2=T_mat*v

v2 =
     0
     3
     2
     1
```

At this point, the resulting vector is $v_2 = (0, 3, 2)^T$. The rotation motions are applied in the following way (the matrix shape depends on the axis):

```
>> T_mat=[1 0 0 0; 0 cos(-pi/4) -sin(-pi/4) 0;
0 sin(-pi/4) cos(-pi/4) 0; 0 0 0 1];
>> v3=T_mat*v2

v3 =
         0
    3.5355
   -0.7071
    1.0000

>> T_mat=[cos(3*pi/5) -sin(3*pi/5) 0 0; sin(3*pi/5) cos(3*pi/5) 0 0;
0 0 1 0; 0 0 0 1];
>> v4=T_mat*v3

v4 =
   -3.3625
   -1.0925
   -0.7071
    1.0000
```

Hence, the results are:

- Initial vector: $v = (1, 0, 0)^T$
- After a translation ($t_x = -1$, $t_y = 3$, $t_z = 2$): $v_2 = (0, 3, 2)^T$
- After a rotation by $-\pi/4$ around the x-axis: $v_3 = (0, 3.5355, -.7071)^T$
- After a rotation by $3\pi/5$ around the z-axis: $v_4 = (-3.3625, -1.0925, -0.7071)$

The graphical representation of the motions above can be observed in the following figure 4.9. The script with the detail to code the plot are included in **Appendix C.4**.

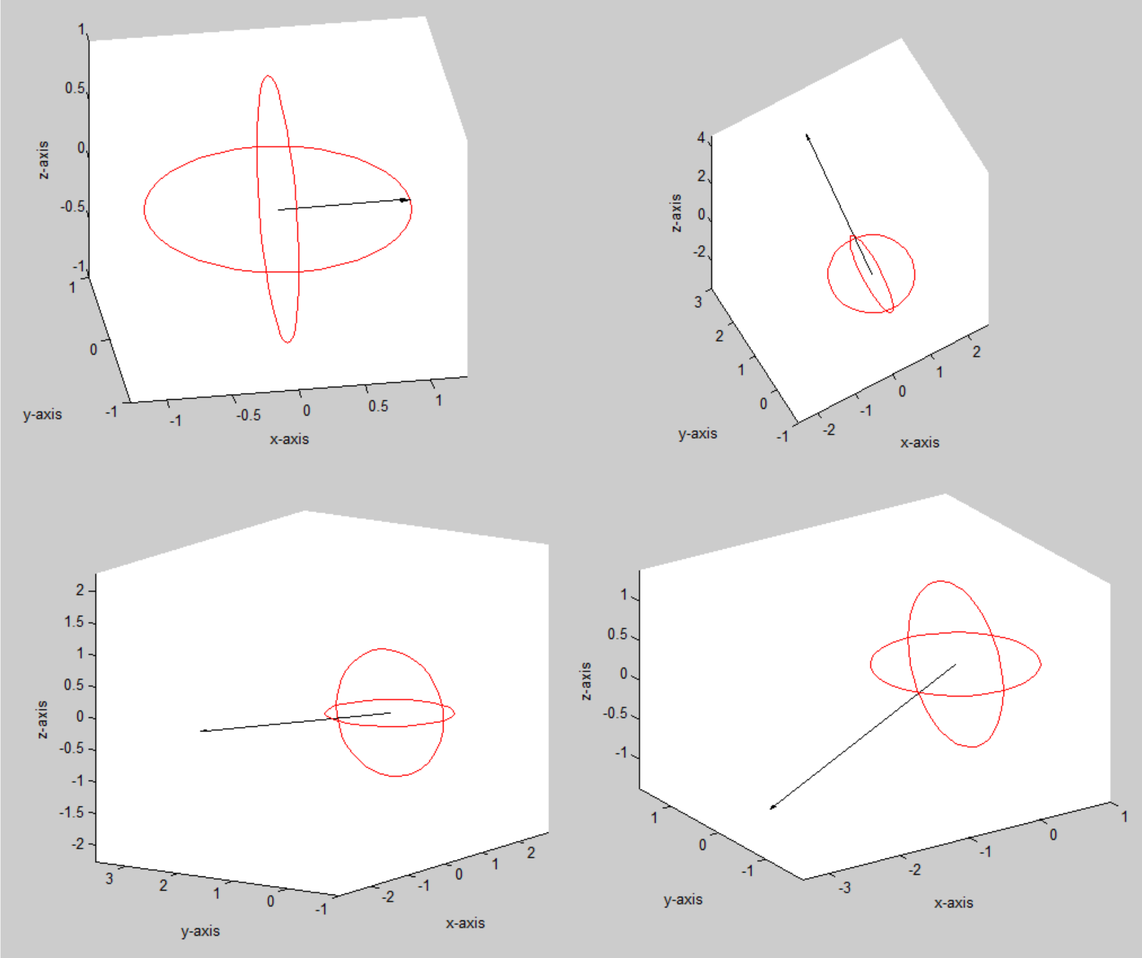


Figure 4.9: Plots of an arrow which have been moved according to the specified transformations: a translation ($t_x = -1$, $t_y = 3$, $t_z = 2$) and two consecutive rotations ($-|pi/4$ around the x-axis and $3\pi/5$ around the z-axis). The two perpendicular circumferences can be used to visualize how the norm 1-limits are overpassed after a translation.

Chapter 5

Forward Kinematics

Definition 31 ***Kinematics*** *is the study of the possible movement and configurations of a system. Thus, it is really concerned with its geometry. The task of understanding how a system can move in given circumstances requires knowledge about forces and inertias. In order to explain it, the* ***forward kinematics*** *tries to describe the position and orientation of the last link of a kinematic chain in terms of joint variables* $q(t) = (q_1(t), q_2(t), ..., q_n(t))$.

This chapter will treat the concordance of all the previous transformation matrices (using Matrix algebra) with the physical constitution of a robot. **Forward Kinematics** is, at the end, only intended to match matrices and movements about each joint of the kinematic chain.

5.1 Structure and components of the kinematic chain

From the mechanical point of view, a robot is a kinematic chain formed by **links** and **joints**, which are designed to allow a relative movement between two consecutive links. The free end of the chain is called **end effector**. Industrial robots base their anatomy in a similar structure to an human arm (see [36]).

5.1.1 Links

Definition 32 *A **link** is an (assumed) rigid body that possesses at least two nodes that are **points for attachment to other links** (see [37]). It can be (5.1):*

- ***Binary link**: One with two nodes.*
- ***Ternary link**: One with three nodes.*
- ***Quaternary link**: One with four nodes.*

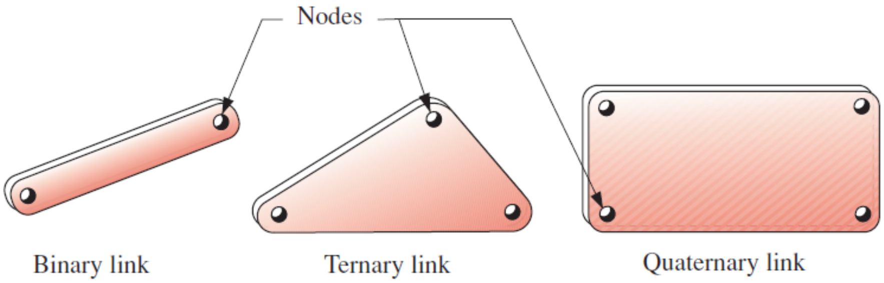


Figure 5.1: Most used kind of links.

5.1.2 Joints

Definition 33 *According to the definition of joint in [38], "A **joint** is a connection between two or more links (at their nodes), which allows some movement, or potential movement, between the connected links". Joints (also called **kinematic pairs**) can be classified in several ways:*

- *By the type of contact between the elements: line, point or surface.*
- *By the number of **degrees of freedom (DOF)** allowed at the joint.*
- *By the type of the joint physical closure: either force or form closed.*
- *By the number of links joined (**joint order**).*

Definition 34 *A **degree of freedom**, also called **DOF**, is the maximum number of possible directions a joint can move.*

There are several kinds of joints, but the most used ones are **prismatic joints** and **revolute joints** (figure 5.2). These joints only allow movement in one direction, so they only have one degree of freedom.

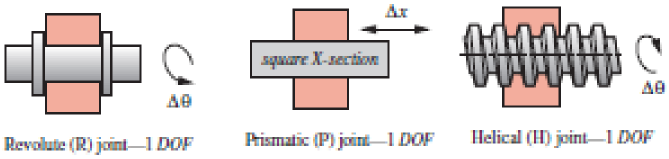

Δθ
Revolute (R) joint—1 DOF
Δx
square X-section
Prismatic (P) joint—1 DOF
Δθ
Helical (H) joint—1 DOF

Figure 5.2: Most known one DOF joints.

Structures with more than one degree of freedom are usually considered complex joints (figure 5.3).

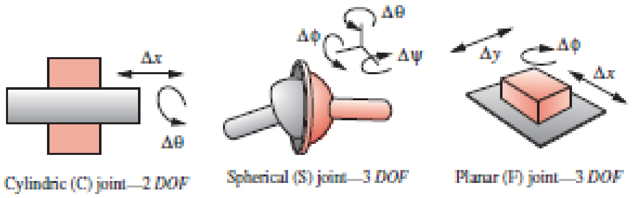

Δx
Δθ
Cylindric (C) joint—2 DOF
Δθ
Δφ
Δψ
Spherical (S) joint—3 DOF
Δy
Δφ
Δx
Planar (F) joint—3 DOF

Figure 5.3: Most known joints with **more** than one DOF.

At this point, it is vital to know that moving a specific joint will change the following joint position because the displacement of a joint affects all the joints after it. Therefore, an easy method to represent the conversions and effects on the kinematic chain (given by 4×4 transformation matrices) is required to be as clear as possible.

5.2 Forward kinematics resolution by homogeneous transformation matrices

The forward kinematics problem has been previously reduced to find a transformation matrix (**T**) which connects the position and orientation of the end of the robot considering the fixed reference system placed in its base.

The homogeneous transformation matrix which represents the relative position and orientation between the associated systems to two consecutive joints is called ${}^{i-1}A_i$ matrix. In this way, ${}^{0}A1$ describes the position and orientation of the reference system corresponding to the first joint, ${}^{1}A_2$ describes the position and orientation of the second one in relation to the first, etc. Similarly, naming ${}^{0}A_k$ the resulting matrices from the product of matrices ${}^{i-1}A_i$ from $i = 1$ to $i = k$, the robot kinematic chain can be represented in a partial or total way (see [39]).

For example, the position and orientation of the third joint with respect to the base coordinated reference system is:

$$ {}^{0}A_3 =^{0} A_1 \, {}^{2}A_2 \, {}^{0}A_3 \tag{5.1} $$

When all the DOF are considered, the ${}^{0}A_n$ matrix is then called **T** matrix. Hence, given a robot with **n** DOF, the position and orientation of the last joint will be expressed by the **T** matrix in this way:

$$ T =^{0} A_n =^{0} A_1 \, {}^{1}A_2 \, {}^{2}A_3 \, ... \, {}^{n-1}A_n \tag{5.2} $$

In order to describe the relationship between two consecutive elements, the most used procedure in robotics is the **Denavit-Hartenberg (D-H)** representation, whose vitality has been preserved despite of being proposed in 1955.

5.3 Denavit-Hartenberg representation

Denavit-Hartenberg representation **(D-H)** is a systematic procedure in order to describe the kinematic structure of an articulated chain (open kinematic chain) composed by **one DOF** joints (see [40]).

Before applying the D-H method it is important to consider the following aspects (see [41]):

- It is possible to start with any configuration of the robot, but placing the robot in an easy initial position is recommended.
- The orthogonal coordinated system in the base (X_0, Y_0, Z_0) is placed in the Z_0-axis, located along the first joint axis of movement and pointing at out of the arm of the robot shoulder.
- The reference system of each link is placed at the end of the link where the following one is joined.
- The angle or displacement of each link is always measured by taking as base the previous link reference system.
- When establishing the coordinated reference system in the robotic hand, the **Pieper's Principle** must be considered: the three last reference systems must be intercepted in a point in order to obtain a closed solution for these links in the Inverse Kinematics problem.

5.3.1 Denavit-Hartenberg parameters

According to this representation, by choosing correctly the coordinate systems assigned to each link, it is possible to go from one link to the following by four basic transformations which depend only on the geometrical features of the link.

This basic transformation consists of a sequence of rotations and translations which allows to relate the element i reference system with the $i-1$ reference system.

The required transformations named above are the following ones (see [42]):

- Rotation around the z_{i-1}-axis an angle θ_i
- Translation around the z_{i-1}-axis a distance d_i; vector $d_i = (0, 0, d_i)$.
- Translation around the x_i-axis a distance a_i; vector $a_i = (a_i, 0, 0)$.
- Rotation around the x_i-axis an angle α_i.

Since the product of matrices is not commutative (see 2.4.1), these transformations must be implemented in a particular order, so:

$$^{i-1}A_i = T(z, \theta_i)T(0, 0, d_i)T(a_i, 0, 0)T(x, \alpha_i) \tag{5.3}$$

Calculating:

$$^{i-1}A_i = \begin{pmatrix} cos\theta_i & -sin\theta_i & 0 & 0 \\ sin\theta_i & cos\theta_i & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & a_i \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & cos\alpha_i & -sin\alpha_i & 0 \\ 0 & -sin\alpha_i & cos\alpha_i & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{5.4}$$

Finally it can be expressed as:

$$^{i-1}A_i = \begin{pmatrix} cos\theta_1 & -cos\alpha_i sin\theta_i & sin\alpha_i sin\theta_i & a_i cos\theta_1 \\ sin\theta_1 & cos\alpha_i cos\theta_i & -sin\alpha_i cos\theta_i & a_i sin\theta_i \\ 0 & -sin\alpha_i & cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{pmatrix}, \tag{5.5}$$

where θ_i, a_i, d_1, α_1 are the the D-H parameters for the joint i.

Therefore, only by identifying these parameters is it possible to obtain the **A-matrices** relating all the robot links.

5.3.2 Denavit-Hartenberg algorithm for obtaining the forward kinematics model:

The kinematic model can be easily applied by following these steps (see [43]):

- **D-H 1.** Links must be enumerated starting with 1 for the first one and ending with n. The base of the robot will be enumerated as 0.

- **D-H 2.** In the same way, joints must be enumerated starting with 1 and ending with n.

- **D-H 3.** The axis of each joint must be located. If the joint is rotative, the axis will be its own rotation axis. If it is prismatic, it will be the axis along the displacement is carried out.

- **D-H 4.** For i from 0 to $n-1$, choose z_i-axis along the axis of joint $i+1$ (fig. 5.4, first picture).

- **D-H 5.** Place the origin of the base system O_0 in any point along z_0-axis. The axes x_0 and y_0 will be placed in order to form a dextrorotation (clockwise-moved) system with z_0.

- **D-H 6.** For i from 1 to $n-1$, place system O_i (corresponding to the link i) in the intersection of z_i-axis with the common normal line to z_{i-1}. If both axes are crossing then O_i would be placed in the crossing point. If they were parallel, O_i would be placed in joint $i+1$ (fig. 5.4, second picture).

- **D-H 7.** Choose x_i in the common normal line to axes z_{i-1} and z_i (fig. 5.4, third picture).

- **D-H 8.** Choose y_i in order to form a dextrorotation system with x_i and z_i, completing the right-handed frame fourth.

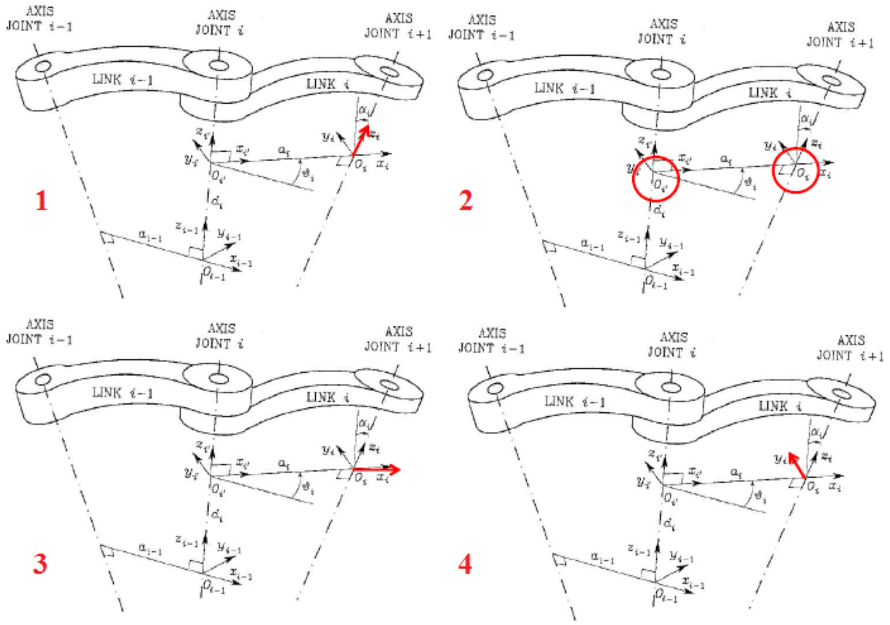

AXIS JOINT i−1
AXIS JOINT i
AXIS JOINT i+1
LINK i−1
LINK i
α_i
z_{i'}
x_{i'}
y_{i'}
O_{i'}
a_i
ϑ_i
y_i
z_i
x_i
O_i
d_i
a_{i−1}
z_{i−1}
y_{i−1}
x_{i−1}
O_{i−1}
1
2
3
4

Figure 5.4: Graphical example to explain how to place axes in Denavit-Hartenberg representation (see [44]).

At this point (see [45]):

- θ_i**:** It is the angle between the axes x_{i-1} and x_i measured in a perpendicular plane to the z_{i-1}-axis, using the right-handed rule. It is a variable parameter in rotative joints.
- d_i**:** It is the distance along the z_{i-1}-axis from the $(i-1)-th$ coordinate system origin to the intersection between the axes z_{i-1} and x_i. It is a variable parameter in prismatic joints.
- a_i**:** It is the distance along the x_i-axis from the intersection of the z_{i-1}-axis with the x_i to the $i-th$ coordinate system origin, in the case of rotative joint. For prismatic joints, it is the shortest distance between the axes z_{i-1} and z_i.
- α_i**:** It is the angle between the axes z_{i-1} and z_i, measured in a perpendicular plane to the x_i axis, using the right-handed rule.

- **D-H 9.** Place system O_n at the end of the robot in a way to make z_n concordant with the direction of z_{n-1} and x_0 to be normal to z_{n-1} and z_n.

- **D-H 10.** Obtain θ_i as the angle necessary in the rotation of z_{i-1} to do x_{i-1} and x_i parallel.

- **D-H 11.** Obtain d_i as the distance, measured along the axis z_{i-1}, that it is needed to move the new O_{i-1} for aligning x_i and x_{i-1} each other.

- **D-H 12.** Obtain a_i as the distance measured along the x_i-axis (now coincident with x_{i-1}) that the new O_{i-1} must be displaced to be totally concordant with O_i in origin.

- **D-H 13.** Obtain α_i as the angle that it must be revolved around x_i (concordant with x_{i-1}) to make the new O_{i-1} totally concordant with O_i.

- **D-H 14.** Obtain the transformation matrices ${}^{i-1}A_i$ previously defined in 5.3.1.

- **D-H 15.** Obtain the transformation matrix which relates the base system to the end of the robot ($T =^0 A_1 \; ^1A_2 \; ... \; ^{n-1}A_n$).

- **D-H 16.** The matrix T defines the orientation (submatrix of rotation) and position (submatrix of translation) from the referred extreme to the base depending on the n joint coordinates.

5.3.2.1 Some examples for better understanding: Three-link planar manipulator (see [46])

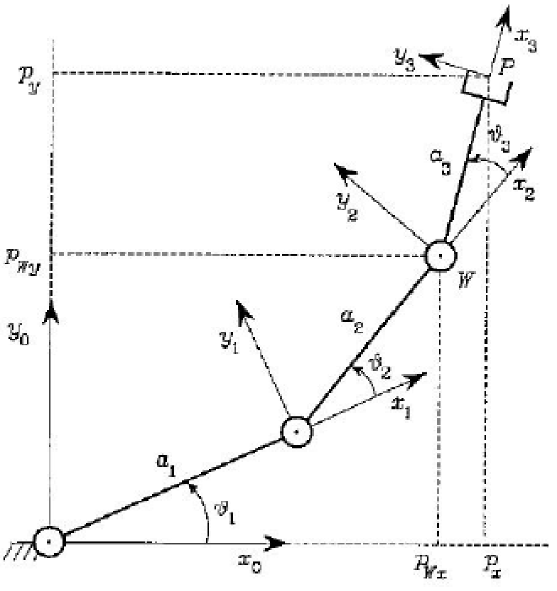

p_y
y_3
x_3
P
ϑ_3
a_3
x_2
y_2
p_{Wy}
W
y_0
y_1
a_2
ϑ_2
x_1
a_1
ϑ_1
x_0
p_{Wx}
p_x

Figure 5.5: Three-link planar manipulator schematic.

Link	a_i	α_i	d_i	θ_i
1	a_1	0	0	θ_1
2	a_2	0	0	θ_2
3	a_3	0	0	θ_3

Table 5.1: Three-link planar manipulator D-H parameters.

The corresponding **A-matrices** to this example are:

$$
{}^{i-1}A_i(\theta_i) = \begin{pmatrix} cos\theta_i & -sin\theta_i & 0 & a_i cos\theta_i \\ sin\theta_i & cos\theta_i & 0 & a_i sin\theta_i \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{5.6}
$$

$$
for\ i = 1, 2, 3.
$$

Then:

$$T =^0 A_3 =^0 A_1 \; ^1A_2 \; ^2A_3 \tag{5.7}$$

$$\begin{pmatrix} cos(\theta_1+\theta_2+\theta_3) & -sin(\theta_1+\theta_2+\theta_3) & 0 & a_1cos\theta_1 + a_2cos(\theta_1+\theta_2) + a_3cos(\theta_1+\theta_2+\theta_3) \\ sin(\theta_1+\theta_2+\theta_3) & cos(\theta_1+\theta_2+\theta_3) & 0 & a_1sin\theta_1 + a_2sin(\theta_1+\theta_2) + a_3sin(\theta_1+\theta_2+\theta_3) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{5.8}$$

5.3.2.2 Some examples for better understanding: Anthropomorphic arm

