Meccanica (2022) 57:1413–1428
https://doi.org/10.1007/s11012-022-01512-w

Iterative inverse kinematics for robot manipulators using quaternion algebra and conformal geometric algebra

L. Lechuga-Gutierrez · E. Macias-Garcia · G. Martínez-Terán · J. Zamora-Esquivel · E. Bayro-Corrochano

Received: 11 July 2021 / Accepted: 26 January 2022 / Published online: 13 April 2022
© Springer Nature B.V. 2022

Abstract This paper presents a set of generalized iterative algorithms to find the inverse position kinematics of n-degree-of-freedom kinematic chains with revolute joints. As a first approach, an iterative algorithm is developed using the gradient descent method in Quaternion Algebra to find both the inverse position and velocity kinematics solution in redundant systems closest to their initial configuration. Additionally, a generalized extension of this approach is developed employing screw rotors and Conformal Geometric Algebra, where efficient update rules are obtained to solve the problem of inverse position kinematics. Simulation experiments using different degree-of-freedom models as well as real-time experiments using a Geomagic Touch Haptic device are carried out to demonstrate the effectiveness of the proposed methods.

Keywords Quaternion algebra · Conformal geometric algebra · Inverse kinematics · Robot manipulators

L. Lechuga-Gutierrez
Cuerpo Académico de Electrónica y Control, Universidad Autónomia del Estado de Hidalgo, Mineral de la Reforma, Hidalgo, Mexico
e-mail: lrlechuga@uaeh.edu.mx

E. Macias-Garcia · G. Martínez-Terán · E. Bayro-Corrochano (✉)
Departamento de Ingeniería Eléctrica y Ciencias de la Computación, Centro de Investigación y de Estudios Avanzados del Instituto Politécnico Nacional, Zapopan, Jalisco, México
e-mail: eduardo.bayro@cinvestav.mx

G. Martínez-Terán
e-mail: gerardo.martinez@cinvestav.mx

J. Zamora-Esquivel
Intel Labs, Zapopan, Jalisco, Mexico
e-mail: julio.c.zamora.esquivel@intel.com

1 Introduction

The problem of forward kinematics consists of determining the position and orientation of the end-effector on a kinematic chain according to a reference frame. This problem is usually solved by employing trigonometric formulas or by the Denavit-Hartenberg algorithm for large kinematic chains. Although the methods work correctly, there is a loss of geometric meaning as the rigid body transformations are restricted to variations in generic rotations on the pitch, raw, or roll axes, hindering representation of an arbitrary rotation angle, as the number of degrees of freedom of the kinematic chain is increased. *Hamilton* [1] manages to express orientations in three dimensions by employing quaternions, making it possible to obtain a very intuitive geometric sense to represent orientations at any reference axis [2, 3].

Inverse kinematics is a technique that allows determining the required movement of the joints in a kinematic chain to ensure a desired end-effector position. The inverse kinematics calculation is a complex problem that usually requires solving equations series whose solution is generally not unique [4]. Traditional kinematic algorithms employ linear transformations using matrices or tensors, which, despite their simplicity, involve redundant coefficients, such as rotation matrices which require nine coefficients to represent one rotation through an axis. The formulation of robot kinematics within the geometric algebra framework has shown attractive advantages, as the motion of 3D Euclidean points can be represented by different entities such as rotors and motors among others [5, 6]. Since geometric algebra is a coordinate-free mathematical system, it facilitates representing a robot configuration by employing its geometric structure directly.

The main contributions of this paper can be described as follows:

- A novel set of algorithms based on Quaternion Algebra and Conformal Geometric Algebra is proposed to solve the inverse position kinematics of n-degree-of-freedom kinematic chains with revolute joints, by employing the gradient descent algorithm and a proposed error function between the end-effector and a desired position. For inverse velocity kinematics, an iterative algorithm based on Quaternion Algebra is also proposed.
- A Conformal Geometric Algebra library is developed to perform basic operations (such as sum, substraction as well as wedge, Clifford, and dot products) in MATLAB [7], employing geometric entities.

The rest of the paper is organized as follows: In Sect. 2, a general background is provided for understanding the basic concepts referred to in this work. In Sect. 3, a set of algorithms using Quaternion Algebra is developed to solve both the inverse position and velocity kinematics for different n-degree-of-freedom kinematic chains, while in Sect. 4, a generalized extension using Conformal Geometric Algebra is also developed for the inverse position kinematics problem. In Sect. 5 both proposed algorithms are compared using different update rules and performance test. In Sect. 6, real-time experiments are carried out using a Geomagic Haptic Touch device with a PID controller, employing the algorithms described in Sect. 2. Finally, in Sect. 7, the conclusions and future work of the present paper are presented.

2 Preliminaries

2.1 Quaternion algebra

Quaternion Algebra $\mathbb{H}$ was invented by W. Hamilton in 1843 [8] while attempting to find an algebraic system which would work for the $\mathbb{R}^4$ space.

The current formalism of vector algebra was simply extracted from the quaternion product [9] of two vectors by Gibbs in 1901 . Unit quaternions provide a mathematical notation to represent orientations and rotations of objects in three dimensions. Compared to rotation matrices, they are more efficient and numerically stable. The quaternions are useful in robotics and navigation, among other applications [10–12]. A quaternion can be expressed as the following set:

$$\mathbb{H} = \{a + bi + cj + dk \,:\, a, b, c, d \in \mathbb{R}\} \subset \mathbb{C}^2 \subset \mathbb{R}^4, \tag{1}$$

where i, j, k are called the main imaginary, which obeys the Hamilton's rules

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

2.2 Rotations employing quaternions

One of the most prominent applications for quaternions remains their suitability for rotating vectors through an arbitrary axis [13]. A vector $\mathbf{v} = [x, y, z]$ can be represented as a quaternion $v \in \mathbb{H}$ through the following transformation:

$$v = 1 + xi + yj + zk, \tag{3}$$

while a rotation of magnitude θ through a unitary axis $\mathbf{n}$ can be represented by the following quaternion:

$$q = \cos\frac{\theta}{2} + \sin\frac{\theta}{2}\mathbf{n}, \tag{4}$$

or likewise by Euler's formula:

$$e^{\pm\frac{\theta}{2}\mathbf{n}} = \cos\frac{\theta}{2} \pm \sin\frac{\theta}{2}\mathbf{n}. \tag{5}$$

Then, a rotation of the vector v through an axis $\mathbf{n}$ with θ magnitude is given by the quaternion product:

$$v' = qv\widetilde{q}, \tag{6}$$

where $qp\widetilde{q}$ represents the quaternion multiplication and $\widetilde{q}$ refers to the following quaternion conjugate:

$$\tilde{q} = \cos\frac{\theta}{2} - \sin\frac{\theta}{2}\mathbf{n}. \tag{7}$$

A mathematical framework derived from Quaternion Algebra, known as Dual-Quaternion Algebra can be employed to obtain the forward kinematics of a manipulator with prismatic elements [14].

