BRNO UNIVERSITY OF TECHNOLOGY
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FACULTY OF MECHANICAL ENGINEERING
FAKULTA STROJNÍHO INŽENÝRSTVÍ

INSTITUTE OF MATHEMATICS
ÚSTAV MATEMATIKY

APPLICATIONS OF QUATERNIONS IN ROBOT KINEMATICS
APLIKACE QUATERNIONŮ V KINEMATICE ROBOTU

MASTER'S THESIS
DIPLOMOVÁ PRÁCE

AUTHOR — **Diana Doctor**
AUTOR PRÁCE

SUPERVISOR — **doc. Ing. Radomil Matoušek, Ph.D.**
VEDOUCÍ PRÁCE

BRNO 2019

BRNO UNIVERSITY OF TECHNOLOGY
FACULTY OF MECHANICAL ENGINEERING

Specification Master's Thesis

Department:	Institute of Mathematics
Student:	**Diana Doctor**
Study programme:	Applied Sciences in Engineering
Study field:	Mathematical Engineering
Supervisor:	**doc. Ing. Radomil Matoušek, Ph.D.**
Academic year:	2018/19

Pursuant to Act no. 111/1998 concerning universities and the BUT study and examination rules, you have been assigned the following topic by the institute director Master's Thesis:

Applications of Quaternions in Robot Kinematics

Concise characteristic of the task:

This thesis deals with the usefulness of the quaternion representation of kinematics for robotics applications. Both the direct and inverse kinematics for a robotic manipulator are derived using quaternions.

Goals Master's Thesis:

The review of 6DOF robots and direct and inverse kinematic models for control design.
State of the art in robot control using quaternion algebra.
Simulation of robot (Universal Robot) and practical programming application.

Recommended bibliography:

SICILIANO, Bruno a KHATIB, Oussama, ed. Springer handbook of robotics. 2nd edition. Berlin: Springer, [2016]. ISBN 978-3-319-32550-7.

Deadline for submission Master's Thesis is given by the Schedule of the Academic year 2018/19

In Brno,

L. S.

prof. RNDr. Josef Šlapal, CSc. Director of the Institute	doc. Ing. Jaroslav Katolický, Ph.D. FME dean

Abstract

This thesis deals with the usefulness of the application of quaternions in representing robot kinematics. It begins by showing the relationship of quaternions to the more commonly-known complex numbers and how it can represent rotations in three-dimensions. Then, the dual quaternions are introduced to represent both the three-dimensional rotation and translation. It will then be used to derive the forward and inverse kinematics, particularly, for the Universal Robot UR3 which is a 6-DOF robotic arm. Lastly, an actual application of dual quaternions in robot programming will be demonstrated.

Keywords

quaternions, dual quaternions, robotics, universal robot, UR3, forward kinematics, inverse kinematics, three-dimensional rotations

DOCTOR, D. B. *Applications of Quaternions in Robot Kinematics*. Brno: Brno University of Technology, Faculty of Mechanical Engineering, 2019. 105p. Diploma thesis supervisor prof. doc. Ing. Radomil Matoušek, Ph.D.

I declare that I have written this diploma thesis *Applications of Quaternions in Robot Kinematics* on my own under the direction of my supervisor, prof. doc. Ing. Radomil Matoušek, Ph.D., and assistance from Ing. Roman Parák, and using the sources listed in the bibliography.

May 31, 2019 **Diana B. Doctor**

I would like to dedicate this thesis, first, to my father, Mateo, who raised me as the person I am today who is full of hopes and dreams and to the people behind the InterMaths programme who gave me multiple chances to fulfill these dreams.

Diana B. Doctor

Contents

Chapter 1

Introduction

The concept of robotics has existed even during the ancient times. However, the term "robot" was conceived only in the twentieth century and originated in the country of Czech Republic. It was first introduced by the Czech playwright Karel Čapek in his play "Rossum's Universal Robots (R.U.R)" which premiered in the city of Prague in the 1920s. However, it was Karel's brother, Josef Čapek, who was really the inventor of the term "robot" and he derived it from the slavic word "robota" which means subordinate or forced labor [2]. The play quickly became influential that the term "robot" was then added to the English language and has since been used in fictions and in scientific researches as well. In the modern era, we refer to robot as a man-made machine that can perform work and mimic actions performed by humans. We have seen various applications of robots which are most commonly in the manufacturing industry. Most advanced application of them would be the sea and space explorations.

Modelling a robot is where robot kinematics comes into play. Formulating the right kinematics of a robot is important in analyzing their behaviour and manipulating them. Robots are composed of joints and links. The two most common types of joints are revolute and prismatic joints. These joints perform rotations and translations, respectively. We can compare the revolute joins with human joints such as joints in the shoulder, elbow, wrists, and knees. A robot is also defined by its degrees of freedom. This corresponds to all the possible movements of its joints. A planar robot is described as having three degrees of freedom. This is because it can only move in a plane. These are left or right and either forward or backward. The other degree of freedom is the rotation that can be performed in a plane. If a robot can also do a movement of up and down which is an added axis of movement then it now has four degrees of freedom. If a robot can also change its pose (position

and orientation) by means of rotating about another axis in the plane then it also adds to its degrees of freedom. These rotations are most commonly known as pitch, yaw, and roll. If a robot can do all of these translations and rotations, then it has the maximum six degrees of freedom.[1]

Robot Kinematics is the study of all the possible motions of a robot without considering the forces that cause these motions as stated in [21] and [12]. Kinematics can be divided into Forward and Inverse Kinematics. Forward Kinematics determines the end-effector pose given the joint angles. This can be easily derived and the solution is always unique. However, its complement, the Inverse Kinematics is generally much more difficult to derive since it determines the joint angles needed for a certain pose. The equations are nonlinear and may have singularities. Apart from this, there may exists also multiple solutions. Our application involves verification of these solutions; however, determining the optimal solution is not in the scope of the study.

Robot Kinematics are mainly described by the transformation done by each of the robot's joints. These transformations are decomposed into rotations and translations. A robotic joint may only perform translation such as prismatic joints which means it doesn't affect the orientation of the end-effector, only its position. There is also a joint that can perform both such as revolute joints. These joints not only change the orientation of the end-effector but may also change its position. An example is if this joint is connected to a link with the end-effector. This link constitutes the translation between this joint frame and the end-effector frame. Thus, rotating the joint will also affect the length of the translation with respect to the base coordinate frame as well as the orientation of the end-effector.

For rotations, there are various ways of representing them. The most common ways are rotation matrices, euler angles, axis-angle representation, and unit quaternions. This study focuses on unit quaternions which gives us an elegant and compact representation of rotation. While a 3x3 rotation matrix needs 9 elements to represent a rotation, quaternions need only 4. However, this representation is still isn't used as much as homogeneous transformations in robotics. This maybe because of the little knowledge we have about it. Quaternions are an extension of complex numbers. Thus, in order to grasp the basic idea of quaternions, we will first need to understand the concept of complex numbers. The purpose of this research is to make it easier to understand the relationship between quaternions and complex numbers and to show how quaternions, most specifically dual quaternions, are able to best represent transformations in three-dimensional space.

[1] From edX Course: "Robotics: Kinematics and Mathematical Foundations" in [1]

Chapter 2

Complex Numbers

Before we begin the study of quaternions, we will need to do some recall on the basic ideas and some important properties of complex numbers which are very beneficial in understanding the concept of quaternions. We will first begin by explaining the reason on why complex numbers were formulated and then we will proceed on describing this number system. Then, we will discuss the important properties of the complex numbers by defining them in the algebraic sense and also in the geometric sense. [1]

2.1 History of Complex Numbers

The real number system $\mathbb{R}$ resulted from the search for a system that included the rational numbers and provides solutions to various polynomial equations. However, there are some polynomial equations such as $x^2 + 2x + 2 = 0$ that cannot be satisfied by any real number x. Using the quadratic formula $\frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$ gives formal expressions for the two solutions for the system $ax^2 + bx + c = 0$; but in the cases of the above equations, this formula involves square roots of negative numbers which were before not defined. By the sixteenth century, the Italian mathematician, Geronimo Cardano, began considering this type of equations and emphasized the need for both negative and "complex numbers". He noticed that if these expressions were to be factored by $\sqrt{-1}$ with the property that $\sqrt{-1} \cdot \sqrt{-1} = -1$, they did indeed satisfy as solutions to these equations. We now use the widely accepted designation $i = \sqrt{-1}$ for these expressions such that $i^2 = -1$.

[1] The contents of this chapter including the images are all referenced from the book *"Basic Complex Analysis" by J.E. Marsden and M.J. Hoffman* [18]

2.2 The Complex Number System

We begin by defining a complex number and explaining the complex number multiplication rule which is necessary in understanding quaternion multiplication.

2.2.1 Definition

The ***system of complex numbers***, denoted by $\mathbb{C}$, is the set $\mathbb{R}^2$ which is the xy-plane consisting of all ordered pairs (x, y) of real numbers together with the usual rules of ***vector addition***,

$$(x_1, y_1) + (x_2, y_2) = (x_1 + x_2, y_1 + y_2) \tag{2.1}$$

and ***scalar multiplication*** by a real number a,

$$a(x, y) = (ax, ay) \tag{2.2}$$

and with the added operation of ***complex multiplication*** defined by,

$$(x_1, y_1)(x_2, y_2) = (x_1x_2 - y_1y_2, x_1y_2 + y_1x_2) \tag{2.3}$$

For the representation of complex numbers, we first identify the real numbers x as points on the real axis x and y as the points on the imaginary axis y where the unit point (0,1) represents i. So now, we can represent a complex number as $(x, y) = x+iy$ which is the more standard notation. Thus, a single symbol such as $z = a + ib$ can also be used to indicate a complex number where $Rez = a$ is called the ***real part*** and $Imz = b$ is called the ***imaginary part***. Therefore, a real number is a complex number z where $Imz = b = 0$. If we have the opposite, $Rez = a = 0$, then we have a ***pure imaginary number***. The notation $z \in \mathbb{C}$ implies that z belongs to the set of complex numbers.

2.2.2 Complex Number Multiplication

To derive the rule for complex number multiplication,

$$(a, b)(c, d) = (ac - bd, ad + bc) \tag{2.4}$$

we use the standard notations for the complex numbers (a, b) and (c, d), and then applying the property that $i^2 = -1$, we get:

$$\begin{aligned}(a + ib)(c + id) &= ac + iad + ibc + i^2bd \\ &= ac + i(ad + bc) + (-1)bd \\ &= ac - bd + i(ad + bc)\end{aligned} \tag{2.5}$$

2.3 Graphical Representation of $z \in \mathbb{C}$

As we have stated in the definition, a complex number may also be thought of as a two-dimensional vector in $\mathbb{R}^2$ from the origin to a point in the xy-plane with coordinates given by the real and the imaginary part of the complex number. See Figure 4.6.

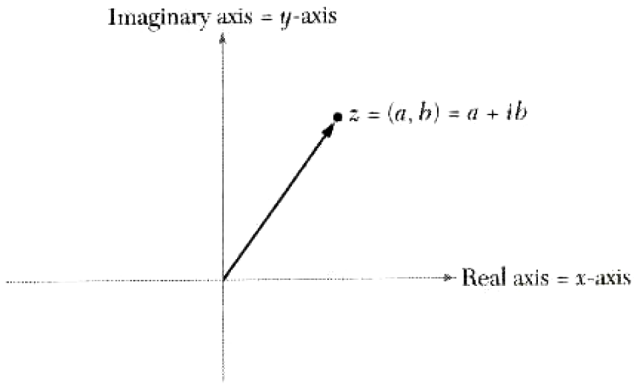


Figure 2.1: Vector representation of $z = a + ib$ with $Rez = a$ and $Imz = b$.

Polar Coordinate and Exponential Representation. We write a complex number in the ***polar coordinate form*** by taking the ***length r*** of the vector $(a, b) = a + ib$ by $r = \sqrt{a^2 + b^2}$. Then, we suppose that the vector makes a positive angle θ of the real axis where $0 \leq \theta < 2\pi$. See Figure 2.2.

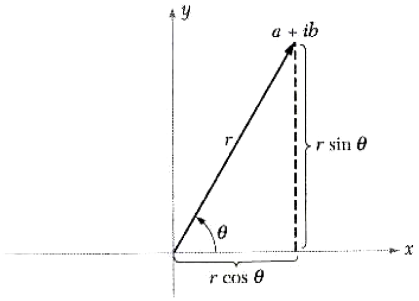


Figure 2.2: Polar coordinate representation of complex number $z = a + ib$ with length r and angle θ.

By *SOHCAHTOA*, we get that $\tan\theta = \frac{b}{a}$ where $a = r\cos\theta$ and $b = r\sin\theta$, therefore we have, [2]

$$z = a + ib = r\cos\theta + i(r\sin\theta) = r(\cos\theta + i\sin\theta) = \mathbf{re^{i\theta}} \tag{2.6}$$

[2]We derived the exponential form using Euler's formula, $e^{i\theta} = \cos\theta + i\sin\theta$.

2.4 Properties of Complex Numbers

In this section, we will present all the basic and important properties of complex numbers necessary for the understanding of the corresponding properties of quaternions.

2.4.1 Norm and Argument of $z \in \mathbb{C}$

In Figure 2.2, the vector representing the complex number $z = a + ib$ has length r which can also be denoted as $|z|$ and is called the ***norm*** of z. The formal computation is given by:

$$|z| = \sqrt{a^2 + b^2} \tag{2.7}$$

The angle θ is called the ***argument*** of z and is denoted by $argz = \theta$. The formal computation is given by:

$$\theta = \tan^{-1} \frac{b}{a} \tag{2.8}$$

2.4.2 Complex Conjugate of $z \in \mathbb{C}$

Given a complex number $z = a + ib$, then $\bar{z}$, which is called the ***complex conjugate*** of z, is defined as $\bar{z} = a - ib$. This can be pictured geometrically as a reflection in the real axis x which is illustrated in Figure 2.3.

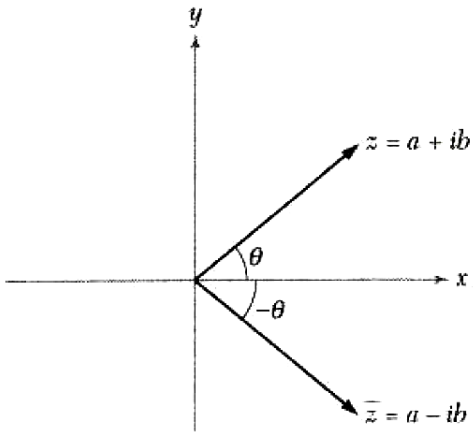


Figure 2.3: Complex conjugation.

Therefore, in polar coordinate form, the complex conjugate can also be represented by using the same magnitude but opposing angle.

2.4.3 Algebraic Rules for Complex Numbers

In this section, we will show that all the usual algebraic rules in manipulating real numbers can also be applied to complex numbers.

Addition Rules

Complex numbers z and w satisfy the following addition rules:

1. $z + w = w + z$
2. $z + (w + s) = (z + w) + s$
3. $z + 0 = z$
4. $z + (-z) = 0$

Multiplication Rules

Complex numbers z and w satisfy the following multiplication rules:

1. $zw = wz$
2. $(zw)s = z(ws)$
3. $1z = z$
4. $z(z^{-1}) = 1$ for $z \neq 0$

   We can solve for the multiplicative inverse $z^{-1} = z' = a' + ib'$ by:

$$a' = \frac{a}{a^2 + b^2} \qquad \text{and} \qquad b' = \frac{-b}{a^2 + b^2} \tag{2.9}$$

   Therefore, for $z = a + ib \neq 0$, we can write:

$$z^{-1} = \frac{a}{a^2 + b^2} - \frac{ib}{a^2 + b^2} \tag{2.10}$$

Distributive Law

Complex numbers z, w, and s satisfy the following distribution rules,

$$z(w + s) = zw + zs \tag{2.11}$$

It is important to note here that same as the real numbers, complex numbers also satisfies the associative, commutative, and distributive properties of addition and multiplication. We will compare this case to the case of quaternion operations which will be discussed in the next chapter.

2.4.4 Addition of Complex Numbers

We can illustrate the addition of complex numbers as the addition of two-dimensional vectors representing the two complex numbers. This operation is illustrated in Figure 2.4.

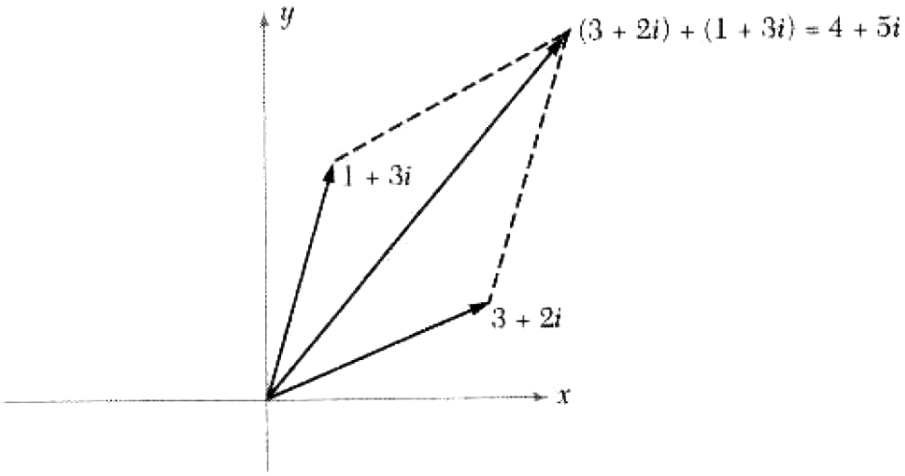


Figure 2.4: Addition of complex numbers $3 + 2i$ and $1 + 3i$.

The same representation can also be used for the subtraction of complex numbers where addition can also be used by adding the first vector to the reflection in the origin of the other vector.

2.4.5 Multiplication of Complex Numbers

Representing a complex number in polar coordinate or exponential form helps to understand the product of two complex numbers in the geometric sense. If we let $z_1 = r_1(\cos\theta_1 + i\sin\theta_1)$ and $z_2 = r_2(\cos\theta_2 + i\sin\theta_2)$, we have,

$$\begin{aligned} z_1 z_2 &= r_1(\cos\theta_1 + i\sin\theta_1) \cdot r_2(\cos\theta_2 + i\sin\theta_2) \\ &= r_1 r_2[(\cos\theta_1 \cdot \cos\theta_2 - \sin\theta_1 \cdot \sin\theta_2) + i(\cos\theta_1 \cdot \sin\theta_2 + \cos\theta_2 \cdot \sin\theta_1)] \\ &= r_1 r_2[\cos(\theta_1 + \theta_2) + i\sin(\theta_1 + \theta_2)] = \mathbf{r_1 r_2 e^{i(\theta_1 + \theta_2)}} \end{aligned} \tag{2.12}$$

Note that we applied the following trigonometric formulas for the sum and difference of the products of the cosines and sines of the angles:

$$\cos(\theta_1 + \theta_2) = \cos\theta_1 \cos\theta_2 - \sin\theta_1 \sin\theta_2 \tag{2.13}$$

$$\sin(\theta_1 + \theta_2) = cos\theta_1 \sin\theta_2 + \cos\theta_2 \sin\theta_1 \tag{2.14}$$

Proposition. For any complex number z_1 and z_2, we propose that,

$$|z_1 z_2| = |z_1| \cdot |z_2| \quad \text{and} \quad arg(z_1 z_2) = arg z_1 + arg z_2 (mod 2\pi) \tag{2.15}$$

This means that the product of two complex numbers is a complex number that has a length equal to the products of the lengths of the two complex numbers that were multiplied and has an argument equal to the sum of the arguments of those two complex numbers. This product can be illustrated in Figure 2.5 and an example in 2.6.

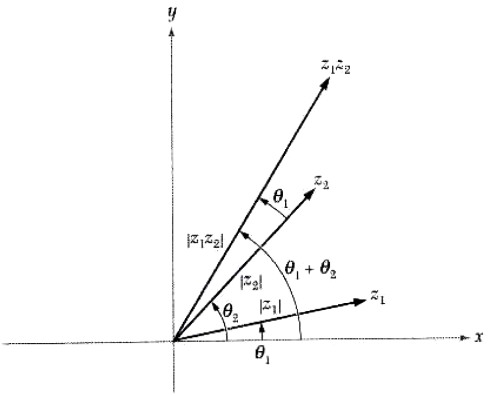


Figure 2.5: Multiplication of complex numbers z_1 and z_2.

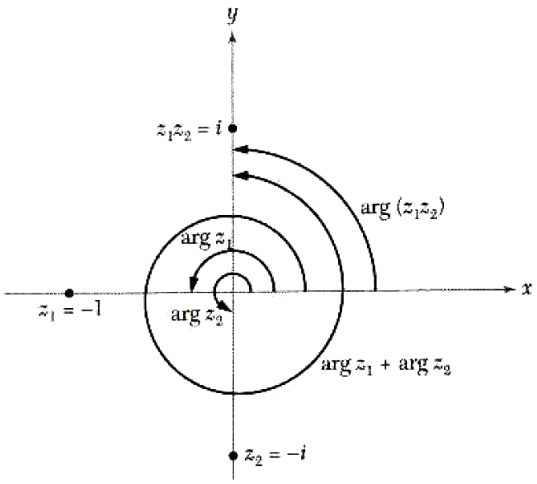


Figure 2.6: Multiplication of complex numbers -1 and $-i$.

2.5 Rotations in the Complex Plane

The multiplication of complex numbers can also be analyzed by a mapping ψ_z that represents the multiplication by z such that $\psi_z : \mathbb{C} \to \mathbb{C}$ and is defined by $\psi_z(w) = wz$. By the proposition, the effect of this map is to ***rotate a complex number w through an angle which is equal to arg z in the counterclockwise direction and to stretch its length by the factor*** $|z|$. This can be seen in Figure 2.7 and also in the previous example in Figure 2.6.

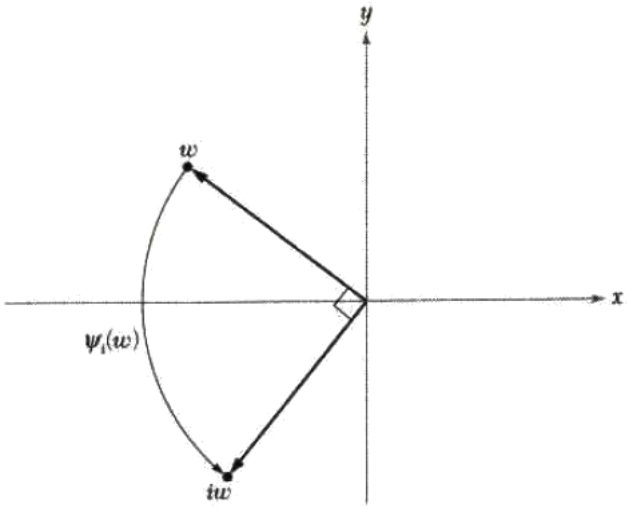


Figure 2.7: Multiplication of wz where $z = i$.

In the next chapter, we will begin with the discussion of quaternions and how its multiplication can also represent rotations in three-dimensions.

Chapter 3

Quaternions

This chapter deals with the main discussion of quaternions and its basic properties. We begin by detailing the dramatic discovery of quaternions and how it has shaken the world of mathematics during its time. Then, we will proceed with the definition and the important properties of quaternions which will give us basic understanding of this number system. It is also important to show how quaternions can represent rotations in three-dimensions and how it relates to other rotation representations. Lastly, we will be discussing about the dual quaternions and how it is used in robotics.