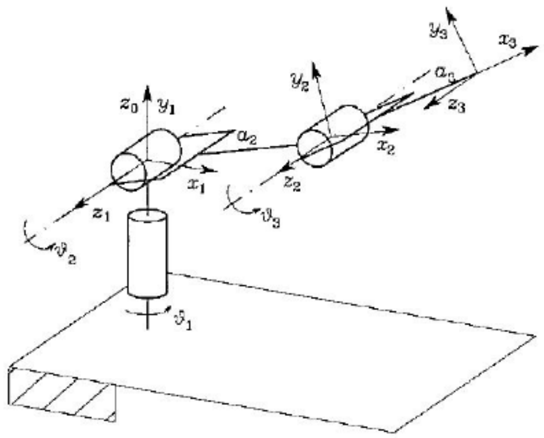


Figure 5.6: Anthropomorphic arm schematic.

Link	a_i	α_i	d_i	θ_i
1	0	$\pi/2$	0	θ_1
2	a_2	0	0	θ_2
3	a_3	0	0	θ_3

Table 5.2: Anthropomorphic arm D-H parameters.

In this case, the corresponding **A-matrices** are:

$$^0A_1(\theta_1) = \begin{pmatrix} cos\theta_1 & sin\theta_1 & 0 & 0 \\ sin\theta_i & 0 & -cos\theta_1 & 0) \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{5.9}$$

$$^{i-1}A_i(\theta_i) = \begin{pmatrix} cos\theta_i & -sin\theta_i & 0 & a_i cos\theta_i \\ sin\theta_i & cos\theta_i & 0 & a_i sin\theta_i \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{5.10}$$

$$for\ i = 2, 3.$$

Then:

$$T =^0 A_3 =^0 A_1\ ^1A_2\ ^2A_3 = \tag{5.11}$$

$$\begin{pmatrix} cos\theta_1 cos(\theta_2 + \theta_3) & -cos\theta_1 sin(\theta_2 + \theta_3) & sin\theta_1 & cos\theta_1(a_2 cos\theta_2) + a_3 cos(\theta_2 + \theta_3)) \\ sin\theta_1 cos(\theta_2 + \theta_3) & -sin\theta_1 sin(\theta_2 + \theta_3) & -cos\theta_1 & sin\theta_1(a_2 cos(\theta_2) + a_3 cos(\theta_2 + \theta_3)) \\ sin(\theta_2 + \theta_3) & cos(\theta_2 + \theta_3) & 0 & a_2 sin\theta_2 + a_3 sin(\theta_2 + \theta_3) \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{5.12}$$

Chapter 6

Inverse Kinematics

The chapter before has studied how the position of the **end effector** (at the end of the robotic chain) can be perfectly placed by using Forward Kinematics and specifying the displacements and angles to move. Now, the desired position and orientation of the **end effector** will be given by the user, and the rotation angles of the required joints will have to be calculated. This problem can have none, one or more solutions depending on its complexity. Therefore, if there are so many configuration limitations and the system has no solution, then the system is **overconstrained**; if there are relatively few restrictions and a lot of solutions can be found to solve the initial problem, then the system is called **underconstrained**.

Definition 35 *The* ***reachable workspace*** *is the volume of space which can be reached by the end effector (see figure 6.2).*

Definition 36 *The* ***dexterous workspace*** *is the volume of space which can be reached by the end effector with all the different orientations (see [47]).*

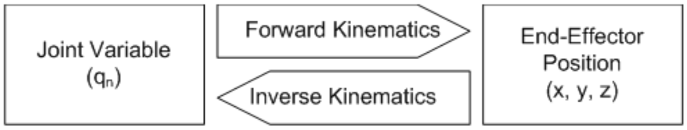
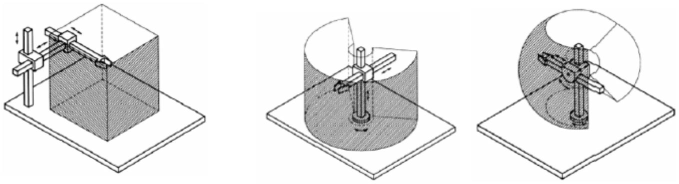
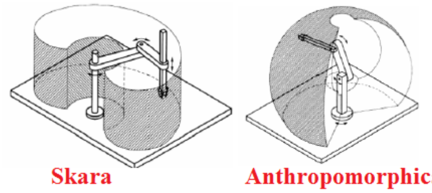


Figure 6.1: Forward Kinematics vs Inverse Kinematics.



Cartesian **Cylindrical** **Spherical**



Skara **Anthropomorphical**

Figure 6.2: Reachable workspace depending on different configurations.

The mechanical complexity is important for determining the different kinds of solution which can be implemented in order to solve the problem, from analytical (very easy systems) to differential ones (with much more difficult configurations), as it will be explained next.

6.1 Important points when solving Inverse Kinematics

Definition 37 *It is possible to define the* ***inverse kinematics solution*** *as the set of variables which allow to locate the end effector in a concrete position and orientation. Unfortunately, there are no general algorithms for solving this dilemma systematically.*

Some problems must be taken into account (see [48]):

- A set of equations must be solved.
    - These equations are non-linear (sin, cos) in rotation matrices.
    - More than one solution can be found (**singularities**).
    - In addition, the possibility of finding no solution to the problem also exists.

- When the solutions are multiple:
  - The solution with a lower number of possible movements is the one which must be chosen.
  - At the same time, the nearest solution must also be considered.
  - The lighter links must have priority to move.
  - Obstacles must also be taken into account in order to avoid collisions.

Another important aspect is the proportion between the number of degrees of freedom (DOF) in the system and the numbers of DOF required by the particular task.

If (see figure 6.3):

- **System degrees of freedom $=$ Required degrees of freedom to perform the task $\Rightarrow$ Two (2) solutions**
- **System degrees of freedom $>$ Required degrees of freedom to perform the task $\Rightarrow$ Infinite (∞) solutions**

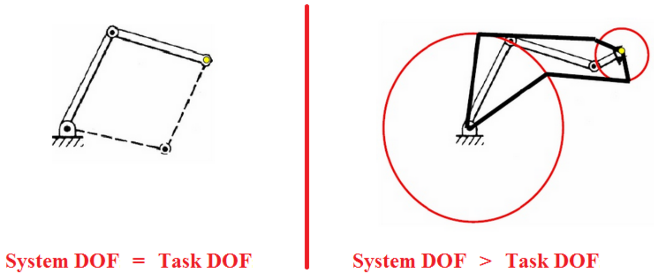


Figure 6.3: Graph to exemplify the dependence between system and task degrees of freedom.

Depending on the kinematics problem complexity, the three main kind of solutions which are usually used are: using geometric resolution, using homogeneous transformation matrices analysis or using iterative methods. In the last case, the solution by Jacobians will be the one studied here.

6.2 Analytical methods

6.2.1 Solution by geometric methods

Most of the simplest robots have relatively simple kinematic chains. This feature makes the inverse kinematics problem resolution easier. Only considering the first three degrees of freedom in a robot, the elements are placed in the same plane. Moreover, in the case of many robots the last three degrees of freedom are mainly used in the end effector orientation. In these cases is possible to find a systematic method to approach and solve the problem, making the demand of computer resources smaller (see [48]).

The geometric procedure is based in finding an enough number of geometric relations between the end effector coordinates, the coordinates of its joints and the physical dimensions of its elements to solve the problem.

6.2.1.1 Example of a three-DOF-robot geometric resolution

In order to make the way of solving this kind of problem clearer, the resolution to the robot in the figure (see 6.4) will be explained (see [49]):

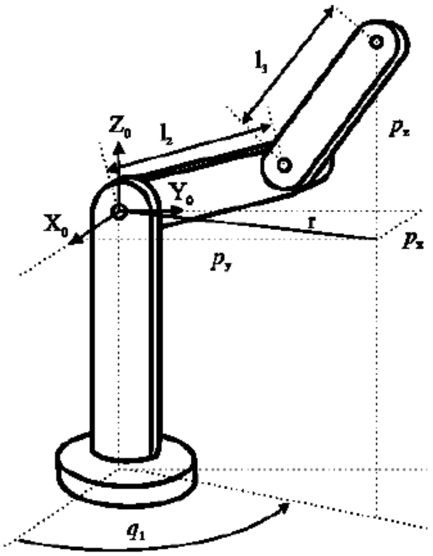


Figure 6.4: Robot to solve: Three-DOF-arm robot.

The features of this robot are shown below:

- Three DOF robot.
- Coordinates P_x, P_y, P_z.
- Planar structure robot.

The coordinates are referred to the point where the end effector must be positioned. Since the robot has a planar structure, it is defined by the the first joint variable angle, which can be easily obtained in the following way:

$$q_1 = arctan\left(\frac{P_y}{P_x}\right) \tag{6.1}$$

Considering the elements 2 and 3 (in the same plane) and using the **Law of cosines**:

$$r^2 = (P_x)^2 + (P_y)^2 \quad \Rightarrow \quad r^2 + (P_x)^2 = (I_2)^2 + (I_3)^2 + 2(I_2)(I_3)cosq_3 \Rightarrow \tag{6.2}$$

$$cosq_3 = \frac{(P_x)^2 + (P_y)^2 + (P_z)^2 - (I_2)^2 - (I_3)^2}{2(I_2)(I_3)} \tag{6.3}$$

This expression allows to obtain q_1 depending on the position vector at **P**. In order to save computational efforts, the function *arctan* will be calculated:

$$senq_3 = \pm\sqrt{1 - cos^2q_3} \quad \Rightarrow q_3 = arctan\left(\pm\frac{1 - cos^2q_3}{cosq_3}\right) \tag{6.4}$$

$$cosq_3 = \frac{(P_x)^2 + (P_y)^2 + (P_z)^2 - (I_2)^2 - (I_3)^2}{2(I_2)(I_3)} \tag{6.5}$$

The two solutions for q_3 are due to the two different configurations which the robot can deal with (see figure 6.5):

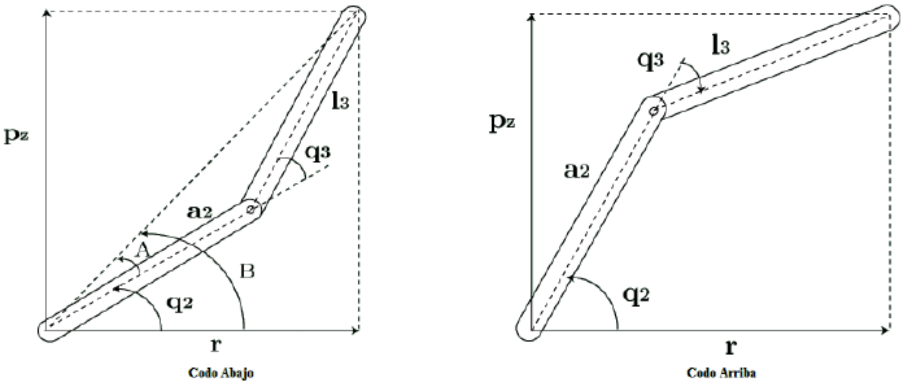


Figure 6.5: Possible two choices for the same purpose.

At this point, q_2 is calculated from the difference between β and α:

$$q_2 = \beta - \alpha \quad \Rightarrow \quad \beta = arctan\left(\frac{P_z}{r}\right) = arctan\left(\frac{P_z}{\mp\sqrt{P_x^2 + P_y^2}}\right) \tag{6.6}$$

Finally:

$$q_2 = arctan\left(\frac{P_z}{\mp\sqrt{P_x^2 + P_y^2}}\right) - arctan\left(\frac{l_3 senq_3}{l_2 + l_3 cosq_3}\right) \tag{6.7}$$

These two possible values depend again on the possible possibilities to choose, as specified in (see 6.1), when the number of solutions was detailed.

6.2.2 Solution by algebraic methods

A priori, it is possible to determine the inverse kinematics model of a robot when knowing the forward kinematics parameters. In other words, the task of obtaining the inverse relations taking as starting-point the known expressions for position and orientation can be developed (see [50]).

However, this method is not always easy and is sometimes so complex that it must be rejected. Besides, as the forward kinematics problem solved by transformation matrices has **twelve equations** (in the case of a six-DOF robot), six final relations are expected, but some dependencies between the twelve initial expressions will be possible to find. The decision of this equations must be made carefully, because sometimes transcendental equations appear, and these ones are not possible to solve by using purely algebraic procedures. In order to avoid that, instead of isolating directly **T** other combinations are searched. This is an **heuristic method**, not a systematic one.

The goal of this procedure is to deduce the values of the articular variables $q_k(t) = q_1(t), q_2(t)...q_k(t)$ depending on the vectors n**,** s**,** a **and** p, which define the **end effector** location (position and orientation) as part of the **transformation matrix T**.

$$\begin{pmatrix} n_x & s_x & a_x & p_x \\ n_y & s_y & a_y & p_y \\ n_z & s_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{6.8}$$

where:

- $\vec{n}$: Normal vector at the end effector. If the hand is like a clamp, this vector is orthogonal to the fingers.

- $\vec{s}$: Sliding vector at the end effector. It is pointing to the direction of the fingers movement when these are moving.

- $\vec{a}$: Approaching vector at the end effector. It is pointing to the normal direction to the back of the hand.

- $\vec{p}$: Position vector at the end effector. It is pointing from the coordinate system origin in the base to the coordinate system origin in the hand.

6.2.2.1 Example of a Three-DOF-spherical-arm robot algebraic resolution

A proper example which can be solved in a relatively easy way by using this method is detailed next. The robot which is aspired to be solved is shown in figure 6.6:

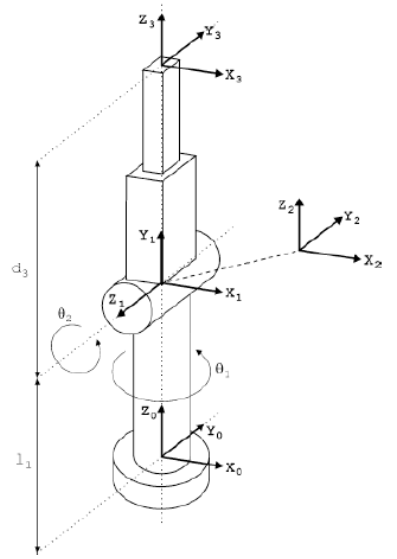


Figure 6.6: Robot to solve: Three-DOF-spherical-arm robot.

The table with the Denavit-Hartenberg parameters is shown below:

Link	a_i	α_i	d_i	θ_i
1	0	$\pi/2$	l_1	θ_1
2	0	$-\pi/2$	0	θ_2
3	0	0	d_3	0

Table 6.1: D-H parameters to solve the given robot.

Thanks to these parameters previously defined, it will be possible to find equivalences in the following way (see [51]):

$$T = {}^0A_1\,{}^1A_2\,{}^2A_3 \;\Rightarrow\; ({}^0A_1)^{-1}\,T = {}^1A_2\,{}^2A_3 \;\Rightarrow\; ({}^1A_2)^{-1}\,({}^0A_1)^{-1}\,T = {}^2A_3 \quad (6.9)$$

Where:

$$
{}^{0}A_1 = \begin{pmatrix} cos\theta_1 & 0 & sin\theta_1 & 0 \\ sin\theta_1 & 0 & -cos\theta_1 & 0 \\ 0 & 1 & 0 & l_1 \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad {}^{1}A_2 = \begin{pmatrix} cos\theta_2 & 0 & sin\theta_2 & 0 \\ sin\theta_2 & 0 & -cos\theta_2 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{6.10}
$$

$$
{}^{2}A_3 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_3 \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad {}^{0}A_2 = \begin{pmatrix} cos\theta_1 cos\theta_2 & -sin\theta_1 & -cos\theta_1 sin\theta_2 & 0 \\ sin\theta_1 cos\theta_2 & co(\theta_1 & -sin\theta_1 sin\theta_2 & 0 \\ sin\theta_2 & 0 & cos\theta_2 & l_1 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{6.11}
$$

$$
T = \ {}^{0}A_3 = \begin{pmatrix} cos\theta_1 cos\theta_2 & -sin\theta_1 & -cos\theta_1 sin\theta_2 & -d_3 cos\theta_1 sin\theta_2 \\ sin\theta_1 cos\theta_2 & cos\theta_1 & -sin\theta_1 sin\theta_2 & -d_3 sin\theta_1 sin\theta_2 \\ sin\theta_2 & 0 & cos\theta_2 & c_3 cos\theta_2 + l_1 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{6.12}
$$

Performing as described before:

$$
({}^{0}A_1)^{-1}\ T =^{1} A_2\ {}^{2}A_3 \tag{6.13}
$$

$$
\begin{pmatrix} cos\theta_1 & sin\theta_1 & 0 & 0 \\ 0 & 0 & 1 & -l_1) \\ sin\theta_1 & -cos\theta_1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} n_x & s_x & a_x & p_x \\ n_y & s_y & a_y & p_y \\ n_z & s_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{pmatrix} = \tag{6.14}
$$

$$
\begin{pmatrix} cos\theta_2 & 0 & -sin\theta_2 & 0 \\ sin\theta_2 & 0 & cos\theta_2 & 0) \\ 0 & -l_1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0) \\ 0 & 0 & 1 & d_3 \\ 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} cos\theta_2 & 0 & -sin\theta_2 & -sin\theta_2 d_3 \\ sin\theta_2 & 0 & cos\theta_2 & cos\theta_2 d_3 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{6.15}
$$

Choosing element $(3,4)$:

$$sin\theta_1\ p_x - cos\theta_1\ p_y = 0\ \Rightarrow\ tan\theta_1 = \frac{p_y}{p_x} \qquad \Rightarrow \qquad \theta_1 = arctan\left(\frac{p_y}{p_x}\right) \tag{6.16}$$

In addition:

$$({}^1A_2)^{-1}\ ({}^0A_1)^{-1}\ T =\ {}^2A_3 \tag{6.17}$$

$$\begin{pmatrix} cos\theta_2 & sin\theta_2 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ -sin\theta_2 & cos\theta_2 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} cos\theta_1 & sin\theta_1 & 0 & 0 \\ 0 & 0 & -1 & -l_1 \\ sin\theta_1 & -cos\theta_1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} n_x & s_x & a_x & p_x \\ n_y & s_y & a_y & p_y \\ n_z & s_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_3 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{6.18}$$

Calculating:

$$\begin{pmatrix} cos\theta_2 cos\theta_1 & cos\theta_2 sin\theta_1 & sin\theta_2 & -l_1 sin\theta_2 \\ -sin\theta_1 & cos\theta_1 & 1 & 0 \\ -sin\theta_2 cos\theta_1 & -sin\theta_2 sin\theta_1 & cos\theta_2 & -l_1 cos\theta_2 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} n_x & s_x & a_x & p_x \\ n_y & s_y & a_y & p_y \\ n_z & s_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_3 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{6.19}$$

Choosing element $(1,4)$:

$$cos\theta_2 cos\theta_1 p_x + cos\theta_2 sin\theta_1 p_y + sin\theta_2 p_z - l_1 sin\theta_2 = 0 \quad \Rightarrow \tag{6.20}$$

$$\theta_2 = arctan\left(\frac{-cos\theta_1 p_x + sin\theta_1 p_y}{p_z - l_1}\right) \tag{6.21}$$

Choosing element $(3,4)$:

$$- sin\theta_2 cos\theta_1 p_x - sin\theta_2 sin\theta_1 p_y + cos\theta_2 p_z - cos\theta_2 l_1 = d_3 \quad \Rightarrow \tag{6.22}$$

$$d_3 = cos(\theta_2)(p_z - 1) - sin(\theta_2)(cos\theta_1 p_x + sin\theta_1 p_y) \tag{6.23}$$

The same result can reached by using geometric procedures.

6.3 Iterative methods

Most of the movement chains are too complex to allow an analytic solution. In these cases, the inverse kinematics problem can be solved by using a sequence of steps leading to incrementally find a better solution for the joint angles. The aim is to minimize the difference between the current and desired positions of the end effector.