2.3 Conformal geometric algebra

Conformal Geometric Algebra $\mathbb{G}_{4,1}$ is an algebra that employs the sphere as a basis element. Entities are represented in this algebra through five basis e_i with the following properties:

$$e_i^2 = 1, \quad i \in \{1,2,3,4\}, \tag{8}$$

$$e_5^2 = -1, \tag{9}$$

$$e_0 = \frac{1}{2}(e_4 - e_5), \quad e_\infty = \quad e_4 + e_5, \tag{10}$$

where e_0 is called the point at origin, and e_∞ the point at infinity. A Euclidean point $p_e \in \mathbb{G}_3$ can be represented as a conformal point $p \in \mathbb{G}_{4,1}$ through the following transformation :

$$p = p_e + \frac{1}{2}p_e^2 e_\infty + e_0. \tag{11}$$

2.4 Screw rotors

A screw rotor can be defined as an entity $M \in \mathbb{G}_{4,1}$ of the following form:

$$M = \cos\left(\frac{\theta}{2}\right) - \sin\left(\frac{\theta}{2}\right)L = e^{-\frac{\theta}{2}L}, \tag{12}$$

where $L \in \mathbb{G}_{4,1}$ is a geometric entity called line, which is given by a set of two Euclidean vectors $a, b \in \mathbb{G}_3$ as :

$$L = (a - b)I_e - e_\infty(a \wedge b)I_e, \tag{13}$$

where $I_e = e_1e_2e_3$ is denominated the pseudoscalar, and the operator $\wedge$ the wedge product. Given a conformal point $p \in \mathbb{G}_{4,1}$, a rotation of $\theta/2$ around a line L can be calculated through the following transformation (Fig. 1) :

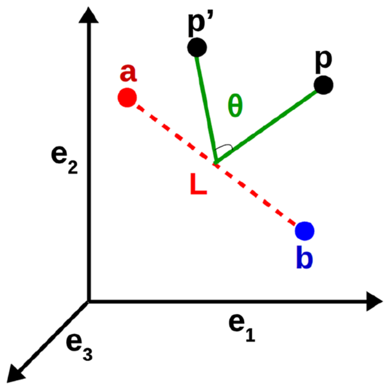

Fig. 1 Rotation of a conformal point employing screw rotors

where $\tilde{M} \in \mathbb{G}_{4,1}$ is the screw rotor conjugate.

2.5 Gradient descent algorithm

The gradient of a multidimensional function $f(x)$ with $x = [x_1, ..., x_d] \in \mathbb{R}^d$ (where d is the number of dimensions), represents how the function varies with respect to every one of its d dimensions. In this way, the gradient g_{x_1} expresses how the function $f(x)$ varies with respect to x_1 [15, 16]. Said gradient is appropriately defined as

$$g_{x_1} = \frac{\partial f(x)}{\partial x_1}. \tag{15}$$

The gradient descent algorithm seeks the minimum $f^* = f(x^*)$ of the function (whether it is local or not) through the following update rule

$$x_1(h) = x_1(h-1) - \alpha\frac{\partial f(x)}{\partial x_1}, \tag{16}$$

where $x_1(h-1)$ is the previous value of the variable x_1 in the function $f(x)$ and $x_1(h)$ is the next value of x_1. By applying this update rule, the function $f(x)$ decreases iteratively once a minimum is reached.

3 Forward and inverse kinematics using quaternion algebra

In robotics, kinematics is the study of operational coordinates or articular movements in robots. There are two types of kinematics: inverse kinematics and forward kinematics. In this section, a generalized algorithm based on Quaternion Algebra is proposed to solve the inverse position and velocity kinematics of revolute-based kinematic chains.

3.1 Forward kinematics

Forward kinematics is the technique employed to compute the operating coordinates of every part on articulated structures, using the configurations associated with each link. As a mathematical framework, Quaternion Algebra offers important advantages over matrices, as it does not present problems such as matrix indeterminacy (e.g, the Gimbal Lock [13]). As an introduction for subsequent sections, the forward kinematics of a two-degree-of-freedom model employing Quaternion Algebra is discussed.

3.1.1 Two-degree-of-freedom model

As an example, the two-degree-of-freedom system shown in Fig. 2 is employed. In this system the rotation axes for the links (v_1 and v_2) are parallel to the Z axis and can be represented by the set:

$$v_1 = 0i + 0j + 1k, \tag{17}$$

$$v_2 = 0i + 0j + 1k \tag{18}$$

which generates a quaternion for every rotation axis (v_1 and v_2), as follows:

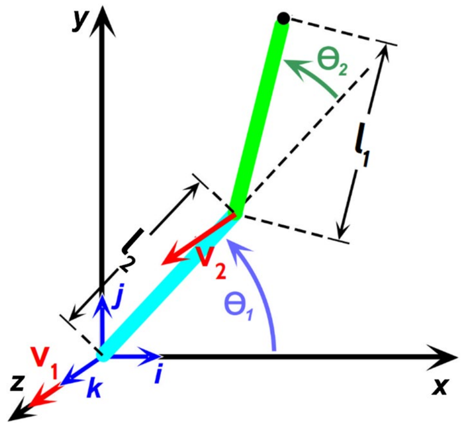

Fig. 2 Two-degree-of-freedom robot model, with two joints; v_1, and v_2

$$q_1 = \cos\frac{\theta_1}{2} + \sin\frac{\theta_1}{2}k, \tag{19}$$

$$q_2 = \cos\frac{\theta_2}{2} + \sin\frac{\theta_2}{2}k. \tag{20}$$

The equation that initializes the position of the links in quaternion form is obtained when all the angles are initialized (in this case at zero value):

$$p_1 = 0 + l_1 i + 0j + 0k, \tag{21}$$

$$p_2 = 0 + l_2 i + 0j + 0k. \tag{22}$$

By assembling equations (20) to (22), the forward kinematics of the model implementing quaternions is given by

$$p_f = q_1 p_1 \widetilde{q_1} + q_1 q_2 p_2 \widetilde{q_2} \widetilde{q_1}. \tag{23}$$

As can be seen in (23), to obtain the model forward kinematics, each link is associated with the quaternions that represent the axes of rotation which affect it (sum of previous movements), and ultimately, only the corresponding quaternion products are made to obtain a vector representation of the form

$$p_f = \begin{bmatrix} 0 \\ l_1\cos(\theta_1) + l_2\cos(\theta_1+\theta_2) \\ l_1\sin(\theta_1) + l_2\sin(\theta_1+\theta_2) \\ 0 \end{bmatrix} \begin{bmatrix} 1 & i & j & k \end{bmatrix}. \tag{24}$$