3.1 History of Quaternions

As we have seen in the previous chapter, complex numbers models two-dimensional space by having a real and imaginary part that corresponds to the xy-plane in which multiplication and addition of these numbers still occur in the same space. For years, Irish mathematician Willian Rowan Hamilton had been trying to model three-dimensional space that has this same structure. He was trying to represent it with a real and two-dimensional imaginary part but it didn't satisfy the structure that he wanted to achieve. His son, William Edward Hamilton, would often ask him about his progress in the multiplication of triplets to which he would reply that he can only add or subtract them but never multiply them. [24]

The most famous act of mathematical vandalism. The breakthrough came on the 16th day of October 1843 while Hamilton was walking along the Brougham Bridge (now Broom Bridge) in Dublin. He had a flash of insight and couldn't help but carve into the bridge the following equations:

$$i^2 = j^2 = k^2 = ijk = -1 \tag{3.1}$$

In his journal [10], Hamilton detailed this moment with the following writing:

"On the 16th day of October - which happened to be a Monday, and a Council day of the Royal Irish Academy - I was walking in to attend a preside, and your mother was walking with me, along the Royal Canal, to which she had perhaps driven; and although she talked with me now and then, yet an under-current of thought was going on in my mind, which gave at last a result, whereof it is not too much to say that I felt at once the importance. An electric circuit seemed to close; and a spark flashed forth, the herald (as I foresaw, immediately) of many long years to come of definitely direted thought and work, by myself if spared, and at all events on the part of others, if I should even be allowed to live long enough distinctly to communicate the discovery. Nor could I resist the impluse - unphilosophical as it may have been - to cut with a knife on a stone of Brougham Bridge, as we passed it, the fundamental formula with the symbols i, j, k***; namely,*** $i^2 = j^2 = k^2 = ijk = -1$ ***which contains the Solution of the Problem."***

His discovery of these multiplication laws was also a defining moment in the history of algebra. In this moment, he realized that a fourth dimension was needed and he later coined the term ***quaternion*** to represent the real space spanned by the elements 1, i, j, and k subject to the above multiplication laws.
His carvings have worn away in time but a plaque was put into the bridge to replace it and to commemorate the discovery in mathematical history.

Figure 3.1: A portrait of Sir William Rowan Hamilton[1]

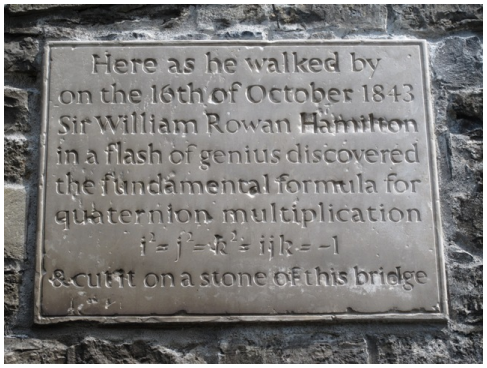


Figure 3.2: Plaque commemorating the discovery of quaternions.[2]

[12]According to the exact words of Hamilton [10],

"A Quaternion may be symbolically defined to be a Quadrinomial Expression of the form, q = w +ix + jy + kz in which w, x, y, z are four scalars or ordinary algebraic quantities, while i, j, and k are three new symbols obeying all laws contained in the above formula and therefore not subject to all the usual rules of algebra."

Hamilton defined the quaternions as a quotient of two vectors. Quaternions being a system of numbers that do not satisfy the usual commutative rule of multiplication inevitably met with some resistance in the mathematical world when it was first discovered and explained by Hamilton. The rivalry between those who prefer quaternions and those who prefer vectors as physical notations flared into a war between those who sided with quaternions and those who sided with the vector notation with some even referring to quaternions as unmixed evil.

With the introduction of quaternions, it opened new algebraic possibilities such as the rediscovery of octaves or octonions, an algebra in eigth dimensions, of John T. Graves done by successfully generalizing quaternions. Other well-known result of quaternions is its generalizations to higher dimensions by William Clifford who developed a way to build algebras from quadratic forms so the Hamilton quaternions arise as this algebra which is more known as the Clifford Algebra.

From the mid-1880s, quaternions began to be displaced by vector analysis which described the same phenomena as quaternions but conceptually simpler and notationally cleaner. However, quaternions had a revival since the late 20th century due to its usefulness in describing spatial rotations. This is because quaternions give an elegant way to describe and compute rotations which is compact and computationally more efficient and avoids *Gimbal Lock* which occurs when a degree of freedom is lost when certain rotations are applied. Euler Angles, in particular, is susceptible to this mechanical problem.

For these reasons, quaternions are still now being used in computer graphics, computer vision, robotics, control theory, signal processing, attitude control, quantum mechanics and many other fields that involves working in three-dimensional space.

[1]https://en.wikipedia.org/wiki/William_Rowan_Hamilton

[2]http://www.irelandtravelkit.com/broome-bridge-and-sir-hamiltons-eureka-moment-cabra-co-dublin/broombridge-plaque-bdolan-2010-800-label/

3.2 The Quaternion Number System

We begin with the definition of quaternion and its expanded multiplication laws as also stated in [13].

3.2.1 Definition

The quaternions is the set defined as,

$$\mathbb{H} = \{w + ix + jy + kz \mid w, x, y, z \in \mathbb{R}\} \tag{3.2}$$

which is the set consisting all the numbers that can be in the form,

$$\mathbf{q} = w + ix + jy + kz \tag{3.3}$$

where w, x, y, and z are real numbers and i, j, and k are what we call the ***fundamental quaternion units*** which represent the three imaginary dimensions.

3.2.2 Quaternion Representations

Standard Quaternional Form. A quaternion is generally represented using its standard quaternional form written as,

$$\mathbf{q} = q_0 + q_1 i + q_2 j + q_3 k \tag{3.4}$$

in which all the variables correspond to the same interpretations as in the definition.

Scalar and Vector Form. Aside from the standard form, a quaternion can also be described as consisting of a scalar part and a vector part where in the above case, the quaternion part, $q_1 i + q_2 j + q_3 k$, represents the vector part and q_0 the scalar part. This can be represented in the so-called **scalar-vector form** written as,

$$\mathbf{q} = (q_0, \vec{q}) \tag{3.5}$$

where q_0 is called as the *scalar part* and the vector $\vec{q}$ represents the three-dimensional vector formed by the imaginary part of the quaternion and is called the *vector part*.

A quaternion with no real part is called a ***pure quaternion*** or ***vector quaternion*** and a quaternion with no imaginary parts is then equivalent to its corresponding real number which implies that the real and complex numbers are a subset of the quaternions. ($\mathbb{R} \subset \mathbb{C} \subset \mathbb{H}$))

3.2.3 Quaternion Multiplication Laws

These quaternion units i, j, and k, as stated before, satisfy the following multiplication rule:

$$i^2 = j^2 = k^2 = -1 \tag{3.6}$$

which is expanded to the following multiplication laws:

$$ij = k = -ji \tag{3.7}$$

$$jk = i = -kj \tag{3.8}$$

$$ki = j = -ik \tag{3.9}$$

These laws are summarized in Table 4.1.

$\times$	1	i	j	k
1	1	i	j	k
i	i	-1	k	$-j$
j	j	$-k$	-1	i
k	k	j	$-i$	-1

Table 3.1: Quaternion Multiplication Table

These can also be visualized in the diagram shown in Figure 3.3.[3]

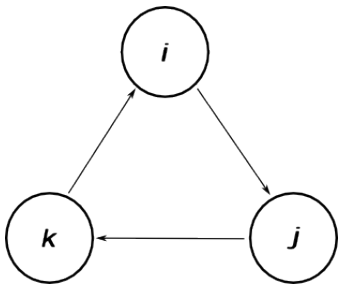


Figure 3.3: Quaternion Multiplication Diagram

As we can see from the table and the diagram above, the multiplication of the quaternion units are not commutative ($ij \neq ji$). If the multiplication of these units go against the flow in the diagram, the product will be the negative of the corresponding product in the normal flow. This will be understood more clearly once we have discussed the representation of rotations by quaternions.

[3]http://hforsten.com/quaternion-roots-of-modern-vector-calculus.html

3.3 Properties of Quaternions

In this section, we will present all the basic and important properties of a quaternion given as $\mathbf{q} = q_0 + q_1 i + q_2 j + q_3 k$ as also stated in [17].

3.3.1 Quaternion Conjugate

The quaternion conjugate of $\mathbf{q}$ is given by negating its vector part shown as,

$$\mathbf{q}^* = q_0 - q_1 i - q_2 j - q_3 k = (q_0, -\vec{q}) \tag{3.10}$$

3.3.2 Norm of a Quaternion

The quaternion norm also called the magnitude is the length of the quaternion from the origin. It is defined by:

$$||\mathbf{q}|| = \sqrt{\mathbf{q}\mathbf{q}^*} = \sqrt{q_0^2 + q_1^2 + q_2^2 + q_3^2} \tag{3.11}$$

3.3.3 Unit Quaternion

A quaternion is called a ***unit quaternion*** if it has a norm of 1, meaning,

$$||\mathbf{q}|| = \sqrt{q_0^2 + q_1^2 + q_2^2 + q_3^2} = 1 \tag{3.12}$$

Normalization . Any quaternion can be normalized to transform it to a unit quaternion. This is done by dividing the quaternion by its magnitude or norm, such as:

$$\hat{\mathbf{q}} = \frac{\mathbf{q}}{||\mathbf{q}||} = \frac{q_0 + q_1 i + q_2 j + q_3 k}{\sqrt{q_0^2 + q_1^2 + q_2^2 + q_3^2}} \tag{3.13}$$

3.3.4 Inverse of a Quaternion

The inverse (reciprocal) of a quaternion can be computed by,

$$\mathbf{q^{-1}} = \frac{\mathbf{q}^*}{||\mathbf{q}||^2} = \frac{q_0 - q_1 i - q_2 j - q_3 k}{q_0^2 + q_1^2 + q_2^2 + q_3^2} \tag{3.14}$$

This can be verified using the definition of the norm, ($||q||^2 = qq^*$),

$$\mathbf{q}\mathbf{q^{-1}} = \mathbf{q}\left(\frac{\mathbf{q}^*}{||\mathbf{q}||^2}\right) = \mathbf{q}\left(\frac{\mathbf{q}^*}{\mathbf{q}\mathbf{q}^*}\right) = 1 \tag{3.15}$$

3.3.5 Addition and Subtraction of Quaternions

Addition of quaternions follows the same rule as in the complex numbers which is done component-wise. If we consider the given quaternion $\mathbf{q}$ and another quaternion $\mathbf{p}$ given as, [4]

$$\mathbf{p} = p_0 + p_1 i + p_2 j + p_3 k \tag{3.16}$$

Then their sum is given by,

$$\mathbf{p} + \mathbf{q} = (p_0 + q_0) + (p_1 + q_1)i + (p_2 + q_2)j + (p_3 + q_3)k \tag{3.17}$$

The same rule is also applied to the subtraction of quaternions such that,

$$\mathbf{p} - \mathbf{q} = (p_0 - q_0) + (p_1 - q_1)i + (p_2 - q_2)j + (p_3 - q_3)k \tag{3.18}$$

3.3.6 Multiplication and Division of Quaternions

The product of two quaternions has to satisfy the fundamental laws that was stated at the beginning along with its non-commutativity property for multiplication. Thus, the product of quaternions $\mathbf{p}$ and $\mathbf{q}$ is given by,

$$\begin{aligned}
\mathbf{pq} &= (p_0 + p_1 i + p_2 j + p_3 k)(q_0 + q_1 i + q_2 j + q_3 k) \qquad (3.19)\\
&= p_0 q_0 + (p_0 q_1 + p_1 q_0)i + (p_0 q_2 + p_2 q_0)j + (p_0 q_3 + p_3 q_0)k \\
&\quad + (p_1 q_2)ij + (p_2 q_1)ji + (p_2 q_3)jk + (p_3 q_2)kj + (p_3 q_1)ki + (p_1 q_3)ik \\
&\quad + (p_1 q_1)i^2 + (p_2 q_2)j^2 + (p_3 q_3)k^2 \\
&= p_0 q_0 + (p_0 q_1 + p_1 q_0)i + (p_0 q_2 + p_2 q_0)j + (p_0 q_3 + p_3 q_0)k \\
&\quad + (p_1 q_2)(k) + (p_2 q_1)(-k) + (p_2 q_3)(i) + (p_3 q_2)(-i) + (p_3 q_1)(j) + (p_1 q_3)(-j) \\
&\quad + (p_1 q_1)(-1) + (p_2 q_2)(-1) + (p_3 q_3)(-1) \\
&= p_0 q_0 - p_1 q_1 - p_2 q_2 - p_3 q_3 + (p_0 q_1 + p_1 q_0 + p_2 q_3 - p_3 q_2)i \\
&\quad + (p_0 q_2 + p_2 q_0 + p_3 q_1 - p_1 q_3)j + (p_0 q_3 + p_3 q_0 + p_1 q_2 - p_2 q_1)k
\end{aligned}$$

This is a complicated way of taking the product of two quaternions. There is a much simpler way which involves using the scalar and vector form of a quaternion and applying the dot and cross product for vectors. This is done by the following,

$$\mathbf{pq} = (p_0, \vec{p})(q_0, \vec{q}) = p_0 q_0 - \vec{p} \cdot \vec{q} + p_0 \vec{q} + q_0 \vec{p} + \vec{p} \times \vec{q} \tag{3.20}$$

For the division between quaternions, this can be done by taking the multiplicative inverse of the divisor such that,

$$\frac{\mathbf{p}}{\mathbf{q}} = \mathbf{pq^{-1}} \tag{3.21}$$

and applying the above method for the multiplication of two quaternions.

[4] This Addition and Multiplication of Quaternions is referenced from [6]

3.3.7 Summarized Properties of Quaternions

Listed here are the summarized properties of quaternions.

Addition Properties

Quaternions $\mathbf{q}$, $\mathbf{p}$ and $\mathbf{r}$ satisfy the following addition properties:

- Closure: $\mathbf{p} + \mathbf{q} \in \mathbb{H}$
- Associativity: $\mathbf{p} + (\mathbf{q} + \mathbf{r}) = (\mathbf{p} + \mathbf{q}) + \mathbf{r}$
- Commutativity: $\mathbf{p} + \mathbf{q} = \mathbf{q} + \mathbf{p}$
- Identity: $\mathbf{p} + 0 = 0 + \mathbf{p} = \mathbf{p}$
- Sum: $\mathbf{p} + \mathbf{q} = (p_0 + q_0) + (p_1 + q_1)i + (p_2 + q_2)j + (p_3 + q_3)k$
- Difference: $\mathbf{p} - \mathbf{q} = (p_0 - q_0) + (p_1 - q_1)i + (p_2 - q_2)j + (p_3 - q_3)k$

Multiplication Properties

Quaternions $\mathbf{q}$, $\mathbf{p}$ and $\mathbf{r}$ satisfy the following multiplication properties:

- Closure: $\mathbf{pq} \in \mathbb{H}$
- Associativity: $(\mathbf{pq})\mathbf{r} = \mathbf{p}(\mathbf{qr})$
- Non-commutativity: $\mathbf{pq} \neq \mathbf{qp}$
- Distributivity: $\mathbf{p}(\mathbf{q} + \mathbf{r}) = \mathbf{pq} + \mathbf{pr}$
- Inverse: If $\mathbf{q} \neq 0$ then $\exists \mathbf{q^{-1}} \in \mathbb{H}$ such that $\mathbf{qq^{-1}} = \mathbf{q^{-1}q} = 1$
- Product: $\mathbf{pq} = p_0 q_0 - \vec{p} \cdot \vec{q} + p_0 \vec{q} + q_0 \vec{p} + \vec{p} \times \vec{q}$
- Division: $\mathbf{p/q} = \mathbf{pq^{-1}}$

Conjugate: $\mathbf{q}^* = q_0 - q_1 i - q_2 j - q_3 k$

Norm: $||\mathbf{q}|| = \sqrt{q_0^2 + q_1^2 + q_3^2 + q_4^2}$

Unit Quaternion: $||\mathbf{q}|| = 1$ and $\forall \mathbf{p} \neq 0$, $\frac{\mathbf{p}}{||p||}$ is a *unit quaternion*

Inverse: $\mathbf{q^{-1}} = \frac{\mathbf{q}^*}{||\mathbf{q}||^2}$, then for a *unit quaternion* $\mathbf{u}$, $\mathbf{u^{-1}} = \mathbf{u}^*$

Dot Product: $\mathbf{q} \cdot \mathbf{p} = q_0 p_0 + q_1 p_1 + q_2 p_2 + q_3 p_3$

3.4 3D Rotations Using Quaternions

In this section, we will show how unit quaternions are able to represent rotations in three-dimensions and its relationship with other rotation representations. [5]

3.4.1 3D Rotation Representations

The concept of expressing a rotation in three dimensions as a mathematical formulation extends to classical mechanics where rotational kinematics is the science of describing the purely rotational motion of an object with numbers.

According to Euler's rotation theorem, a general displacement of a rigid body with one point fixed is described by a rotation about an axis. Moreover, this rotation can be uniquely described by a minimum of three parameters. There are various ways to represent 3D rotations with some needing more than just three parameters even if all of them only has three degrees of freedom. The most used rotation representations are detailed here and are each discussed.

Rotation Matrix Representation

We know that two-dimensional rotations can be represented by the matrix,

$$R = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \tag{3.22}$$

where θ is the angle of rotation about the origin. Having a single angle parameter also implies that plane rotations only have one degree of freedom.

In three-dimensions, we are dealing with rotations on the principal axes x, y, and z. Rotating about the x-axis corresponds to a rotation in the yz-plane. On the other hand, rotating about the y-axis corresponds to a rotation in the xz-plane and rotating about the z-axis corresponds to a rotation in the xy-plane.

These rotations are represented by the following matrices,

$$R_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix} \tag{3.23}$$

[5] This subsection is a compilation of contents from the following sources: [5][6][8][9][17]

$$R_y = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix} \tag{3.24}$$

$$R_z = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.25}$$

In general, if we consider a rigid body with three orthogonal unit vectors fixed to it (also referred to as basis). Specifying the coordinates of this basis in its current rotated position in terms of the reference non-rotated coordinate axes will describe the rotation applied to it. These three unit vectors $\mathbf{r_1}, \mathbf{r_2}$ and $\mathbf{r_3}$, consisting of 3 coordinates each, yields a total of 9 parameters. These parameters form a 3x3 matrix which is referred to as the rotation matrix.

The elements of a rotation matrix are defined as follows:

$$R = \begin{bmatrix} \mathbf{r_1} & \mathbf{r_2} & \mathbf{r_3} \end{bmatrix} \tag{3.26}$$

$$= \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \tag{3.27}$$

A rotation matrix has the following special properties:

1. The columns of a rotation matrix are orthogonal unit vectors.
2. The rows of a rotation matrix are also orthogonal unit vectors.
3. The transpose of a rotation matrix is its inverse, $R^T = R^{-1}$. Thus, $RR^T = R^T R = I$.
4. The determinant of a rotation matrix is $+1$.

A general three-dimensional rotation is achieved by combining these rotation matrices into a single matrix by performing matrix multiplication. Since matrix multiplication are not commutative, it is important to note the order of rotations such as,

$$R_x R_y R_z \neq R_z R_y R_x \tag{3.28}$$

This implies that rotating first about the x-axis and then the y-axis and then z-axis does not produce the same rotation as rotating first about the z-axis and then the y-axis and then x-axis. This property agrees on how rotations are also not commutative. The final matrix will have three angle parameters from each rotation matrix which corresponds to having three degrees of freedom (these angle parameters are often referred to as the Euler Angles $[\phi, \theta, \psi]$).

Performing a matrix-vector multiplication between the rotation matrix and a vector in space results to a rotation of that vector in the space while having its length preserved. This can be expressed as:

$$\mathbf{z}' = R\mathbf{z} \tag{3.29}$$
$$\begin{bmatrix} z_1' \\ z_2' \\ z_3' \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \begin{bmatrix} z_1 \\ z_2 \\ z_3 \end{bmatrix}$$

where $\mathbf{z}'$ is the vector $\mathbf{z}$ rotated by the rotation matrix R.

The ability to combine several rotations into a single matrix and the ease of applying the rotation to a vector makes the rotation matrix a useful and popular way to represent rotations even if it is less precise than other representations.

Euler Angles Representation

Euler Angles is historically the most popular rotation representation where a general rotation is described as a sequence of rotations about three orthogonal coordinate axes namely the x, y, and z axes. The idea behind Euler Angles is to split a rotation into three simpler rotations $[\phi, \theta, \psi]$ corresponding to rotations on the principal axes and are commonly referred to as ***pitch, yaw,*** and ***roll***.

The definition of Euler Angles is not unique and the conventions depend on the frame of reference, axes and the sequence on how the rotations are being applied. Therefore, for a *global frame of reference*, a rotation of ϕ in the x-axis followed by a rotation of θ in the fixed y-axis, and then a rotation of ψ in the fixed z-axis can be represented in the rotation matrix form by pre-multiplying the succeeding rotations (for *local frame of reference*, post-multiplying is done). This can be expressed as:

$$\begin{aligned} R_{xyz} &= R_z(\psi)R_y(\theta)R_x(\phi) \\ &= \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi & \cos\phi \end{bmatrix} \end{aligned} \tag{3.30}$$

Euler Angles became the most common way to represent the rotation of a rigid body in three-dimensional space since it is the easiest to use and understand. However, the main disadvantages of Euler angles such as being susceptible to gimbal lock and low accuracy in integrating incremental changes overtime, led researchers to use other rotation representations such as unit quaternions as a way of effectively representing rotations.

Axis-Angle Representation

Euler's rotation theorem implies that any rotation can be expressed as a single rotation about some axis. Thus, a more generalized expression of a rotation is by the axis-angle representation which is expressed simply as,

$$rot(\mathbf{n}, \theta) \text{ , where } \mathbf{n} = \begin{bmatrix} n_x & n_y & n_z \end{bmatrix}^T \text{ and } \theta \in \mathbb{R} \tag{3.31}$$

The axis of rotation $\mathbf{n}$ is defined as any unit vector in three-dimensional space which is not changed by the rotation, and the angle θ represents the amount of rotation about this axis which follows the right hand rule to determine the positive direction of the rotation.

To rotate a vector $\mathbf{z}$ to $\mathbf{z}'$ using the axis-angle representation, we apply the *Rodrigues' Rotation Formula* defined as,

$$\mathbf{z}' = \mathbf{z}\cos\theta + \mathbf{n}(\mathbf{n}\cdot\mathbf{z})(1-\cos\theta) + (\mathbf{z}\times\mathbf{n})\sin\theta \tag{3.32}$$

Note that if the axis is not one of the principal axes then it is usually hard to identify the axis of rotation. Moreover, combining two successive rotations represented by an axis-angle is not straightforward and it is better to employ rotation matrix or quaternion multiplication to get the final rotation and then convert it to the axis-angle representation.

Unit Quaternion Representation

A unit quaternion $\mathbf{q} = q_0 + q_1 i + q_2 j + q_3 k$ is defined as a quaternion with a unity norm such that,

$$||\mathbf{q}|| = \sqrt{q_0^2 + q_1^2 + q_2^2 + q_3^2} = 1 \tag{3.33}$$

Representing a rotation by a unit quaternion is generally done by converting from other rotation representations into quaternions and normalizing it. For an axis-angle representation $rot(\mathbf{n}, \theta)$ where the axis of rotation is given by a unit vector $\mathbf{n}$ and angle of rotation given by θ, its equivalent quaternion representation is formulated as,

$$\mathbf{q} = \left[\cos\left(\frac{\theta}{2}\right), \sin\left(\frac{\theta}{2}\right)\mathbf{n}\right] = \cos\left(\frac{\theta}{2}\right) + \sin\left(\frac{\theta}{2}\right)[n_x i + n_y j + n_z k] \tag{3.34}$$

It is important to note that $\mathbf{q}$ and $-\mathbf{q}$ is equivalent to the same rotation even if their axis-angle representations are different. This is due to the fact that rotating by an angle θ around an axis $\mathbf{n}$ is the same as rotating by an angle of $2\pi - \theta$ around the axis $-\mathbf{n}$.

For sequences of rotations that are represented by multiple quaternions, deriving the final quaternion rotation is just as simple as with rotation matrices. If we are given a sequence of quaternion rotations $\mathbf{q_1}, \mathbf{q_2}, ..., \mathbf{q_n}$, the final rotation $\mathbf{q_F}$ is computed by simply getting the product of the quaternions noting the order of rotation and type of reference frame.

Thus, for a local reference frame, we have,

$$\mathbf{q_F} = \mathbf{q_1 q_2 ... q_n} \tag{3.35}$$

For rotating a three-dimensional vector $\mathbf{p}$ to $\mathbf{p'}$ by $\mathbf{q}$, the initial vector $\mathbf{p}$ is first represented as a *pure quaternion* having only its vector part as the imaginary part and zero real part. In order to get $\mathbf{p'}$, the following is applied to the initial vector $\mathbf{p}$:

$$\mathbf{p'} = \mathbf{qpq^{-1}} \text{ , where } \mathbf{q^{-1}} \text{ is the inverse of } \mathbf{q} \tag{3.36}$$

As an example, we have the following rotation illustrated in Figure 3.4 where we want to rotate the vector representing the point (2,1,1) along the z-axis by 90°, $\mathbf{rot}(\mathbf{z}, 90°)$. The rotated vector is then determined by using the formula above and applying the multiplication rules of the quaternion units i, j, and k. This will yield to another pure quaternion which then represents the rotated vector.