The most known method is the **inverse Jacobian matrix**.

6.3.1 Solution by the Jacobian inversion method

Definition 38 *The* ***Jacobian*** *is the multidimensional extension to the differentiation of a single variable. As there is a relation between the Cartesian space of the* ***end effector*** *and the joint space of the joint angles, the Jacobian transforms the differential angle changes to the differential movements of the* ***end effector****. It shows how each coordinate changes with respect to each* ***joint angle*** *in the system. As a result, the Jacobian is extremely useful, since it describes the first order linear behaviour of a system.*

$$\dot{X} = J(\theta)\dot{\theta}, \tag{6.24}$$

where the vector $\dot{X}$ represents the expected change in the **end effector**. It is composed by the linear velocities ($\hat{x}$, $\hat{y}$, $\hat{z}$) and rotational velocities (θ_x, θ_y, θ_z). The desired change is based on the difference between the current position/orientation and what is specified in the goal configuration. $\dot{\theta}$ represents the vector of angular velocities, which are the equation unknown variables. **J** is the matrix which relates the two matrices before and it depends on the current position (see [47]).

Hence, the equation (6.24) can be built in the following way, in relation to the system degrees of freedom (DOF). Normally the number of DOF is six (from θ_1 to θ_6): three for rotations and three for translations.

$$[\hat{x}, \hat{y}, \hat{z}, \theta_1, \theta_2, \theta_3] = \begin{pmatrix} \frac{\partial \hat{x}}{\partial \theta_1} & \frac{\partial \hat{x}}{\partial \theta_2} & \cdots & \frac{\partial \hat{x}}{\partial \theta_n} \\ \frac{\partial \hat{y}}{\partial \theta_1} & \frac{\partial \hat{y}}{\partial \theta_2} & \cdots & \frac{\partial \hat{y}}{\partial \theta_n} \\ \frac{\partial \hat{z}}{\partial \theta_1} & \frac{\partial \hat{z}}{\partial \theta_2} & \cdots & \frac{\partial \hat{z}}{\partial \theta_n} \end{pmatrix} = [\theta_1, \theta_2, \theta_3, \theta_4, \cdots, \theta_n] \tag{6.25}$$

The Jacobian can be obtained column by column from the transformation matrices A_i.

Since the variable is $\dot{\theta}$, the Jacobian inversion is needed. Therefore, the equation before (6.24) is transformed to the form:

$$\dot{\theta} = J^{-1}(\theta)\dot{X} \tag{6.26}$$

6.3.1.1 Iterative model

The Jacobian inversion method works in two phases. Partial transformations on the joint angles are computed first. After that, the end effector position and the Jacobian are computed. Then the end effector locations is changed.

The second part contains Jacobian matrix inversion and joint angles changes. The next step consists of the repetition of the first step and of the change of the end effector position. The obtained differential position of the end effector position (dX) is inserted in phase two. These steps are repeated until the **error** (difference between the current and the desired location) comes below a defined value ϵ or the maximal number of iterations are reached (see [52]):

$$\|J(d\theta) - dX\| \leq \epsilon \quad \vee \quad iterations \geq maximal \tag{6.27}$$

The algorithm for developing the **Jacobian inversion method** is summarized in the following figure (6.7):

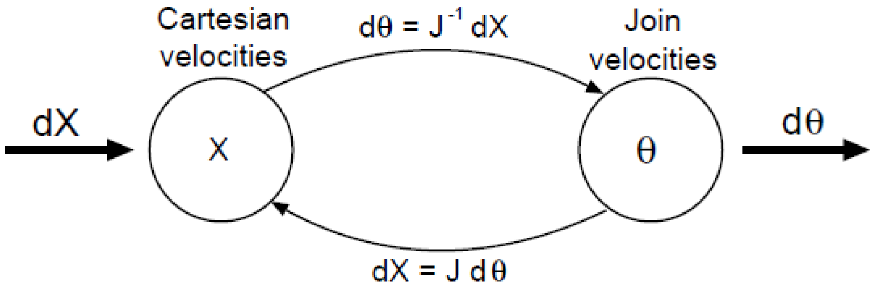


Figure 6.7: Iterative model for the Jacobian inversion method.

Within this kind of solutions, the Jacobian inversion method can be implemented in many ways: **Jacobian Pseudo-inverse, Jacobian Transpose, Singular Value Decomposition, Damped Least Squares or the Feedback Inverse Kinematics method**. The big complexity of these methods and the huge computational efforts that they demand make another family of methods, based in numerical solutions, strongly appear. The possible algorithm to find these numerical solutions will be detailed in the following chapter.

Chapter 7

Numerical methods and their applications to solve the Inverse Kinematics Problem

The task of finding the angles which are needed to move the robot joints to the desired position is a very complex mathematical problem (as it has been specified before) and all the possible methods are intended to reach the best value possible in order to get the minimum difference between the **desired position** (the coordinates where theoretically the **end effector** must go) and the obtained position (the coordinates where it is in the real world). The Inverse Kinematics problem is destined to minimize this difference (before called ϵ as much as possible).

As it has studied in previous chapters, not always the solution for the Inverse Kinematics problem can be reached by an analytic way, specially when the robot configuration is extremely difficult to solve. At this point, **iterative methods** are used in order to find a solution for **non-linear system**, but many iterations are required to delimit the error.

Consequently, the number of calculations (it is very important to remember the huge computational necessity for the **Jacobian inversion**) and the reliability of the final results make the method choice a vital aspect.

7.1 Taylor's Theorem

Definition 39 ***Taylor's Theorem*** *gives an approximation of a* ***defined and k-times differentiable function*** *(with so many variables as wanted) around a neighbourhood of the point* ***a*** $(a-\epsilon, a+\epsilon)$ *by a* ***k-th order Taylor polynomial****. The higher is the polynomial order, the better is the approximation at the evaluated point and the higher the number of points inside the neighbourhood which satisfy such approximation (see [53]).*

7.1.1 Taylor's theorem in one variable

Given a function $f \in C^k$, and $a \in dom(f)$, the Taylor k-order series expansion in one variable at the point a is defined as the following **power series**:

$$P_{k,a}f(x) = f(a) + \frac{f'(a)}{1!}(x-a) + \frac{f''(a)}{2!}(x-a)^2 + \cdots + \frac{f^{k)}(a)}{k!}(x-a)^k \qquad (7.1)$$

In general, it can also be written as follows:

$$P_a f(x) = \sum_{k=0}^{\infty} \frac{f^{k)}(a)}{k!}(x-a)^k, \qquad (7.2)$$

where $P_{k,a}(x)$ is a polynomial with an approximated value to the function $f(x)$ at the point a, $f^{k)}(a)$ are the successive derivatives of the function which is being approximated at the point a and k is the polynomial order.
This polynomial satisfies with $f(x) \simeq P_{k,a}(x)$ if $a \in (a-\epsilon, a+\epsilon)$, so the polynomial value in the neighbourhood of the point a is approximately the value of the function $f(x)$ at that point. Moreover, it is trivial to see how the polynomial derivatives in the point a until the order k are coincident with the function $f(x)$ derivatives at that point.

In other words, one can deduce that (see [54]):

$$\lim_{x \to a} \frac{f(x) - P_{k,a(x)}}{(x-a)^k} = 0 \qquad (7.3)$$

This result indicates not only the difference between $f(x)$ and $P_{k,a}(x)$ is lower when x approaches to a, it also means this difference is lower even comparing it with $(x-a)^k$. Definitely, it is possible to say that Taylor polynomials are a very good way to approximate a function in a point.

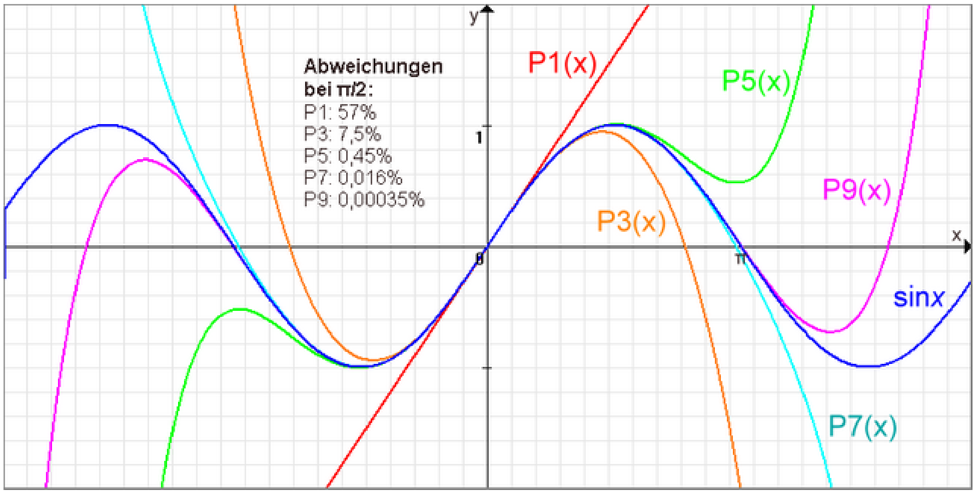


Figure 7.1: Different approaches to the function $sin(x)$ (in dark blue) by several Taylor polynomials (k=1 in red, k=3 in orange, k=5 in green, k=7 in light blue and k=9 in violet).

As it is possible to see, as higher is the order higher is the neighbourhood which satisfy the approximation. In the chart it can be observed the different deviations when the functions are measured in $\frac{\pi}{2}$.

The difference between the obtained value by the polynomial use and real value of the function at the desired point is called **remainder term** ($R_k(x)$):

$$R_k(x) = |f(x) - P_{k,a}(x)| \tag{7.4}$$

The remainder mean-value form can be expressed as follows:

$$R_k(x) = \frac{f^{k+1}(c)}{(k+1)!}(x-a)^{k+1}, \tag{7.5}$$

where $c \in (x-a, x+a)$.

7.1.2 Taylor's Theorem for multivariable functions

Analogously to the series expansion for functions with one variable, the Taylor's Theorem for function with several variables describes the approximation to a multivariable function at a given point (defined in so many coordinates as wanted) by using a multivariable polynomial. In this case the neighbourhood of the point is not an interval in $\mathbb{R}$, it is a ball with so many dimensions as variables in the space (in $\mathbb{R}^n$ for n variables).

Given $f \in C^k$, and $a := (a_1, \cdots, a_k)$, $a \in dim(f)$, the expression which allows the calculation is presented below (see [55]):

$$P_{k,a}f(x) = f(a) + \sum_{i=1}^{n} \frac{\partial f(a)}{\partial x_i}(x_i - a_i) + \frac{1}{2!}\sum_{i,j=1}^{n} \frac{\partial^2 f(a)}{\partial x_i \partial x_j}(x_i - a_i)(x_j - a_j) + \cdots \quad (7.6)$$

$$\cdots + \frac{1}{k!} \sum_{i_1, i_2, i_3, \cdots, i_k = 1}^{n} \frac{\partial^k f(a)}{\partial x_{i_1} \partial x_{i_2} \partial x_{i_3}, \cdots, \partial x_{i_k}} (x_{i_1} - a_{i_1})(x_{i_2} - a_{i_2})(x_{i_3} - a_{i_3}), \cdots, (x_{i_k} - a_{i_k}), \quad (7.7)$$

where a and x are sets with so many variables as dimensions in the space, $P_{k,a}(x)$ is a multivariable polynomial with an approximated value to the multivariable function $f(x)$ at the point a, $\frac{\partial^k f(a)}{\partial x_i}$ are the successive partial derivatives of the function with respect each one of the variables which are composing the set of variables x at the multivariable point a and k is the polynomial order.

As in the case for one variable, is defined the **remainder term** by the equality:

$$R_{k,a}(x) = f(x) - P_{k,a}(x) \quad (7.8)$$

The equation (7.9) is also satisfied, so the similarity between the function and its Taylor polynomial is better the nearer is a and the higher is the order.

$$\lim_{x \to a} \frac{R_{k,a}(x)}{\|x - a\|^k} = 0 \quad (7.9)$$

These **remainder term** can be calculated now in this way:

$$R_{k,a}(x) = \frac{1}{(k+1)!} \sum_{i_1,\cdots,i_{k+1}}^{n} \frac{\partial^{k+1} f(c)}{\partial x_{i_1} \cdots \partial x_{i_{k+1}} (x_i - a_i) \cdots (x_{i_{k+1}} - a_{i_{k+1}})}, \tag{7.10}$$

where c is a multivariable point and $c \in (x - a, x + a)$.

The figure 7.2 shows a graph with two multivariable approximations to a given function at an specific point, in this case in two dimensions:

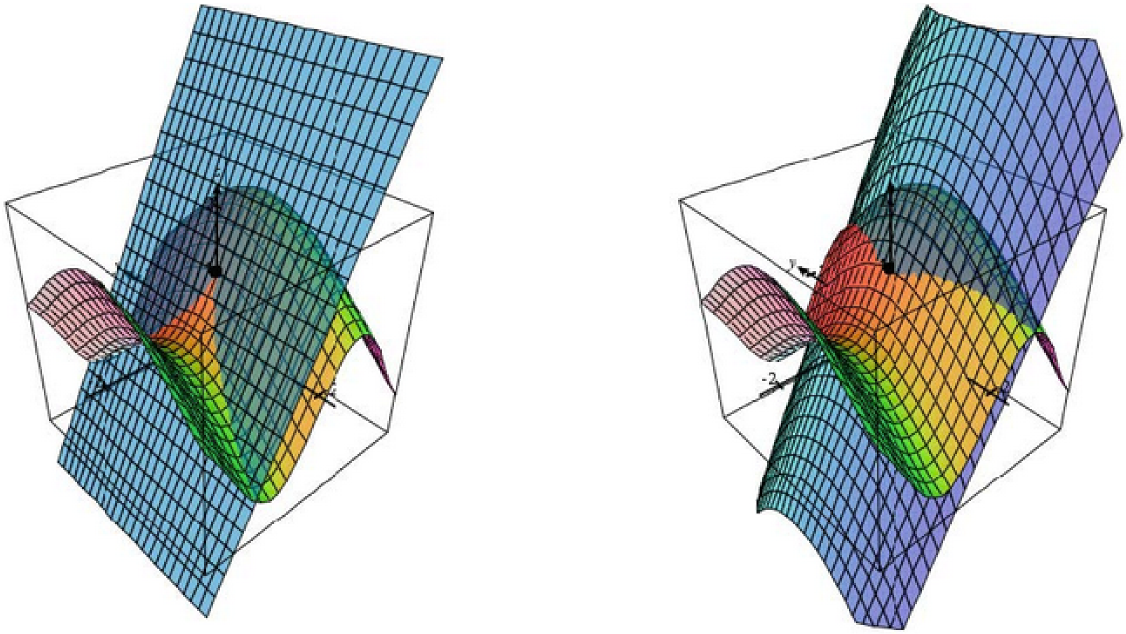


Figure 7.2: First order Taylor polynomial approximation for the sinusoidal function f (left) and second order Taylor polynomial approximation for the same function (right). One can observe that the first degree polynomial in 2D is a plane (in 1D is a line) and the second degree polynomial is corresponded with a curved plane inside a three-dimensional space.

7.2 Optimization methods to solve equations

The great number of variables in the Inverse Kinematics problem and the complexity which their resolution requires make impossible to use an analytic method. Hence, the values must be estimated in order to satisfy as much as possible all the equations at the same time, what is a very difficult task.

Optimization methods are iterative procedures in which the solutions are approximated incrementally having as basis the previous values, so the consequence is that the higher the number of iterations is, the better is the obtained approximation. Of course, there are several possibilities to do it. The most used ones will be seen in this section.

7.2.1 Gradient descent

Definition 40 ***Gradient descent****, also known as steepest descent, is a optimization method based in the iterative application of the first-order multivariable Taylor approximation in order to reach a desired local minimum.*

Given a differentiable and defined scalar field (it can be a multivariable function) $f(x)$ *in a neighbourhood of a point a (which can be a multidimensional point), gradient decreases fastest if one goes from a in the direction of the negative gradient of* $F(a)$ $(-\nabla F(a))$.

This method is evidently necessary when solving asymmetric functions, because for symmetric ones all the possible directions are the same. It is possible to find an example about this aspect in figure 7.3 (left), where the direction to move does not matter because there are infinite number of curves to get the goal. In contrast, when functions are complex (which is unfortunately an inherent circumstance in Inverse Kinematics) the direction to move must be chosen rationally. In the figure 7.3 (right) it is possible to distinguish two possibilities by using the **descent gradient method** because depending on the initial position the method is only able to reach a local minimum, but not always the global one is guaranteed.

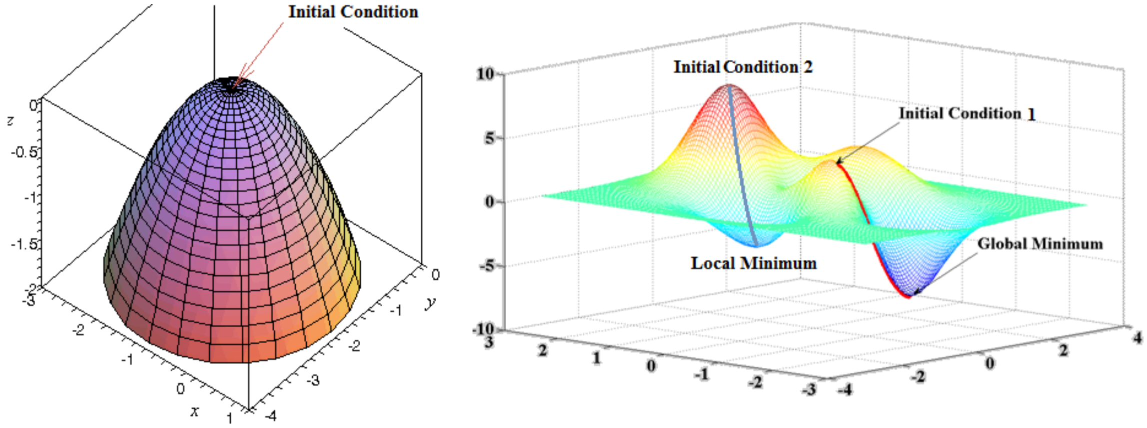


Figure 7.3: Graph example to show the different aspects to consider when using **descent gradient** depending on the function shape and the chosen initial point.

It is not hard to see why this method is one of the most popular ones: it is very simple, easy to use, and repetitions are fast. But the biggest advantage of this method is it **is the guaranty to find a local minimum even when a lot of iterations are needed** (see [56]).

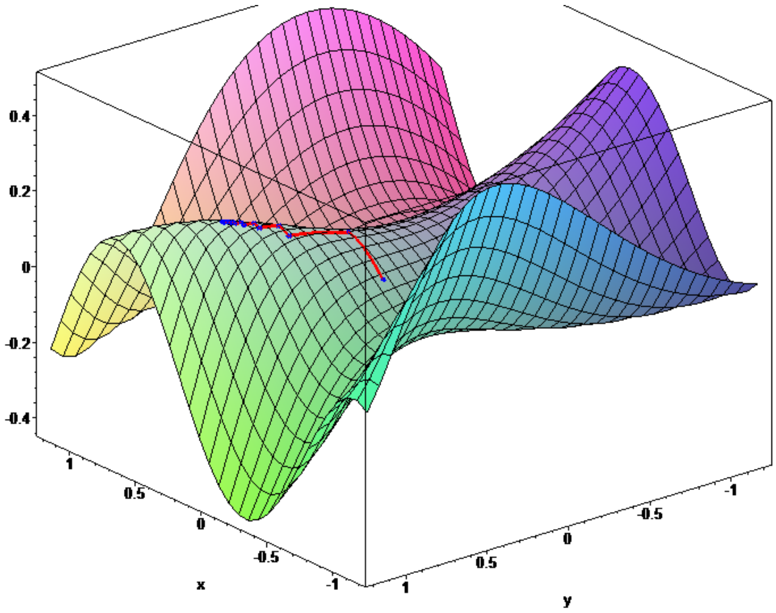


Figure 7.4: Zig-zag movement when reaching the steepest direction by using decent gradient.

Nevertheless, the problem with this procedure is it is **relatively slow in closing to the minimum**: technically, its rate of convergence is inferior to many other methods. The reason is the method implies a right angle ($\frac{\pi}{2}$) turn at the conclusion of each search line, so when the steepest descent is reached moving diagonally it has to **zig-zag** back and forth across the proper direction (see figure 7.4), what **is very inefficient** because the progress is slow.

7.2.1.1 Descent gradient algorithm

The algorithm which this method follows is shown below (see [57]). It is initialized with a guess (x), a maximum iteration count (N_{max}), a gradient norm tolerance (ϵ_g) that is used to determine whether the algorithm has arrived at a critical point, and a step of tolerance (ϵ_x) to determine whether significant progress is being made.