3.2 Inverse position kinematics

The gradient descent represents a standard parameter update rule in many areas of engineering [17], which consists of finding a set of parameters that minimizes a required cost function. This method can be applied to find the inverse position kinematics as the following update rule:

$$\vec{\theta}(h) = \vec{\theta}(h-1) - \alpha \frac{\partial \mathcal{L}(h-1)}{\partial \vec{\theta}}, \tag{25}$$

where $\vec{\theta}(h) \in \mathbb{R}^n$ is the vector of angles associated to the kinematic chain at the current moment h, $\vec{\theta}(h-1) \in \mathbb{R}^n$ is the vector at an earlier time, α is the learning rate, and $\frac{\partial \mathcal{L}}{\partial \theta}$ is the gradient of the error function $\mathcal{L} \in \mathbb{R}^n$ with respect to θ:

$$\frac{\partial \mathcal{L}}{\partial \vec{\theta}} = \begin{bmatrix} \frac{\partial \mathcal{L}}{\partial \theta_1} \\ \frac{\partial \mathcal{L}}{\partial \theta_2} \\ \vdots \\ \frac{\partial \mathcal{L}}{\partial \theta_n} \end{bmatrix}^T, \tag{26}$$

where $\mathcal{L}$ is proposed according to the linear regression criterion [18], so it remains:

$$\mathcal{L} = \frac{(p_d - p_f)^2}{2}, \; \{p_d, p_f\} \in \mathbb{H}, \tag{27}$$

with p_d and p_f the desired point and position of the end-effector manipulator, respectively (in quaternions).

By decomposing the error function along every axis the equation (27) can be rewritten as

$$\mathcal{L} = \frac{[(p_{dx} - p_{fx})i + (p_{dy} - p_{fy})j + (p_{dz} - p_{fz})k]^2}{2}, \tag{28}$$

and since $q^2 = q\tilde{q}$,

$$\mathcal{L} = \frac{(p_{dx} - p_{fx})^2 + (p_{dy} - p_{fy})^2 + (p_{dz} - p_{fz})^2}{2}, \tag{29}$$

$$\mathcal{L} = \frac{\mathcal{L}_x^2}{2} + \frac{\mathcal{L}_y^2}{2} + \frac{\mathcal{L}_z^2}{2}; \tag{30}$$

thus, (26) can be described as

$$\begin{bmatrix} \frac{\partial \mathcal{L}}{\partial \theta_1} \\ \frac{\partial \mathcal{L}}{\partial \theta_2} \\ \frac{\partial \mathcal{L}}{\partial \theta_3} \\ \vdots \\ \frac{\partial \mathcal{L}}{\partial \theta_n} \end{bmatrix}^T = - \begin{bmatrix} \mathcal{L}_x \frac{\partial p_{fx}}{\partial \theta_1} + \mathcal{L}_y \frac{\partial p_{fy}}{\partial \theta_1} + \mathcal{L}_z \frac{\partial p_{fz}}{\partial \theta_1} \\ \mathcal{L}_x \frac{\partial p_{fx}}{\partial \theta_2} + \mathcal{L}_y \frac{\partial p_{fy}}{\partial \theta_2} + \mathcal{L}_z \frac{\partial p_{fz}}{\partial \theta_2} \\ \mathcal{L}_x \frac{\partial p_{fx}}{\partial \theta_3} + \mathcal{L}_y \frac{\partial p_{fy}}{\partial \theta_3} + \mathcal{L}_z \frac{\partial p_{fz}}{\partial \theta_3} \\ \vdots \\ \mathcal{L}_x \frac{\partial p_{fx}}{\partial \theta_n} + \mathcal{L}_y \frac{\partial p_{fy}}{\partial \theta_n} + \mathcal{L}_z \frac{\partial p_{fz}}{\partial \theta_n} \end{bmatrix}^T, \tag{31}$$

where, by factorizing the right matrix, the following equation can be obtained:

$$\frac{\partial \mathcal{L}}{\partial \vec{\theta}} = -[\mathcal{L}_x \; \mathcal{L}_y \; \mathcal{L}_z] \begin{bmatrix} \frac{\partial P_f x}{\partial \theta_1} & \frac{\partial P_f x}{\partial \theta_2} & \cdots & \frac{\partial P_f x}{\partial \theta_n} \\ \frac{\partial P_f y}{\partial \theta_1} & \frac{\partial P_f y}{\partial \theta_2} & \cdots & \frac{\partial P_f y}{\partial \theta_n} \\ \frac{\partial P_f z}{\partial \theta_1} & \frac{\partial P_f z}{\partial \theta_2} & \cdots & \frac{\partial P_f z}{\partial \theta_n} \end{bmatrix}. \tag{32}$$

It can be clearly seen that the right matrix of (32) is the Jacobian matrix of the system.

According to equation (32), the gradient required to update the joint values using (25) is

$$\frac{\partial \mathcal{L}}{\partial \vec{\theta}} = -\mathcal{L}J. \tag{33}$$

Thus, equation (25) can be generalized to define an update rule to obtain the joint adjustment as

$$\vec{\theta}(h) = \vec{\theta}(h-1) + \alpha \mathcal{L}J. \tag{34}$$

3.3 Inverse velocity kinematics

The task of inverse velocity kinematics is to determine the required joint velocity according to the desired position of the end-effector. This model is usually obtained by computing the inverse of the Jacobian matrix and multiplying it by the Euclidean velocity vector; if the Jacobian matrix is non-square, the pseudo-inverse must be employed [19, 20]. This section presents an alternate method for obtaining the inverse kinematics of velocity.

If the joint velocity vector $\vec{\dot{\theta}}(h)$ at sample time h (case in discrete time) is known to be

$$\vec{\dot{\theta}}(h) = \frac{\vec{\theta}(h) - \vec{\theta}(h-1)}{\Delta t}, \tag{35}$$

where Δt is the sampling time. Thus, from equation (34),

$$\vec{\theta}(h) - \vec{\theta}(h-1) = \alpha \mathcal{L}J. \tag{36}$$

If (36) is divided over the sampling time Δt it holds that

$$\frac{\vec{\theta}(h) - \vec{\theta}(h-1)}{\Delta t} = \frac{\alpha \mathcal{L}J}{\Delta t}, \tag{37}$$

thus, if $\beta = \frac{\alpha}{\Delta t}$ is proposed, the final equation to obtain the inverse kinematics of velocity can be described as

$$\vec{\dot{\theta}}(h) = \beta \mathcal{L}J. \tag{38}$$

3.4 Two-degree-of-freedom planar robot example

Equation (34) can be employed to find the inverse kinematics of a two-degree-of-freedom robot manipulator (Fig. 2). As mentioned previously, the main problem of calculating the inverse kinematics of a