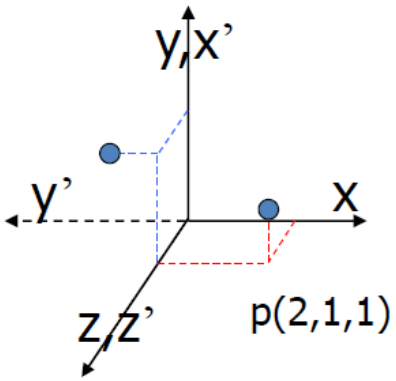


Figure 3.4: Rotation of vector (2,1,1) from [22]

$$\begin{aligned} \mathbf{p} &= \begin{bmatrix} 0 & 2 & 1 & 1 \end{bmatrix} \\ &= 2i + j + k \end{aligned} \tag{3.37}$$

$$\begin{aligned} \mathbf{q} &= \begin{bmatrix} \cos\left(\frac{90°}{2}\right) & 0 & 0 & \sin\left(\frac{90°}{2}\right) \end{bmatrix} \\ &= \frac{\sqrt{2}}{2} + \frac{\sqrt{2}}{2}k \end{aligned} \tag{3.38}$$

$$\begin{aligned} \mathbf{q^{-1}} &= \begin{bmatrix} \cos\left(\frac{90°}{2}\right) & 0 & 0 & -\sin\left(\frac{90°}{2}\right) \end{bmatrix} \\ &= \frac{\sqrt{2}}{2} - \frac{\sqrt{2}}{2}k \end{aligned} \tag{3.39}$$

$$\mathbf{p'} = \mathbf{qpq^{-1}} = -i + 2j + k \tag{3.40}$$

Therefore, the coordinates of the vector (2,1,1) rotated by 90° along the z-axis is (-1,2,1).

3.4.2 Quaternion Conversions

In dealing with unit quaternions, it is necessary to learn to convert other rotation representations into unit quaternions and back. In this study, we restrict our conversions only to what we will be using for deriving the robot kinematics applying the concept of quaternions. [6]

Unit Quaternion $\Longrightarrow$ Rotation Matrix

The conversion from a quaternion $\mathbf{q} = q_0 + q_1 i + q_2 j + q_3 k$ to a rotation matrix R is given by the following matrix,

$$R(q) := \begin{bmatrix} q_0^2 + q_1^2 - q_2^2 - q_3^2 & 2q_1q_2 + 2q_0q_3 & 2q_1q_3 - 2q_0q_2 \\ 2q_1q_2 - 2q_0q_3 & q_0^2 - q_1^2 + q_2^2 - q_3^2 & 2q_2q_3 + 2q_0q_1 \\ 2q_1q_3 + 2q_0q_2 & 2q_2q_3 - 2q_0q_1 & q_0^2 - q_1^2 - q_2^2 + q_3^2 \end{bmatrix} \tag{3.41}$$

Each elements of matrix R are denoted by r_{ij} for $i, j \in 1, 2, 3$.

Unit Quaternion $\Longleftarrow$ Rotation Matrix

To get the reverse mapping, by inspecting the relationship of the elements of R with the quaternion coefficients (q_0, q_1, q_2, q_3) will lead to four different inverse mappings. Some of these mappings will produce complex results. Therefore, we use the following conditions (depending on the parameters of R) to determine which mapping to use:

a) If $r_{22} > -r_{33}$, $r_{11} > -r_{22}$, $r_{11} > -r_{33}$, then,

$$\mathbf{q}(R) = \frac{1}{2} \begin{bmatrix} (1 + r_{11} + r_{22} + r_{33})^{\frac{1}{2}} \\ (r_{23} - r_{32})/(1 + r_{11} + r_{22} + r_{33})^{\frac{1}{2}} \\ (r_{31} - r_{13})/(1 + r_{11} + r_{22} + r_{33})^{\frac{1}{2}} \\ (r_{12} - r_{21})/(1 + r_{11} + r_{22} + r_{33})^{\frac{1}{2}} \end{bmatrix} \tag{3.42}$$

b) If $r_{22} < -r_{33}$, $r_{11} > r_{22}$, $r_{11} > r_{33}$, then,

$$\mathbf{q}(R) = \frac{1}{2} \begin{bmatrix} (r_{23} - r_{32})/(1 + r_{11} - r_{22} - r_{33})^{\frac{1}{2}} \\ (1 + r_{11} - r_{22} - r_{33})^{\frac{1}{2}} \\ (r_{12} + r_{21})/(1 + r_{11} - r_{22} - r_{33})^{\frac{1}{2}} \\ (r_{31} + r_{13})/(1 + r_{11} - r_{22} - r_{33})^{\frac{1}{2}} \end{bmatrix} \tag{3.43}$$

[6] In our application, we particularly used the conversions from rotation matrix and axis-angle representation to unit quaternions.

c) If $r_{22} > r_{33}$, $r_{11} < r_{22}$, $r_{11} < -r_{33}$, then,

$$\mathbf{q}(R) = \frac{1}{2}\begin{bmatrix}(r_{31} - r_{13})/(1 - r_{11} + r_{22} - r_{33})^{\frac{1}{2}} \\ (r_{12} + r_{21})/(1 - r_{11} + r_{22} - r_{33})^{\frac{1}{2}} \\ (1 - r_{11} + r_{22} - r_{33})^{\frac{1}{2}} \\ (r_{23} + r_{32})/(1 - r_{11} + r_{22} - r_{33})^{\frac{1}{2}}\end{bmatrix} \tag{3.44}$$

d) If $r_{22} < r_{33}$, $r_{11} < -r_{22}$, $r_{11} < r_{33}$, then,

$$\mathbf{q}(R) = \frac{1}{2}\begin{bmatrix}(r_{12} - r_{21})/(1 - r_{11} - r_{22} + r_{33})^{\frac{1}{2}} \\ (r_{31} + r_{13})/(1 - r_{11} - r_{22} + r_{33})^{\frac{1}{2}} \\ (r_{23} + r_{32})/(1 - r_{11} - r_{22} + r_{33})^{\frac{1}{2}} \\ (1 - r_{11} - r_{22} + r_{33})^{\frac{1}{2}}\end{bmatrix} \tag{3.45}$$

Choosing among these mappings helps to reduce inaccuracy by avoiding situations in which the denominator is close to zero.

Unit Quaternion $\Leftarrow$ Axis-Angle

The quaternion $\mathbf{q}$ equivalent of a rotation θ about a unit vector $\mathbf{n}$ in three-dimensional space represented by the axis-angle $rot(\mathbf{n}, \theta)$ is defined as,

$$\mathbf{q}(\theta, \mathbf{n}) := \begin{bmatrix}(\cos(\frac{\theta}{2}) & \mathbf{n}\sin(\frac{\theta}{2})\end{bmatrix} \tag{3.46}$$

Note that we are only considering unit vectors, $||\mathbf{n}|| = 1$, in this conversion.

Unit Quaternion $\Longrightarrow$ Axis-Angle

For the inverse mapping from unit quaternion $\mathbf{q}$ to the axis-angle representation $rot(\mathbf{n}, \theta)$, we first consider $\vec{q} = \begin{bmatrix}q_1 & q_2 & q_3\end{bmatrix}^T$ which is the vector part of $\mathbf{q}$. The axis-angle representation can then be computed by:

$$\theta = 2\arccos(q_0) \tag{3.47}$$

$$\mathbf{n} = \frac{\vec{q}}{||\vec{q}||} \tag{3.48}$$

Thus, the angle of rotation θ can be easily computed from the first term of the given unit quaternion while the axis of rotation $\mathbf{n}$ can be derived by simply normalizing the vector part (imaginary part) of the unit quaternion.

3.5 Dual Quaternions

Quaternions are able to represent three-dimensional rotations as we have seen in the previous discussion. However, when dealing with transformations in three-dimensions, there could also be displacement or translation in which our current form of quaternion is not able to represent. For this section, we will introduce the concept of Dual Quaternions which overcomes this limit and will be used in the entire duration of the study.

3.5.1 Definition

Dual Quaternions provide a way to represent both rotation and translation in one transformation vector. For a unit quaternion $\mathbf{q}$ which represents the appropriate rotation and for a vector $\mathbf{p}$ written as a pure quaternion which represents the corresponding displacement, the most common form used in robotics of a dual quaternion is given by the quaternion-vector pair,[7]

$$\mathbf{Q}(\mathbf{q},\mathbf{p}) = \left(\left[\cos\left(\frac{\theta}{2}\right), \sin\left(\frac{\theta}{2}\right) < k_x, k_y, k_z > \right], < p_x, p_y, p_z > \right) \tag{3.49}$$

For the quaternion part, we use the conversion from an axis-angle rotation representation to a unit quaternion. Thus, θ gives the angle of rotation and $< k_x, k_y, k_z >$ denotes the axis of rotation. The vector part is simply the displacement vector given by $\mathbf{p} =< p_x, p_y, p_z >$.

Clearly, for a transformation with no rotation and only displacement, the dual quaternion is represented by:

$$\mathbf{Q}(\mathbf{q},\mathbf{p}) = ([1, < 0, 0, 0 >], < p_x, p_y, p_z >) \tag{3.50}$$

where $[1, < 0, 0, 0 >]$ represents the unit identity quaternion.

3.5.2 Inverse of a Dual Quaternion

The inverse of a dual quaternion $\mathbf{Q}(\mathbf{q},\mathbf{p})$ is denoted as:

$$\mathbf{Q^{-1}} = ([\mathbf{q^{-1}}], < -\mathbf{q^{-1}} * \mathbf{p} * \mathbf{q} >) \tag{3.51}$$

where $\mathbf{q^{-1}}$ denotes the usual inverse of the quaternion $\mathbf{q}$. Therefore, for a quaternion $\mathbf{q} = [s, \vec{v}]$, the vector part can also be written as,

$$\mathbf{q^{-1}} * \mathbf{p} * \mathbf{q} = -\mathbf{p} + [-2s(\vec{v} \times (-\mathbf{p})) + 2\vec{v} \times (\vec{v} \times (-\mathbf{p})) \tag{3.52}$$

Note that $\mathbf{p}$ is a pure quaternion (zero real part) and thus, is also a vector.

[7] Dual Quaternions are generally defined using dual numbers such that $\mathbf{Q} = \mathbf{q} + \epsilon\mathbf{p}$ where $\epsilon^2 = 0$. This is clearly described in [14].

3.5.3 Multiplication of a Dual Quaternion

In dual quaternions, sequence of rotations and displacements are represented by multiplying their corresponding dual quaternion representations. Therefore, for transformations $\mathbf{Q_1}(\mathbf{q_1}, \mathbf{p_1})$ followed by $\mathbf{Q_2}(\mathbf{q_2}, \mathbf{p_2})$ for a local coordinate frame, their dual quaternion product is given by:

$$\mathbf{Q_1 Q_2} = (\mathbf{q_1}, \mathbf{p_1}) * (\mathbf{q_2}, \mathbf{p_2}) = ([\mathbf{q_1} * \mathbf{q_2}], < \mathbf{q_1} * \mathbf{p_2} * \mathbf{q_1^{-1}} + \mathbf{p_1} >) \quad (3.53)$$

where the quaternion part is simply multiplication between quaternions $\mathbf{q_1}$ and $\mathbf{q_2}$ and the vector part can also be written as,

$$\mathbf{q_1} * \mathbf{p_2} * \mathbf{q_1^{-1}} + \mathbf{p_1} = \mathbf{p_2} + 2s_1(\vec{v_1} \times \mathbf{p_2}) + 2\vec{v_1} \times (\vec{v_1} \times \mathbf{p_2}) + \mathbf{p_1} \quad (3.54)$$

where $\mathbf{q_1} = [s_1, \vec{v_1}]$ and $\mathbf{p_2}$ is a pure quaternion and also acts as a vector.

Analyzing the operation for the translation part of the dual quaternion shows that the succeeding translation is rotated first by the previous rotation and the resulting rotated translation is then added to the current translation.

3.5.4 Transformation by a Dual Quaternion

Transforming a vector using dual quaternions follows almost the same process as multiplying two dual quaternions. If we have the final transformation given by the dual quaternion,

$$\mathbf{Q_F} = ([w, < a, b, c >], < p_x, p_y, p_z >) \quad (3.55)$$

where $[w, < a, b, c >] = \mathbf{q_F}$ and $< p_x, p_y, p_z >= \mathbf{p_F}$, and an initial vector to be transformed given by,

$$\mathbf{v} =< v_x, v_y, v_z > \quad (3.56)$$

We get the transformed (rotated and translated) vector $\mathbf{v'} =< v'_x, v'_y, v'_z >$ by,

$$\mathbf{v'} = (\mathbf{q_F} * \mathbf{v} * \mathbf{q_F^{-1}}) + \mathbf{p_F} \quad (3.57)$$

Therefore, a vector is transformed using dual quaternions by first rotating it using the quaternion part $\mathbf{q_F}$ (representing the rotation) of the dual quaternion, and then translating it by adding to it the vector part $\mathbf{p_F}$ (representing the translation) of the dual quaternion. This process is similar to how a transformation is applied using transformation matrices.

In the next chapter, we will demonstrate the application of dual quaternions by using it in the derivation of both a robot's forward and inverse kinematics with practical application on the Universal Robot UR3.

Chapter 4

Application of Quaternions to Universal Robot UR3

The main goal of this study is to be able to apply the concept of quaternions in manipulating robot movements. For this study, we will be working with the Universal Robot UR3 and derive its forward and inverse kinematics using quaternions. At the end of this chapter, an actual programming application of computing for the kinematics of UR3 is tested and presented.

4.1 Universal Robot - UR3

The Universal Robot UR3 is a collaborative robotic arm. It has six degrees of freedom (6DOF) and is composed of six revolute joints, commonly referred to as the (1) base, (2) shoulder, (3) elbow, (4) wrist 1, (5) wrist 2, and (6) wrist 3, each with a working range of 360 degrees. [1]

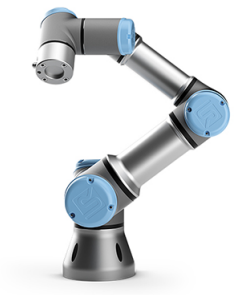


Figure 4.1: Universal Robot UR3

[1] Technical specifications of the robot can be found in Appendix A.

4.2 Forward Kinematics by Dual Quaternions

Forward Kinematics is generally easier to compute than its counterpart and the same goes for its computation by Dual Quaternions. Our goal is to determine the final orientation and position of the end-effector given all the joint angles. Basically, there are at least two ways to derive the Forward Kinematics using Dual Quaternions. It can either be starting from a rotation matrix or from an axis-angle representation. Any of these two will result in the same final rotation but each uses different coordinates for the initial TCP position. In this study, we will be using the approach with the axis-angle representation.[2]

4.2.1 Representations of Joints Transformation

Given the kinematic structure of the UR3 robot in zero position, we derive the dual quaternions representing the transformation in each joints by deriving first the axis-angle representation of each joints with respect to the base frame and set it as the first part of our dual quaternion and the second part by getting the translation from the current joint frame to the next: [3]

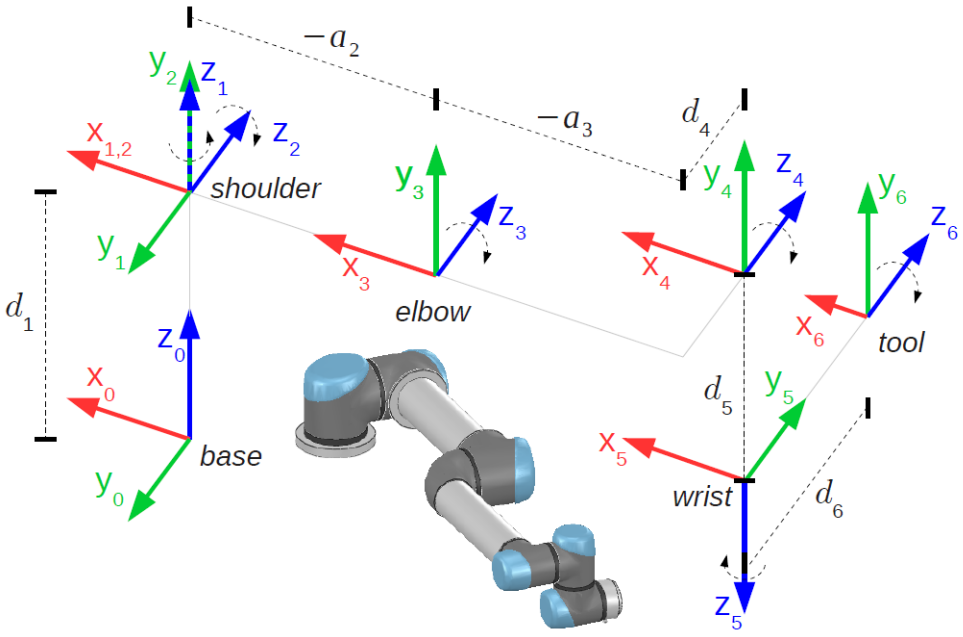


Figure 4.2: Kinematic Structure of UR3 in zero position ($\theta_{1,2,3,4,5,6} = 0$) [3].

[2]This study follows the approach done in [4] and [15]
[3]The usual convention of deriving kinematics for UR3 can be found in [3] and [11]

Note that the z-axis represents the axis of rotation and we are following the right-hand rule to determine the positive direction of the rotation on each rotation axis. We will represent the z-axes with respect to the coordinate axis of the base frame.

For the UR3, we are also given the following DH parameters:[4]

i	α_{i-1}	a_{i-1}	d_i	θ_i
1	0	0	d_1	θ_1
2	$\alpha_1 = 90°$	0	0	θ_2
3	0	a_2	0	θ_3
4	0	a_3	d_4	θ_4
5	$\alpha_4 = 90°$	0	d_5	θ_5
6	$\alpha_5 = -90°$	0	d_6	θ_6

Table 4.1: UR3 Denavit-Harteberg Parameters

For the first joint, in the rotation part, we have the angle of rotation as θ_1 and we have our axis of rotation z_1 aligned with the base axis z_0. For the translation part, starting from the base frame to the shoulder frame, there is a displacement of d_1 in the z-axis. Therefore, our dual quaternion for the first joint is represented by:

$$\mathbf{Q_1} = \left(\left[\cos\left(\frac{\theta_1}{2}\right), \sin\left(\frac{\theta_1}{2}\right) < 0, 0, 1 > \right], < 0, 0, d_1 > \right)$$

For the second joint, the angle of rotation is θ_2 and our axis of rotation z_2 is now parallel with the $-y_0$ axis of the base frame. For the translation part, we have to analyze that rotating the second joint will result to a translation in the x_0-axis and the z_0-axis which is dependent on a_2 and θ_2. Therefore, we have:

$$\mathbf{Q_2} = \left(\left[\cos\left(\frac{\theta_2}{2}\right), \sin\left(\frac{\theta_2}{2}\right) < 0, -1, 0 > \right], < a_2 \cos\theta_2, 0, a_2 \sin\theta_2 > \right)$$

For the third joint, the angle of rotation is θ_3 and our axis of rotation z_3, same as z_2, is parallel with the $-y_0$-axis of the base frame. For the translation part, rotating the third joint will result to a translation in the x_0-axis and the z_0-axis which is dependent on a_3 and θ_3. Aside from this, to get to the fourth joint, there is also a displacement of d_4 in the $-y_0$-axis. Thus:

$$\mathbf{Q_3} = \left(\left[\cos\left(\frac{\theta_3}{2}\right), \sin\left(\frac{\theta_3}{2}\right) < 0, -1, 0 > \right], < a_3 \cos\theta_3, -d_4, a_3 \sin\theta_3 > \right)$$

[4]DH parameters are discussed and illustrated in Section 4.4.1

For the fourth joint, the angle of rotation is θ_4 and our axis of rotation z_4, same as z_2 and z_3, is parallel with the $-y_0$-axis of the base frame. For the translation part, rotating the fourth joint will result to a translation in the x_0-axis and the z_0-axis which is dependent on d_5 and θ_5. Therefore, we have:

$$\mathbf{Q_4} = \left(\left[\cos\left(\frac{\theta_4}{2}\right), \sin\left(\frac{\theta_4}{2}\right) < 0, -1, 0 > \right], < d_5 \sin\theta_4, 0, -d_5 \cos\theta_4 > \right)$$

For the fifth joint, the angle of rotation is θ_5 and our axis of rotation z_5 is parallel with the $-z_0$-axis of the base frame. For the translation part, rotating the fifth joint will result to a translation in the x_0-axis and the y_0-axis which is dependent on d_6 and θ_5. Therefore, we have:

$$\mathbf{Q_5} = \left(\left[\cos\left(\frac{\theta_5}{2}\right), \sin\left(\frac{\theta_5}{2}\right) < 0, 0, -1 > \right], < -d_6 \sin\theta_5, -d_6 \cos\theta_5, 0 > \right)$$

For the sixth and last joint, the angle of rotation is θ_6 and our axis of rotation z_6 is parallel with the $-y_0$-axis of the base frame. There is no translation since it is the last frame. Therefore, we have:

$$\mathbf{Q_6} = \left(\left[\cos\left(\frac{\theta_6}{2}\right), \sin\left(\frac{\theta_6}{2}\right) < 0, -1, 0 > \right], < 0, 0, 0 > \right)$$

Representing the transformations by dual quaternions has a similar concept with the transformation matrices. The main difference is that dual quaternions are more compact in such a way that we only need 7 components whereas for the transformation matrices, we are using 4x4 = 16 components.

4.2.2 UR3 Forward Kinematics Computation

After deriving the corresponding dual quaternions for each joints, we take their products starting from the base frame to the final frame to get the final dual quaternion representing the final transformation and the forward kinematics of the robot. Thus, we have,

$$\mathbf{Q_{FK}} = \mathbf{Q_1} * \mathbf{Q_2} * \mathbf{Q_3} * \mathbf{Q_4} * \mathbf{Q_5} * \mathbf{Q_6} \tag{4.1}$$

For this approach, the coordinates of the point to be transformed has to be also with respect to the base frame. Therefore, during the transformation, the point is rotated by the rotation part of $\mathbf{Q_{FK}}$ and then translated by its translation part starting from the base frame.

4.3 Inverse Kinematics by Dual Quaternions

Deriving the inverse kinematics requires more work and analysis than forward kinematics. We will have to apply the concept of transformations by dual quaternions, deal with complex trigonometric equations, and geometrically analyze the transformations done by each joints to the robot and how each of them affects the coordinates of the succeeding joint frames. Our goal in inverse kinematics is to determine the joint angles needed to move the end-effector to the desired position and orientation which is specified by the final transformation, $\mathbf{Q_{FK}} = ([w, < a, b, c >], < p_x, p_y, p_z >)$.

4.3.1 Representations of Joints Transformations

To begin with the derivation of inverse kinematics, we need the dual quaternions representing the transformation done by each joints $\mathbf{Q}_i$, for $1 \leq i \leq 6$, which we have already derived for the forward kinematics above, and also their corresponding dual quaternion inverses, $\mathbf{Q}_i^{-1}$.

Moving forward, we will be using the following representations for lengthy equations: $\cos\theta_i \rightarrow c_i$, $\sin\theta_i \rightarrow s_i$, $\cos\left(\frac{\theta_i}{2}\right) \rightarrow \bar{c}_i$, $\sin\left(\frac{\theta_i}{2}\right) \rightarrow \bar{s}_i$

Therefore, for each joints, we have:

$$\begin{aligned}
\mathbf{Q_1} &= ([\bar{c}_1 + \bar{s}_1 k], < d_1 k >) \\
\mathbf{Q_2} &= ([\bar{c}_2 - \bar{s}_2 j], < a_2 c_2 i + a_2 s_2 k >) \\
\mathbf{Q_3} &= ([\bar{c}_3 - \bar{s}_3 j], < a_3 c_3 i - d_4 j + a_3 s_3 k >) \\
\mathbf{Q_4} &= ([\bar{c}_4 - \bar{s}_4 j], < d_5 s_4 i - d_5 c_4 k >) \\
\mathbf{Q_5} &= ([\bar{c}_5 - \bar{s}_5 k], < -d_6 s_5 i - d_6 c_5 j >) \\
\mathbf{Q_6} &= ([\bar{c}_6 - \bar{s}_6 j], < 0 >)
\end{aligned}$$

and the inverses as:

$$\begin{aligned}
\mathbf{Q_1}^{-1} &= ([\bar{c}_1 - \bar{s}_1 k], < -d_1 k >) \\
\mathbf{Q_2}^{-1} &= ([\bar{c}_2 + \bar{s}_2 j], < -a_2 i >) \\
\mathbf{Q_3}^{-1} &= ([\bar{c}_3 + \bar{s}_3 j], < -a_3 i + d_4 j >) \\
\mathbf{Q_4}^{-1} &= ([\bar{c}_4 + \bar{s}_4 j], < d_5 k >) \\
\mathbf{Q_5}^{-1} &= ([\bar{c}_5 + \bar{s}_5 k], < d_6 i >) \\
\mathbf{Q_6}^{-1} &= ([\bar{c}_6 + \bar{s}_6 j], < 0 >)
\end{aligned}$$

We will use both of these joints' dual quaternion representations to get the relationship among the succeeding transformation combinations.