```
1. for $i = 1, 2, \ldots, N_{max}$ $x_{i+1} \leftarrow x_i - \alpha_i \nabla f(x_i)$

2. If $\|\nabla f(x_{i+1})\| < \epsilon_g$ then return "Converged on critical point"

3. If $\|x_i - x_{i+1}\| < \epsilon_x$ then return "Converged on a $x$ value"

4. If $f(x_{i+1})\| > f(x)$ then return "Diverging"

5. Return "Maximum number of iterations reached"
```

The variable α_i is known as the **step size**, and should be chosen to maintain a balance between convergence speed and avoiding divergence. As it can seen, this value depends on each one of the iterations i.

7.2.2 Newton-Raphson Method

Definition 41 *The* ***Newton-Raphson Method****, also known as only Newton's Method, is a procedure for finding successively better approximations to the roots of a real-valued function.*

Given $f(x)$ as a continuous function and continuously differentiable, it is possible to reach its roots by approximations using tangents to the function and having as the following approximation the point where the tangent line crosses with the x-axis.

7.2.2.1 Newton-Raphson method to solve one-variable functions

Let x_0 a good guess of a root r and let $r = x_0 + h$. Since the true root is r, and $h = r - x_0$, then h measures how far the guess x_0 is from the true value of the root (see [58]).

Since h is very small, it will be used the tangent line approximation to conclude that:

$$f(r) = 0 = f(x_0 + h) \approx f(x_0) + hf'(x_0) \tag{7.11}$$

This expression can also be obtained from the **Taylor expansion** in $x_0 + h$ (see equation (7.12)):

$$f(x_0 + h) = f(x_0) + f'(x_0)h + \frac{1}{2}f''(x_0)h^2 + \dots, \tag{7.12}$$

where $h = (x - (x_0 + h))$ and it has been shown all the terms until the second-order expansion.

Hence, only keeping the first-order terms and considering that (x_0+h) approaches to r, it can be deduced that $f(x_0+h) = f(r) = 0$ and the same expression as before (7.11).

Therefore, unless $f'(x_0)$ is close to 0:

$$h \approx -\frac{f(x_0)}{f'(x_0)} \tag{7.13}$$

Thus:

$$r = x_0 + h \approx x_0 - \frac{f(x_0)}{f'(x_0)} \tag{7.14}$$

So the first estimation is:

$$x_1 = x_0 - \frac{f(x_0}{f'(x_0)} \tag{7.15}$$

The following i iterations are implemented in exactly the same way as x_1 has been obtained from x_0:

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)} \tag{7.16}$$

A graphical example is presented in figure 7.5:

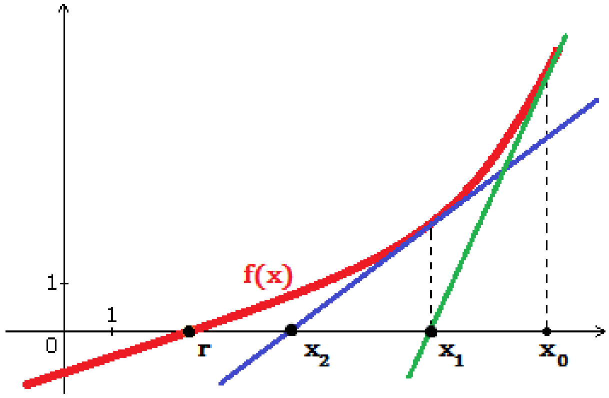


Figure 7.5: Newton-Raphson method for a one-variable funtion, where r is the root and x_i the successive approximations by using tangents.

7.2.2.2 Newton-Raphson method to solve and minimize multivariable functions

The previous procedure can also be applied to solve systems of m equations with n variables. In this case the expression (7.11) can be written as follows (see [59]):

$$g_m(r) = g(x_i) + J_g(x_i)h, \tag{7.17}$$

where $g_m(x_i) = \begin{pmatrix} g_1(x_1, x_2 \ldots x_n) \\ g_1(x_1, x_2 \ldots x_n) \\ \vdots \\ g_m(x_1, x_2 \cdots x_n) \end{pmatrix}$ is a system of n variables and m equations, $J_g(x) = \begin{pmatrix} \frac{\partial g_1}{\partial x_1} \frac{\partial g_1}{\partial x_2} \cdots \frac{\partial g_1}{\partial x_n} \\ \frac{\partial g_2}{\partial x_1} \frac{\partial g_2}{\partial x_2} \cdots \frac{\partial g_2}{\partial x_n} \\ \vdots \\ \frac{\partial g_m}{\partial x_1} \frac{\partial g_m}{\partial x_2} \cdots \frac{\partial g_m}{\partial x_n} \end{pmatrix}$ is the Jacobian matrix of partial derivatives of the component functions of $g(x)$, and h is again the distance from the initial guess to the root which is been searched $(h = (r - x_0))$.

In the same way as before, the final expression of the **Newton-Raphson method for several variables** is:

$$x_{i+1} = x_i - [J_g(x_i)]^{-1} g(x_i) \tag{7.18}$$

In practice, x_{i+1} is not calculated by computing $[J_g(x_i)]^{-1}$ and the multiplying by $g(x_i)$, because the calculations which are needed for the Jacobian inversion make it computationally inefficient. Instead of that, it is more practical to solve the following system of linear equations:

$$J_g(x_i)\ s_n = -g(x_i), \tag{7.19}$$

where the unknown s_i is solved by using a method such a Gaussian elimination, and then setting $x_{i+1} = x_i + s_i$.

In the Inverse Kinematics problem the function $f(x)$ (with $m \geq n$) must be minimized and the system of equations $\nabla f(x) = 0$ solved. Then, $g(x) = \nabla f(x)$, and $J_j(x) = Hf(x)$. Therefore, the Newton-Raphson method adopts the form bellow:

$$x_{i+1} = x_i - [H_g(x_i)]^{-1} \nabla f(x_i), \tag{7.20}$$

where $H_g(x_i)$ is the Hessian matrix, the square matrix with the second-order partial derivatives and $\nabla f(x_i)$ is the gradient.

- If $H_g f(x_i)$ is positive definite, then its critical point is also guaranteed to be the unique strict global minimizer of $f(x)$.

When used for minimization, the Newton-Raphson method approximates $f(x_i)$ by its quadratic approximation near x_i.

Now, computationally the system to solve is:

$$H_f(x_i) p_i = \nabla f(x_i), \tag{7.21}$$

where $p_i = [H_f(x_i)]^{-1} \nabla f(x_i)$ is the vector which solves the system and determines the search direction.

The **main advantage of this method** is the high efficiency it has when the initial guess x_0 is close to r. Besides, it also converges faster than gradient descent, avoiding the zig-zag movement at the final steps (see figure 7.6). **In contrast**, this convergence is not completely guaranteed, and if the initial point is far from the root then the method may not converge. Due to that, this **initial guess must be chosen carefully**.

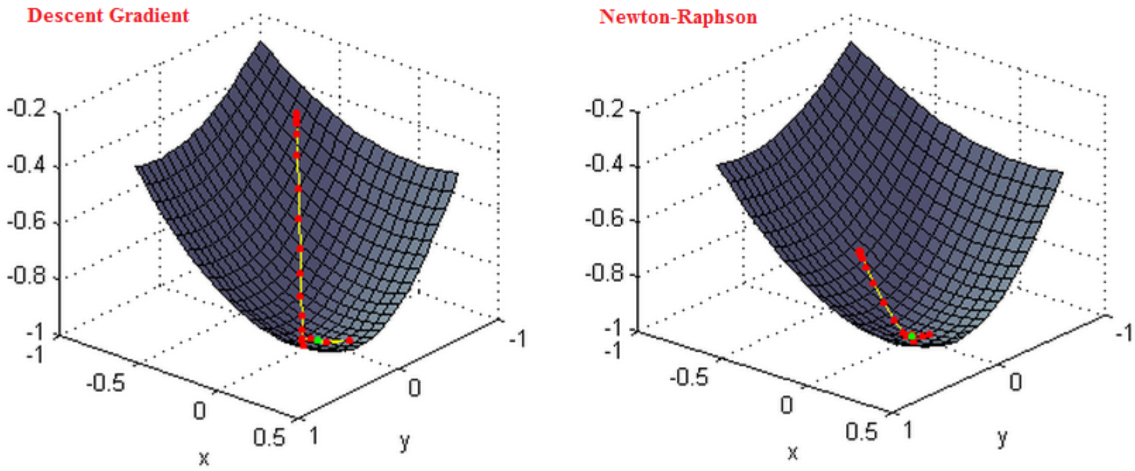


Figure 7.6: Comparison between the methods of Gradient descent and Newton-Raphson. Convergence is much faster in this last one (and because of that the efficiency is higher), but it is not assured if the initial guess is not close to the root.

7.2.3 Gauss-Newton Method

Definition 42 *The* ***Gauss-Newton Algorithm*** *is a method used to solve* ***non-linear least squares problems****. It can be seen as a modification of the* ***Newton-Raphson method*** *for finding a minimum in a function without using second-order derivatives, what Newton-Raphson method cannot do.*

Given m functions $g_m(x) = (g_1(x), g_2(x), \ldots, g_m(x))$ of n variables $x = (x_1, x_2, \ldots, x_n)$ with $m \geq n$, the Gauss-Newton algorithm iteratively finds the minimum of the sum of squares:

$$S(g) = \frac{1}{2}\sum_{i=1}^{m} g_i(x)^2 \tag{7.22}$$

7.2.3.1 Deduction from the Newton-Raphson method

The Gauss-Newton Method is developed from Newton-Raphson by approximation and it is deduced having as starting point the equation (7.20). Taking into account that $S(g) = \sum_{i=1}^{m} g_i(x)^2$, the **gradient vector** of S is given by:

$$\nabla S(g) = \sum_{i=1}^{m} g_i(x) \frac{\partial g_i(x)}{\partial x_j}, \tag{7.23}$$

where $i = 1, 2, \ldots, m$ and $j = 1, 2, \ldots, n$.
The Hessian is calculated by differentiating the gradient elements with respect to x_k by the chain rule:

$$H_g(x) = \sum_{i=1}^{m} \left(\frac{\partial g_i(x)}{\partial x_j} \frac{\partial g_i(x)}{\partial x_k} + g_i(x) \frac{\partial^2 g_i(x)}{\partial x_j \partial x_k} \right) \tag{7.24}$$

At the same time, it is known that the Jacobian is:

$$J_g(x) = \frac{\partial g_i(x)}{\partial x_j} \tag{7.25}$$

The Newton-Gauss method is obtained by ignoring the second-order derivative terms. Knowing the Jacobian, the Hessian is given by:

$$H_g(x) \simeq \sum_{1=1}^{m} \left(\frac{\partial g_i(x)}{\partial x_j} \frac{\partial g_i(x)}{\partial x_k} \right) = J_g(x)^T J_g(x) \tag{7.26}$$

As $\nabla g_i(x) = J_g(x)^T g_i(x)$, the equation (7.20) is the **Gauss-Newton formula**:

$$x_{i+1} = x_i - [H_g(x)]^{-1} \nabla g_i(x) = x_i - [J_g(x)^T J_g(x)]^{-1} J_g(x)^T g_i(x) \tag{7.27}$$

In practice, because of the computational inefficiency described in 7.2.2.2, the inverse matrix is never calculated. Instead of that, it is used:

$$x_{i+i} = x_i + p_i, \tag{7.28}$$

where p_i is the vector which determines the search direction.

The update of p_i is computed by solving the linear system (7.29):

$$J_g(x)^T J_g(x) p_i = -J_g(x)^T g_i(x), \tag{7.29}$$

where $g_i(x)$ is a set of m functions of n variables.

Some aspects must also be mentioned (see [60]):

- Assuming that $J_g(x)$ has full rank (the maximum rank possible according to its rows and its columns), the Hessian approximation $H_g(x) = J_g(x)^T J_g(x)$ is positive definite and the Gauss-Newton search direction is a **descent direction**.

- Otherwise, $J_g(x)^T J_g(x)$ is non invertible and the equation (7.29) has no an unique solution. In this case, **the problem is under-determined or over-parametrized**.

As one can observe, the main advantage of this method is it does not need to solve second-order derivatives like in Newton-Raphson case. However it also has disadvantages: unlike Newton-Raphson, the only application of this method is to minimize a sum of squared function values, and not even local convergence is guaranteed. Besides, the Gauss-Newton method may fail if the initial estimation is very far from the minimum.

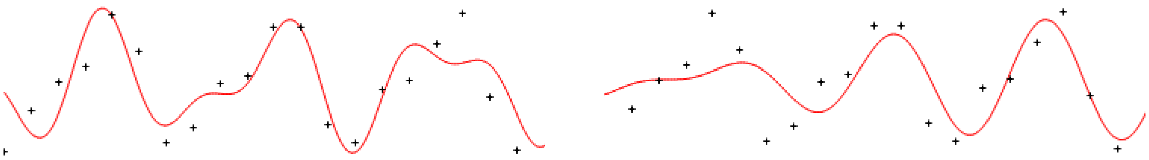


Figure 7.7: Example of the optimization process for one-variable functions by using the Gauss-Newton method.

7.2.4 Levenberg-Marquardt

Definition 43 *The* ***Levenberg-Marquardt Algorithm (LMA)****, also known as* ***damped least-squares*** *(DLS) is the most used optimization algorithm to solve non-linear least squares problems (see* (7.22)*).*

The Levenberg-Marquardt Method interpolates between the ***Gauss-Newton*** *Method (see 7.2.3) and the* ***Gradient descent*** *Method (see 7.2.1). When the current estimation is far from a local minimum, it behaves like the Gradient descent, and when the minimum is close, it behaves like Gauss-Newton, so it converges faster. Furthermore, it is more robust than the Gauss-Newton method, which means that in many cases it can find a solution even when the starting guess is far from the final minimum.*

7.2.4.1 Levemberg-Marquardt's aim

The Levenberg-Marquardt Method tries to combine the advantages of convergence of the previous methods. Hence, Levenberg-Marquardt steps are linear combination of gradient descent and Gauss-Newton, so it comprises of adaptative rules (see [61]).

Gradient descent dominates the algorithm until a valley is reached (in order to avoid zig-zag (see figure 7.4)) and as of that moment Gauss-Newton steps are used because of the good behaviour of this method in that situation.

7.2.4.2 Levenberg-Marquardt's formula

Considering that the initial idea was intended to modify the Newton-Gauss equation to combine it with the gradient descent one, Levenberg proposed this algorithm:

$$x_{i+1} = x_i - (H_g(x_i) + \lambda I)^{-1} \nabla g_m(x), \tag{7.30}$$

where $H_g(x_i)$ is the Hessian matrix evaluated at x_i around the set of functions $g_m(x)$. λ is a positive value named **damping parameter** and it is used to control the influence of gradient descent or Newton-Raphson in the behaviour of the method.

Using the Hessian approximation described in (7.26), the expression above (7.30) can also be written as:

$$x_{i+1} = x_i - ([J_g(x_i)^T J_g(x_i)] + \lambda I)^{-1} \nabla g_m(x) \tag{7.31}$$

The value for λ is initially pre-fixed, normally $\lambda = 10^{-3}$. If the obtained value $\Delta x = x_{i+1} - x_i$ by solving the equations conducts to a **reduction in the error**, then the increment is accepted and λ is divided (usually by 10) in the following iteration. On the other hand, if the value of Δx conducts to increment the error, then λ is multiplied (usually by 10) and the equations are solved again. This process stills until getting a value Δx to decrement the error. The problem of this method is when the value of λ is very large, the calculated Hessian matrix is not used at all.

Maquardt solved this problem by replacing the identity matrix I in the equation before (see (7.31)) for a diagonal of the resulting Hessian, so the **final Levenberg-Marquardt method formula** is (see [62]):

$$x_{i+1} = x_i - (J_g(x_i)^T J_g(x_i) + \lambda diag(H_g(x_i)))^{-1} \nabla g_m(x) \quad \Rightarrow \tag{7.32}$$

$$x_{i+1} = x_i - (J_g(x_i)^T J_g(x_i) + \lambda diag(J_g(x_i)^T J_g(x_i)))^{-1} \nabla g_m(x) \tag{7.33}$$

After this improvement, for large values of λ the Levenberg-Marquardt Algorithm almost behaves like the gradient descent method (with a short step in order to fit low curvatures). This is a good strategy when the current solution is far from the minimum. On the contrary, if λ is a small value (to fit high curvatures), then the Levenberg-Marquardt step is almost identical to the Gauss-Newton step. This is the desired behaviour for the final steps of the algorithm since, near the minimum, the convergence of the Gauss-Newton method can be almost quadratic [63].

The Levenberg-Marquardt Method always converges in a local minimum, but the best solution is not completely guaranteed. In other words, the global minimum may be reached or not depending on the initial considered point, like in the gradient descent method (figure 7.3 (right)).

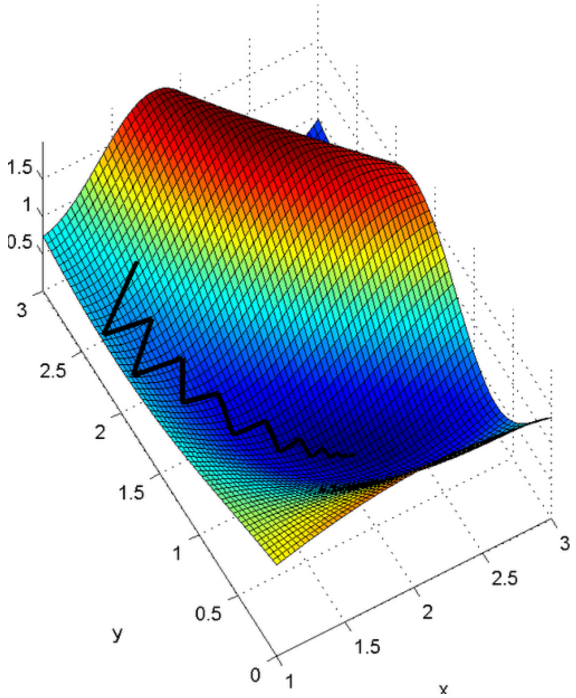


Figure 7.8: The Levenber-Marquardt method applied to a bi-dimensional error function.

Chapter 8

Manifolds in order to avoid non-Euclidean spaces

The most of the methods which have been seen in previous chapters work properly in vector spaces (isomorphic to $\mathbb{R}^n$). Nevertheless, in the case of rotations, this is not always possible and variables to be estimated may not define an **euclidean vector space**.

In the group of rotations $SO(3)$, for example, two optimal approaches can be implemented: by using three parameters or over-parametrizing the system.

When using a representation with three variables (Euler Angles or another kind of representation composed of the one-by-one spatial axis rotation), the problem of **gimbal lock** (see 4.1.3) might appear because **there exist no parametrization of $SO(3)$ with three parameters and no singularities**. When the variable to be estimated approaches such a singularity the inverse of the parametrization becomes discontinuous. Therefore some algorithms's choice to avoid this problem is to change the parametrization if the variables gets close to a singularity.

As it has already been seen, another option is to over-parametrize the system, for example by using quaternions in $SO(3)$ (using four values to parametrize) (see 4.1.5). After that, the parametrization must be re-normalized if needed. Nevertheless, estimation algorithms are not aware of the inner constraints between the parameters, so this is especially problematic when the parametrization of the variable has more parameters than the dimension of the measurement space, since there cannot exist a unique result if ignoring the constraints.

The solution to these problems if to combine both approaches: the variable is globally over-parametrized but local changes are represented with a representation which behaves like a Euclidean space for small values, making possible to operate like in $\mathbb{R}^3$. This is the aim of the **theory of manifolds** (see 2.3).

8.1 Manifolds as state representations

Definition 44 *The* ***dynamic's state representation*** *is a set of physical quantities, the specification of which completely determine the system's temporal evolution.*

The specific physical quantities which define the system's state are not unique, although their number (called the system order) it is (see [65]).

- Example. State for a defined three-dimensional movement.

  $$S = \mathbb{R}^3 \times SO(3) \times \mathbb{R}^3, \tag{8.1}$$

  where three components are represented: position ($\mathbb{R}^3$), orientation ($SO(3)$) and velocity ($\mathbb{R}^3$).