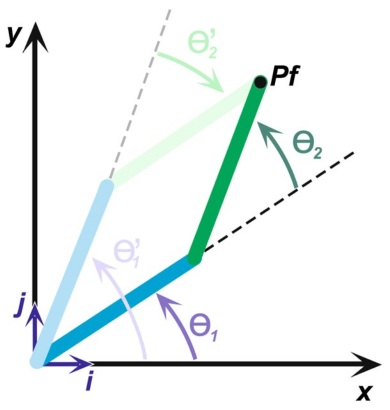

Fig. 3 Both robot configurations are shown (elbow up and elbow down) to reach a desired point P_f

manipulator remains in the configuration of a kinematic chain is not always unique (Fig. 3). To obtain the inverse kinematics, it is necessary to know the quaternion forward kinematics and subsequently derive them with respect to θ_1 and θ_2. Thus, from (23) the jacobian results:

$$J = \begin{bmatrix} q_1 p_1 \widetilde{q_1} + q_1 q_2 p_1 \widetilde{q_2} \widetilde{q_1} \\ q_1 q_2 p_1 \widetilde{q_2} \widetilde{q_1} \end{bmatrix}^T, \quad (39)$$

whose matrix representation (making the corresponding quaternion products) can be represented as

$$J = \begin{bmatrix} -l_1 \sin(\theta_1) - l_2 \sin(\theta_1 + \theta_2) & -l_2 \sin(\theta_1 + \theta_2) \\ l_1 \cos(\theta_1) + l_2 \cos(\theta_1 + \theta_2) & l_2 \cos(\theta_1 + \theta_2) \end{bmatrix}, \quad (40)$$

where the error vector associated with the system is:

$$\mathcal{L} = \begin{bmatrix} p_{dx} - p_{fx} & p_{dy} - p_{fy} \end{bmatrix}, \quad (41)$$

where p_{fx} and p_{fy} are the operational coordinates for x and y, respectively, and the points p_{dx} and p_{dy}, are the desired x and y coordinates for the end-effector. Finally, the joint update rule is obtained through (34) in matrix representation as:

$$\begin{bmatrix} \theta_1(h) \\ \theta_2(h) \end{bmatrix}^T = \begin{bmatrix} \theta_1(h-1) + \alpha(\mathcal{L}_1 J_{11} + \mathcal{L}_2 J_{21}) \\ \theta_2(h-1) + \alpha(\mathcal{L}_1 J_{12} + \mathcal{L}_2 J_{22}) \end{bmatrix}^T, \quad (42)$$

where $\theta_1(h)$ and $\theta_2(h)$ are the new angles at the moment h, $\theta_1(h-1)$ and $\theta_2(h-1)$ represent the same angles at a previous moment, and subindices $\mathcal{L}$ and J represent the elements of their respective vectors and matrices.

As a first experiment, the update rule (42) was employed to move the system described above to the desired end-effector position $p_d = [0.0292, 0.1267]$, starting from the home position $\theta(0) = [-100.0, 30.0]$ with manipulator dimensions $l_1 = 0.1m$ and $l_2 = 0.1m$. Figure 4 shows the evolution of the manipulator kinematics from its initial configuration (gray lines), and how it travels along the trajectory to reach the desired end-effector position (blue line), ending with the configuration that satisfies the conditions (black lines).

The behavior of the mean square error (seen in the right of Fig. 4) depends on the distance between the desired point and the end-effector position. As can be seen, the inverse kinematics using the gradient descent allows the end-effector to reach the desired point after 30 iterations with a step of $\alpha = 10$. The vector field of the error gradient concerning the vector of angles is also presented in Fig. 5. In addition to the error function's curve level being superimposed, it can be observed that there are two minimum zones (in dark blue).

The two minimum zones are created due to the manipulator with the proposed architecture (two links and two degrees of freedom) having two possible configurations for reaching the same point (which is colloquially known as elbow up and elbow down), where the vector field indicates that by employing the proposed initial angles ($\theta_1 = -100^o$ and $\theta_2 = -30^o$), one of the configurations where the manipulator reaches the desired point is $\theta_1 = 27.5416^o$ and $\theta_2 = 98.9168^o$.

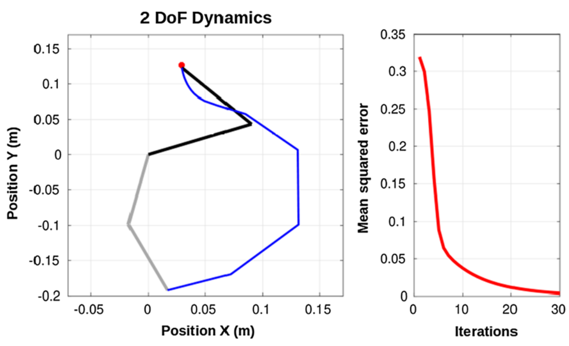

Fig. 4 Evolution of the manipulator kinematics (left) and the mean square error (right)