4.3.2 Joints Transformation Products

Using the dual quaternions $\mathbf{Q}_i$ $(1 \leq i \leq 6)$, which denotes the kinematics transformations that describes the spatial relationships between succeeding coordinate frames, and $\mathbf{Q_i}^{-1}$ as their corresponding inverses, we first define the dual quaternion product $\mathbf{M}_i$ as:

$$\mathbf{M}_i = \mathbf{Q}_i * \mathbf{M}_{i+1} \ \textit{where}\ 1 \leq i \leq 6 \tag{4.2}$$

For $i = 6$, we have that $\mathbf{M}_6 = \mathbf{Q}_6$. Therefore, we have $\mathbf{M}_i$ as,

$$\begin{aligned}
\mathbf{M}_6 &= \mathbf{Q}_6 \\
\mathbf{M}_5 &= \mathbf{Q}_5 * \mathbf{M}_6 = \mathbf{Q}_5 * \mathbf{Q}_6 \\
\mathbf{M}_4 &= \mathbf{Q}_4 * \mathbf{M}_5 = \mathbf{Q}_4 * \mathbf{Q}_5 * \mathbf{Q}_5 \\
\mathbf{M}_3 &= \mathbf{Q}_3 * \mathbf{M}_4 = \mathbf{Q}_3 * \mathbf{Q}_4 * \mathbf{Q}_5 * \mathbf{Q}_6 \\
\mathbf{M}_2 &= \mathbf{Q}_2 * \mathbf{M}_3 = \mathbf{Q}_2 * \mathbf{Q}_3 * \mathbf{Q}_4 * \mathbf{Q}_5 * \mathbf{Q}_6 \\
\mathbf{M}_1 &= \mathbf{Q}_1 * \mathbf{M}_2 = \mathbf{Q}_1 * \mathbf{Q}_2 * \mathbf{Q}_3 * \mathbf{Q}_4 * \mathbf{Q}_5 * \mathbf{Q}_6
\end{aligned}$$

Then, we define another dual quaternion product $\mathbf{N}_i$ as follows:

$$\mathbf{N}_{i+1} = \mathbf{Q}_i^{-1} * \mathbf{N}_i \ \textit{where}\ 1 \leq i \leq 6 \tag{4.3}$$

For $i = 0$, we define $\mathbf{N}_1$ as the final transformation quaternion which means,

$$\mathbf{N}_1 = [R_w, T_w] = ([w, < a, b, c >], < p_x, p_y, p_z >)$$

Therefore, we have $\mathbf{N}_i$ as,

$$\begin{aligned}
\mathbf{N}_1 &= [R_w, T_w] \\
\mathbf{N}_2 &= \mathbf{Q}_1^{-1} * \mathbf{N}_1 = \mathbf{Q}_1^{-1} * [R_w, T_w] \\
\mathbf{N}_3 &= \mathbf{Q}_2^{-1} * \mathbf{N}_2 = \mathbf{Q}_2^{-1} * \mathbf{Q}_1^{-1} * [R_w, T_w] \\
\mathbf{N}_4 &= \mathbf{Q}_3^{-1} * \mathbf{N}_3 = \mathbf{Q}_3^{-1} * \mathbf{Q}_2^{-1} * \mathbf{Q}_1^{-1} * [R_w, T_w] \\
\mathbf{N}_5 &= \mathbf{Q}_4^{-1} * \mathbf{N}_4 = \mathbf{Q}_4^{-1} * \mathbf{Q}_3^{-1} * \mathbf{Q}_2^{-1} * \mathbf{Q}_1^{-1} * [R_w, T_w] \\
\mathbf{N}_6 &= \mathbf{Q}_5^{-1} * \mathbf{N}_5 = \mathbf{Q}_5^{-1} * \mathbf{Q}_4^{-1} * \mathbf{Q}_3^{-1} * \mathbf{Q}_2^{-1} * \mathbf{Q}_1^{-1} * [R_w, T_w]
\end{aligned}$$

Clearly, we can see that $\mathbf{M}_1$ and $\mathbf{N}_1$ are just equivalent to forward kinematics $\mathbf{Q_{FK}}$. Analyzing further the composition of each of these transformations between dual quaternion products $\mathbf{M}$ and $\mathbf{N}$ implies that $\mathbf{M}_i = \mathbf{N}_i$. Therefore, their corresponding terms should also be equal.

The next step is to equate each corresponding dual quaternion terms between $\mathbf{M}_i$ and $\mathbf{N}_i$ which will give us several equations that we can use to compute for the inverse kinematics.

4.3.3 *Base* to *Wrists* Frames Transformations

Computing for the transformation products $\mathbf{M}_i$ and $\mathbf{N}_i$ where $(a \leq i \leq b)$ will lead us to $((b-a)+1) \times 7 \times 2$ equations where the multipliers represent the number of transformations $(b-(a-1))$, the number of terms in the dual quaternion (7), and for both $\mathbf{M}$ and $\mathbf{N}$ correspondingly. We equate the terms for each corresponding transformation products and get $((b-a)+1)) \times 7$ equations and we try to find a relationship among some of these equations and manipulate them to be able to solve explicitly for the joint angles.

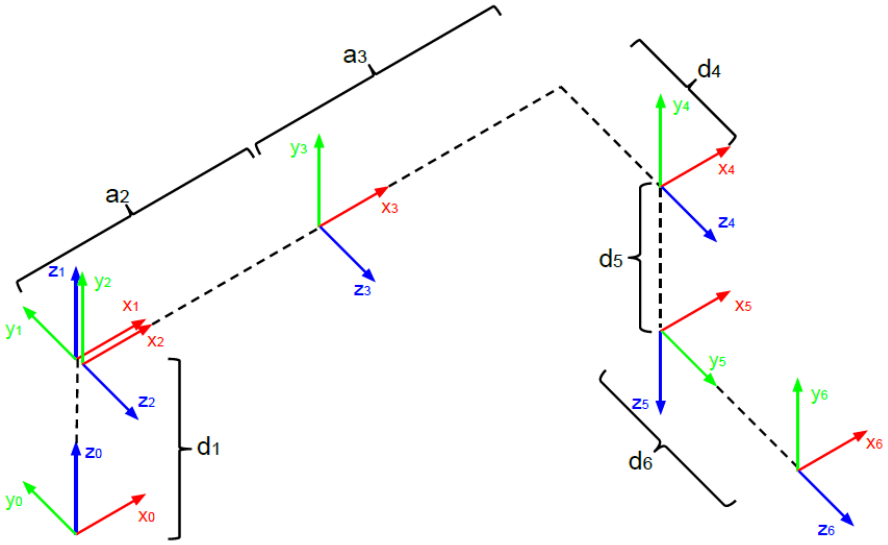


Figure 4.3: Coordinate Frames of UR3 ($\theta_{1,2,3,4,5,6} = 0$) from [11].

Frame 1 (*Base*) to Frame 6 (*Wrist 3*) Transformation

Following [4] and [15], we should start first with the *Base to Wrist 3* Transformation. This will lead us to 84 equations where $(1 \leq i \leq 6)$. We equate the corresponding transformation products and terms between $\mathbf{M}_i$ and $\mathbf{N}_i$. These equations are listed in Appendix B. The ease of getting a straightforward equation to compute for the joint angles depend on the robot structure. In the case of UR3, there are no equations that can straightforwardly derive the value of any joint angle. We can find an equation for θ_5 and then θ_6 but we need θ_1 to solve them. We can apply numerical methods to estimate the values of the joint angles but in our case, we need exact values. To solve this, we can then incorporate the geometrical approach which is commonly used in deriving a robot's inverse kinematics and moving on to the next transformation which is *Base to Wrist 2* Transformation.

Frame 1 (*Base*) to Frame 5 (*Wrist 2*) Transformation

Our main motivation for moving on to this transformation is to solve for θ_1. The geometrical approach is to find the relationship between joints' angles and displacements. The figure below shows the relationship between the joint angle θ_1 with the Frame 1 (*Base*) and Frame 5 (*Wrist 2*) coordinates.

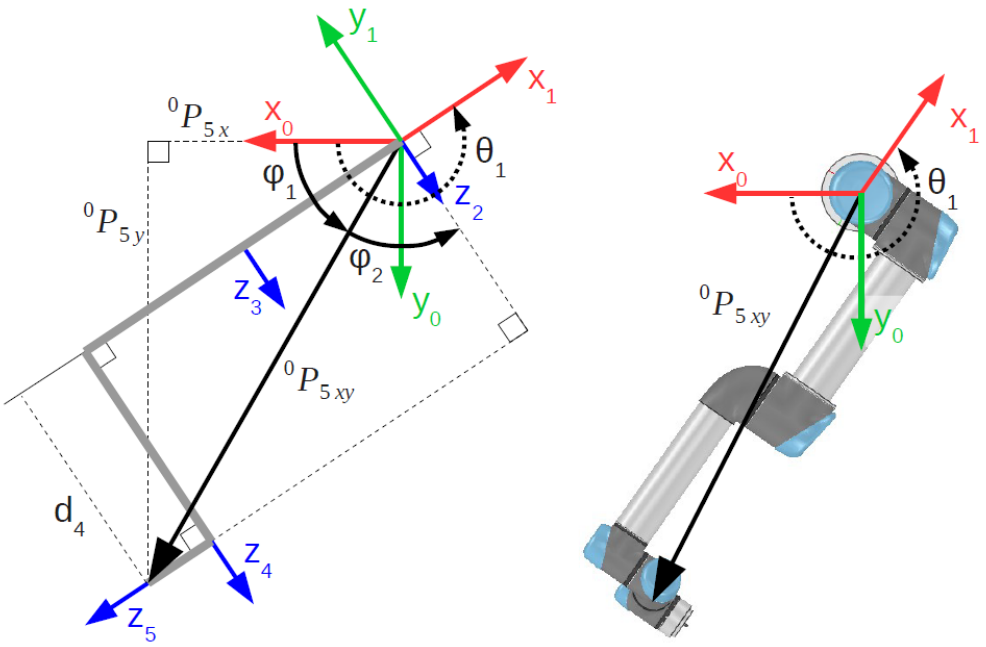


Figure 4.4: UR3 Robot Structure from Frame 1 to Frame 5 (from [3])

As we can see, it will be possible to determine the value of θ_1 using the displacement between Frame 1, $\mathbf{Q}_1$, and Frame 5, $\mathbf{Q}_5$. The *Base to Wrist 2* Transformation will lead us to 70 equations where $(1 \leq i \leq 5)$. Fortunately, instead of recomputing these transformations, we can use the previous transformation and take $\mathbf{N_1}$ to $\mathbf{N_5}$ since they are just equivalent. The only difference is that we are working on the coordinates at Frame 5. Thus, we will only need to do a translation from Frame 6 coordinates to Frame 5. As for the forward transformation products, we should start with $\mathbf{M_5}$ instead of from $\mathbf{M_6}$. This can be done simply by setting $\mathbf{M_5} = \mathbf{Q_5}$ and the translation to zero. Equating the terms of $\mathbf{M_5}$ with its corresponding terms in $\mathbf{N_5}$ and applying trigonometric identities will give us our formula for θ_1. This will allow us to then solve for θ_5 and θ_6 using the equations we got from the Frame 1 (*Base*) to Frame 6 (*Wrist 3*) Transformation.

Deriving Joint Angle θ_1 (*Base*)

To derive a formula for θ_1, we can translate the coordinates first in the final frame (*Wrist 3*) back to Frame 5 (*Wrist 2*) and use these in the formula to be extracted from the transformation products. To translate back to Frame 5, we analyze the transformations by *Wrist 3* ($\mathbf{Q_6}$) and *Wrist 2* ($\mathbf{Q_5}$) and their contributions to the final transformation $\mathbf{Q_{FK}}$.

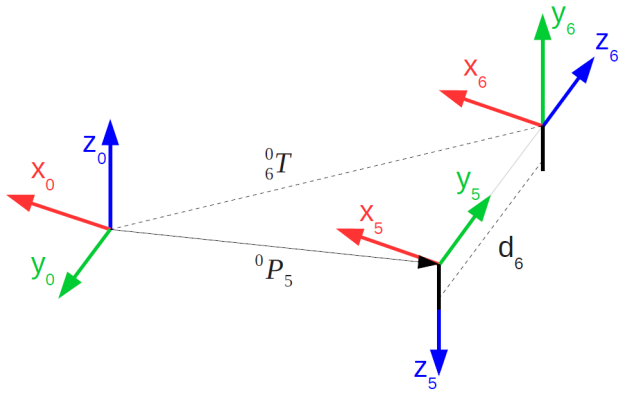


Figure 4.5: Translation from Frame 6 (*Wrist 3*) to Frame 5 (*Wrist 2*)[3]

Frame 5 (*Wrist 2*) Transformation:

$$\mathbf{Q_5} = ([\bar{c}_5 - \bar{s}_5 k], < -d_6 s_5 i - d_6 c_5 j >)$$

Frame 6 (*Wrist 3*) Transformation:

$$\mathbf{Q_6} = ([\bar{c}_6 - \bar{s}_6 j], < 0 >)$$

Final Transformation: Frame 1 (*Base*) to Frame 6 (*Wrist 3*)

$$\mathbf{Q_{FK}} = \mathbf{Q_1} * \mathbf{Q_2} * \mathbf{Q_3} * \mathbf{Q_4} * \mathbf{Q_5} * \mathbf{Q_6} = \mathbf{Q_6^1}$$

Clearly, there are no translations being done by $\mathbf{Q_6}$, only rotation. Therefore, our end-effector position is already determined in the transformation upto $\mathbf{Q_5}$. Thus, for the translation part ($\mathbf{p}$) of $\mathbf{Q_{FK}}$, we have,

$$\begin{aligned}(\mathbf{Q_{FK}})_\mathbf{p} &= (\mathbf{Q_6^1})_\mathbf{p} = (\mathbf{Q_5^1})_\mathbf{p} \\ &= (\mathbf{Q_1} * \mathbf{Q_2} * \mathbf{Q_3} * \mathbf{Q_4} * \mathbf{Q_5})_\mathbf{p}\end{aligned} \tag{4.4}$$

Note that what we need are the coordinates at Frame 5 and we are only dealing with the translation part of the dual quaternion.

We need to revert the translation done by $\mathbf{Q_5}$. We do this by using its inverse,

$$\mathbf{Q_5}^{-1} = ([\bar{c}_5 + \bar{s}_5 k], < d_6 i >)$$

Therefore,

$$\begin{aligned}(\mathbf{Q_{FK}} * \mathbf{Q_5^{-1}})_\mathbf{p} &= (\mathbf{Q_1} * \mathbf{Q_2} * \mathbf{Q_3} * \mathbf{Q_4} * \mathbf{Q_5} * \mathbf{Q_5^{-1}})_\mathbf{p} \\ &= (\mathbf{Q_1} * \mathbf{Q_2} * \mathbf{Q_3} * \mathbf{Q_4})_\mathbf{p} \\ &= (\mathbf{Q_4^1})_\mathbf{p} \end{aligned} \tag{4.5}$$

We do not have a value for θ_5 yet so we cannot completely build $\mathbf{Q_5}$, but since we are only dealing with the translation part, we don't really need it. Instead, we can set an arbitrary value to its quaternion part and define only the translation part. Recall that determining the translation part from a dual quaternion multiplication is defined without using $\mathbf{q_B}$,

$$(\mathbf{Q_A} * \mathbf{Q_B})_\mathbf{p} =< \mathbf{q_A} * \mathbf{p_B} * \mathbf{q_A^{-1}} + \mathbf{p_A} > \tag{4.6}$$

Since $(Q_{FK})_p = (Q_5^1)_p$, we can have,

$$(\mathbf{Q_{FK}} * \mathbf{Q_5^{-1}})_\mathbf{p} =< \mathbf{q_{FK}} * \mathbf{p_5^{-1}} * \mathbf{q_{FK}^{-1}} + \mathbf{p_{FK}} > \tag{4.7}$$

$$=< x_5 i + y_5 j + z_5 k >= (\mathbf{Q_4^1})_\mathbf{p} \tag{4.8}$$

where $< x_5, y_5, z_5 >$ are the coordinates in Frame 5 with respect to Frame 1.

Now, to compute for θ_1, we need the dual quaternion transformation products for the *Base* to *Wrist 2* Transformation. As we have stated in our analysis, we can simply use (B.13) $\underline{\mathbf{M_{56}} = \mathbf{N_{56}}}$, where the $\cos\theta_5$ term is excluded.

Therefore,

$$\begin{aligned} 0 &= y\cos\theta_1 - x\sin\theta_1 + d_4 \\ y\cos\theta_1 - x\sin\theta_1 &= -d_4 \end{aligned} \tag{4.9}$$

We apply the following trigonometric identity,

$$a\sin\theta + b\cos\theta = c$$

$$\theta = atan2(a, b) \pm atan2(\sqrt{a^2 + b^2 - c^2}, c) \tag{4.10}$$

And now we use the coordinates from Frame 5 in deriving the formula for θ_1,

$$\theta_1 = atan2(-x_5, y_5) \pm atan2(\sqrt{x_5^2 + y_5^2 - d_4^2}, -d_4) \tag{4.11}$$

We consider two values for θ_1 corresponding to the *Base* joint rotation of setting the *Shoulder* either to the left or to the right.

Deriving Joint Angle θ_5 (*Wrist 2*)

Now that we are given θ_1, we can use the same pair of equations from the dual quaternion transformation products for the *Base* to *Wrist 3* Transformation (B.13) but without excluding any term to derive the equation for θ_5,

$$\underline{\mathbf{M_{56} = N_{56}}}$$

$$-d_6 \cos\theta_5 = y\cos\theta_1 - x\sin\theta_1 + d_4 \tag{4.12}$$

We can straightforwardly compute for θ_5 by,

$$\theta_5 = \pm acos\left(\frac{x_6\sin\theta_1 - y_6\cos\theta_1 - d_4}{d_6}\right) \tag{4.13}$$

We consider two values for θ_5 for the *Wrist 2* joint rotation corresponding to the wrist being either up or down.

Deriving Joint Angle θ_6 (*Wrist 3*)

We now have θ_1 and θ_5, we can then use the pairs of equations (B.29), (B.30), (B.31) and (B.32) also from the dual quaternion transformation products for the *Base* to *Wrist 3* Transformation to derive the equation for θ_6,

$$\underline{\mathbf{M_{21} = N_{21}}}$$

$$\cos\left(\frac{\theta_5}{2}\right)\cos\left(\frac{\theta_2+\theta_3+\theta_4+\theta_6}{2}\right) = w\cos\left(\frac{\theta_1}{2}\right) + c\sin\left(\frac{\theta_1}{2}\right) \tag{4.14}$$

$$\underline{\mathbf{M_{22} = N_{22}}}$$

$$\sin\left(\frac{\theta_5}{2}\right)\sin\left(\frac{\theta_2+\theta_3+\theta_4-\theta_6}{2}\right) = a\cos\left(\frac{\theta_1}{2}\right) + b\sin\left(\frac{\theta_1}{2}\right) \tag{4.15}$$

$$\underline{\mathbf{M_{23} = N_{23}}}$$

$$-\cos\left(\frac{\theta_5}{2}\right)\sin\left(\frac{\theta_2+\theta_3+\theta_4+\theta_6}{2}\right) = -a\sin\left(\frac{\theta_1}{2}\right) + b\cos\left(\frac{\theta_1}{2}\right) \tag{4.16}$$

$$\underline{\mathbf{M_{24} = N_{24}}}$$

$$-\sin\left(\frac{\theta_5}{2}\right)\cos\left(\frac{\theta_2+\theta_3+\theta_4-\theta_6}{2}\right) = -w\sin\left(\frac{\theta_1}{2}\right) + c\cos\left(\frac{\theta_1}{2}\right) \tag{4.17}$$

Using trigonometric identities *(sum of angles)*, these are equivalent to,

$$\bar{c}_{(2+3+4)}\bar{c}_6 - \bar{s}_{(2+3+4)}\bar{s}_6 = \left(\frac{w\bar{c}_1 + c\bar{s}_1}{\bar{c}_5}\right) \qquad = \alpha \tag{4.18}$$

$$\bar{s}_{(2+3+4)}\bar{c}_6 - \bar{c}_{(2+3+4)}\bar{s}_6 = \left(\frac{a\bar{c}_1 + b\bar{s}_1}{\bar{s}_5}\right) \qquad = \beta \tag{4.19}$$

$$\bar{s}_{(2+3+4)}\bar{c}_6 + \bar{c}_{(2+3+4)}\bar{s}_6 = \left(\frac{a\bar{s}_1 - b\bar{c}_1}{\bar{c}_5}\right) \qquad = \gamma \tag{4.20}$$

$$\bar{c}_{(2+3+4)}\bar{c}_6 + \bar{s}_{(2+3+4)}\bar{s}_6 = \left(\frac{w\bar{s}_1 - c\bar{c}_1}{\bar{s}_5}\right) \qquad = \delta \tag{4.21}$$

Multiplying (4.18) by $\bar{s}_6$ will give us,

$$\bar{c}_{(2+3+4)}\bar{c}_6\bar{s}_6 - \bar{s}_{(2+3+4)}\bar{s}_6^2 = \alpha\bar{s}_6 \tag{4.22}$$

And multiplying (4.19) by $\bar{c}_6$ will give us:

$$\bar{s}_{(2+3+4)}\bar{c}_6^2 - \bar{c}_{(2+3+4)}\bar{s}_6\bar{c}_6 = \beta\bar{c}_6 \tag{4.23}$$

Getting the sum of these two equations will give us,

$$\bar{s}_{(2+3+4)}(\bar{c}_6^2 - \bar{s}_6^2) = \bar{s}_{(2+3+4)}c_6 = \alpha\bar{s}_6 + \beta\bar{c}_6 \tag{4.24}$$

Then, multiplying (4.20) by $\bar{c}_6$ will give us,

$$\bar{s}_{(2+3+4)}\bar{c}_6^2 + \bar{c}_{(2+3+4)}\bar{s}_6\bar{c}_6 = \gamma\bar{c}_6 \tag{4.25}$$

And multiplying (4.21) by $-\bar{s}_6$ will give us:

$$-\bar{c}_{(2+3+4)}\bar{c}_6\bar{s}_6 - \bar{s}_{(2+3+4)}\bar{s}_6^2 = -\delta\bar{s}_6 \tag{4.26}$$

Getting the sum of these two equations will give us,

$$\bar{s}_{(2+3+4)}(\bar{c}_6^2 - \bar{s}_6^2) = \bar{s}_{(2+3+4)}c_6 = \gamma\bar{c}_6 - \delta\bar{s}_6 \tag{4.27}$$

We can subtract equations (4.24) and (4.27) to get the following simplified equation in which only θ_6 is undetermined:

$$\alpha\bar{s}_6 + \beta\bar{c}_6 = \gamma\bar{c}_6 - \delta\bar{s}_6 \longrightarrow \bar{s}_6(\alpha + \delta) = \bar{c}_6(\gamma - \beta)$$

Rearranging this equation to solve for θ_6 will give us the following formula,

$$\theta_6 = 2atan2(\gamma - \beta, \alpha + \delta) \tag{4.28}$$

There is only one value for θ_6 which corresponds to the *Wrist 3* joint rotation.