In the simple case in which a state consists of a single component (for example, a three-dimensional orientation), a state can be represented as a single manifold (see 2.3). If a state comprises of multiple different components, then the result is also a manifold since the Cartesian product of the manifolds representing each individual component returns another manifold.

- Example. State S for a defined three-dimensional movement.

$$S = \mathbb{R}^3 \times SO(3) \times \mathbb{R}^3, \tag{8.2}$$

where three components are represented: position ($\mathbb{R}^3$), orientation ($SO(3)$) and velocity ($\mathbb{R}^3$).

8.2 Encapsulation of manifolds

Manifolds are encapsulated when used in standard algorithms in order to avoid their entire handling by the main algorithm. In other words, the goal is to help the estimation algorithm handle the manifold as a **black box** (see [66]).

The algorithm can only access to the manifold in two different ways:

$$\boxplus : M \times \mathbb{R}^m \longrightarrow M, \tag{8.3}$$

where $\delta \mapsto x \boxplus \delta$ is a homeomorphism from a neighbourhood of $0 \in \mathbb{R}^m$ to a neighbourhood of $x \in M$.

The second way is to find the difference between the elements in M:

$$\boxminus : M \times M \longrightarrow \mathbb{R}^m, \tag{8.4}$$

where $y \mapsto y \boxminus x$ is the inverse of $\boxplus$.

Therefore:

$$x \boxplus (y \boxminus x) = y, \tag{8.5}$$

where $x, y \in M$.

Definition 45 *A bijective function f defined around a topological space A and with values in the space B is called a* ***homeomorphism*** *if the function and its inverse are continuous. Obviously, if f is a homeomorphism, then f^{-1} is another one.*

Two topological spaces A and B are ***homeomorphic*** *if there is an homeomorphism between A and B. When two spaces are homeomorphic is possible to exchange them reasoning and demonstrating without modifying conclusions. In that case, those two spaces are considered the same topological object (see [64]).*

Hence, every point of a manifold M has a neighbourhood which can be mapped bidirectionally to $\mathbb{R}^n$, and the algorithm only sees a locally mapped part of M in $\mathbb{R}^{\ltimes}$ at any moment.

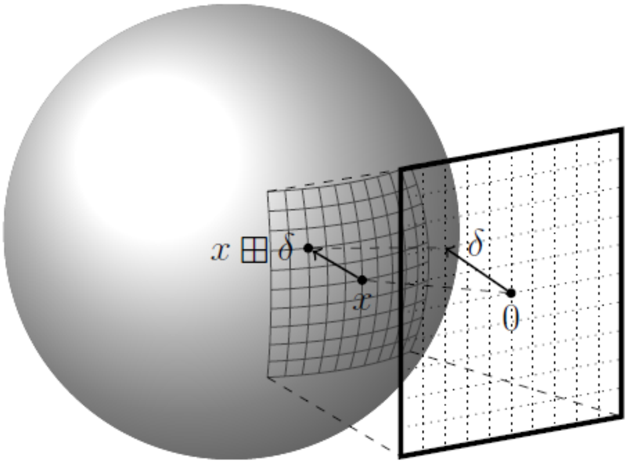


Figure 8.1: Local neighbourhood in the manifold M (here the unit sphere) mapped into $\mathbb{R}^n$ (here $\mathbb{R}^2$, the plane).

As $\delta \mapsto x \boxplus \delta$ is supposed to be a homeomorphism, the dimension f M has to be m. In addition, as the domain of the homeomorphism is a neighbourhood of 0, then $x \boxplus 0 = x$.

The main advantage of this approach is that most standard algorithms working on $\mathbb{R}^m$ can now work in the same way on M, after replacing $+$ with $\boxplus$ when adding small updates to the state, and $-$ with $\boxminus$ when calculating the difference between two states.

Moreover, algorithms do not have to deal with singularities or denormalized over-parametrizations and, after adapting the algorithm, it is not necessary to make further adaptations.

8.2.1 Summary of properties

Given a $\boxplus$-manifold and an open neighbourhood $V \subset \mathbb{R}^n$ of 0 (see [67]):

1. $\delta \mapsto x \boxplus \delta$ must be smooth on $\mathbb{R}^n$
2. $\forall x \in M$, $y \mapsto y \boxminus x$ must be smooth on $x \boxplus V$
3. $x \boxplus 0 = x$
4. $\forall y \in M$: $x \boxplus (y \boxminus x) = y$
5. $\forall \delta \in V$: $(x \boxplus \delta) \boxminus x = \delta$
6. $\forall \delta_1, \delta_2 \in \mathbb{R}^n$: $\|(x \boxplus \delta_1) \boxminus (x \boxplus \delta_2)\| \leq \|\delta_1 - \delta_2\|$

The operators $\boxplus$ and $\boxminus$ allow a generic algorithm to modify and compare manifold states as if they were flat vectors and they can do it without knowing about the internal structure of the manifold, which works as a **black box** for the algorithm.

It is needed the operators to be smooth to make limits and derivatives of δ correspond to limits and derivatives of $x \boxplus \delta$, something vital for any estimation algorithm. Finally, it is important to note that it is not required $x \boxplus \delta$ or $y \boxminus y$ to be smooth in x.

8.2.1.1 Consequences

- The **neutral element** of $\boxplus$ is 0.

- The fourth axiom ensure that from an element x, every other element $y \in M$ can be reached via $\boxplus$, making in this way $\delta \mapsto x \boxplus \delta$ surjective.

- The fifth axiom makes $\delta \mapsto x \boxplus \delta$ injective on V, so the range of perturbations for the parametrization by $\boxplus$ is unique. $\boxplus$ and *boxminus* create a local vectorized view of the state space. Intuitively x is a reference point which defines the center of a neighbourhood in the manifold. Thus, also the coordinate system of δ in the part of $\mathbb{R}^n$ onto which the local neighbourhood in the manifold is mapped.

- The sixth axiom allows to define a metric: the real distance $d(x \boxplus \delta_1, x \boxplus \delta_2)$ is less or equal to the distance $\|\delta_1 - \delta_2\|$ in the parametrization.

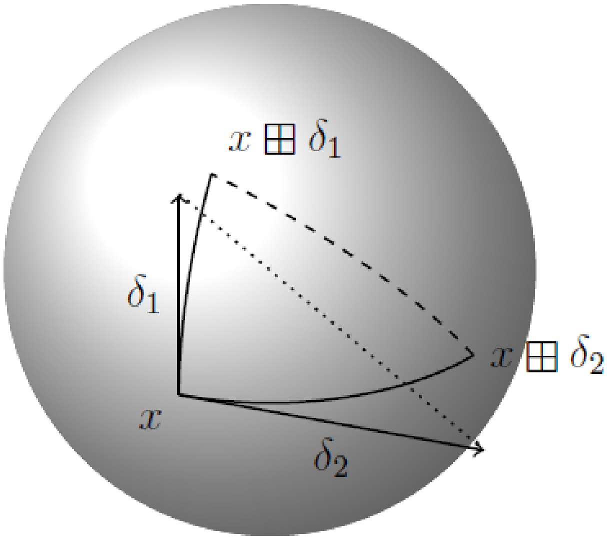


Figure 8.2: Graph to exemplify that the distance between $x \boxplus \delta_1$ and $x \boxplus \delta_2$, represented with the dashed line, is less or equal to the distance in the parametrization around x, which is represented with the dotted line.

8.2.2 Generic definition

The way to define $x \boxplus \delta$ is to choose a chart (see 2.3.1.1) including x, within which a vector $\vec{x}$ can be added, and project the result back to M (as it is shown in figure 9.1). On the other hand, $y \boxminus x$ does the inverse of the previous operation finding the value y in a chart including x and getting its coordinates. The choice of the chart has to be determined by x.

Given ϕ_x as the chosen chart around x, it is possible to define:

$$x \boxplus \delta := \phi_x^{-1}(\phi_x(x)\delta) \tag{8.6}$$

$$y \boxminus x := \phi_x(y) - \phi_x(x) \tag{8.7}$$

This definitions can also be proving in the following way, demonstrating the equation 8.5:

$$x \boxplus (y \boxminus x) = x \boxplus (\phi_x(y) - \phi_x(x)) \Rightarrow \tag{8.8}$$

$$x \boxplus (y \boxminus x) = \phi_x^{-1}(\phi_x(y) + \phi_x(y) - \phi_x(x)) \Rightarrow \tag{8.9}$$

$$x \boxplus (y \boxminus x) = \phi_x^{-1}(\phi_x(y)) = y \tag{8.10}$$

8.3 Examples

8.3.1 Manifold approach to vector space $\mathbb{R}^n$

In this case, the $\boxplus$ and $\boxminus$ operators are planar projections of a planar space, so vector addition and subtraction are implemented as follows:

$$x \boxplus \delta = x + \delta \tag{8.11}$$

$$y \boxminus x = y - x \tag{8.12}$$

8.3.2 Manifold approach to $SO(2)$

The special group of rotations in the plane, $SO(2)$ (see 3), can be described as the set matrix which verify:

$$SO(2) := \{R \in \mathbb{R}^{2\times 2}; R^T R = RR^T = I_{2\times 2} \wedge det(R) = 1\} \tag{8.13}$$

At the same time, this group is also a subgroup of $GL(n)$, the invertible matrices using the standard multiplication. $SO(2)$ has one DOF **(degree of freedom)** (see 5.1.2) and, as it has been studied in previous chapter, it can be parametrized by using one variable:

$$SO(2) = \{ \begin{pmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{pmatrix} with\theta \in \mathbb{R}\} \tag{8.14}$$

A single parametrization can be made defining:

$$\theta \boxplus \delta := \theta + \delta \tag{8.15}$$

$$\theta \boxminus \gamma := \phi(\theta - \gamma) \tag{8.16}$$

where θ is a single rotation angle, γ is the value in a chart whose coordinates are subtracted and $\phi(\delta) = \delta - 2\pi[\frac{\delta + \pi}{2\pi}]$ is the normalization of the difference $\theta \boxminus \gamma = \theta - \gamma$ to the interval $(-\pi, \pi)$ in order to take into account that $\theta + 2k\pi$ represent the same rotation for any integer k.

The procedure before must be developed manually in algorithms which do not use a manifold approach.

Chapter 9

Conclusions

The report which has been developed combining different matters and methods, allows the analysis of several systems with diverse usefulness and features. That is why the task of distinguishing the most relevant points or of making interesting comparisons in relation to the way these systems work is very important when intending to get a critical conclusion. This chapter, consequently, is intended to summarize the points which are vital for the comprehension of the Inverse Kinematics problem and simultaneously define the relationships between the procedures presented here. Since it is not a lab work, it is not possible to include any analysis or obtained value in a practical way. In lieu thereof, the analytical and compared information of the studied theory is the best tool to acquire an useful knowledge about the contents which have been discussed.

9.1 A world of rotations

The Cartesian plane is the most limited space where rotations make sense. Therefore, it is first important to study this case to further understand the three-dimensional world we live in. The group $SO(2)$ (3) defines the meaning of rotations in the plane. In contrast to higher dimensions, the main feature of bi-dimensional rotations is that they have a commutative product due to the group being Abelian. This fact can be checked making use of the methods for rotations in 2D, among which it is possible to highlight rotation matrices and complex numbers.

Rotation matrices are matrices by the form $\begin{pmatrix} cos(\theta) & -sin(\theta) \\ sin(\theta) & cos(\theta) \end{pmatrix}$, which apply a θ rotation around a planar object. The determinant must be 1 in order not to distort or reflect the object shape. On the other hand, complex numbers are based in Euler's formula to group the different arguments (angles) in the exponential exponent, getting the resultant rotation. This method requires the conversion from Cartesian form to polar form and vice versa before and after being implemented.

In a similar way, the group rotations in 3D $SO(3)$ (see 4) also defines all the possible spatial rotations, but there are several methods to do it. The traditional way consists of implementing different rotations independently around each axis, defining the rotation matrix similarly to the bi-dimensional one but adapted to 3D space, making the idea more intuitive. The most frequently used system of partial rotations are Euler Angles, rotation matrices whose application order and axes may vary depending on the convention which has been adopted (here the 3-1-3 case has been explained, see **??**). The problem of this method is known as gimbal lock, which implies the loss of a degree of freedom (any possible direction of rotation) when two axes are lined up. This problem resides in the fact that finding a minimal three-variable representation is not possible without ambiguity. Fortunately, to solve this problem other kind of representations have been created. They act in higher spaces but they remove this possibility. The two main methods of this sort are axis/angle representation, which requires knowing the resultant direction of the rotation (by using Euler Rodrigues' formula) and the use of quaternions. Quaternions are an extension of complex numbers to a four-dimensional one by the form $Q = q_0 + iq_1 + jq_2 + kq_3$, and where the representation of the point is made by using the three imaginary units. Hence, the rotation quaternion is defined as follows:

$$Q_r = cos\left(\frac{\pi}{2}\right) + (v_1 i + v_2 j + v_3 k)\, sin\left(\frac{\pi}{2}\right)$$

The main advantage of quaternions over other kind of representations (like Euler angles) is that they offer a solution to the singularity or **gimbal lock** problem, already mentioned. The problem is solved by the rotation in a four-dimensional space around an inertial fixed system, which gives the axis no possibility to alignment (section 4.1.3).

Furthermore the quaternion system is very much compact and because of that it requires less computational capacity than the representations angle/axis or Euler angles, as it is possible to see in the following comparative graphs (see [32]):

Method	Storage
Rotation matrices	9
Quaternions	4
Axis/angle	3

Table 9.1: Comparative graph about the storage requirements.

Method	Multiplies	Add/Subtracts	Total operations
Rotation matrices	27	18	45
Quaternions	16	12	28

Table 9.2: Comparative graph detailing the comparison of rotation chaining operations.

Method	Multiplies	Add/Subtracts	sin/cos	Total operations
Rotation matrices	9	6	0	15
Quaternions	15	15	0	30
Axis/angle	23	16	2	41

Table 9.3: Comparative graph detailing the vector rotating operations.

9.2 The Inverse Kinematics problem

Furthermore, another three dimensions to define the possible translations must be considered in addition to the previous three ones, resulting in an hexa-dimensional space for those elements which can rotate and translate themselves freely in every direction in space. In the field of robotics these elements, are named robotic joints. The concept of kinematic chain as concatenation of joints and links (it may be easier by thinking of it as an human arm) is important to realize that all the elements which take part in the chain must be solved in order to get the end effector (the endpoint) in the desired position. Moreover, a change in a joint affects all the joints after it.

The Inverse Kinematics problem can be solved by geometric or algebraic method if the chain is simple, but for most applications that becomes impossible. Due to that reason, it is necessary to use iterative methods to minimize as much as possible the error between the reached and the desired position when solving this system of several equations with several variables. One of the most used procedures is the Jacobian inversion, but it requires a considerable computational effort.

9.2.1 Numerical methods and other improvements

In order to reduce this computational effort to the minimum, the choice of the optimization algorithm applied on the error function (comprises of several funtions and variables) becomes of vital use for this method. There are some algorithms to find the minimum, among the most important of which are the gradient descent, Newton-Raphson, Gauss-Newton and the Levenberg-Marquardt algorithm. Gradient descent is maybe the most intuitive one because it goes down in the direction of the maximum slope until reaching a local minimum. On the other hand, Newton-Raphson uses successive tangents to approach the root. This method has a faster convergence but the initial estimation must be chosen carefully, because if the initial point is far from the minimum, the method may not converge (a comparison bewteen these algorithms can be seen in figure 7.6).

Finally, Levenberg-Marquardt interpolates between Gauss-Newton method (a Newton-Raphson adaptation but without using second-order derivatives) and gradient descent. When the current estimation is far from a local minimum, it behaves like the gradient descent, and when it is close, like Gauss-Newton. This method is widely considered as the best one, because it is always possible to find a minimum and the convergence is faster and guaranteed.

In addition, another way of improving the algorithms is to work on the problem by using encapsulation of manifolds. In this way, complex state representations (which are not Euclidean) can be approached by finding a neighbourhood of the point where another kind of algebra can be used without distorting the results. This procedure, which is a kind of mapping, is used to solve the problem which appears when the parametrization of the variable has more parameters than the dimension of the measurement space, making it impossible to find an unique result.

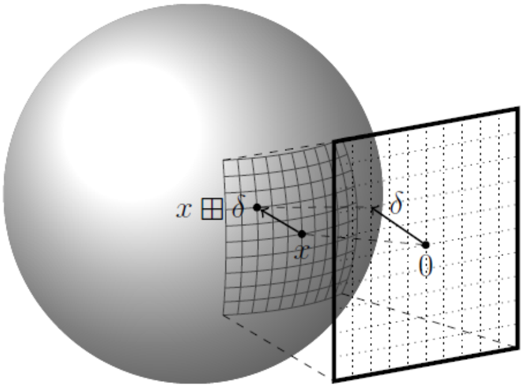


Figure 9.1: Local neighbourhood in the manifold M (here the unit sphere) mapped into $\mathbb{R}^n$ (here $\mathbb{R}^2$, the plane).