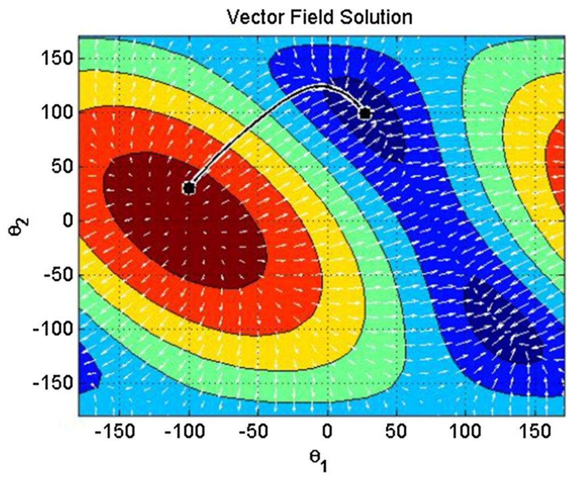

Fig. 5 Vector field of the error gradient

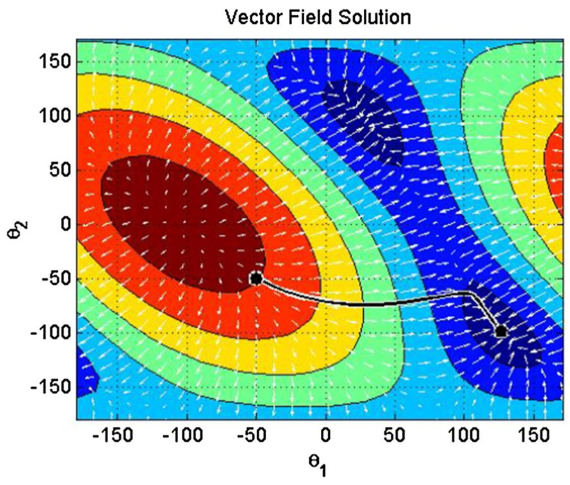

Fig. 7 Vector field of the error gradient

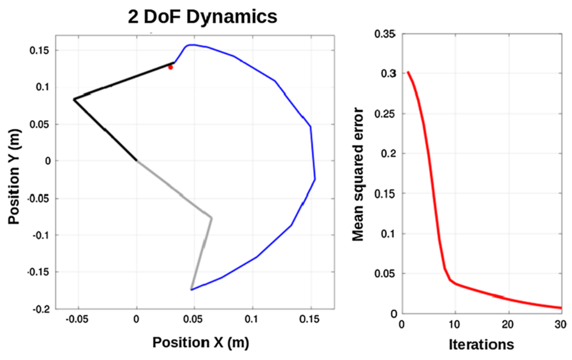

Fig. 6 Evolution of manipulator kinematics (left) and evolution of the mean square error (right)

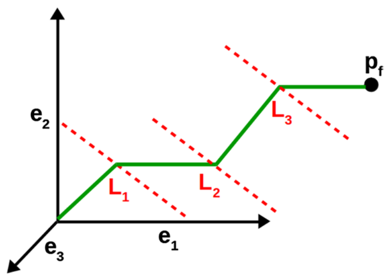

Fig. 8 Forward kinematics of a kinematic chain employing screw rotors

Figure 6 shows the evolution of the manipulator kinematics by staring from a different home position $\theta(0) = [-50.0, -50.0]$, and the same desired end-effector position ($p_d = [0.0292, 0.1267]$). As can be seen, the desired point is reached through a different configuration; this happens because the vector field carries the initial conditions to the closest solution (see Fig. 7), so the presented algorithm can obtain the solution with the configuration closest to its initial condition, as noted in the previous experiments.

4 Conformal geometric algebra algorithm extension

In this section, an extension of the previously presented algorithm is developed employing Conformal Geometric Algebra, using screw rotors and the gradient descent algorithm to find the inverse position kinematics for n-degree-of-freedom kinematic chains with revolute joints.

4.1 Forward kinematics

The forward kinematics for a serial robot arm of n joints (Fig. 8) can be represented by a succession of screw rotor's operations as [21]

$$p_f = \prod_{i=1}^{n} M_i p_0 \prod_{i=1}^{n} \tilde{M}_{n-i+1} \tag{43}$$

where :

$$M_i = e^{-\frac{\theta_i}{2}L_i} = \cos\left(\frac{\theta_i}{2}\right) - \sin\left(\frac{\theta_i}{2}\right)L_i, \tag{44}$$

$$L_i = (a_i - b_i)I_e - e_\infty(a_i \wedge b_i)I_e, \tag{45}$$

$$p_0 = p_{0e} + \frac{1}{2}p_{0e}^2 e_\infty + e_0. \tag{46}$$

4.2 Inverse position kinematics

By considering p_d and $p_f \in \mathbb{G}_{4,1}$ as the desired, and the current end-effector position respectively, in Conformal Geometric Algebra the distance between both can be calculated through the squared root [9]:

$$d(p_d, p_f) = \sqrt{-2p_d \cdot p_f}, \tag{47}$$

where equivalently, for the Euclidean points $p_{de}, p_{fe} \in \mathbb{G}_{3,0}$:

$$\begin{aligned}\mathcal{L}(p_d, p_f) &= -2(p_d \cdot p_f)\\ &= p_{de} \cdot p_{fe} - 2p_{de} \cdot p_{fe} + p_{de}^2\\ &= (p_{de} - p_{fe})^2,\end{aligned} \tag{48}$$

According to the error function (48), its partial derivate over the end-effector position can be defined as:

$$\frac{\partial \mathcal{L}}{\partial p_f} = \frac{\partial}{\partial p_{fe}}\left(\frac{1}{2}(p_{de} - p_{fe})^2\right) = -(p_{de} - p_{fe}), \tag{49}$$

and the partial derivates of the end-effector position over the joint angles as [21]

$$\begin{aligned}\frac{\partial p_f}{\partial \theta_j} &= \frac{\partial}{\partial \theta_j}\prod_{i=1}^{n} M_i p_0 \prod_{i=1}^{n} \tilde{M}_{n-i+1}\\ &= p_f \cdot \prod_{i=1}^{j-1} M_i L_j \prod_{i=1}^{j-1} \tilde{M}_{j-i} = p_f \cdot L_j'.\end{aligned} \tag{50}$$

Thus, according to the chain rule the full gradient of the error function over the joint angles can be defined as:

$$\frac{\partial \mathcal{L}}{\partial \theta_j} = \frac{\partial \mathcal{L}}{\partial p_f} \cdot \frac{\partial p_f}{\partial \theta_j} = -(p_{de} - p_{fe}) \cdot p_f \cdot L_j', \tag{51}$$

where L_j' is the line L_j rotated by the previous joints in the kinematic chain;

$$L_j' = \prod_{i=1}^{j-1} M_i L_j \prod_{i=1}^{j-1} \tilde{M}_{j-i} \tag{52}$$

By considering the previous gradient definition (51), and by employing the gradient descent algorithm, an optimization rule to get the robot configurations θ_i can be defined as

$$\begin{aligned}\theta_i(h) &= \theta_i(h-1) - \alpha\frac{\partial \mathcal{L}(p_d, p_f)}{\partial \theta_i(h)}\\ &= \theta_i(h-1) - \alpha(p_{fe} - p_{de}) \cdot p_f \cdot L_i',\end{aligned} \tag{53}$$

where

$$p_f = \prod_{i=1}^{n} M_i p_0 \prod_{i=1}^{n} \tilde{M}_{n-i+1}, \tag{54}$$

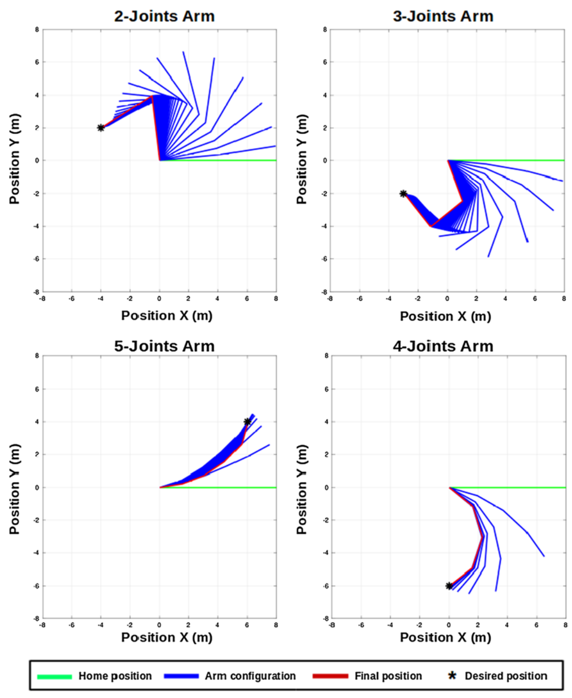

Fig. 9 Algorithm employed to solve the inverse position kinematics from two to six joints kinematic chains