4.3.4 *Shoulder* to *Wrist* Frames Transformation

We are now left with only θ_2, θ_3 and θ_4 to be determined. Notice that their corresponding joints forms a 3R-planar manipulator as seen in the figure below,

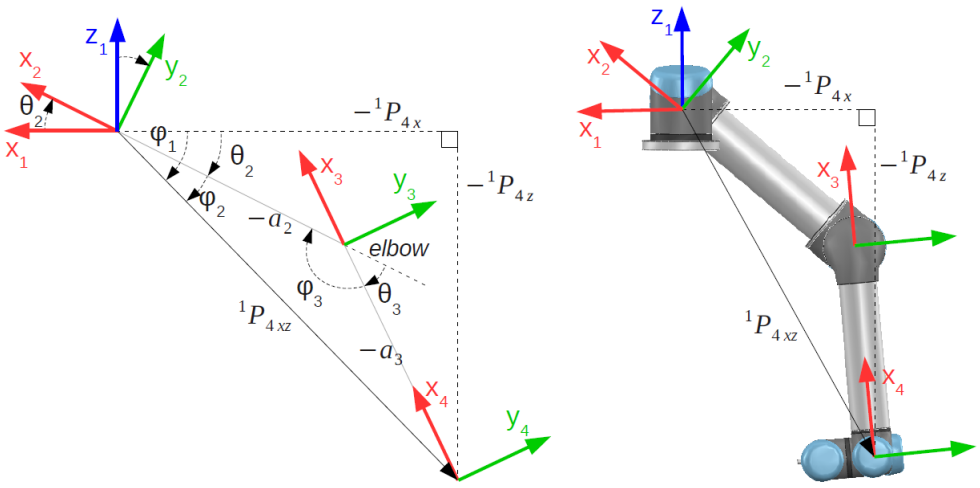


Figure 4.6: 3R-Planar Manipulator formed by UR3 Joints 2 to 4 from [3].

Therefore, we may get the transformation products $\mathbf{M}_i$ and $\mathbf{N}_i$ where $(2 \leq i \leq 4)$. This will lead us to 42 equations. Equating the terms for each corresponding transformation products will leave us with $\frac{42}{2} = 21$ equations and we try to find a relationship among some of these equations and manipulate them to be able to solve explicitly for our remaining joint angles to be determined.

We thus have the following dual quaternion representation for each joints,

$$\begin{aligned}
\mathbf{Q_2} &= ([\bar{c}_2 - \bar{s}_2 j], < a_2 c_2 i + a_2 s_2 k >) \\
\mathbf{Q_3} &= ([\bar{c}_3 - \bar{s}_3 j], < a_3 c_3 i - d_4 j + a_3 s_3 k >) \\
\mathbf{Q_4} &= ([\bar{c}_4 - \bar{s}_4 j], < 0 >)
\end{aligned}$$

and the inverses as:

$$\begin{aligned}
\mathbf{Q_2}^{-1} &= ([\bar{c}_2 + \bar{s}_2 j], < -a_2 i >) \\
\mathbf{Q_3}^{-1} &= ([\bar{c}_3 + \bar{s}_3 j], < -a_3 i + d_4 j >) \\
\mathbf{Q_4}^{-1} &= ([\bar{c}_4 + \bar{s}_4 j], < 0 >)
\end{aligned}$$

Therefore, our dual quaternion transformation product $\mathbf{M}_i$ is formulated as:

$$\begin{aligned}\mathbf{M}_4 &= \mathbf{Q}_4 \\ \mathbf{M}_3 &= \mathbf{Q}_3 * \mathbf{M}_4 = \mathbf{Q}_3 * \mathbf{Q}_4 \\ \mathbf{M}_2 &= \mathbf{Q}_2 * \mathbf{M}_3 = \mathbf{Q}_2 * \mathbf{Q}_3 * \mathbf{Q}_4\end{aligned}$$

And the dual quaternion product $\mathbf{N}_i$ as:

$$\begin{aligned}\mathbf{N}_2 &= \mathbf{Q_4^2} = ([w_4, < a_4, b_4, c_4 >], < x_4, y_4, z_4 >) \\ \mathbf{N}_3 &= \mathbf{Q}_2^{-1} * \mathbf{N}_2 = \mathbf{Q}_2^{-1} * \mathbf{Q_4^2} \\ \mathbf{N}_4 &= \mathbf{Q}_3^{-1} * \mathbf{N}_3 = \mathbf{Q}_3^{-1} * \mathbf{Q}_2^{-1} * \mathbf{Q_4^2}\end{aligned}$$

These equations are listed in Appendix C.

Aside from this, we will also need to translate our base coordinates and final coordinates since we are working only from Frame 2 (*Shoulder*) to Frame 4 (*Wrist 1*). To translate the coordinates, we use the same technique we did in determining the coordinates for θ_1,

$$\mathbf{Q_{FK}} = \mathbf{Q_1} * \mathbf{Q_2} * \mathbf{Q_3} * \mathbf{Q_4} * \mathbf{Q_5} * \mathbf{Q_6} \quad = \mathbf{Q_6^1} \tag{4.29}$$

$$\mathbf{Q_1^{-1}} * \mathbf{Q_{FK}} = \mathbf{Q_2} * \mathbf{Q_3} * \mathbf{Q_4} * \mathbf{Q_5} * \mathbf{Q_6} \quad = \mathbf{Q_6^2} \tag{4.30}$$

$$\mathbf{Q_1^{-1}} * \mathbf{Q_{FK}} * \mathbf{Q_6^{-1}} = \mathbf{Q_2} * \mathbf{Q_3} * \mathbf{Q_4} * \mathbf{Q_5} \quad = \mathbf{Q_5^2} \tag{4.31}$$

$$\mathbf{Q_1^{-1}} * \mathbf{Q_{FK}} * \mathbf{Q_6^{-1}} * \mathbf{Q_5^{-1}} = \mathbf{Q_2} * \mathbf{Q_3} * \mathbf{Q_4} \quad = \mathbf{Q_4^2} \tag{4.32}$$

Note that we can already completely form $\mathbf{Q_1}$, $\mathbf{Q_5}$, and $\mathbf{Q_6}$. For $\mathbf{Q_4}$, we only need to revert its translation. To do this, we use its inverse $\mathbf{Q_4^{-1}}$ in the same way we did in the previous translation. We have,

$$\mathbf{Q_4}^{-1} = ([\bar{c}_5 + \bar{s}_5 k], < d_6 i >)$$

And we revert the translation by,

$$\begin{aligned}(\mathbf{Q_4^2} * \mathbf{Q_4^{-1}})_\mathbf{p} &= (\mathbf{Q_2} * \mathbf{Q_3} * \mathbf{Q_4} * \mathbf{Q_4^{-1}})_\mathbf{p} \\ &= (\mathbf{Q_2} * \mathbf{Q_3})_\mathbf{p} \\ &= (\mathbf{Q_3^2})_\mathbf{p}\end{aligned} \tag{4.33}$$

We get the coordinates by the translation part of this product,

$$\begin{aligned}(\mathbf{Q_4^2} * \mathbf{Q_4^{-1}})_\mathbf{p} &= < \mathbf{q_4^2} * \mathbf{p_4^{-1}} * (\mathbf{q_4^2})^{-1} + \mathbf{p_4^2} > \\ &= < x_4 i + y_4 j + z_4 k >\end{aligned} \tag{4.34}$$

where $< x_4, y_4, z_4 >$ are the coordinates in Frame 4 with respect to Frame 2.

Deriving Joint Angle θ_3 (*Elbow*)

To derive a formula for θ_3, we make use of the equations (C.19) and (C.21) in the *Shoulder* to *Wrist 1* Transformation along with the previously computed coordinates $< x_4, y_4, z_4 >$.

$$\underline{\mathbf{M_{25} = N_{25}}}$$

$$a_3 \cos(\theta_2 + \theta_3) + a_2 \cos\theta_2 = x_4 \tag{4.35}$$

$$\underline{\mathbf{M_{27} = N_{27}}}$$

$$a_3 \sin(\theta_2 + \theta_3) + a_2 \sin\theta_2 = z_4 \tag{4.36}$$

Squaring these two equations and adding them will give us,

$$a_3^2 + 2a_2a_3 \cos\theta_3 + a_2^2 = x_4^2 + z_4^2 \tag{4.37}$$

Now, we can directly compute for θ_3 by,

$$\theta_3 = \pm acos\left(\frac{x_4^2 + z_4^2 - a_3^2 - a_2^2}{2a_2a_3}\right) \tag{4.38}$$

We consider two values for θ_3 which corresponds to the *Elbow* joint rotation of either being up or down.

Deriving Joint Angle θ_2 (*Shoulder*)

Since we already have θ_3, we can compute for θ_2 using equations (C.12) and (C.14) also in the *Shoulder* to *Wrist 1* Transformation using the same coordinates.

$$\underline{\mathbf{M_{35} = N_{35}}}$$

$$a_3 \cos\theta_3 = -a_2 + x_4 \cos\theta_2 + z_4 \sin\theta_2 \tag{4.39}$$

$$\underline{\mathbf{M_{37} = N_{37}}}$$

$$a_3 \sin\theta_3 = -x_4 \sin\theta_2 + z_4 \cos\theta_2 \tag{4.40}$$

And using similar manipulations, we can directly compute for θ_2 by,

$$\begin{aligned}\theta_2 = atan2(&z_4(a_2 + a_3\cos\theta_3) - x_4a_3\sin\theta_3,\\ &x_4(a_2 + a_3\cos\theta_3) + z_4a_3\sin\theta_3)\end{aligned} \tag{4.41}$$

We consider one value for θ_2 which corresponds to the *Shoulder* rotation.

Deriving Joint Angle θ_4 (*Wrist 1*)

Given θ_3 and θ_2, we can compute for θ_4 using equations (C.1) and (C.3) and the rotation part of the transformation $\mathbf{Q_4^2}$ given by $\mathbf{q_4} = w_4 + a_4 i + b_4 j + c_4 k$.

$$\underline{\mathbf{M_{41}} = \mathbf{N_{41}}}$$

$$\cos\left(\frac{\theta_4}{2}\right) = w_4 \cos\left(\frac{\theta_2+\theta_3}{2}\right) - b_4 \sin\left(\frac{\theta_2+\theta_3}{2}\right) \tag{4.42}$$

$$\underline{\mathbf{M_{43}} = \mathbf{N_{43}}}$$

$$-\sin\left(\frac{\theta_4}{2}\right) = w_4 \sin\left(\frac{\theta_2+\theta_3}{2}\right) + b_4 \cos\left(\frac{\theta_2+\theta_3}{2}\right) \tag{4.43}$$

We can straightforwardly compute for θ_4 by,

$$\theta_4 = -2atan2(w_4 \bar{s}_{(2+3)} + b_4 \bar{c}_{(2+3)}, w_4 \bar{c}_{(2+3)} - b_4 \bar{s}_{(2+3)}) \tag{4.44}$$

We consider one value for θ_4 which corresponds to the *Wrist 1* joint rotation.

4.3.5 Joint Angles Solutions

Using the equations (4.11), (4.13), (4.28), (4.38), (4.41) and (4.44) will allow us to compute for an angle combination $(\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6)$ that will give us our desired transformation $\mathbf{Q_6^1}$. Each of our joint angles from these equations has the following number of possible values,

$$\theta_1 \times 2 \qquad \theta_2 \times 1 \qquad \theta_3 \times 2$$

$$\theta_4 \times 1 \qquad \theta_5 \times 2 \qquad \theta_6 \times 1$$

Therefore, there are a total of $2 \times 1 \times 2 \times 1 \times 2 \times 1 = 8$ possible joint angle combinations. It is still necessary to do some filtration of these values in our practical programming application because it can still happen that only some of these combinations will give us our exact desired transformation. In this study, it is done by using the principle of forward kinematics. Now, we have our input as the joint angles from our current combination and we get their corresponding transformation,

$$\mathbf{Q_1}(\theta_1) * \mathbf{Q_2}(\theta_2) * \mathbf{Q_3}(\theta_3) * \mathbf{Q_4}(\theta_4) * \mathbf{Q_5}(\theta_5) * \mathbf{Q_6}(\theta_6) = \mathbf{Q_{FK}} \tag{4.45}$$

We check if this transformation is equivalent to our desired transformation where we also need to consider some small discrepancies due to numerical errors. This procedure is demonstrated in our programming application.

4.4 Practical Programming Application

The Python programming application for deriving the forward and inverse kinematics of UR3 is presented in this section.

4.4.1 Denavit-Hartenberg Parameters

The Denavit-Hartenberg Parameters of UR3 is given in the following table:

UR3				
Kinematics	theta [rad]	a [m]	d [m]	alpha [rad]
Joint 1	0	0	0.1519	π/2
Joint 2	0	-0.24365	0	0
Joint 3	0	-0.21325	0	0
Joint 4	0	0	0.11235	π/2
Joint 5	0	0	0.08535	-π/2
Joint 6	0	0	0.0819	0

Table 4.2: Denavit-Hartenberg Parameters for UR3 from [20]

These parameters are described by the following diagram:[5]

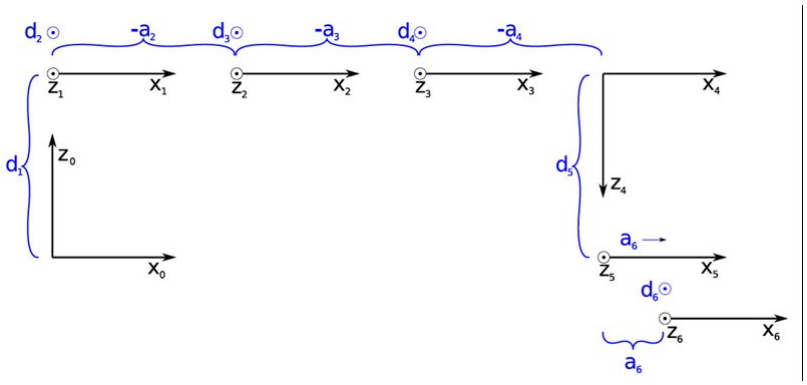


Figure 4.7: UR3 DH-parameters Diagram from [20]

The setup of the DH parameters in the program follows the order in Table 4.1 using the values given by Table 4.2.

```
# UR3 Denavit-Harteberg Parameters
d = [0.1519,0,0,0.11235,0.08535,0.0819]
a = [0,0,-0.24365,-0.21325,0,0]
alpha = [0,pi/2,0,0,pi/2,-pi/2]
```

[5]The use and derivation of the Denavit-Hartenberg parameters can be found in [23]

4.4.2 Python Programming Application

We use the **urx** python library to communicate and control the UR3 robot.

```
import urx
from urx import urrobot
rob = urx.Robot("147.229.132.249")   #IP Address of UR3
```

The IP address of the UR3 robot can be seen in the 'About' tab in the UR3 Polyscope Graphical User Interface.

Dual Quaternion Class

We first define our dual quaternion object. As we have described, it consists of two quaternions where the first quaternion represents the rotation and the second quaternion represents the translation. The **pyquaternion** library is used to define these quaternion objects and we created a class for the Dual Quaternion following the properties we listed in the quaternion chapter. The multiplication of dual quaternions is also defined in this class following the properties we listed in the dual quaternion chapter.

```
class DualQuaternion:
   # Initializes a Dual Quaternion
   def __init__(self, q=Quaternion(), p=0):
      self.q = q
      self.p = p

   # Dual Quaternion Multiplication
   def mult(self, dq):
      # For the rotation part
      q1 = self.q
      q2 = dq.q

      # For the translation part
      p1 = self.p
      p2 = dq.p

      return DualQuaternion(q1*q2, q1*p2*q1.conjugate+p1)

   # Checks equality of Dual Quaternions
   def equals(self,dq):
      # For the rotation part
      q1 = self.q.vector
      q2 = dq.q.vector

      # For the translation part
      p1 = self.p.vector
      p2 = dq.p.vector

```

```
29      # Checks the equality
30      print("\nComparing Dual Quaternions:")
31      print("Rotation:", q1,"~",q2)
32      print("Translation: ",p1,"~",p2)
33
34      if((np.allclose(p1,p2,atol=1e-03)
35          or np.allclose(p2,p1,atol=1e-03))
36         and (np.allclose(q1,q2,atol=1e-03)
37              or np.allclose(q2,q1,atol=1e-03))):
38         return True
39      return False
40
41   # Transforms a point in the space
42   def transform(self, pose):
43      # Rotated Pose
44      rotPose = self.q.rotate(pose)
45      print("\nAfter Rotation:",rotPose)
46
47      # Translated Pose
48      transPose = self.p.vector + rotPose
49      print("After Translation:",transPose)
50
51      return transPose
```

In rotating a vector, we used the **pyquaternion** method **rotate** since it also normalizes the quaternion before it applies the rotation. This is done in the following codes in the **pyquaternion** library:

```
1     def rotate(self, vector):
2         """Rotate a 3D vector by the rotation stored in the
    Quaternion object.
3
4         Params:
5             vector: A 3-vector specified as any ordered sequence
     of 3 real numbers corresponding to x, y, and z values.
6                 Some types that are recognised are: numpy arrays
    , lists and tuples.
7                 A 3-vector can also be represented by a
    Quaternion object who's scalar part is 0 and vector part is
    the required 3-vector.
8                 Thus it is possible to call `Quaternion.rotate(q
    )` with another quaternion object as an input.
9
10        Returns:
11            The rotated vector returned as the same type it was
    specified at input.
12
13        Raises:
14            TypeError: if any of the vector elements cannot be
    converted to a real number.
```

```
15              ValueError: if `vector` cannot be interpreted as a
    3-vector or a Quaternion object.
16
17          """
18          if isinstance(vector, Quaternion):
19              return self._rotate_quaternion(vector)
20          q = Quaternion(vector=vector)
21          a = self._rotate_quaternion(q).vector
22          if isinstance(vector, list):
23              l = [x for x in a]
24              return l
25          elif isinstance(vector, tuple):
26              l = [x for x in a]
27              return tuple(l)
28          else:
29              return a
30
31      def _rotate_quaternion(self, q):
32          """Rotate a quaternion vector using the stored rotation.
33
34          Params:
35              q: The vector to be rotated, in quaternion form (0 +
     xi + yj + kz)
36
37          Returns:
38              A Quaternion object representing the rotated vector
    in quaternion from (0 + xi + yj + kz)
39          """
40          self._normalise()
41          return self * q * self.conjugate
42
43      def _normalise(self):
44          """Object is guaranteed to be a unit quaternion after
    calling this
45          operation UNLESS the object is equivalent to Quaternion
    (0)
46          """
47          if not self.is_unit():
48              n = self.norm
49              if n > 0:
50                  self.q = self.q / n
```

We also note than in the previous code, we defined how to determine if two dual quaternions are approximately equal. We used the **allclose** method of the **numpy** library that checks if two arrays are element-wise equal within a tolerance. We do this for the inverse kinematics when we want to filter out the resulting angle combinations by their corresponding transformations and we must consider the numerical errors that may arise during computations.

Forward Kinematics

For the forward kinematics, our goal is to determine the final transformation (orientation and position) of the end-effector of the robot given its joints' angle rotations. First, we need to get the current joint angles of our robot. We do this by using the method **getj** from the **urx** library.

```
# Gets the current joint angles
jangles = rob.getj()
```

This returns the value of the six joint angles starting from the *Base* to *Wrist 3*. We also define two approaches of deriving a dual quaternion. One is by the axis-angle representation (AA) and by the transformation matrix (TM). We define different initial TCP coordinates for each of these approaches.

```
# Set TCP initial position in space based on the corresponding frame
TCP_endf = np.array([0,0,0.2]) # based on end frame (0,0,1)
TCP_basef = np.array([0,-0.2,0]) # based on base frame (0,-1,0)
```

The reason for this is for the first approach which is by AA, we must consider the TCP coordinates with respect to the *Base* frame. And for the TM approach, we must consider the TCP coordinates with respect to its frame.

The method **get forward kinematics** now calls the derivation of the dual quaternions for each six joints based on the approach chosen and computes the final dual quaternion representing the final transformation by multiplying them following the order of transformation (joints).

```
# Gets the current pose using Forward Kinematics
def get_forward_kinematics(ans,jangles):

    print("\nCurrent Joint Angles:", jangles)

    # Derives the corresponding Dual Quaternions for each joints from their Angle-Axis Representation
    if(ans == "AA"):
        print("\nDeriving Forward Kinematics from Angle-Axis Representation...")
        DualQuatList = derive_dual_quat_from_AA(jangles)
        initial_pose = TCP_basef
    # Derives the corresponding Dual Quaternions for each joints from their Transformation Matrices
    elif (ans == "TM"):
        print("\nDeriving Forward Kinematics from Transformation Matrix...")
        DualQuatList = derive_Dual_Quaternion_from_TM(jangles)
        initial_pose = TCP_endf
    else:
        print("Wrong Input!")
```

```
18          return
19
20      # Computes the Final Dual Quaternion by multiplying all the
        Dual Quaternions
21      FKDualQuat = DualQuaternion()
22      for dq in DualQuatList:
23          FKDualQuat = FKDualQuat.mult(dq)
24
25      print("\nForward Kinematics (by Dual Quaternion):\n([",
        FKDualQuat.q,"],<", FKDualQuat.p,">)")
26
27      # Transform initial pose by Quaternion-Vector Multiplication
28      print("\nInitial TCP Coordinates (at zero pose):",
        initial_pose)
29      curr_pose = FKDualQuat.transform(initial_pose)
30      print("\nTransformed TCP Coordinates:\n", np.round(curr_pose
        ,3))
31
32      return FKDualQuat
```

The output of this method is the current pose or the TCP coordinates of the robot after it has been transformed (rotated and translated) and with respect to the *Base* frame.

Dual Quaternions from Axis-Angle Representation

The derivation of the Forward Kinematics by the Axis-Angle approach is done in the method **derive dual quat from AA**:

```
1  # Derive Dual Quaternion from Angle and Axis of Rotation
2  def derive_dual_quat_from_AA(jangles):
3      # Gets joint angles from current pose
4      theta = jangles
5
6      # Set translation for each joints
7      trans = ([[0, 0, d[0]],                                    # P1
8                [a[2]*cos(theta[1]), 0, a[2]*sin(theta[1])],     # P2
9                [a[3]*cos(theta[2]), -d[3], a[3]*sin(theta[2])],# P3
10               [d[4]*sin(theta[3]), 0, -d[4]*cos(theta[3])],    # P4
11               [-d[5]*sin(theta[4]), -d[5]*cos(theta[4]), 0],   # P5
12               [0,0,0]])                                        # P6
13
14     # Builds the Dual Quaternions
15     DQList = []
16     alp = 0
17
18     for i in range(6):
19         # Builds the dual quaternion
20         qscalar = cos(theta[i]/2)
```

```
21          alp += alpha[i]
22          qvector = np.multiply(sin(theta[i]/2), [0, -sin(alp), cos(
    alp)])
23          quatRot = np.concatenate(([qscalar],qvector))
24
25          DualQuat = DualQuaternion(Quaternion(quatRot),Quaternion(
    vector=trans[i]))
26          print("Dual Quaternion:([",DualQuat.q,"],<",DualQuat.p,">)
    ")
27
28          # Store the Dual Quaternion to the list
29          DQList.append(DualQuat)
30
31      return DQList
```

To compute for the final transformation, we first define the translation for each joints. This corresponds to the translation part of our dual quaternions in which we use the Denavit-Hartenberg parameters **(a, d,** θ**)**. Then, we build the rotation quaternions for all six joints using the angles from the robot and its corresponding translation quaternion by the derived translations. The variable **alp** follows the **alpha** parameter in the DH parameters and we use this in determining the axis of rotation. This can have values 0 or $\frac{\pi}{2}$ which will make either of the terms **-sin(alp)** and **cos(alp)** zero or ± 1. Then, we create the dual quaternion object by combining these two quaternions. Lastly, we store these dual quaternions to a list and returns it to the previous method which performs the transformation.