Taking into account all the mentioned problems, the best combination possible is to use quaternions globally and the previous kind of mapping when local changes are needed, operating them like in Euclidean spaces.

9.3 Further study fields

As one can infer, Mathematics offers engineering an almost infinite field of possibilities which are not difficult at all, so the possible ways of improving the current methods are only a problem of man's imagination, not an infrastructural one. Following along the same line as some of the studied chapters, probably the most clear path to walk on consists of working on the optimization algorithms, because when speaking about computational calculations, efficiency is as important as getting a good result. In this way, studying the way algorithms work and thinking about new ideas to get them better is a requirement in order to face the future, and Levenberg-Marquart is a very good example of improvement by combining previous methods. There are many possibilities of improving an algorithm: making the code more efficient, parallelizing as much as possible, solving eventual errors, etc.

Another problem which has been treated is that non-Euclidean spaces may appear during local working, and this reason explains how mathematical tools become vital to solve resulting problems. In this line, the field based on topological research makes possible the task of finding a way out when the current system shows weakness. Many problems have been solved so far by using new conceptions of space and reality, as one can realize only by looking at quaternions. In addition, when disposing of many options to choose, the decision can be made reasonably depending on the application, because the system features and the nature of the problem may require particular conditions. Mathematics allows the improvement of every imperfect system, the only thing to do is to firstly imagine how to start this.

To conclude, while working in the computer science field, programming new applications related to Inverse Kinematics may result also interesting. Before starting the analysis which has been developed, the previous idea was intended to find a way of calibrating a robotic arm by using graph factors (see [68]). It would be interesting to go other ways by using optimization algorithms in depth to make the range of possible applications wider. Similarly to that preceding project, many others can be thought of for improving systems or creating new uses for the current ones. As told at the beginning, only imagination can set the limits to human progress.

Acknowledgements

Cuando un camino, esclavo de su definición, llega a su fin, siempre es conveniente echar la vista atrás. La retrospectiva nos ayuda a mejorar, a aprender y a madurar, es la que nos brinda la oportunidad de valorar las cosas. Es también la manera de contemplar el trabajo realizado y distinguir la dedicación de aquellas personas que han contribuido a procurar un éxito que, lejos de ser de uno sólo, es de todos los que han querido que así fuese. Existe la creencia popular de que una persona es el resultado de sus influencias, y en esta afirmación yo no podría estar más de acuerdo. También en el ámbito académico. Es por tanto mi intención, aprovechando estas breves líneas, la de agraceder de corazón a todos aquellos que me han ayudado a llegar hasta aquí.

En primer lugar, al Colegio San José de Villanueva de la Serena, porque ése fue el lugar donde comencé a descubrir la naturaleza de un mundo que ahora me parece mucho más complejo que entonces, el lugar donde más años he pasado en mi vida y el lugar donde conocí a algunos de mis mejores amigos.

Al Instituto San José, en el que tuve la suerte de disfrutar de la mejor combinación de profesores posible y que cualquiera hubiera deseado. A ellos les agradezco haber tratado de inculcarme una de las cosas que más aprecio: la curiosidad. Y la mención es digna porque consiguieron atraer mi interés hacia casi todo lo que se propusieron. Mil gracias por ello, pocas cosas existen para mí más valiosas que ésa.

A la Universidad de Extremadura y a todos los profesores que he tenido, por demostrar que una buena enseñanza puede basarse en una relación cercana y por responder de la mejor manera a la ingente cantidad de preguntas que he formulado durante estos cuatro años. Especial es la alusión a Carmen Ortiz, Pedro Núñez y Pablo Bustos por su implicación en este proyecto, que pese a algunos nervios me ha encantado realizar. Gracias por supervisar mi trabajo y proporcionarme tanto información como respuesta con interés y entusiasmo.

Al sistema de Educación Pública que ha guiado mi formación, que me ha permitido estudiar una carrera ayudado por las becas y contribuido así a mi futuro, al mío y al de tantos otros.

Muy especialmente al Colegio Mayor Francisco de Sande por ser mi casa durante todo este tiempo. Gracias a todas las personas que he conocido allí y que han contribuido a los mejores años de mi vida, gracias por descubrirme la ciudad de Cáceres, por las innumerables conversaciones y por enseñarme todo lo que esta experiencia me ha enseñado. Los amigos son para siempre. Aunque el Colegio cierre, nadie podrá borrarlo jamás de mi recuerdo.

Por último, y de manera particular, gracias a mis padres por apoyarme en todo momento, por ofrecerme todo lo que he necesitado, por confiar en mí y por estar a mi lado siempre compartiendo mis problemas. Su apoyo y sustento han estado y estarán para mí presentes durante toda la vida.

Appendices

Appendix A

Conversions

A.1 2D rotations

A.1.1 From 2×2 rotation matrices to complex numbers

In order to rotate by complex numbers a given a vector $v = (a, b)^T$, the first step to follow is expressing it in Cartesian form, in this way:

$$v = (a, b)^T \longrightarrow v = a + ib \tag{A.1}$$

After that, it must be converted to Polar form, calculating the modulus and argument of the vector in the complex plane:

$$|v| = sqrt(a^2 + b^2) \quad \theta = arctan\left(\frac{b}{a}\right) \tag{A.2}$$

The vector in Polar form is now $v = |v|e^{i\theta}$. Let be a rotation matrix $\begin{pmatrix} cos\phi & -sin\phi \\ sin\phi & cos\phi \end{pmatrix}$, the equivalent expression when rotating with complex numbers is $rot = e^{i\phi}$, so the rotated complex vector in Polar form is $v_2 = rot\ cdotv = |v|e^{i(\phi+\theta)}$.

A.1.2 From complex numbers to 2×2 rotation matrices

This process is the opposite of the previous one. Given a complex vector in Polar form $v = |v|e^{i\theta}$, the way to find the Cartesian equivalence is using the Euler identity: $e^{i\theta} = cos\theta + isin\theta$. Hence:

$$v = |v|(cos\theta + isin\theta) \longrightarrow v = (a, b), \tag{A.3}$$

where a is the real part and b the imaginary one resulting from the product.

The rotation element $rot = e^{i\phi}$ is identical to the 2×2 matrix $\begin{pmatrix} cos\phi & -sin\phi \\ sin\phi & cos\phi \end{pmatrix}$, as it was seen before. After that, the way of acting is the usual one.

A.2 3D rotations

The following methods are valid for a right-hand system. The way to adapt them to a left-hand system will be explained in.

A.2.1 From rotation matrix to Euler angles

The Euler angles can be extracted from the rotation matrix R by inspecting the rotation matrix in analytical form.

Using the **x-convention (3-1-1)**, which have been used in this document, the Euler angles can be deduced as:

$$\Phi = arctan\left(\frac{R_{31}}{R_{32}}\right) \tag{A.4}$$

$$\Theta = arccos R_{33} \tag{A.5}$$

$$\Psi = -arctan\left(\frac{R_{13}}{R_{23}}\right) \tag{A.6}$$

Some considerations must be taken into account:

- There are generally two solutions in $(-\pi, \pi]^3$. The above formula works only when Θ is from the interval $[0, \Pi)^3$.
- In case $R_{33} = 0$, Φ,Ψ shall be derived from R_{11}, R_{12}.
- There are many solutions outside of the interval $(-\pi, \pi]^3$.

A.2.2 From Euler angles to rotation matrix

The rotation matrix R is generated from the Euler angles by multiplying the three matrices built by rotations about the axes. The order depends on the convention used. In case of the **x-convention (3-1-3)**:

$$R(\Phi, \Theta, \Psi) = R(\Psi, z) \cdot R(\Theta, x) \cdot R(\Phi, z) \tag{A.7}$$

$$R(\Phi, \Theta, \Psi) = \begin{pmatrix} cos\Psi cos\Phi - cos\Theta sin\Phi sin\Psi & cos\Psi sin\Phi + cos\Theta cos\Phi cos\Psi & sin\Psi sin\Theta \\ -sin\Psi cos\Phi - cos\Theta sin\Phi cos\Psi & -sin\Psi sin\Phi + cos\Theta cos\Phi cos\Psi & cos\Psi sin\Theta \\ sin\Theta sin\Phi & -sin\Theta cos\Phi & cos\Theta \end{pmatrix} \tag{A.8}$$

A.2.3 From rotation matrix to Euler axis/angle

If the Euler angle θ is not a multiple of π, the Euler axis $\hat{r} = [r_x, r_y, r_z]^T$ and the angle θ can be computed from the elements of the rotation matrix R in the following way:

$$\theta = arccos\left(\frac{1}{2}[R_{11} + R_{22} + R_{33} - 1]\right) \tag{A.9}$$

$$r_x = \frac{R_{32} - R_{23}}{2sin\theta} \quad r_y = \frac{R_{13} - R_{31}}{2sin\theta} \quad r_z = \frac{R_{21} - R_{12}}{2sin\theta} \tag{A.10}$$

A.2.4 From Euler axis/angle to rotation matrix

The matrix for a rotation by an angle θ around an axis in the direction of $\hat{r} = [r_x, r_y, r_z]^T$ is defined as:

$$R = \begin{pmatrix} cos\theta + r_x^2(1 - cos\theta) & r_x r_y(1 - cos\theta) - r_z sin\theta & r_x r_z(1 - cos\theta) + r_y sin\theta \\ r_y r_x(1 - cos\theta) + r_z sin\theta & cos\theta + r_y^2(1 - cos\theta) & r_y r_z(1 - cos\theta) - r_x sin\theta \\ r_z r_x(1 - cos\theta) - r_y sin\theta & r_z r_y(1 - cos\theta) + r_x sin\theta & cos\theta + r_z^2(1 - cos\theta) \end{pmatrix} \tag{A.11}$$

A.2.5 From rotation matrix to quaternions

Given a rotation matrix R, it can be converted to a quaternion $Q = q_0 + iq_1 + jq_2 + kq_3$ in the way that follows:

$$q_0 = \frac{1}{2}\sqrt{1 + R_{11} + R_{22} + R_{33}} \tag{A.12}$$

$$q_1 = \frac{1}{4q_0}(R_{32} - R_{23}) \tag{A.13}$$

$$q_2 = \frac{1}{4q_0}(R_{13} - R_{31}) \tag{A.14}$$

$$q_3 = \frac{1}{4q_0}(R_{21} - R_{12}) \tag{A.15}$$

A.2.6 From quaternions to rotation matrix

On the other hand, the rotation matrix corresponding to the quaternion $Q = q_0 + iq_1 + jq_2 + kq_3$ is given by:

$$R = \begin{pmatrix} q_0^2 + q_1^2 - q_2^2 - q_3^2 & 2q_1q_2 - 2q_0q_3 & 2q_1q_3 + 2q_0q_2 \\ 2q_0q_3 + 2q_1q_2 & q_0^2 - q_1^2 + q_2^2 - q_3^2 & -2q_0q_1 + 2q_2q_3 \\ -2q_0q_2 + 2q_1q_3 & 2q_0q_1 + 2q_2q_3 & q_0^2 - q_1^2 - q_2^2 + q_3^2 \end{pmatrix} \tag{A.16}$$

A.2.7 From Euler angles to quaternions

Considering the **x-convention (3-1)** in Euler angles, the quaternion $Q = q_0 + iq_1 + jq_2 + kq_3$ can be computed from (Φ, Θ, Ψ) in the following way:

$$q_0 = cos\left(\frac{\Phi + Psi}{2}\right) cos\left(\frac{\Theta}{2}\right) \tag{A.17}$$

$$q_1 = cos\left(\frac{\Phi - Psi}{2}\right) sin\left(\frac{\Theta}{2}\right) \tag{A.18}$$

$$q_2 = sin\left(\frac{\Phi - Psi}{2}\right) sin\left(\frac{\Theta}{2}\right) \tag{A.19}$$

$$q_3 = sin\left(\frac{\Phi + Psi}{2}\right) cos\left(\frac{\Theta}{2}\right) \tag{A.20}$$

$$\tag{A.21}$$

A.2.8 From quaternions to Euler angles

In the same way, given the rotation quaternion $Q = q_0 + iq_1 + jq_2 + kq_3$, the **x-convention (3-1-3)** of Euler angles (Φ, Θ, Ψ) van be computed by:

$$\Phi = arctan\left(\frac{q_1q_3 + q_2q_0}{-(q_2q_3 - q_1q_0)}\right) \tag{A.22}$$

$$\Theta = arccos(q_0^2 - q_1^2 - q_2^2 + q_3^2) \tag{A.23}$$

$$\Psi = arctan\left(\frac{q_1q_3 - q_2q_0}{q_2q_3 + q_1q_4}\right) \tag{A.24}$$

A.2.9 From Euler axis/angle to quaternions

Given the Euler axis $\hat{r}$ and the angle θ, the associated quaternion $Q = q_0 + iq_1 + jq_2 + kq_3$ can be calculated as:

$$q_0 = cos\left(\frac{\theta}{2}\right) \quad q_1 = \hat{r}_x sin\left(\frac{\theta}{2}\right) \quad q_2 = \hat{r}_y sin\left(\frac{\theta}{2}\right) \quad q_3 = \hat{r}_z sin\left(\frac{\theta}{2}\right) \tag{A.25}$$

A.2.10 From quaternions to Euler axis/angle

The opposite conversion, from a rotation quaternion $Q = q_0 + iq_1 + jq_2 + kq_3$, to an Euler axis/angle system with axis $\hat{r}$ and angle θ, can be expressed as:

$$\hat{r} = \frac{q}{\|q\|}, \tag{A.26}$$

where $q = iq_1 + jq_2 + kq_3$

$$\theta = 2arccos(q_0) \tag{A.27}$$

Appendix B

Coordinate systems

Since the Euclidean space has no preferred origin or direction, a coordinate system must be created in order to assign numerical values to points and objects in the space.

In the three-dimensional space, the right-handed and the left-handed coordinate systems are used to describe positions. The main properties they have are:

- Any right-handed coordinate system can be converted to any other right-handed coordinate system by rotating it.
- Any left-handed coordinate system can be converted to any other right-handed coordinate system by rotating it.
- A right-handed coordinate system cannot be converted to a left-handed coordinate system by rotating it.
- A right-handed coordinate system can be converted to a left-handed coordinate system by using the method which is explained in B.3.

B.1 Right-handed coordinate system

It is the most used system, and the system which has been applied in this document. In a right-handed coordinate system:

- $+x$ is placed in the **right** direction.
- $+y$ is placed in the **up** direction.
- $+z$ is placed in the **direction from the plane of the page.**

B.2 Left-handed coordinate system

This system is not so used as the previous one, but it is the system adopted for **Robolab (Universidad de Extremadura)**. In a left-handed coordinate system:

- $+x$ is placed in the **left** direction.
- $+y$ is placed in the **up** direction.
- $+z$ is placed in the **direction from the plane of the page.**

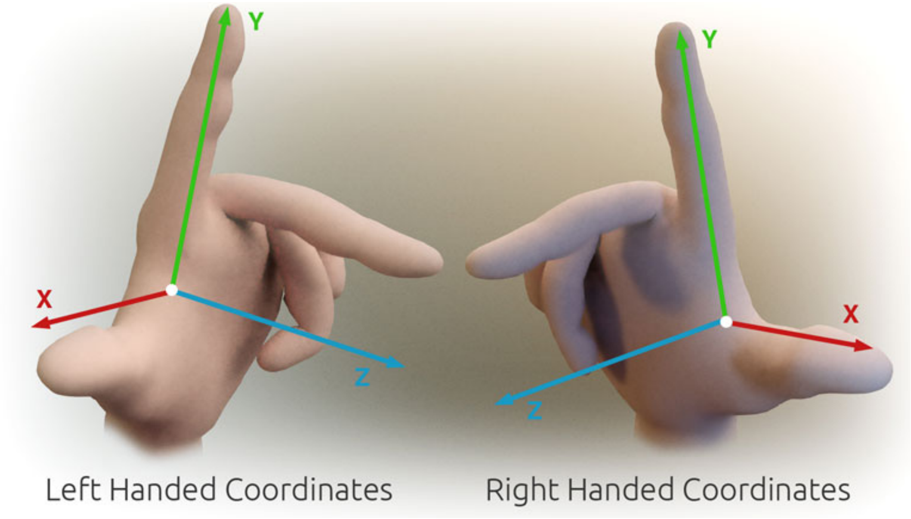


Figure B.1: Comparative between right-handed and left-handed coordinate systems: x-axis in red, y-axis in green and z-axis in blue.

B.3 Conversion from Right-Handedness to Left-Handedness

B.3.1 Conversion of points

Looking at the two systems defined, the only difference between them is the x-axis orientation. As a consequence, only changing the direction of this axis is possible to convert a system in the another one: a point (x, y, z) in the right-handed coordinate system is converted to a point $(-x, y, z)$ in the left-handed system.

In the left-handed coordinate system, the x-component must be negative, so expressing this conversion in matrix-vector form the result is:

$$P_{left} = \begin{pmatrix} -x \\ y \\ z \end{pmatrix} = \begin{pmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} = D_x P_{right}, \tag{B.1}$$

where D_x is defined to be the diagonal matrix $D_x = \begin{pmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$ used when converting from a system to the another one.

B.3.2 Conversion of rotations

In this case, a right-handed point $P_{right} = (x, y, z)$ must be transformed to a left-handed one $P_{left} = (x', y', z')$, where the effect of a rotation is taken into account.

As it has been seen in 4, a rotation over a coordinate axis is written in the following way:

$$P'_{right} = R P_{right}, \tag{B.2}$$

where R is the rotation matrix, whose shape depends on the axis around which the rotation is implemented.

Considering that the conversion of the point after the rotation must be put into practice as before:

$$P'_{left} = D_x P'_{right} \tag{B.3}$$

Then:

$$P'_{left} = D_x P'_{right} = D_x R P_{right} \tag{B.4}$$

In the same way, as the conversion is equivalent in both sides, $P_{right} = D_x P_{left}$.

Hence:

$$P'_{left} = D_x R D_x P_{left} = (D_x R D_x) P_{left} = R' P_{left}, \tag{B.5}$$

where the new rotation matrix is $R' = D_x R D_x$, where R is the rotation matrix in the original system, whose shape depends on the axis around which the rotation is implemented.

Appendix C

MATLAB Code

C.1 Rotation matrices in 2D