$$L'_j = \prod_{i=1}^{j-1} M_i L_j \prod_{i=1}^{j-1} \tilde{M}_{j-i}, \tag{55}$$

and $\tilde{M}$ is the screw rotor conjugate of M, defined as

$$\tilde{M}_i = e^{\frac{\theta_i}{2} L_i} = \cos\left(\frac{\theta_i}{2}\right) + \sin\left(\frac{\theta_i}{2}\right) L_i. \tag{56}$$

4.3 Algorithm implementation

To evaluate the algorithm different kinematics chains were employed to solve a proposed set of end-effector trajectories according the following scheme:

– As a first experiment, different 2D kinematic chains were employed to reach a set of end-effector positions, by considering different joint numbers.
– As a second experiment, the kinematic arm presented in Fig. 10 was employed to solve two end-effector trajectories: an inclined circle and a lemniscate of Bernoulli with a length of $m = 126$ points.
– According to the proposed update rule (53), the joint position was updated iteratively once the distance between the end-effector and every point (47) fell below the threshold $\mathcal{L} < 0.01\, m^2$.
– For the second experiment, once a point on the trajectory is reached, the current joint positions are taken as a starting point for the next calculation.

Simulations were carried out using MATLAB by developing a new library [7]. These results are presented in Fig. 9 for the first experiment in a 2D space, and in Figs. 11 and 12 for the second experiment using the five-joint kinematic arm in a 3D space. For visual purposes in the second experiment, an additional quadratic interpolation was

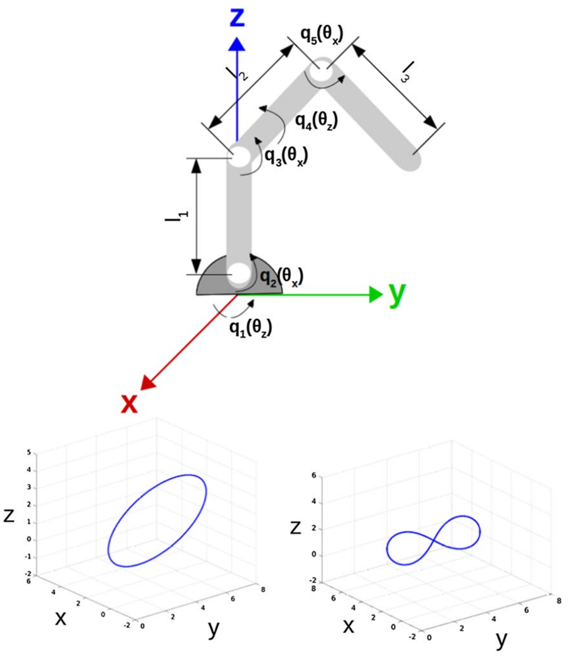

Fig. 10 Use of a 5-DoF kinematic arm, where $l_1 = l_2 = l_3 = 3.0$ m. The required end-effector trajectories are presented on the bottom

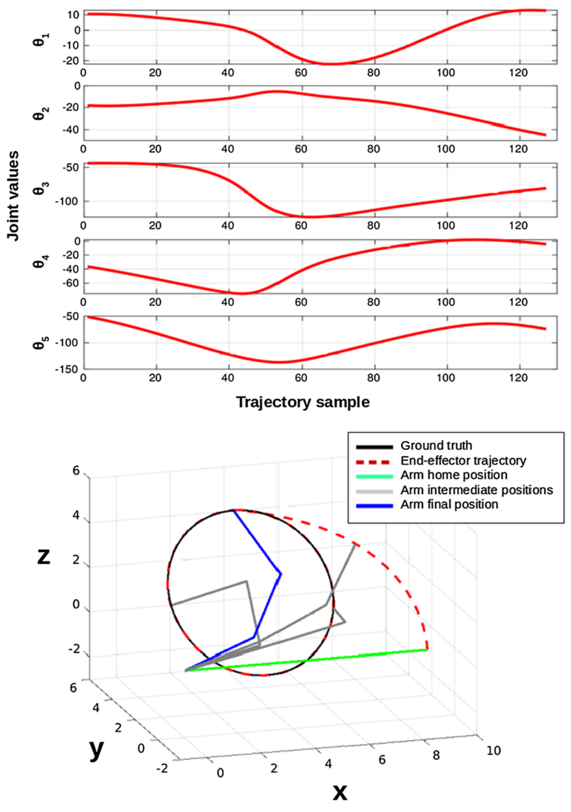

Fig. 11 Kinematic arm behavior by employing the inverse kinematics solution for the inclined circle trajectory. Top: Joint trajectories calculated for the circular trajectory. Bottom: Arm behavior

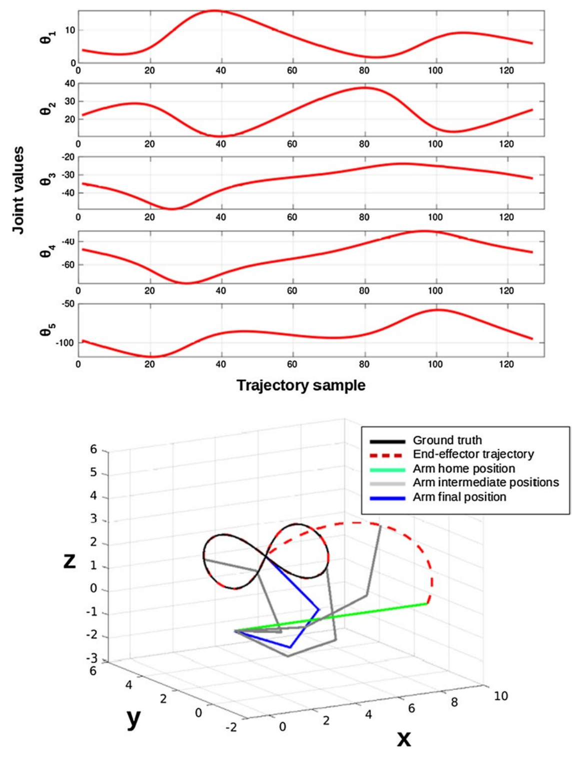

Fig. 12 Kinematic arm behavior by employing the inverse kinematics solution for Bernoulli's lemniscate trajectory. Top: Joint trajectories. Bottom: Arm behavior

carried out from the home position to the first point solved on the trajectories.