Dual Quaternions from Transformation Matrix

Another approach to derive forward kinematics can be by a transformation matrix. This is done in the method **derive dual quat from TM**. The usual convention of deriving forward kinematics is by using rotation matrices and translation vectors which are combined to form a (4x4) transformation matrix. From this transformation matrix, we can convert it to a dual quaternion by converting the rotation matrices (3x3) into unit quaternions and the translation vectors (3x1) as a pure quaternion. The rotation matrix to quaternion conversion was discussed in Section 3.4.2. For the transformation matrix, we follow its derivation in [3] and [16].

```
# Gets the Dual Quaternions for each joints
def derive_Dual_Quaternion_from_TM(jangles):
    print("Computing Transformation Matrix by Denavit-Hartenberg
    Parameters...")
    # Gets joint angles from current pose
    theta = jangles
    # Gets transformation matrices for each 6 frames
```

```
7     T = np.zeros(shape=(6,4,4))
8
9     # Builds the Dual Quaternions
10    DQList = []
11
12    for i in range(6):
13        T[i] = ([[cos(theta[i]), -sin(theta[i]), 0, a[i]],
14                 [sin(theta[i])*cos(alpha[i]), cos(theta[i])*cos(
      alpha[i]), -sin(alpha[i]), -sin(alpha[i])*d[i]],
15                 [sin(theta[i])*sin(alpha[i]), cos(theta[i])*sin(
      alpha[i]), cos(alpha[i]), cos(alpha[i])*d[i]],
16                 [0,0,0,1]])
17
18        print("\nTransformation Matrix for Frame",i,"to", i+1,"\n"
      ,T[i])
19
20        # Derives the Dual Quaternion from the Transformation
      Matrix
21        DualQuat = DualQuaternion(Quaternion(matrix = T[i]),
      Quaternion(vector = T[i][0:3,3]))
22        print("Dual Quaternion:([",DualQuat.q,"],<",DualQuat.p,">)
      ")
23
24        # Store the Dual Quaternion to the list
25        DQList.append(DualQuat)
26
27    return DQList
```

Same as the previous approach, we store the dual quaternions into a list and pass it to the first method that performs the transformation. The final dual quaternion representing the final transformation will be different from the other approach. This is because in the derivation by AA, we are basing the dual quaternions only on the base frame while for the TM approach, each dual quaternions are derived with respect to their own frames. In any of these approaches, the final TCP coordinates will always be the same.

Practical Application

For the application, we use a method **record pose** that takes the current pose of the robot and records it to a text file. The coefficients of this dual quaternion derived from the robot's joint angles is then stored into a text file. Multiple poses can be recorded into the text file. Aside from this, the forward kinematics is also used to filter the angle combinations derived in the inverse kinematics by checking if their transformations equates to the desired transformation written in the text file.

Inverse Kinematics

For the inverse kinematics, our goal is to determine the joint angles that will give us our desired pose (rotation and translation). Thus, we are now given the final transformation and we use it to derive the values of the corresponding six joint angles. We use the **get inverse kinematics** method to perform the derivation. We begin with defining the coordinates at Frame 6 (*Wrist 3*) and the corresponding coordinates at Frame 5 (*Wrist 2*).

```
# Outputs joint angles from Inverse Kinematics
def get_inverse_kinematics(desired_pose):
    print("\nDesired Pose:",desired_pose.q, desired_pose.p);
    # Desired Transformation from Frame 1 to 6
    DQ16 = desired_pose
    # Rotation Part
    w6 = DQ16.q[0]
    a6 = DQ16.q[1]
    b6 = DQ16.q[2]
    c6 = DQ16.q[3]
    # Translation Part
    x6 = DQ16.p[1];
    y6 = DQ16.p[2];
    z6 = DQ16.p[3];

    # Remove Translation from Frame 5
    DQ5i = DualQuaternion(p=Quaternion(vector=[0,d[5],0]))
    DQ15p = DQ16.mult(DQ5i)
    x5 = DQ15p.p[1]
    y5 = DQ15p.p[2]
    z5 = DQ15p.p[3]
```

The translation from *Wrist 3* to *Wrist 2* was demonstrated in Section 4.3.3.

Computation for *Base* (θ_1)

To compute for θ_1, we used the equation (4.11):

$$\theta_1 = atan2(-x_5, y_5) \pm atan2(\sqrt{x_5^2 + y_5^2 - d_4^2}, -d_4)$$

This is expressed by the following code,

```
    # Compute possible values for Theta 1
    theta1=[]
    t11 = atan2(-x5,y5)+atan2(sqrt(x5**2+y5**2-d[3]**2),-d[3])
    t12 = atan2(-x5,y5)-atan2(sqrt(x5**2+y5**2-d[3]**2),-d[3])
    # (Base) Theta 1: 2 Values
    theta1.append(t11)
    theta1.append(t12)
```

This gives at most two possible values for θ_1.

Computation for *Wrist 2* (θ_5)

To compute for θ_5, we used the equation (4.13):

$$\theta_5 = \pm acos\left(\frac{x_6 \sin\theta_1 - y_6 \cos\theta_1 - d_4}{d_6}\right)$$

This is expressed by the following code,

```
# Compute possible values for Theta 5
theta15 = []
for t1 in theta1: # For each values of Theta 1
    # Evaluate expression for Theta 5
    exp5 = (x6*sin(t1)-y6*cos(t1)-d[3])/d[5]
    # Checks if within the domain for arccos
    if (exp5<=1 and exp5>=-1):
        t5 = acos(exp5)
    elif (round(exp5)<=1 and round(exp5)>=-1):
        t5 = acos(round(exp5))
    else: continue
    # (Wrist 2) Theta 5: 2 Values
    theta15.append([t1,t5])
    theta15.append([t1,-t5])
```

We consider that *arccos* has domain of only $[-1, 1]$. Therefore, to prevent runtime errors, we round-off the values and check if they are now inside this domain. This gives again two possible values for θ_5.

Computation for *Wrist 3* (θ_6)

To compute for θ_6, we used the equation (4.28):

$$\theta_6 = 2atan2(\gamma - \beta, \alpha + \delta)$$

where,

$$\alpha = \left(\frac{w\bar{c}_1 + c\bar{s}_1}{\bar{c}_5}\right) \quad \beta = \left(\frac{a\bar{c}_1 + b\bar{s}_1}{\bar{s}_5}\right) \quad \gamma = \left(\frac{a\bar{s}_1 - b\bar{c}_1}{\bar{c}_5}\right) \quad \delta = \left(\frac{w\bar{s}_1 - c\bar{c}_1}{\bar{s}_5}\right)$$

This is expressed by the following code,

```
# Compute possible values for Theta 6
theta156 = []
for t1t5 in theta15:
    t1 = t1t5[0]
    t5 = t1t5[1]

    # Avoids singularity for Theta 6
    if(t5==0):
```

```
            t156 = [t1,t5,0]
            if t156 not in theta156:
                theta156.append(t156)
            continue

        # Evaluate expression for Theta 6
        tm1 = ((w6*cos(t1/2)+c6*sin(t1/2))/cos(t5/2))
        tm2 = ((a6*cos(t1/2)+b6*sin(t1/2))/sin(t5/2))
        tm3 = ((a6*sin(t1/2)-b6*cos(t1/2))/cos(t5/2))
        tm4 = ((w6*sin(t1/2)-c6*cos(t1/2))/sin(t5/2))

        # (Wrist 3) Theta 6: 1 Value
        t6 = 2*atan2(tm3-tm2,tm1+tm4)
        theta156.append([t1,t5,t6])
```

This gives only one possible value for θ_6.

Now, in order to define the next angles. We need to perform a translation again. We need to get the transformation from Frame 2 (*Shoulder*) to Frame 4 (*Wrist 1*), therefore we need $\mathbf{Q_4^2}$. This was discussed in Section 4.3.4.

```
# Derive transformation from Frame 2 to Frame 4
DQ24f = []
for t156 in theta156:
    t1 = t156[0]
    t5 = t156[1]
    t6 = t156[2]
    # Derives the Inverse Dual Quaternions of Joints 1, 5, 6
    q1 = np.concatenate(([cos(t1/2)], np.multiply(sin(t1/2),
[0, 0, -1])))
    DQ1i = DualQuaternion(Quaternion(q1),Quaternion(vector
=[0,0,-d[0]]))
    q5 = np.concatenate(([cos(t5/2)], np.multiply(sin(t5/2),
[0, 0, 1])))
    DQ5i = DualQuaternion(Quaternion(q5),Quaternion(vector=[0,
d[5],0]))
    q6 = np.concatenate(([cos(t6/2)], np.multiply(sin(t6/2),
[0, 1, 0])))
    DQ6i = DualQuaternion(Quaternion(q6),Quaternion(vector
=[0,0,0]))
    # Derives Transformation from Frame 2 to Frame 6
    DQ26 = DQ1i.mult(DQ16)
    # Derives Transformation From Frame 2 to Frame 5
    DQ25 = DQ26.mult(DQ6i)
    # Derives Transformation from Frame 2 to Frame 4
    DQ24 = DQ25.mult(DQ5i)
    # Remove Translation from Frame 4
    DQ4i = DualQuaternion(p=Quaternion(vector=[0,0,d[4]]))
    DQ24f.append([DQ24.mult(DQ4i),t156])
```

For each angle combinations of θ_1, θ_5 and θ_6, we derive their corresponding dual quaternion inverses $\mathbf{Q_1^{-1}}, \mathbf{Q_5^{-1}}$ and $\mathbf{Q_6^{-1}}$. We use these inverses to derive the transformation $\mathbf{Q_4^2}$. Then, we revert the translation done by $\mathbf{Q_4}$ by also using its inverse $\mathbf{Q_4^{-1}}$. The transformations are then stored into a list along with the set of angles $[\theta_1, \theta_5, \theta_6]$ that generated them. All of these will be used in the computation for the remaining joint angles.

Computation for *Elbow* (θ_3)

To compute for θ_3, we used the equation (4.38):

$$\theta_3 = \pm acos\left(\frac{x_4^2 + z_4^2 - a_3^2 - a_2^2}{2a_2a_3}\right)$$

This is expressed by the following code,

```
# Compute possible values for Theta 3
theta1356q=[]
for dq in DQ24f:
    w4 = dq[0].q[0]
    b4 = dq[0].q[2]
    x4 = dq[0].p[1]
    y4 = dq[0].p[2]
    z4 = dq[0].p[3]
    q4 = [w4,b4,x4,y4,z4]

    t1 = dq[1][0]
    t5 = dq[1][1]
    t6 = dq[1][2]

    # Evaluate expression for Theta 3
    exp3 = ((x4**2+z4**2)-a[3]**2-a[2]**2)/(2*a[2]*a[3])

    # Checks if within the domain for arccos
    if (exp3<=1 and exp3>=-1):
        t3 = acos(exp3)
    elif (round(exp3)<=1 and round(exp3)>=-1):
        t3 = acos(round(exp3))
    else: continue

    # (Elbow) Theta 3: 2 Values
    theta1356q.append([t1,t3,t5,t6,q4])
    theta1356q.append([t1,-t3,t5,t6,q4])
```

Same with θ_5, we also consider the results to be inside the domain of *arccos*. This gives at most two possible values for θ_3.

Computation for *Shoulder* (θ_2)

To compute for θ_2, we used the equation (4.41):

$$\theta_2 = atan2(z_4(a_2 + a_3\cos\theta_3) - x_4 a_3 \sin\theta_3, x_4(a_2 + a_3\cos\theta_3) + z_4 a_3 \sin\theta_3)$$

This is expressed by the following code,

```
# Compute possible values for Theta 2
theta12356q=[]
for t1356q in theta1356q:
    t1 = t1356q[0]
    t3 = t1356q[1]
    t5 = t1356q[2]
    t6 = t1356q[3]
    x4 = t1356q[4][2]
    z4 = t1356q[4][4]
    # Evaluate expression for Theta 2
    t2 = atan2((z4*(a[2]+a[3]*cos(t3))-x4*a[3]*sin(t3)),(x4*(a
[2]+a[3]*cos(t3))+z4*a[3]*sin(t3)))
    # (Shoulder) Theta 2: 1 Value
    theta12356q.append([t1,t2,t3,t5,t6,t1356q[4]])
```

This gives only one possible value for θ_2.

Computation for *Wrist 1* (θ_4)

To compute for θ_4, we used the equation (4.44):

$$\theta_4 = -2atan2(w_4\bar{s}_{(2+3)} + b_4\bar{c}_{(2+3)}, w_4\bar{c}_{(2+3)} - b_4\bar{s}_{(2+3)})$$

This is expressed by the following code,

```
# Compute possible values for Theta 4
theta123456=[]
for t12356q in theta12356q:
    t1 = t12356q[0]
    t2 = t12356q[1]
    t3 = t12356q[2]
    t5 = t12356q[3]
    t6 = t12356q[4]
    w4 = t12356q[5][0]
    b4 = t12356q[5][1]
    # Evaluate expression for Theta 4
    t4 = -2*atan2((w4*sin((t2+t3)/2))+(b4*cos((t2+t3)/2)),(w4*
cos((t2+t3)/2))-(b4*sin((t2+t3)/2)))
    # (Wrist 1) Theta 4: 1 Value
    theta123456.append([t1,t2,t3,t4,t5,t6])
```

This gives only one possible value for θ_4.

Filtration of Angle Combinations and Practical Application

To filter and verify the resulting angle combinations, we derive their corresponding forward kinematics which represents the final transformation done by these angles. If this transformation is not approximately equal to our desired transformation, then we filter out these angles and leave only those who satisfy our equality within the tolerance of $1e-3$. This is done below.

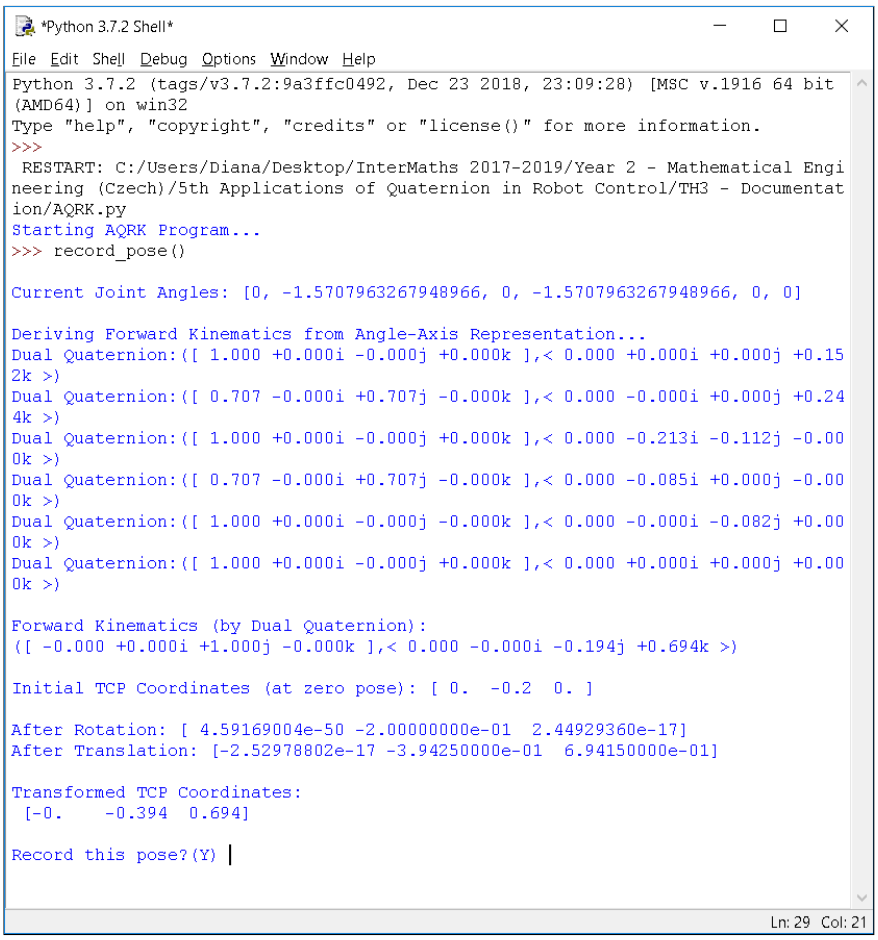
```
    # Get the corresponding Final Dual Quaternion based on the
    combination of angles
    jangles=[]
    print("\nSet of Possible Joint Angles:\n",theta123456)
    for theta in theta123456:
        FinalDQ = get_forward_kinematics("AA",theta)

        print("\nComparing Dual Quaternions: ")
        print("Desired Pose:\n([", DQ16.q,"],<", DQ16.p,">)")
        print("Current Pose:\n([", FinalDQ.q,"],<",FinalDQ.p,">)")

        if(DQ16.equals(FinalDQ)):
            print("PASSED criteria!")
            jangles.append(theta)
        else: print("FAILED criteria!")

    print("\nFinal Joint Angles:",jangles)

    return jangles
```

For the application, we first build the dual quaternion represented by the coefficients from the text file generated by the **record pose** method. Then, we apply the inverse kinematics and use the first resulting joint angles as the input for the movement of the robot. (An added criteria can also be made in order to find the optimal joint angles combination)

```
# Moves robot from given Dual Quaternion Transformation
def move_robot(dq_coeff):
        # Takes the dual quaternion coefficients and create the
    dual quaternion
        DQ_desired_pose = DualQuaternion(Quaternion(array=dq_coeff
    [0:4]),Quaternion(array=dq_coeff[4:8]))

        # Applies Inverse Kinematics to derive the joint angles
        joint_angles = get_inverse_kinematics(DQ_desired_pose)

        print("\nThere are ",len(joint_angles) ," possible poses!"
    );

        # Moves the robot given the joint angles
        rob.movej(joint_angles[0], acc=1, vel=1, wait=False)
```

4.5 Verification of Results

The program begins with the **record pose** function:



This is the base pose of the robot given by the following set of angles:

$$\text{Base: } \theta_1 = 0 \qquad \text{Shoulder: } \theta_2 = \frac{-\pi}{2} \qquad \text{Elbow: } \theta_3 = 0$$

$$\text{Wrist 1: } \theta_4 = \frac{-\pi}{2} \qquad \text{Wrist 2: } \theta_5 = 0 \qquad \text{Wrist3: } \theta_6 = 0$$

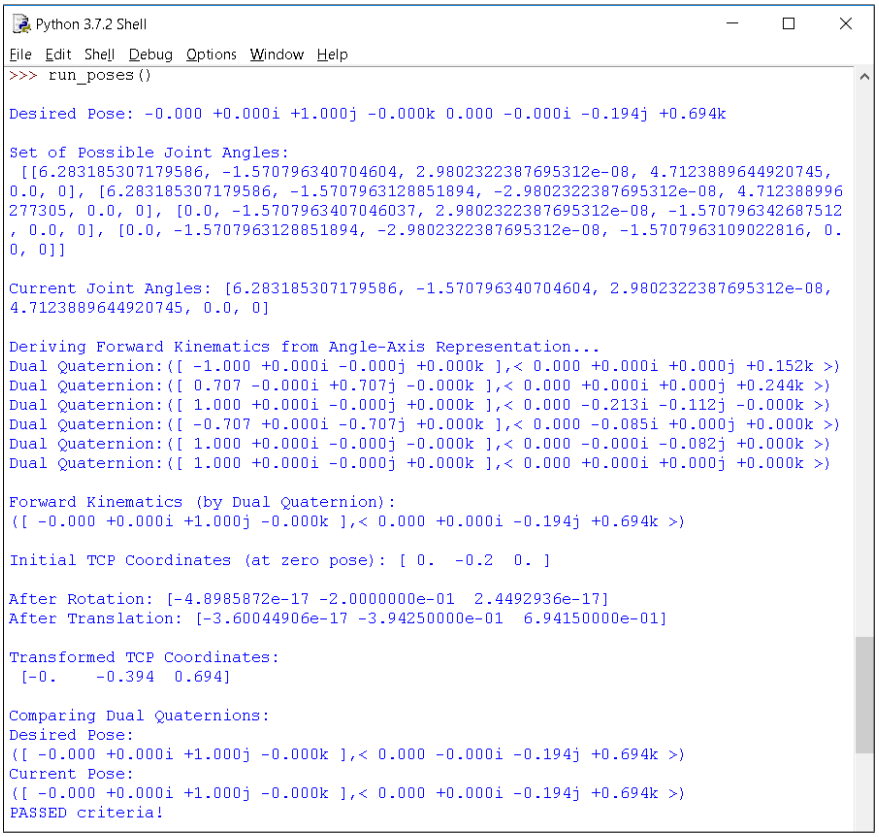
The dual quaternions above are derived with respect to these angles along with their axis of rotation and their corresponding translations. This can be verified in Section 4.2.1. These are then multiplied and the product is set as the dual quaternion representing the final transformation. The tool center point of the end-effector is then transformed from the coordinates (0, -0.2, 0) to (0, -0.394, 0.694) with respect to the base frame.

The function **record pose** is set to use the axis-angle representation. For demonstration purposes, the derivation of forward kinematics from a transformation matrix can also be done by calling the **get forward kinematics** function setting the joint angles as parameters and TM as the mode of transformation. The implementation are demonstrated below:

```
>>> get_forward_kinematics("TM",rob.getj())

 Current Joint Angles: [0, -1.5707963267948966, 0, -1.5707963267948966, 0, 0]

 Deriving Forward Kinematics from Transformation Matrix...
 Computing Transformation Matrix by Denavit-Hartenberg Parameters...

Transformation Matrix for Frame 0 to 1
  [[ 1.     -0.      0.      0.    ]
  [ 0.      1.     -0.     -0.    ]
  [ 0.      0.      1.      0.1519]
  [ 0.      0.      0.      1.    ]]
Dual Quaternion:([ 1.000 +0.000i +0.000j +0.000k ],< 0.000 +0.000i -0.000j +0.152k >)

Transformation Matrix for Frame 1 to 2
  [[ 6.12323400e-17  1.00000000e+00  0.00000000e+00  0.00000000e+00]
  [-6.12323400e-17  3.74939946e-33 -1.00000000e+00 -0.00000000e+00]
  [-1.00000000e+00  6.12323400e-17  6.12323400e-17  0.00000000e+00]
  [ 0.00000000e+00  0.00000000e+00  0.00000000e+00  1.00000000e+00]]
Dual Quaternion:([ 0.500 +0.500i +0.500j -0.500k ],< 0.000 +0.000i -0.000j +0.000k >)

Transformation Matrix for Frame 2 to 3
  [[ 1.      -0.       0.      -0.24365]
  [ 0.       1.      -0.      -0.     ]
  [ 0.       0.       1.       0.     ]
  [ 0.       0.       0.       1.     ]]
Dual Quaternion:([ 1.000 +0.000i +0.000j +0.000k ],< 0.000 -0.244i -0.000j +0.000k >)

Transformation Matrix for Frame 3 to 4
  [[ 6.123234e-17  1.000000e+00  0.000000e+00 -2.132500e-01]
  [-1.000000e+00  6.123234e-17 -0.000000e+00 -0.000000e+00]
  [-0.000000e+00  0.000000e+00  1.000000e+00  1.123500e-01]
  [ 0.000000e+00  0.000000e+00  0.000000e+00  1.000000e+00]]
Dual Quaternion:([ 0.707 +0.000i +0.000j -0.707k ],< 0.000 -0.213i -0.000j +0.112k >)

Transformation Matrix for Frame 4 to 5
  [[ 1.00000000e+00 -0.00000000e+00  0.00000000e+00  0.00000000e+00]
  [ 0.00000000e+00  6.12323400e-17 -1.00000000e+00 -8.53500000e-02]
  [ 0.00000000e+00  1.00000000e+00  6.12323400e-17  5.22618022e-18]
  [ 0.00000000e+00  0.00000000e+00  0.00000000e+00  1.00000000e+00]]
Dual Quaternion:([ 0.707 +0.707i +0.000j +0.000k ],< 0.000 +0.000i -0.085j +0.000k >)

Transformation Matrix for Frame 5 to 6
  [[ 1.00000000e+00 -0.00000000e+00  0.00000000e+00  0.00000000e+00]
  [ 0.00000000e+00  6.12323400e-17  1.00000000e+00  8.19000000e-02]
  [-0.00000000e+00 -1.00000000e+00  6.12323400e-17  5.01492864e-18]
  [ 0.00000000e+00  0.00000000e+00  0.00000000e+00  1.00000000e+00]]
Dual Quaternion:([ 0.707 -0.707i +0.000j +0.000k ],< 0.000 +0.000i +0.082j +0.000k >)

Forward Kinematics (by Dual Quaternion):
([ 0.000 +0.000i +0.707j -0.707k ],< 0.000 +0.000i -0.194j +0.694k >)

Initial TCP Coordinates (at zero pose): [0.  0.  0.2]

After Rotation: [ 0.  -0.2  0. ]
After Translation: [ 0.      -0.39425  0.69415]

Transformed TCP Coordinates:
  [ 0.    -0.394  0.694]
```

The dual quaternion representing the final transformation in both of these approaches are clearly different but the transformed coordinates are the same. Note that we are also using different initial TCP coordinates.