```
function final_vector = rotation(vx,vy,theta)

    s=[0:1:40]*2*pi/40;
    x1=cos(s);
    y1=sin(s);

    plot(x1,y1,'r');
    arrow([0 0],[vx vy]);
    xlabel('x-axis');
    ylabel('y-axis');
    axis equal

    vec=[vx;vy];
    rot=[cos(theta) -sin(theta); sin(theta) cos(theta)];
    vecR=rot*vec;

    vx=vecR(1,1);
    vy=vecR(2,1);
    pause;
    figure;
    plot(x1,y1,’r’);
    arrow([0 0],[vx vy]);
    xlabel(’x-axis’);
    ylabel(’y-axis’);
    axis equal

final_vector=[vx,vy];
```

C.2 Homogeneous transformation matrices in 2D

```
function final_vector = rotran(vx, vy, theta, tx, ty)
    vz=1;
    s=[0:1:40]*2*pi/40;
    x1=cos(s);
    y1=sin(s);

    plot(x1,y1,’r’);
    arrow([0 0],[vx vy]);
    xlabel(’x-axis’);
    ylabel(’y-axis’);
    axis equal

    vec=[vx;vy;vz];
    Tran=[cos(theta) -sin(theta) tx; sin(theta) cos(theta) ty; 0 0 1];
    vecR=Tran*vec;

    vx=vecR(1,1);
    vy=vecR(2,1);
    pause;
    figure;
    plot(x1,y1,’r’);
    arrow([0 0],[vx vy]);
    xlabel(’x-axis’);
    ylabel(’y-axis’);
    axis equal

final_vector=[vx,vy];
```

C.3 Rotation matrices in 3D

```
function final_vector = rotation3(vx, vy, vz, psi, theta, phi)

    s=[0:1:40]*2*pi/40;
    x1=cos(s);
    y1=sin(s);
    z1=cos(s);
    zero=[0:1:40]*0;

    hold on
    plot3(x1,y1,zero,’r’);
    plot3(zero,y1,z1,’r’);
    arrow([0 0 0],[vx vy vz]);
    xlabel(’x-axis’);
    ylabel(’y-axis’);
    zlabel(’z-axis’);
    axis equal
    hold off
```