As can be seen in Fig. 9, the algorithm is capable of solving the position inverse kinematics for all cases, making it possible to reach the desired end-effector position iteratively. On the other hand, the algorithm is capable of calculating the required joint positions for both trajectories (Figs. 11, 12). For additional multimedia resources, consult [22].

5 Algorithm comparison

To compare the algorithms presented in Sects. 3 and 4, the 3-DoF kinematic arm presented in Fig. 13 is employed, where $l_1 = l_2 = 0.4$ m. In the following subsections, the arm's forward and inverse kinematics are solved using both algorithms in order to compare their performance.

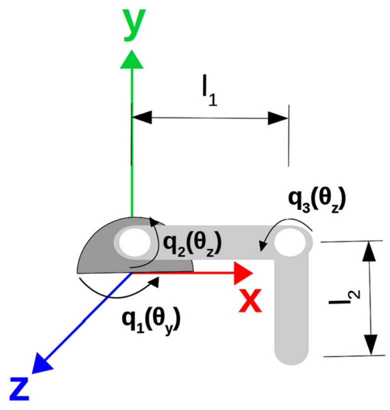

Fig. 13 Three-degree-of-freedom kinematic chain model

5.1 Forward kinematics using quaternions

As seen in Sect. 3, the forward kinematics of the arm can be modeled by defining the set of quaternions

$$q_1 = \cos\frac{\theta_1}{2} + \sin\frac{\theta_1}{2}j, \tag{57}$$

$$q_2 = \cos\frac{\theta_2}{2} + \sin\frac{\theta_2}{2}k, \tag{58}$$

$$q_3 = \cos\frac{\theta_2}{2} + \sin\frac{\theta_2}{2}k, \tag{59}$$

and links:

$$p_1 = 0 + 0i + 0j + 0k, \tag{60}$$

$$p_2 = 0 + l_1 i + 0j + 0k, \tag{61}$$

$$p_3 = 0 + 0i - l_2 j + 0k, \tag{62}$$

which produces the end-effector position in Euclidean space:

$$p_f = \begin{bmatrix} \cos(\theta_1)(l_1\cos(\theta_2) + l_2\sin(\theta_2+\theta_3)) \\ l_1\sin(\theta_2) - l_2 cos(\theta_2+\theta_3) \\ \sin(\theta_1)(l_1\cos(\theta_2) + l_2\sin(\theta_2+\theta_3)) \end{bmatrix} \tag{63}$$

5.2 Forward kinematics using conformal algebra

In a similar way, the forward kinematics of the arm can be modeled by defining the set of screw rotors:

$$p_f = \prod_{i=1}^{3} e^{-\frac{\theta_i}{2}L_i} p_0 \prod_{i=1}^{3} e^{\frac{\theta_i}{2}L_{3-i+1}}, \tag{64}$$

where:

$$p_0 = l_1 e_1 - l_2 e_2 + \frac{l_1^2 + l_2^2}{2} e_i + e_0, \tag{65}$$

$$L_1 = -e_{13}, \tag{66}$$

$$L_2 = -e_{12}, \tag{67}$$

$$L_3 = -e_{12} + l_1 e_{24} + l_2 e_{25}. \tag{68}$$

5.3 Inverse position kinematics

Once the forward kinematics are calculated usingh both methods, the gradients presented in Eqs. (32) and (51) are used to move the end-effector to the desired position $p_{des} = [0.2, 0.4, 0.2]$ using the same home position $(\theta(0) = [180.0, 0.0, 90.0])$, learning rate $(\alpha = 0.5)$ and update rule (gradient descent).

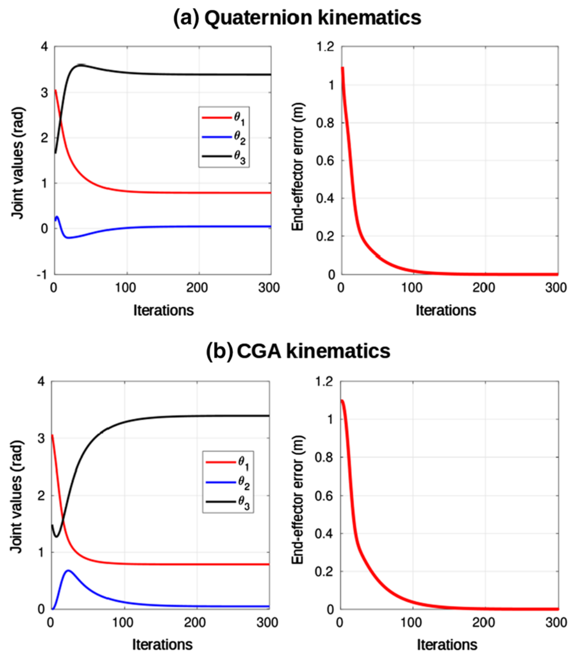

Fig. 14 Iterative solution of the inverse position kinematics using the proposed methods; Quaternion Algebra (Top), and Conformal Geometric Algebra (Bottom)

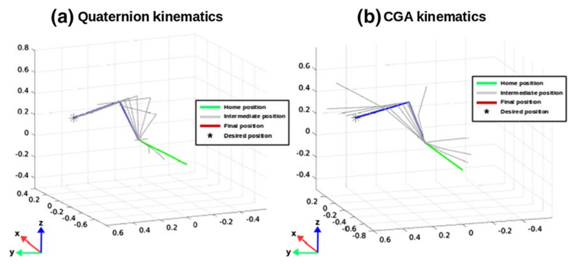

Fig. 15 Iterative solution of the inverse position kinematics using every algorithm. Left: Quaternion Algebra (QA) solution. Right: Conformal Geometric Algebra (CGA) solution

These results are presented in Figs. 14 and 15 for both algorithms: The first shows the error comparison and joints evolution through the update iterations, while the second some 3D positions of the arm behavior. As can be seen, both algorithms are able to displace the end-effector to the desired position, where the same joint values are reached by the algorithms using a similar number of iterations.

As the Gradient Descent is a practical and easily implementable update rule for optimization, depending on the initial conditions and the existence of local minimums, reaching an optimal solution can be difficult, to afford this problem some improved algorithm variations have been developed [23, 24] . In practice any update rule is easily implementable, as the gradient expressions (Eqs. (32) and (51)) are determined analytically using both methods.