The **record pose** function stores the poses into a file and the **run poses** function retrieves the coefficients of the dual quaternions stored into the file and generates the angles corresponding to these transformations. This is demonstrated below:



There were 4 possible angle combinations. This is further verified by checking if its corresponding transformation (represented by the dual quaternion) is approximately equal to our desired transformation as also seen above.

The final results at the end are given by,

```
Final Joint Angles: [[6.283185307179586, -1.570796340704604, 2.9802322387695312e-08, 4
.7123889644920745, 0.0, 0], [6.283185307179586, -1.5707963128851894, -2.98023223876953
12e-08, 4.712388996277305, 0.0, 0], [0.0, -1.5707963407046037, 2.9802322387695312e-08,
-1.570796342687512, 0.0, 0], [0.0, -1.5707963128851894, -2.9802322387695312e-08, -1.57
07963109022816, 0.0, 0]]

There are  4  possible pose(s)!
>>>
```

Therefore, all combinations were valid and all of them will transform the tool center point of our robot to the desired pose (same transformed TCP coordinates) which is the solution of the inverse kinematics.

Chapter 5

Conclusions and Recommendations

5.1 Conclusion

The thesis was able to successfully demonstrate the use of quaternion algebra in the representation of kinematics for robotic applications in particular with the 6-DOF Universal Robot UR3. Both its direct and inverse kinematics were derived using the principles of quaternions. These are contained in Section 4.2 *Forward Kinematics by Dual Quaternions* and Section 4.3 *Inverse Kinematics by Dual Quaternions*. A practical programming application was also developed shown in Section 4.4.2 *Python Programming Application* to confirm the accuracy of the derived kinematics.

Aside from this, based on our analyzation of results, we can also conclude the following statements:

1. Dual Quaternions present a more compact way of representing three-dimensional transformations. Thus, also offers a considerable advantage in terms of storage efficiency.

    (a) This is due to quaternions requiring only four elements in representing a rotation compared to the most commonly used rotation matrices which involves 9 elements.

    (b) For the transformation in three-dimensions which involves rotation and translation. A homogeneous transformation matrix consists of 16 elements on which four elements are trivial while a dual quaternion only needs 7 elements to represent the same transformation.

2. Performing multiple transformations by dual quaternions also has an advantage in terms of computational robustness and numerical accuracy especially in dealing with the kinematics of a robot.

   (a) Combining transformations using homogeneous transformation matrices involves multiplying 4x4 matrices which involves several arithmetic and trigonometric operations that may greatly decrease the numerical precision of the solution.

   (b) Performing quaternion multiplication is comparable to polynomial multiplication while also incorporating the multiplication laws for the quaternion units. This makes it more efficient, and also more numerically precise since it involves less arithmetic and trigonometric operations.

3. Quaternions also offer a great advantage in terms of normalization and ease of derivation.

   (a) Quaternions are easier to normalize than matrices. A rotation matrix has to be orthogonal and must have a determinant of 1. For quaternions, it has to be a unit quaternion which means it has to have a unity norm. This can be simply done by dividing all four elements of the quaternions by its magnitude.

   (b) Dual Quaternions are easily derived by using axis-angle representation in where we only need to determine the relationship of the current axis of rotation to the base coordinate frame.

4. Dual Quaternions are not susceptible to gimbal lock.

   (a) Although not discussed in this study, gimbal lock is a serious problem in dealing with rotations in three-dimensional space. Imagine three gimbals which represents rotations about each three axes; thus, represents the three degrees of freedom. These are defined as roll, pitch, and yaw. It happens when the outer and inner gimbals align with each other and loses one degree of freedom. This is due to the fact that the rotation that can be done by any of these two gimbals are now considered the same. Euler Angles is famously known as being susceptible to this rare occurrence.

   (b) Quaternions same as rotation matrices are known to be not susceptible to gimbal lock. This is due to them representing orientation as a value rather than three separate and related values.

5. The ease of representing robot kinematics by dual quaternions depends on the type of kinematics being derived and the robot's kinematic structure.

   (a) For forward kinematics, it is advisable to derive the rotations by an axis-angle approach compared to starting from a rotation matrix and then convert it to unit quaternions. In this way, there will clearly be less operations to be performed compared to the traditional homogeneous transformation.

   (b) For inverse kinematics, the difficulty of deriving formulas for the angles are still preserved in dual quaternions. This is generally dependent on the robot kinematic structure, as in our case. It pertains to the relationship of the transformation that can be done by each joints to the final transformation. In our case, we had to divide the transformations into three. However in some cases, it can happen that certain formulas can already be derived only by the equations generated in the final transformation.

   (c) In order to still apply the concept of quaternions in the inverse kinematics, we also had to analyze and compute several equations that can represent the transformations done by each joints. This makes it more tedious compared to the normal convention which is usually done using the geometric solution approach.

Hopefully, these advantages will be deemed sufficient for dual quaternions to be used more extensively not only in the robotics and computer graphics field but also in other areas that involves working in three-dimensional space.

5.2 Recommendation

For future research, it is recommended to find if there is a better and more efficient way of representing robot kinematics particularly for the inverse kinematics. For the Universal Robot UR3, the inverse kinematics solution can be further filtered by considering only a certain range of movements. Certain conditions can also be applied to determine the optimal solution that will give the desired pose. As this research focused only on the application of quaternions in robot kinematics, it can be further extended to its application in robot control in general. This can involve an added step of performing quaternion interpolation. This contributes in finding the optimal path for the trajectory of the robot. This will be good for future research since it is also one of the useful and well-known applications of quaternions.

Appendices

Appendix A

Universal Robot UR3 Technical Specifications

UR3

Performance

Repeatability	±0.1 mm / ±0.0039 in (4 mils)
Ambient temperature range	0-50° *
Power consumption	Min 90W, Typical 125W, Max 250W
Collaboration operation	15 advanced adjustable safety functions. TüV NORD Approved Safety Function Tested in accordance with: EN ISO 13849:2008 PL d

Specification

Payload	3 kg / 6.6 lbs
Reach	500 mm / 19.7 in
Degrees of freedom	6 rotating joints
Programming	Polyscope graphical user interface on 12 inch touchscreen with mounting

Movement

Axis movement robot arm	Working range	Maximum speed
Base	± 360°	± 180°/Sec.
Shoulder	± 360°	± 180°/Sec.
Elbow	± 360°	± 180°/Sec.
Wrist 1	± 360°	± 360°/Sec.
Wrist 2	± 360°	± 360°/Sec.
Wrist 3	Infinite	± 360°/Sec.
Typical tool		1 m/Sec. / 39.4 in/Sec.

Features

IP classification	IP64	
ISO Class Cleanroom	5	
Noise	70dB	
Robot mounting	Any	
I/O ports	Digital in	2
	Digital out	2
	Analog in	2
	Analog out	0
I/O power supply in tool	12 V/24 V 600 mA in tool	

Physical

Footprint	Ø 128mm
Materials	Aluminium, PP plastics
Tool connector type	M8
Cable length robot arm	6 m / 236 in
Weight with cable	11 kg /24.3 lbs

* The robot can work in a temperature range of 0-50°C. At high continuous joint speed, ambient temperature is reduced.

CONTROL BOX

Features

IP classification	IP20	
ISO Class Cleanroom	6	
Noise	<65dB(A)	
I/O ports	Digital in	16
	Digital out	16
	Analog in	2
	Analog out	2
I/O power supply	24V 2A	
Communication	TCP/IP 100Mbit, Modbus TCP, Profinet, EthernetIP	
Power source	100-240 VAC, 50-60 Hz	
Ambient temperature range	0-50°	

Physical

Control box size (WxHxD)	475mm x 423mm x 268mm / 18.7 x 16.7 x 10.6 in
Weight	15 kg / 33.1 lbs
Materials	Steel

TEACH PENDANT

Features

IP classification	IP20

Physical

Materials	Aluminium, PP
Weight	1,5 kg / 3.3 lbs
Cable length	4,5 m / 177 in

Appendix B

Transformation Products for *Base* to *Wrist 3*

This part shows the transformation products for the UR3 robot kinematic structure starting from Frame 1 (*Base*) to Frame 6 (*Wrist 3*).

We define the dual quaternion product $\mathbf{M}_i$ as:

$$\mathbf{M}_i = \mathbf{Q}_i * \mathbf{M}_{i+1} \ \textit{where} \ 1 \leq i \leq 6$$

$$\begin{aligned}
\mathbf{M}_6 &= \mathbf{Q}_6 \\
\mathbf{M}_5 &= \mathbf{Q}_5 * \mathbf{M}_6 = \mathbf{Q}_5 * \mathbf{Q}_6 \\
\mathbf{M}_4 &= \mathbf{Q}_4 * \mathbf{M}_5 = \mathbf{Q}_4 * \mathbf{Q}_5 * \mathbf{Q}_5 \\
\mathbf{M}_3 &= \mathbf{Q}_3 * \mathbf{M}_4 = \mathbf{Q}_3 * \mathbf{Q}_4 * \mathbf{Q}_5 * \mathbf{Q}_6 \\
\mathbf{M}_2 &= \mathbf{Q}_2 * \mathbf{M}_3 = \mathbf{Q}_2 * \mathbf{Q}_3 * \mathbf{Q}_4 * \mathbf{Q}_5 * \mathbf{Q}_6 \\
\mathbf{M}_1 &= \mathbf{Q}_1 * \mathbf{M}_2 = \mathbf{Q}_1 * \mathbf{Q}_2 * \mathbf{Q}_3 * \mathbf{Q}_4 * \mathbf{Q}_5 * \mathbf{Q}_6
\end{aligned}$$

We define the dual quaternion product $\mathbf{N}_i$ as follows:

$$\mathbf{N}_{i+1} = \mathbf{Q}_i^{-1} * \mathbf{N}_i \ \textit{where} \ 1 \leq i \leq 6$$

$$\begin{aligned}
\mathbf{N}_1 &= \mathbf{Q_{FK}} = ([w, < a, b, c >], < x, y, z >) \\
\mathbf{N}_2 &= \mathbf{Q}_1^{-1} * \mathbf{N}_1 = \mathbf{Q}_1^{-1} * \mathbf{Q_{FK}} \\
\mathbf{N}_3 &= \mathbf{Q}_2^{-1} * \mathbf{N}_2 = \mathbf{Q}_2^{-1} * \mathbf{Q}_1^{-1} * \mathbf{Q_{FK}} \\
\mathbf{N}_4 &= \mathbf{Q}_3^{-1} * \mathbf{N}_3 = \mathbf{Q}_3^{-1} * \mathbf{Q}_2^{-1} * \mathbf{Q}_1^{-1} * \mathbf{Q_{FK}} \\
\mathbf{N}_5 &= \mathbf{Q}_4^{-1} * \mathbf{N}_4 = \mathbf{Q}_4^{-1} * \mathbf{Q}_3^{-1} * \mathbf{Q}_2^{-1} * \mathbf{Q}_1^{-1} * \mathbf{Q_{FK}} \\
\mathbf{N}_6 &= \mathbf{Q}_5^{-1} * \mathbf{N}_5 = \mathbf{Q}_5^{-1} * \mathbf{Q}_4^{-1} * \mathbf{Q}_3^{-1} * \mathbf{Q}_2^{-1} * \mathbf{Q}_1^{-1} * \mathbf{Q_{FK}}
\end{aligned}$$

Transformation Products from Base to Wrist 3 ($\mathbf{M_6} = \mathbf{N_6}$)

$$\mathbf{M_6} = ([M_{61} + M_{62}i + M_{63}j + M_{64}k], < M_{65}i + M_{66}j + M_{67}k >)$$
$$\mathbf{N_6} = ([N_{61} + N_{62}i + N_{63}j + N_{64}k], < N_{65}i + N_{66}j + N_{67}k >)$$

$$\mathbf{M_{61} = N_{61}}$$

$$\begin{aligned}\cos\left(\frac{\theta_6}{2}\right) = w\cos\left(\frac{\theta_2+\theta_3+\theta_4}{2}\right)\cos\left(\frac{\theta_1-\theta_5}{2}\right) \\ +a\sin\left(\frac{\theta_2+\theta_3+\theta_4}{2}\right)\sin\left(\frac{\theta_1+\theta_5}{2}\right) \\ -b\sin\left(\frac{\theta_2+\theta_3+\theta_4}{2}\right)\cos\left(\frac{\theta_1+\theta_5}{2}\right) \\ +c\cos\left(\frac{\theta_2+\theta_3+\theta_4}{2}\right)\sin\left(\frac{\theta_1-\theta_5}{2}\right)\end{aligned} \tag{B.1}$$

$$\mathbf{M_{62} = N_{62}}$$

$$\begin{aligned}0 = -w\sin\left(\frac{\theta_2+\theta_3+\theta_4}{2}\right)\sin\left(\frac{\theta_1+\theta_5}{2}\right) \\ +a\cos\left(\frac{\theta_2+\theta_3+\theta_4}{2}\right)\cos\left(\frac{\theta_1-\theta_5}{2}\right) \\ +b\cos\left(\frac{\theta_2+\theta_3+\theta_4}{2}\right)\sin\left(\frac{\theta_1-\theta_5}{2}\right) \\ +c\sin\left(\frac{\theta_2+\theta_3+\theta_4}{2}\right)\cos\left(\frac{\theta_1+\theta_5}{2}\right)\end{aligned} \tag{B.2}$$

$$\mathbf{M_{63} = N_{63}}$$

$$\begin{aligned}-\sin\left(\frac{\theta_6}{2}\right) = w\sin\left(\frac{\theta_2+\theta_3+\theta_4}{2}\right)\cos\left(\frac{\theta_1+\theta_5}{2}\right) \\ -a\cos\left(\frac{\theta_2+\theta_3+\theta_4}{2}\right)\sin\left(\frac{\theta_1-\theta_5}{2}\right) \\ +b\cos\left(\frac{\theta_2+\theta_3+\theta_4}{2}\right)\cos\left(\frac{\theta_1-\theta_5}{2}\right) \\ +c\sin\left(\frac{\theta_2+\theta_3+\theta_4}{2}\right)\sin\left(\frac{\theta_1+\theta_5}{2}\right)\end{aligned} \tag{B.3}$$

$$\underline{\mathbf{M_{64} = N_{64}}}$$

$$\begin{aligned} 0 = &-w\cos\left(\frac{\theta_2+\theta_3+\theta_4}{2}\right)\sin\left(\frac{\theta_1-\theta_5}{2}\right) \\ &-a\sin\left(\frac{\theta_2+\theta_3+\theta_4}{2}\right)\cos\left(\frac{\theta_1+\theta_5}{2}\right) \\ &-b\sin\left(\frac{\theta_2+\theta_3+\theta_4}{2}\right)\sin\left(\frac{\theta_1+\theta_5}{2}\right) \\ &+c\cos\left(\frac{\theta_2+\theta_3+\theta_4}{2}\right)\cos\left(\frac{\theta_1-\theta_5}{2}\right) \end{aligned} \tag{B.4}$$

$$\underline{\mathbf{M_{65} = N_{65}}}$$

$$\begin{aligned} 0 = &-d_1\cos\theta_5\sin\left(\theta_2+\theta_3+\theta_4\right) - d_4\sin\theta_5 - a_2\cos\left(\theta_3+\theta_4\right)\cos\theta_5 \\ &-a_3\cos\theta_4 cos\theta_5 + x\left[\cos\theta_1\cos\left(\theta_2+\theta_3+\theta_4\right)\cos\theta_5 + \sin\theta_1\sin\theta_5\right] \\ &+y\left[\cos\left(\theta_2+\theta_3+\theta_4\right)\cos\theta_5\sin\theta_1 - \cos\theta_1\sin\theta_5\right] \\ &+z\cos\theta_5\sin\left(\theta_2+\theta_3+\theta_4\right) \end{aligned} \tag{B.5}$$

$$\underline{\mathbf{M_{66} = N_{66}}}$$

$$\begin{aligned} 0 = &-d_1\sin\left(\theta_2+\theta_3+\theta_4\right)\sin\theta_5 + d_4\cos\theta_5 - a_2\cos\left(\theta_3+\theta_4\right)\sin\theta_5 \\ &-a_3\cos\theta_4 sin\theta_5 + x\left[\cos\theta_1\cos\left(\theta_2+\theta_3+\theta_4\right)\sin\theta_5 - \cos\theta_5\sin\theta_1\right] \\ &+y\left[\cos\left(\theta_2+\theta_3+\theta_4\right)\sin\theta_1\sin\theta_5 + \cos\theta_1\cos\theta_5\right] \\ &+z\sin\left(\theta_2+\theta_3+\theta_4\right)\sin\theta_5 + d_6 \end{aligned} \tag{B.6}$$

$$\underline{\mathbf{M_{67} = N_{67}}}$$

$$\begin{aligned} 0 = &-d_1\cos\left(\theta_2+\theta_3+\theta_4\right) - x\cos\theta_1\sin\left(\theta_2+\theta_3+\theta_4\right) \\ &-y\sin\theta_1\sin\left(\theta_2+\theta_3+\theta_4\right) + z\cos\left(\theta_2+\theta_3+\theta_4\right) \\ &+a_2\sin\left(\theta_3+\theta_4\right) + a_3\sin\theta_4 + d_5 \end{aligned} \tag{B.7}$$

Transformation Products from Base to Wrist 3 ($\mathbf{M_5} = \mathbf{N_5}$)

$$\mathbf{M_5} = ([M_{51} + M_{52}i + M_{53}j + M_{54}k], < M_{55}i + M_{56}j + M_{57}k >)$$
$$\mathbf{N_5} = ([N_{51} + N_{52}i + N_{53}j + N_{54}k], < N_{55}i + N_{56}j + N_{57}k >)$$

$\mathbf{M_{51}} = \mathbf{N_{51}}$

$$\begin{aligned}\cos\left(\frac{\theta_5}{2}\right)\cos\left(\frac{\theta_6}{2}\right) = w\cos\left(\frac{\theta_1}{2}\right)\cos\left(\frac{\theta_2+\theta_3+\theta_4}{2}\right) \\ +a\sin\left(\frac{\theta_1}{2}\right)\sin\left(\frac{\theta_2+\theta_3+\theta_4}{2}\right) \\ -b\cos\left(\frac{\theta_1}{2}\right)\sin\left(\frac{\theta_2+\theta_3+\theta_4}{2}\right) \\ +c\sin\left(\frac{\theta_1}{2}\right)\cos\left(\frac{\theta_2+\theta_3+\theta_4}{2}\right)\end{aligned} \tag{B.8}$$

$\mathbf{M_{52}} = \mathbf{N_{52}}$

$$\begin{aligned}-\sin\left(\frac{\theta_5}{2}\right)\sin\left(\frac{\theta_6}{2}\right) = -w\sin\left(\frac{\theta_1}{2}\right)\sin\left(\frac{\theta_2+\theta_3+\theta_4}{2}\right) \\ +a\cos\left(\frac{\theta_1}{2}\right)\cos\left(\frac{\theta_2+\theta_3+\theta_4}{2}\right) \\ +b\sin\left(\frac{\theta_1}{2}\right)\cos\left(\frac{\theta_2+\theta_3+\theta_4}{2}\right) \\ +c\cos\left(\frac{\theta_1}{2}\right)\sin\left(\frac{\theta_2+\theta_3+\theta_4}{2}\right)\end{aligned} \tag{B.9}$$

$\mathbf{M_{53}} = \mathbf{N_{53}}$

$$\begin{aligned}-\cos\left(\frac{\theta_5}{2}\right)\sin\left(\frac{\theta_6}{2}\right) = w\cos\left(\frac{\theta_1}{2}\right)\sin\left(\frac{\theta_2+\theta_3+\theta_4}{2}\right) \\ -a\sin\left(\frac{\theta_1}{2}\right)\cos\left(\frac{\theta_2+\theta_3+\theta_4}{2}\right) \\ +b\cos\left(\frac{\theta_1}{2}\right)\cos\left(\frac{\theta_2+\theta_3+\theta_4}{2}\right) \\ +c\sin\left(\frac{\theta_1}{2}\right)\sin\left(\frac{\theta_2+\theta_3+\theta_4}{2}\right)\end{aligned} \tag{B.10}$$

$$\underline{\mathbf{M_{54} = N_{54}}}$$

$$\begin{aligned} -\sin\left(\frac{\theta_5}{2}\right)\cos\left(\frac{\theta_6}{2}\right) = & -w\sin\left(\frac{\theta_1}{2}\right)\cos\left(\frac{\theta_2+\theta_3+\theta_4}{2}\right) \\ & -a\cos\left(\frac{\theta_1}{2}\right)\sin\left(\frac{\theta_2+\theta_3+\theta_4}{2}\right) \\ & -b\sin\left(\frac{\theta_1}{2}\right)\sin\left(\frac{\theta_2+\theta_3+\theta_4}{2}\right) \\ & +c\cos\left(\frac{\theta_1}{2}\right)\cos\left(\frac{\theta_2+\theta_3+\theta_4}{2}\right) \end{aligned} \tag{B.11}$$

$$\underline{\mathbf{M_{55} = N_{55}}}$$

$$\begin{aligned} -d_6\sin\theta_5 = & -d_1\sin(\theta_2+\theta_3+\theta_4) - a_2\cos(\theta_3+\theta_4) - a_3\cos\theta_4 \\ & + x\cos\theta_1\cos(\theta_2+\theta_3+\theta_4) + y\sin\theta_1\cos(\theta_2+\theta_3+\theta_4) \\ & + z\sin(\theta_2+\theta_3+\theta_4) \end{aligned} \tag{B.12}$$

$$\underline{\mathbf{M_{56} = N_{56}}}$$

$$-d_6\cos\theta_5 = y\cos\theta_1 - x\sin\theta_1 + d_4 \tag{B.13}$$

$$\underline{\mathbf{M_{57} = N_{57}}}$$

$$\begin{aligned} 0 = & -d_1\cos(\theta_2+\theta_3+\theta_4) + d_5 + a_2\sin(\theta_3+\theta_4) + a_3\sin\theta_4 \\ & - x\cos\theta_1\sin(\theta_2+\theta_3+\theta_4) - y\sin\theta_1\sin(\theta_2+\theta_3+\theta_4) \\ & + z\cos(\theta_2+\theta_3+\theta_4) \end{aligned} \tag{B.14}$$

Transformation Products from Base to Wrist 3 ($\mathbf{M_4} = \mathbf{N_4}$)

$$\mathbf{M_4} = ([M_{41} + M_{42}i + M_{43}j + M_{44}k], < M_{45}i + M_{46}j + M_{47}k >)$$
$$\mathbf{N_4} = ([N_{41} + N_{42}i + N_{43}j + N_{44}k], < N_{45}i + N_{46}j + N_{47}k >)$$

$\mathbf{M_{41}} = \mathbf{N_{41}}$

$$\begin{aligned}\cos\left(\frac{\theta_5}{2}\right)\cos\left(\frac{\theta_4+\theta_6}{2}\right) = w\cos\left(\frac{\theta_1}{2}\right)\cos\left(\frac{\theta_2+\theta_3}{2}\right) \\ +a\sin\left(\frac{\theta_1}{2}\right)\sin\left(\frac{\theta_2+\theta_3}{2}\right) \\ -b\cos\left(\frac{\theta_1}{2}\right)\sin\left(\frac{\theta_2+\theta_3}{2}\right) \\ +c\sin\left(\frac{\theta_1}{2}\right)\cos\left(\frac{\theta_2+\theta_3}{2}\right)\end{aligned} \tag{B.15}$$

$\mathbf{M_{42}} = \mathbf{N_{42}}$

$$\begin{aligned}\sin\left(\frac{\theta_5}{2}\right)\sin\left(\frac{\theta_4-\theta_6}{2}\right) = -w\sin\left(\frac{\theta_1}{2}\right)\sin\left(\frac{\theta_2+\theta_3}{2}\right) \\ +a\cos\left(\frac{\theta_1}{2}\right)\cos\left(\frac{\theta_2+\theta_3}{2}\right) \\ +b\sin\left(\frac{\theta_1}{2}\right)\cos\left(\frac{\theta_2+\theta_3}{2}\right) \\ +c\cos\left(\frac{\theta_1}{2}\right)\sin\left(\frac{\theta_2+\theta_3}{2}\right)\end{aligned} \tag{B.16}$$