```
%% Psi (rotation around the x-axis)
vec=[vx;vy;vz];
Rot_psi=[cos(psi) -sin(psi) 0; sin(psi) cos(psi) 0; 0 0 1];
vecR=Rot_psi*vec;

vx=vecR(1,1);
vy=vecR(2,1);
vz=vecR(3,1);

pause;
figure;
hold on
plot3(x1,y1,zero,’r’);
plot3(zero,y1,z1,’r’);
arrow([0 0 0],[vx vy vz]);
xlabel(’x-axis’);
ylabel(’y-axis’);
zlabel(’z-axis’);
axis equal
hold off

%% Theta (rotation aroud the y-axis)
vec=[vx;vy;vz];
Rot_theta=[cos(theta) 0 sin(theta); 0 1 0; -sin(theta) 0 cos(theta)];
vecR=Rot_theta*vec;

vx=vecR(1,1);
vy=vecR(2,1);
vz=vecR(3,1);

pause;
figure;
```

```
    hold on
    plot3(x1,y1,zero,’r’);
    plot3(zero,y1,z1,’r’);
    arrow([0 0 0],[vx vy vz]);
    xlabel(’x-axis’);
    ylabel(’y-axis’);
    zlabel(’z-axis’);
    axis equal
    hold off

    %% Phi (rotation around the z-axis)
    vec=[vx;vy;vz];
    Rot_phi=[cos(phi) -sin(phi) 0; sin(phi) cos(phi) 0; 0 0 1];
    vecR=Rot_phi*vec;

    vx=vecR(1,1);
    vy=vecR(2,1);
    vz=vecR(3,1);
    pause;
    figure;
    hold on
    plot3(x1,y1,zero,’r’);
    plot3(zero,y1,z1,’r’);
    arrow([0 0 0],[vx vy vz]);
    xlabel(’x-axis’);
    ylabel(’y-axis’);
    zlabel(’z-axis’);
    axis equal
    hold off

final_vector=[vx,vy,vz];
```

C.4 Homogeneous transformation matrices in 3D

```
function final_vector = rotran3(vx, vy, vz, psi, tht, phi, tx, ty, tz)
  s=[0:1:40]*2*pi/40;
  x1=cos(s);
  y1=sin(s);
  z1=cos(s);
  zero=[0:1:40]*0;

  hold on
  plot3(x1,y1,zero,'r');
  plot3(zero,y1,z1,'r');
  arrow([0 0 0],[vx vy vz]);
  xlabel('x-axis');
  ylabel('y-axis');
  zlabel('z-axis');
  axis equal
  hold off

if   psi =0 then %% Rotation around the x-axis
     vec=[vx;vy;vz;1];
     Tran_psi=[1 0 0 tx; 0 cos(psi) -sin(psi) ty; 0 sin(psi) cos(psi) tz; 0 0 0 1];
     vecR=Tran_psi*vec;

     vx=vecR(1,1);
     vy=vecR(2,1);
     vz=vecR(3,1);

     pause;
     figure;
```

```
    hold on
    plot3(x1,y1,zero,'r');
    plot3(zero,y1,z1,'r');
    arrow([0 0 0],[vx vy vz]);
    xlabel('x-axis');
    ylabel('y-axis');
    zlabel('z-axis');
    axis equal
    hold off
end if

if  tht =0 then %% Rotation around the y-axis
    vec=[vx;vy;vz;1];
    Tran_tht=[cos(tht) 0 sin(tht) tx; 0 1 0 ty; -sin(tht) 0 cos(tht) tz; 0 0 0 1];
    vecR=Tran_tht*vec;

    vx=vecR(1,1);
    vy=vecR(2,1);
    vz=vecR(3,1);

    pause;
    figure;
    hold on
    plot3(x1,y1,zero,'r');
    plot3(zero,y1,z1,'r');
    arrow([0 0 0],[vx vy vz]);
    xlabel('x-axis');
    ylabel('y-axis');
    zlabel('z-axis');
    axis equal
    hold off
end if
```

```
if    phi =0 then %% Rotation around the z-axis
      vec=[vx;vy;vz;1];
      Tran_phi=[cos(phi) -sin(phi) 0 tx; sin(phi) cos(phi) 0 ty; 0 0 1 tz; 0 0 0 1];
      vecR=Tran_phi*vec;

      vx=vecR(1,1);
      vy=vecR(2,1);
      vz=vecR(3,1);

      pause;
      figure;
      hold on
      plot3(x1,y1,zero,'r');
      plot3(zero,y1,z1,'r');
      arrow([0 0 0],[vx vy vz]);
      xlabel('x-axis');
      ylabel('y-axis');
      zlabel('z-axis');
      axis equal
      hold off
end if

final_vector=[vx,vy,vz];
```

Bibliography

[1] B. Ash, Robert *Algebra: Group Fundamentals, p. 1*, Department of Mathematics, University of Illinois, Urbana-Champaign.

[2] University of Syracuse *Self-instructional Mathematics Tutorials, Unit 1: Algebra*, Department of Mathematics, University of Syracuse, USA.

[3] Stanislav, Jabuka *Chapter 1: Topological spaces, p. 1*, Department of Mathematics and Statistics, University of Nevada, Reno, 2009.

[4] Campos González, Neila *El Espacio Euclídeo, p. 1-3*, Departamento de Matemática Aplicada y Ciencias de la Computación, Universidad de Cantabria.

[5] Euclidean Space - Mathematics and Computing *Euclidean space definitions: http://www.euclideanspace.com/maths/geometry/space/euclidean/*, specialized website.

[6] Department of Mathematics *Lecture notes on Linear Algebra: Inner (scalar) products: Euclidean space, p. 3-4*, College of Liberal Arts & Sciences, University of Kansas, 2008.

[7] Melving, George *Linear Algebra, p. 101-102*, Mathematics Department, University of California, Berkeley, USA, 2012.

[8] L. Rueda, Sonia *Matemáticas I: Espacio Euclídeo, p. 2-4*, Departamento de Matemática Aplicada, Escuela Técnica Superior de Arquitectura, Universidad Politécnica de Madrid, 2008.

[9] Sjamaar, Reyer *Manifolds and Differential Forms, p. 1-5*, Department of Mathematics, Cornell University, Ithaca, New York.

[10] Ramírez Galarza, Ana Irene *Geometría Analítica: Una introducción a la geometría, p. 303-304*, UNAM, 2004.

[11] G. Ivancevic, Vladimir; T. Ivancevic, Tijana *Lecture Notes in Lie Groups, p. 18*, Cornell University Library, 2011.

[12] Departamento de Xeometría e Topoloxía *Grupos de Lie, p. 4*, Facultade de Matemáticas, Universidade de Santiago de Compostela.

[13] Rodríguez, Miguel A. *Álgebras de Lie, p. 7*, Departamento de Física Teórica II, Facultad de Ciencias, Universidad Complutense de Madrid, 2007.

[14] Blanco Caraco, José Luis *A tutorial on $SE(3)$ transformation parametrizations and on-manifold optimization, p. 37-41*, Departamento de Ingeniería de Sistemas y Automática, MAPIR Group, Universidad de Málaga, 2010.

[15] Kumar, V. *Rigid Body Motion and the Euclidean Group, p. 1-4*, University of Pennsylvania.

[16] López de Teruel, Pedro E.; Ruiz, A. *Introducción a las Lie Algebras (aplicadas a la visión por computador), s. 4-14*, Artificial Perception and Pattern Recognition Research Group (PARP), Departamento de Ingenería y Tecnología de Computadores y Departamento de Informática y Sistemas, Universidad de Murcia.

[17] Ballan, Luca *Rigid Transformations: The geometry of SO(3) and SE(3), s. 38-43*, Mathematical Foundations of Computer Graphics and Vision, Computer Vision and Geometry Lab, Institute of Visual Computing, Zürich.

[18] Mas Sole, Javier *Física Matemática, p. 122-123*, Universidade de Santiago de Compostela, 2011.

[19] Ballan, Luca *Metrics on SO(3) and Inverse Kinematics, s. 4-5*, Mathematical Foundations of Computer Graphics and Vision, Computer Vision and Geometry Lab, Institute of Visual Computing, Zürich.

[20] van Beveren, Eef *Some notes on group theory, p. 57-58*, Faculdade de Ciências e Tecnologia, Universidade de Coimbra, 1998.

[21] Triola, Cristopher *Special Orthogonal groups and rotations, p. 4*, University of Mary Washington, 1998.

[22] Selig, J.M. *Introductory Robotics, p. 9*, Department of Electrical and Electronic Engineering, South Bank Polytechnic, 1992.

[23] Selig, J.M. *Introductory Robotics, p. 8-9*, Department of Electrical and Electronic Engineering, South Bank Polytechnic, 1992.

[24] Quiroga-Barranco, Raúl *La geometria de dos fórmulas de Euler, Revista Miscelánea Matemática, p. 106-110*, Centro de Investigación en Matemáticas (CIMAT), México, 2007.

[25] Selig, J.M. *Introductory Robotics, p. 10-12*, Department of Electrical and Electronic Engineering, South Bank Polytechnic, 1992.

[26] Sánchez Castaño, Alberto; Sánchez Hernández, José Germán *Manipulación del espacio: Transformaciones y proyecciones, p. 12-14*, Ingeniería en Informática, Facultad de Ciencias, Universidad de Salamanca.

[27] Carreras, Ángel M. *Tópicos en Álgebra Abstracta: Ángulos de Euler, s. 2-4*, University of Georgia, 2009.

[28] Math World *Euler Angles: http://mathworld.wolfram.com/EulerAngles.html*, specialized website.

[29] Deep Mesh *Gimbal Lock: http://www.deepmesh3d.com/help/axis.htm*, specialized website.

[30] Gandullo Ávila, José Alberto *Simocap: Sistema de captura del movimiento, p. 85-86*, Escuela Técnica Superior de Ingeniería Informática, Universidad de Sevilla, 2010.

[31] Perumal, Logah *Quaternion and Its Application in Rotation Using Sets of Regions, p. 40-43*, Multimedia University, Faculty of Engineering and Technology, Melaka (Malaysia), 2011.

[32] 3D Rendering *3D Rendering by Wikipedians*, PediaPress.

[33] Van Verth, Jim *Game Developers Conference: Understanding Quaternions, s. 24-26*, Software Engineering Department, Google, 2013.

[34] T. Mason, Matthew *Lecture 7: Representing Rotation, Mechanics of Manipulation, s. 13*, Robotics Institute, Carnegie Mellon University, Pittsburgh, 2013.

[35] M. Brannon, Rebecca *Rotation: A review of useful theorems involving proper orthogonal matrices referenced to three-dimensional physical space, p. 20-21*, Computational Physics and Mechanics Department, Sandia National Laboratories, Albuquerque, 2002.

[36] de J. Ramírez, Miguel *Automatización de Sistemas de Manufactura, s. 2*, Instituto Tecnológico de Monterrey, México.

[37] Norton, R. L. *Kinematics and Dynamics of Machinery, C.2, p. 32*, McGraw-Hill Companies, 1961.

[38] Norton, R. L. *Kinematics and Dynamics of Machinery, C.2, p. 33-34*, McGraw-Hill Companies, 1961.

[39] Barrientos, Antonio; Peñín, Luis Felipe; Balaguer, Carlos, Aracil, Rafael *Fundamentos de Robótica, p. 95*, Universidad Politécnica de Madrid, McGraw-Hill, 1997.

[40] Cortés Parejo, José *La representación Denavit-Hartenberg, p. 1*, Departamento de Matemática Aplicada I, Universidad de Sevilla, 2008.

[41] Grupo de Investigación de Neuroingeniería biomédica y bioingeniería *Prácticas de Robótica utilizando Matlab: Cinemática de robots*, Universidad Miguel Hernández de Elche.

[42] Moya Pinta, Diego Armando *Modelo y análisis cinemático de un robot manipulador esférico industrial aplicando Matlab, p. 34-35*, Escuela Politécnica Nacional, Facultad de Ingeniería Mecánica, Universidad de Quito, 2010.

[43] Barrientos, Antonio; Peñín, Luis Felipe; Balaguer, Carlos, Aracil, Rafael *Fundamentos de Robótica, p. 97-98*, Universidad Politécnica de Madrid, McGraw-Hill, 1997.

[44] Atrash H., Amin *Lecture 8: Direct Kinematics, s. 7-26*, School of Computer Science, University of Southern California.

[45] Barrientos, Antonio; Peñín, Luis Felipe; Balaguer, Carlos, Aracil, Rafael *Fundamentos de Robótica, p. 99*, Universidad Politécnica de Madrid, McGraw-Hill, 1997.

[46] Íñigo Madrigal, Rafael; Vidal Idiarte, Enric *Robots industriales manipuladores, p. 52-53*, Universidad Politécnica de Catalunya, Edicions UPC, Barcelona, 2002.

[47] San Martín López, José *Tesis Doctoral: Aportaciones al diseño mecánico de los entrenadores basados en realidad virtual*, Departamento de Arquitectura y Tecnología de Computadores, Ciencias de la Computación e Inteligencia Artificial, Universidad Rey Juan Carlos, Madrid, 2007.

[48] Ramírez Benavides, Kryscia Daviana *Robótica: Cinemática Inversa del Robot, s. 6-13*, Escuela de Ciencias de la Computación e Informática, Universidad de Costa Rica.

[49] Iles, Patricio; Ramos, Gabriel *Capítulo II: Locomoción de Sistemas Robóticos, p.30-32*, Universidad Técnica del Norte, Ibarra (Ecuador).

[50] Ruíz González, Julio Douglas Antonio *Robotics, p.65*, Atlantic International University, North Miami, Florida, 2006.

[51] Ramírez Benavides, Kryscia Daviana *Robótica: Cinemática Inversa del Robot, s. 31-38*, Escuela de Ciencias de la Computación e Informática, Universidad de Costa Rica.

[52] Bařinka, Lukáš; Berka, Roman *Inverse Kinematics: Basic Methods, p. 4-5*, Department of Computer Science and Engineering, Czech Technical University, Prague.

[53] Serna Martín, Alberto; Yuste Gallego, Pedro *Caso Práctico IV: Derivación Numérica, p. 6*, Métodos Numéricos de la Ingeniería, Escuela Politécnica, Universidad de Extremadura, 2013.

[54] Abia Vian, José Antonio *Tema 7: Polinomios de Taylor, p. 1*, Departamento de Matemática Aplicada, Escuela de Ingenierías Industriales, Universidad de Valladolid.

[55] Departament de Matemàtica Aplicada II *Tema 9: Polinomio de Taylor en varias variables, p. 2*, Universidad Politècnica de Catalunya, Barcelona.

[56] Wang, Xu *Method of Steepest Descent and its Applications, p.1*, Department of Engineering, University of Tenessee, 2008.

[57] Hauser, Kris *Lecture 4: Gradient Descent, p. 1-2*, Department of Computer Science and Informatics, Indiana University, 2012.

[58] Anstee, Richard *The Newton-Raphson method, p. 1-2*, Department of Computer Science, University of British Columbia.

[59] Lambers, James *Lecture 9 Notes: Newton's Method, p. 1-7*, Department of Mathematics, University of Southern Mississippi, 2012.

[60] Börlin, Niclas *Non-linear Optimization: Least-squares Problems - The Gauss-Newton method, s. 10-14*, Department of Computing Science, Umeå University, 2007.

[61] Zinn-Björkman, Leif *Numerical Optimization using the Levenberg-Marquardt Algorithm, s. 1-6*, Department of Mathematics, University of Utah.

[62] Ranganathan, Ananth *The Levenberg-Marquardt Algorithm, p. 1-3*, Senior Research Scientist, Honda Research Institute, USA.

[63] Brunet, Florent *Contribution to Parametric Image Registration and 3D Surface Reconstruction: Basics on Continuous Optimization*, Thesis for the degree of Doctor of the Université d'Auvergne.

[64] Casarrubias Segura, Fidel; Tamariz Mascarúa, Ángel *Elementos de Topología de Conjuntos, p. 95*, Sociedad Matemática Mexicana, Universidad Nacional Autónoma de México, 2011.

[65] G. Landers, Robert *State Space Representation: Mechanical and Aerospace Control Systems, s.2*, University of Science & Technology, Missouri.

[66] Hertzberg, Christoph *DIPLOMARBEIT in Mathematik und Informatik: A Framework for Sparse, Non-Linear Least Squares Problems on Manifolds, Manifolds, p. 3-8* , Universität Bremen, 2008.

[67] Hertzberg, Christoph; Wagner, René; Frese, Udo; Schröder, Lutz *Integrating Generic Sensor Fusion Algorithms with Sound State Representations through Encapsulations of Manifolds, p. 1-8* , Universität Bremen.

[68] Paoletti Ávila, Mercedes; Yuste Gallego, Pedro *Calibración del robot Loki mediante factores gráficos*, Escuela Politécnica, Universidad de Extremadura, 2014.

[69] Paoletti Ávila, Mercedes; Yuste Gallego, Pedro *Introducción al funcionamiento de los factores gráficos de GTSAM*, Escuela Politécnica, Universidad de Extremadura, 2014.

[70] Paoletti Ávila, Mercedes *Trabajo Fin de Grado: Cinemática Inversa en Robots Sociales*, Escuela Politécnica, Universidad de Extremadura, 2014.