$\mathbf{M_{43}} = \mathbf{N_{43}}$

$$\begin{aligned}-\cos\left(\frac{\theta_5}{2}\right)\sin\left(\frac{\theta_4+\theta_6}{2}\right) = w\cos\left(\frac{\theta_1}{2}\right)\sin\left(\frac{\theta_2+\theta_3}{2}\right) \\ -a\sin\left(\frac{\theta_1}{2}\right)\cos\left(\frac{\theta_2+\theta_3}{2}\right) \\ +b\cos\left(\frac{\theta_1}{2}\right)\cos\left(\frac{\theta_2+\theta_3}{2}\right) \\ +c\sin\left(\frac{\theta_1}{2}\right)\sin\left(\frac{\theta_2+\theta_3}{2}\right)\end{aligned} \tag{B.17}$$

$$\underline{\mathbf{M_{44} = N_{44}}}$$

$$\begin{aligned} -\sin\left(\frac{\theta_5}{2}\right)\cos\left(\frac{\theta_4-\theta_6}{2}\right) = &-w\sin\left(\frac{\theta_1}{2}\right)\cos\left(\frac{\theta_2+\theta_3}{2}\right) \\ &-a\cos\left(\frac{\theta_1}{2}\right)\sin\left(\frac{\theta_2+\theta_3}{2}\right) \\ &-b\sin\left(\frac{\theta_1}{2}\right)\sin\left(\frac{\theta_2+\theta_3}{2}\right) \\ &+c\cos\left(\frac{\theta_1}{2}\right)\cos\left(\frac{\theta_2+\theta_3}{2}\right) \end{aligned} \tag{B.18}$$

$$\underline{\mathbf{M_{45} = N_{45}}}$$

$$\begin{aligned} -d_6\cos\theta_4\sin\theta_5 + d_5\sin\theta_4 = &-d_1\sin(\theta_2+\theta_3) - a_2\cos\theta_3 - a_3 \\ &+x\cos\theta_1\cos(\theta_2+\theta_3) \\ &+y\sin\theta_1\cos(\theta_2+\theta_3) \\ &+z\sin(\theta_2+\theta_3) \end{aligned} \tag{B.19}$$

$$\underline{\mathbf{M_{46} = N_{46}}}$$

$$-d_6\cos\theta_5 = y\cos\theta_1 - x\sin\theta_1 + d_4 \tag{B.20}$$

$$\underline{\mathbf{M_{47} = N_{47}}}$$

$$\begin{aligned} -d_6\sin\theta_4\sin\theta_5 - d_5\cos\theta_4 = &-d_1\cos(\theta_2+\theta_3) + a_2\sin\theta_3 \\ &-x\cos\theta_1\sin(\theta_2+\theta_3) \\ &-y\sin\theta_1\sin(\theta_2+\theta_3) \\ &+z\cos(\theta_2+\theta_3) \end{aligned} \tag{B.21}$$

Transformation Products from Base to Wrist 3 ($\mathbf{M_3} = \mathbf{N_3}$)

$$\mathbf{M_3} = ([M_{31} + M_{32}i + M_{33}j + M_{34}k], < M_{35}i + M_{36}j + M_{37}k >)$$
$$\mathbf{N_3} = ([N_{31} + N_{32}i + N_{33}j + N_{34}k], < N_{35}i + N_{36}j + N_{37}k >)$$

$\mathbf{M_{31}} = \mathbf{N_{31}}$

$$\begin{aligned}\cos\left(\frac{\theta_5}{2}\right)\cos\left(\frac{\theta_3+\theta_4+\theta_6}{2}\right) = {} & w\cos\left(\frac{\theta_1}{2}\right)\cos\left(\frac{\theta_2}{2}\right) \\ & +a\sin\left(\frac{\theta_1}{2}\right)\sin\left(\frac{\theta_2}{2}\right) \\ & -b\cos\left(\frac{\theta_1}{2}\right)\sin\left(\frac{\theta_2}{2}\right) \\ & +c\sin\left(\frac{\theta_1}{2}\right)\cos\left(\frac{\theta_2}{2}\right)\end{aligned} \tag{B.22}$$

$\mathbf{M_{32}} = \mathbf{N_{32}}$

$$\begin{aligned}\sin\left(\frac{\theta_5}{2}\right)\sin\left(\frac{\theta_3+\theta_4-\theta_6}{2}\right) = {} & -w\sin\left(\frac{\theta_1}{2}\right)\sin\left(\frac{\theta_2}{2}\right) \\ & +a\cos\left(\frac{\theta_1}{2}\right)\cos\left(\frac{\theta_2}{2}\right) \\ & +b\sin\left(\frac{\theta_1}{2}\right)\cos\left(\frac{\theta_2}{2}\right) \\ & +c\cos\left(\frac{\theta_1}{2}\right)\sin\left(\frac{\theta_2}{2}\right)\end{aligned} \tag{B.23}$$

$\mathbf{M_{33}} = \mathbf{N_{33}}$

$$\begin{aligned}-\cos\left(\frac{\theta_5}{2}\right)\sin\left(\frac{\theta_3+\theta_4+\theta_6}{2}\right) = {} & w\cos\left(\frac{\theta_1}{2}\right)\sin\left(\frac{\theta_2}{2}\right) \\ & -a\sin\left(\frac{\theta_1}{2}\right)\cos\left(\frac{\theta_2}{2}\right) \\ & +b\cos\left(\frac{\theta_1}{2}\right)\cos\left(\frac{\theta_2}{2}\right) \\ & +c\sin\left(\frac{\theta_1}{2}\right)\sin\left(\frac{\theta_2}{2}\right)\end{aligned} \tag{B.24}$$

$$\underline{\mathbf{M_{34} = N_{34}}}$$

$$\begin{aligned} -\sin\left(\frac{\theta_5}{2}\right)\cos\left(\frac{\theta_3+\theta_4-\theta_6}{2}\right) = & -w\sin\left(\frac{\theta_1}{2}\right)\cos\left(\frac{\theta_2}{2}\right) \\ & -a\cos\left(\frac{\theta_1}{2}\right)\sin\left(\frac{\theta_2}{2}\right) \\ & -b\sin\left(\frac{\theta_1}{2}\right)\sin\left(\frac{\theta_2}{2}\right) \\ & +c\cos\left(\frac{\theta_1}{2}\right)\cos\left(\frac{\theta_2}{2}\right) \end{aligned} \tag{B.25}$$

$$\underline{\mathbf{M_{35} = N_{35}}}$$

$$\begin{aligned} -d_6\cos(\theta_3+\theta_4)\sin\theta_5 + d_5\sin(\theta_3+\theta_4) + a_3\cos\theta_3 = & \\ -d_1\sin\theta_2 - a_2 + x\cos\theta_1\cos\theta_2 & \\ + y\sin\theta_1\cos\theta_2 + z\sin\theta_2 & \end{aligned} \tag{B.26}$$

$$\underline{\mathbf{M_{36} = N_{36}}}$$

$$-d_6\cos\theta_5 - d_4 = y\cos\theta_1 - x\sin\theta_1 \tag{B.27}$$

$$\underline{\mathbf{M_{37} = N_{37}}}$$

$$\begin{aligned} -d_6\sin(\theta_3+\theta_4)\sin\theta_5 - d_5\cos(\theta_3+\theta_4) + a_3\sin\theta_3 = & \\ -d_1\cos\theta_2 - x\cos\theta_1\sin\theta_2 & \\ - y\sin\theta_1\sin\theta_2 + z\cos\theta_2 & \end{aligned} \tag{B.28}$$

Transformation Products from Base to Wrist 3 ($\mathbf{M_2} = \mathbf{N_2}$)

$$\mathbf{M_2} = ([M_{21} + M_{22}i + M_{23}j + M_{24}k], < M_{25}i + M_{26}j + M_{27}k >)$$
$$\mathbf{N_2} = ([N_{21} + N_{22}i + N_{23}j + N_{24}k], < N_{25}i + N_{26}j + N_{27}k >)$$

$\mathbf{M_{21}} = \mathbf{N_{21}}$

$$\cos\left(\frac{\theta_5}{2}\right)\cos\left(\frac{\theta_2+\theta_3+\theta_4+\theta_6}{2}\right) = w\cos\left(\frac{\theta_1}{2}\right) + c\sin\left(\frac{\theta_1}{2}\right) \tag{B.29}$$

$\mathbf{M_{22}} = \mathbf{N_{22}}$

$$\sin\left(\frac{\theta_5}{2}\right)\sin\left(\frac{\theta_2+\theta_3+\theta_4-\theta_6}{2}\right) = a\cos\left(\frac{\theta_1}{2}\right) + b\sin\left(\frac{\theta_1}{2}\right) \tag{B.30}$$

$\mathbf{M_{23}} = \mathbf{N_{23}}$

$$-\cos\left(\frac{\theta_5}{2}\right)\sin\left(\frac{\theta_2+\theta_3+\theta_4+\theta_6}{2}\right) = -a\sin\left(\frac{\theta_1}{2}\right) + b\cos\left(\frac{\theta_1}{2}\right) \tag{B.31}$$

$\mathbf{M_{24}} = \mathbf{N_{24}}$

$$-\sin\left(\frac{\theta_5}{2}\right)\cos\left(\frac{\theta_2+\theta_3+\theta_4-\theta_6}{2}\right) = -w\sin\left(\frac{\theta_1}{2}\right) + c\cos\left(\frac{\theta_1}{2}\right) \tag{B.32}$$

$\mathbf{M_{25}} = \mathbf{N_{25}}$

$$\begin{aligned} -d_6\cos(\theta_2+\theta_3+\theta_4)\sin\theta_5 + d_5\sin(\theta_2+\theta_3+\theta_4) \\ +a_3\cos(\theta_2+\theta_3) + a_2\cos\theta_2 = x\cos\theta_1 + y\sin\theta_1 \end{aligned} \tag{B.33}$$

$\mathbf{M_{26}} = \mathbf{N_{26}}$

$$-d_6\cos\theta_5 - d_4 = y\cos\theta_1 - x\sin\theta_1 \tag{B.34}$$

$\mathbf{M_{27}} = \mathbf{N_{27}}$

$$\begin{aligned} -d_6\sin(\theta_2+\theta_3+\theta_4)\sin\theta_5 - d_5\cos(\theta_2+\theta_3+\theta_4) \\ +a_3\sin(\theta_2+\theta_3) + a_2\sin\theta_2 = -d_1 + z \end{aligned} \tag{B.35}$$

Transformation Products from Base to Wrist 3 ($\mathbf{M_1} = \mathbf{N_1}$)

$$\mathbf{M_1} = ([M_{11} + M_{12}i + M_{13}j + M_{14}k], < M_{15}i + M_{16}j + M_{17}k >)$$
$$\mathbf{N_1} = ([N_{11} + N_{12}i + N_{13}j + N_{14}k], < N_{15}i + N_{16}j + N_{17}k >)$$

$$\underline{\mathbf{M_{11}} = \mathbf{N_{11}}}$$

$$\cos\left(\frac{\theta_1 - \theta_5}{2}\right)\cos\left(\frac{\theta_2 + \theta_3 + \theta_4 + \theta_6}{2}\right) = w \tag{B.36}$$

$$\underline{\mathbf{M_{12}} = \mathbf{N_{12}}}$$

$$\sin\left(\frac{\theta_1 + \theta_5}{2}\right)\sin\left(\frac{\theta_2 + \theta_3 + \theta_4 - \theta_6}{2}\right) = a \tag{B.37}$$

$$\underline{\mathbf{M_{13}} = \mathbf{N_{13}}}$$

$$-\cos\left(\frac{\theta_1 + \theta_5}{2}\right)\sin\left(\frac{\theta_2 + \theta_3 + \theta_4 + \theta_6}{2}\right) = b \tag{B.38}$$

$$\underline{\mathbf{M_{14}} = \mathbf{N_{14}}}$$

$$-\sin\left(\frac{\theta_5 - \theta_1}{2}\right)\cos\left(\frac{\theta_2 + \theta_3 + \theta_4 - \theta_6}{2}\right) = c \tag{B.39}$$

$$\underline{\mathbf{M_{15}} = \mathbf{N_{15}}}$$

$$\begin{aligned} -d_6\cos\theta_1\cos(\theta_2+\theta_3+\theta_4)\sin\theta_5 + d_6\cos\theta_5\sin\theta_1 + d_4\sin\theta_1 \\ +d_5\cos\theta_1\sin(\theta_2+\theta_3+\theta_4) + a_3\cos\theta_1\cos(\theta_2+\theta_3) \\ +a_2\cos\theta_1\cos\theta_2 = x \end{aligned} \tag{B.40}$$

$$\underline{\mathbf{M_{16}} = \mathbf{N_{16}}}$$

$$\begin{aligned} -d_6\cos\theta_1\cos\theta_5 - d_4\cos\theta_1 + d_6\cos(\theta_2+\theta_3+\theta_4)\sin\theta_1\sin\theta_5 \\ +d_5\sin(\theta_2+\theta_3+\theta_4) + a_3\cos(\theta_2+\theta_3) + a_2\cos\theta_2 = y \end{aligned} \tag{B.41}$$

$$\underline{\mathbf{M_{17}} = \mathbf{N_{17}}}$$

$$\begin{aligned} -d_6\sin(\theta_2+\theta_3+\theta_4)\sin\theta_5 - d_5\cos(\theta_2+\theta_3+\theta_4) + a_3\sin(\theta_2+\theta_3) \\ +a_2\sin\theta_2 + d_1 = z \end{aligned} \tag{B.42}$$

Appendix C

Transformation Products for *Shoulder* to *Wrist 1*

This part shows the transformation products for the UR3 robot kinematic structure starting from Frame 2 (*Shoulder*) to Frame 4 (*Wrist 1*).

We define the dual quaternion product $\mathbf{M}_i$ as:

$$\mathbf{M}_i = \mathbf{Q}_i * \mathbf{M}_{i+1} \ \textit{where}\ 2 \leq i \leq 4$$

$$\begin{aligned}
&\mathbf{M}_4 = \mathbf{Q}_4 \\
&\mathbf{M}_3 = \mathbf{Q}_3 * \mathbf{M}_4 = \mathbf{Q}_3 * \mathbf{Q}_4 \\
&\mathbf{M}_2 = \mathbf{Q}_2 * \mathbf{M}_3 = \mathbf{Q}_2 * \mathbf{Q}_3 * \mathbf{Q}_4
\end{aligned}$$

We define the dual quaternion product $\mathbf{N}_i$ as follows:

$$\mathbf{N}_{i+1} = \mathbf{Q}_i^{-1} * \mathbf{N}_i \ \textit{where}\ 2 \leq i \leq 4$$

$$\begin{aligned}
&\mathbf{N}_2 = \mathbf{Q_4^2} = ([w_4, < a_4, b_4, c_4 >], < x_4, y_4, z_4 >) \\
&\mathbf{N}_3 = \mathbf{Q}_2^{-1} * \mathbf{N}_2 = \mathbf{Q}_2^{-1} * \mathbf{Q_4^2} \\
&\mathbf{N}_4 = \mathbf{Q}_3^{-1} * \mathbf{N}_3 = \mathbf{Q}_3^{-1} * \mathbf{Q}_2^{-1} * \mathbf{Q_4^2}
\end{aligned}$$

Transformation Products from Shoulder to Wrist 1 ($\mathbf{M_4} = \mathbf{N_4}$)

$$\mathbf{M_4} = ([M_{41} + M_{42}i + M_{43}j + M_{44}k], < M_{45}i + M_{46}j + M_{47}k >)$$
$$\mathbf{N_4} = ([N_{41} + N_{42}i + N_{43}j + N_{44}k], < N_{45}i + N_{46}j + N_{47}k >)$$

$\mathbf{M_{41}} = \mathbf{N_{41}}$

$$\cos\left(\frac{\theta_4}{2}\right) = w_4 \cos\left(\frac{\theta_2 + \theta_3}{2}\right) - b_4 \sin\left(\frac{\theta_2 + \theta_3}{2}\right) \tag{C.1}$$

$\mathbf{M_{42}} = \mathbf{N_{42}}$

$$0 = a_4 \cos\left(\frac{\theta_2 + \theta_3}{2}\right) + c_4 \sin\left(\frac{\theta_2 + \theta_3}{2}\right) \tag{C.2}$$

$\mathbf{M_{43}} = \mathbf{N_{43}}$

$$-\sin\left(\frac{\theta_4}{2}\right) = w_4 \sin\left(\frac{\theta_2 + \theta_3}{2}\right) + b_4 \cos\left(\frac{\theta_2 + \theta_3}{2}\right) \tag{C.3}$$

$\mathbf{M_{44}} = \mathbf{N_{44}}$

$$0 = -a_4 \sin\left(\frac{\theta_2 + \theta_3}{2}\right) + c_4 \cos\left(\frac{\theta_2 + \theta_3}{2}\right) \tag{C.4}$$

$\mathbf{M_{45}} = \mathbf{N_{45}}$

$$0 = -\,a_2 \cos\theta_3 - a_3 + x_4 \cos(\theta_2 + \theta_3) + z_4 \sin(\theta_2 + \theta_3) \tag{C.5}$$

$\mathbf{M_{46}} = \mathbf{N_{46}}$

$$0 = y_4 + d_4 \tag{C.6}$$

$\mathbf{M_{47}} = \mathbf{N_{47}}$

$$0 = a_2 \sin\theta_3 - x_4 \sin(\theta_2 + \theta_3) + z_4 \cos(\theta_2 + \theta_3) \tag{C.7}$$

Transformation Products from Shoulder to Wrist 1 ($\mathbf{M_3} = \mathbf{N_3}$)

$$\mathbf{M_3} = ([M_{31} + M_{32}i + M_{33}j + M_{34}k], < M_{35}i + M_{36}j + M_{37}k >)$$
$$\mathbf{N_3} = ([N_{31} + N_{32}i + N_{33}j + N_{34}k], < N_{35}i + N_{36}j + N_{37}k >)$$

$$\underline{\mathbf{M_{31}} = \mathbf{N_{31}}}$$

$$\cos\left(\frac{\theta_3 + \theta_4}{2}\right) = w_4 \cos\left(\frac{\theta_2}{2}\right) - b_4 \sin\left(\frac{\theta_2}{2}\right) \tag{C.8}$$

$$\underline{\mathbf{M_{32}} = \mathbf{N_{32}}}$$

$$0 = a_4 \cos\left(\frac{\theta_2}{2}\right) + c_4 \sin\left(\frac{\theta_2}{2}\right) \tag{C.9}$$

$$\underline{\mathbf{M_{33}} = \mathbf{N_{33}}}$$

$$-\sin\left(\frac{\theta_3 + \theta_4}{2}\right) = w_4 \sin\left(\frac{\theta_2}{2}\right) + b_4 \cos\left(\frac{\theta_2}{2}\right) \tag{C.10}$$

$$\underline{\mathbf{M_{34}} = \mathbf{N_{34}}}$$

$$0 = -a_4 \sin\left(\frac{\theta_2}{2}\right) + c_4 \cos\left(\frac{\theta_2}{2}\right) \tag{C.11}$$

$$\underline{\mathbf{M_{35}} = \mathbf{N_{35}}}$$

$$a_3 \cos\theta_3 = -a_2 + x_4 \cos\theta_2 + z_4 \sin\theta_2 \tag{C.12}$$

$$\underline{\mathbf{M_{36}} = \mathbf{N_{36}}}$$

$$-d_4 = y_4 \tag{C.13}$$

$$\underline{\mathbf{M_{37}} = \mathbf{N_{37}}}$$

$$a_3 \sin\theta_3 = -x_4 \sin\theta_2 + z_4 \cos\theta_2 \tag{C.14}$$

Transformation Products from Shoulder to Wrist 1 ($\mathbf{M_2} = \mathbf{N_2}$)

$$\mathbf{M_2} = ([M_{21} + M_{22}i + M_{23}j + M_{24}k], < M_{25}i + M_{26}j + M_{27}k >)$$
$$\mathbf{N_2} = ([N_{21} + N_{22}i + N_{23}j + N_{24}k], < N_{25}i + N_{26}j + N_{27}k >)$$

$$\underline{\mathbf{M_{21}} = \mathbf{N_{21}}}$$

$$\cos\left(\frac{\theta_2 + \theta_3 + \theta_4}{2}\right) = w_4 \tag{C.15}$$

$$\underline{\mathbf{M_{22}} = \mathbf{N_{22}}}$$

$$0 = a_4 \tag{C.16}$$

$$\underline{\mathbf{M_{23}} = \mathbf{N_{23}}}$$

$$-\sin\left(\frac{\theta_2 + \theta_3 + \theta_4}{2}\right) = b_4 \tag{C.17}$$

$$\underline{\mathbf{M_{24}} = \mathbf{N_{24}}}$$

$$0 = c_4 \tag{C.18}$$

$$\underline{\mathbf{M_{25}} = \mathbf{N_{25}}}$$

$$a_3 \cos(\theta_2 + \theta_3) + a_2 \cos\theta_2 = x_4 \tag{C.19}$$

$$\underline{\mathbf{M_{26}} = \mathbf{N_{26}}}$$

$$-d_4 = y_4 \tag{C.20}$$

$$\underline{\mathbf{M_{27}} = \mathbf{N_{27}}}$$

$$a_3 \sin(\theta_2 + \theta_3) + a_2 \sin\theta_2 = z_4 \tag{C.21}$$

Appendix D

List of Tables and Figures

List of Tables

List of Figures

Bibliography

[1] Robotics: Kinematics and Mathematical Foundations. https://www.edx.org/course/robotics-kinematics-mathematical-pennx-robo1x.

[2] M. Ahr. The Legacy of Rossum's Universal Robots. https://www.denofgeek.com/us/culture/279175/the-legacy-of-rossums-universal-robots, 2019.

[3] R. Andersen. Kinematics of a UR5. Technical report, Aalborg University, 2018.

[4] Y. Aydin and S. Kucuk. Quaternion Based Inverse Kinematics for Industrial Robot Manipulators with Euler Wrist. *2006 IEEE International Conference on Mechatronics, ICM*, 2006.

[5] J. Barbic. Quaternions and Rotations. Technical report, University of Southern California, 2011.

[6] M. Ben-Ari. A Tutorial on Euler Angles and Quaternions. 2018.

[7] Z. Chen and J. Hung. Application of Quaternion in Robot Control. *IFAC 10th Triennial World Congress*, 1987.

[8] J. Diebel. Representing Attitude: Euler Angles, Unit Quaternions, and Rotation Vectors. Technical report, Stanford University, 2006.

[9] P. R. Evans. Rotations and Rotation Matrices. Technical report, MRC Laboratory of Molecular Biology, 2001.

[10] W. R. Hamilton. *Elements of Quaternions*. Cambridge University Press, 1866.

[11] K. Hawkins. Analytic Inverse Kinematics for the Universal Robots UR-5/UR-10 Arms. Technical report, 2013.

[12] A. O. Hill. Kinematics: Why Robots Move Like They Do. https://blog.robotiq.com/kinematics-why-robots-move-like-they-do, 2015.

[13] J. Huerta. Introducting the Quaternions. Technical report, Fullerton College, 2010.

[14] B. Kenwright. A Beginners Guide to Dual-Quaternions: What They Are, How They Work, and How to Use Them for 3D. Technical report, Newcastle University, United Kingdom.

[15] S. Kucuk and Z. Bingul. Robot Kinematics: Forward and Inverse Kinematics. Technical report, Kocaell University, 2006.

[16] S. LaValle. Planning Algorithms - The Homogenous Transformation Matrix. http://planning.cs.uiuc.edu/node111.html, 2006.

[17] A. Lerios. Rotations and Quaternions. Technical report, Stanford University, 1995.

[18] J. E. Marsden and M. J. Hoffman. *Basic Complex Analysis*. W.H.Freeman & Co Ltd, New York, United States, 3 edition, 1999.

[19] Matarić. *The Robotics Primer*. Massachusetts Institute of Technology, London, England, 2007.

[20] U. Robots. Parameters for Calculations of Kinematics and Dynamics. https://www.universal-robots.com/how-tos-and-faqs/faq/ur-faq/parameters-for-calculations-of-kinematics-and-dynamics-45257/.

[21] B. Siciliano and O. Khatib. *Springer Handbook of Robotics*. Springer, Berlin, 2nd edition, 2016.

[22] M. Sunardi. 3D Kinematics. Technical report, 2006.

[23] Tekkotsu. Kinematics. http://www.tekkotsu.org/Kinematics.html, 2010.

[24] J. Voight. *Quaternion Algebras*, volume 0.9.14. 2018.

[25] A. Watt and M. Watt. *Advanced Animation and Rendering Techniques*. ACM Press, New York, United States, 1992.