As a final test, the algorithms were employed to find the joints required to reach a set of points using the Adam [23] update rule ($\alpha = 0.05$ and recommended hyperparameters), according the following procedure:

– A total set of 200 points distributed in a sphere of radius $r = 0.2$ m centered at [0.4, 0.6, 0.0] is employed.
– The kinematic arm presented in Fig. 13 with home position at $\theta(0) = [0.0, 90.0, 0.0]$ was employed, by considering the dimensions $l_1 = l_2 = 0.6$ m.
– For every point, both algorithms were employed to find the joints required to reach them, starting from the home position.
– The algorithms were stopped once an error measurement below the limit $e < 0.005$ m was reached, then an average of the iterations required to reach every point was calculated.

Table 1 Average iterations required to find a solution for the sphere of points, employing Adam [23] through the QA and CGA algorithms

Algorithm	Required iterations
QA Algorithm	82.52
CGA Algorithm	64.41

These results are presented in Table 1, where both methods requires a similar number of iterations to find the solution. By comparing the methods:

– An advantage of the CGA algorithm is that the gradients are determined analytically by employing generic formulas (Eqs. (52) and (51)), while for the QA method, the forward kinematics expression must be determine first, and then derivated according the particular configuration of the arm.
– An advantage of the QA algorithm is that the forward kinematics can be easily transformed into Euclidean coordinates, while for the CGA method additional transformations must be developed to go from the Conformal to the Euclidean space.

6 Geomagic touch implementation

In this section, a Geomagic Touch haptic device is employed using the Quaternion Algebra algorithm presented in Sect. 3 to solve its inverse position kinematics, following a proposed end-effector trajectory in a 3D space.

6.1 Forward kinematics

As a first step, the forward kinematics model of a Geomagic Touch haptic device is obtained using quaternions. The device is presented in Fig. 16 [25], where $l_1 = l_2 = 0.135$ m represent the length of its links, while $a = 0.035$ m, $l_4 = l_1' + a$ and $l_3 = 0.025$ m represent auxiliary physical measurements to obtain the kinematic model.

Figure 17 shows the model's free-body scheme, from which the respective forward kinematics equations will be generated. The first task is to propose the equation of each joint when all the angles are at the initial position (zero), which generates, in the quaternion notation, the following:

$$p_1 = 0 + 0i + 0j + l_1 k, \tag{69}$$

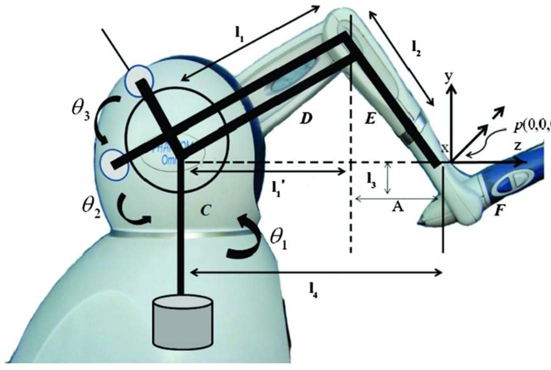

Fig. 16 General structure of the Geomagic Touch haptic device [25]

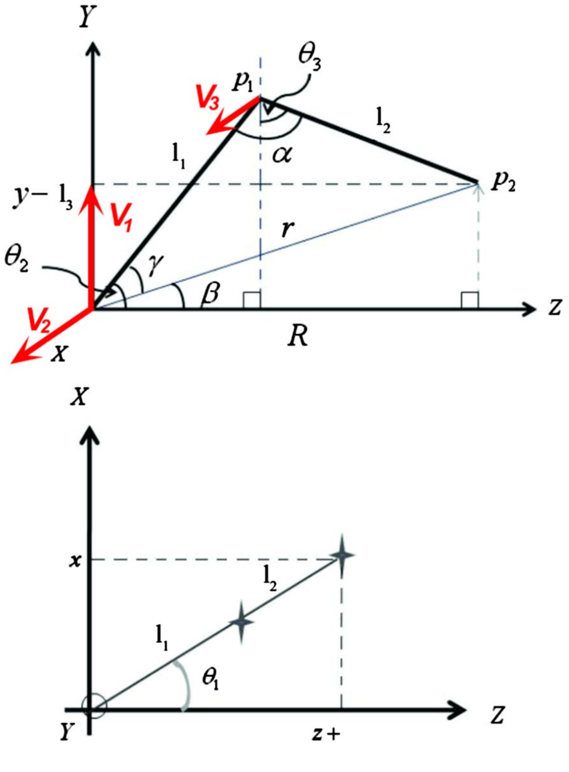

Fig. 17 Free-body model of the Geomagic Touch haptic device, in the Y-X / Z plane (top figure) and in the XZ plane (bottom figure)

$$p_2 = 0 + 0i - l_2 j + 0k, \tag{70}$$

whose rotation vectors v_1, v_2 and v_3 are:

$$v_1 = 0i + 1j + 0k, \tag{71}$$

$$v_2 = 1i + 0j + 0k, \tag{72}$$

$$v_3 = 1i + 0j + 0k, \tag{73}$$

where the rotation quaternions remain

$$q_1 = \cos\frac{\theta_1}{2} + \sin\frac{\theta_1}{2} j, \tag{74}$$

$$q_2 = \cos\frac{\theta_2}{2} + \sin\frac{\theta_2}{2} i, \tag{75}$$

$$q_3 = \cos\frac{\theta_3}{2} + \sin\frac{\theta_3}{2} i. \tag{76}$$

In this particular case, the origin of the coordinates is out of phase with the origin of the kinematic chain, so the following quaternion is employed to correct this offset:

$$q_d = 0 + 0i + l_3 j - l_4 k. \tag{77}$$

Finally, the equation that describes the movement of the end-effector (p_f) is obtained using

$$\begin{aligned} p_f &= q_1 q_2 L_1 \widetilde{q_2} \widetilde{q_1} + q_1 q_2 q_3 L_2 \widetilde{q_3} \widetilde{q_2} \widetilde{q_1} + q_d \\ &= \begin{bmatrix} 0 \\ -\sin(\theta_1)(l_1\cos(\theta_2) + l_2\sin(\theta_3)) \\ l_3 + l_1\sin(\theta_2) - l_2\cos(\theta_3) \\ -l_4 + \cos(\theta_1)(l_1\cos(\theta_2) + l_2\sin(\theta_3)) \end{bmatrix}, \end{aligned} \tag{78}$$

which is the same forward kinematics model used for the Geomagic Touch haptic device obtained in [25] using traditional techniques.

6.2 Inverse position kinematics

In this section, the inverse position kinematics of the Geomagic Touch haptic device are obtained using the algorithm presented in Sect. 3. The initial conditions of the angles of the haptic device are $\theta(0) = [57.29, 0, -57.29]$ (arbitrarily proposed); this configuration results in the end-effector being located at the initial position $p_f = [-0.0614, -0.2236, -0.1306]$ m in cartesian coordinates, with a desired position at $p_d = [0, 0, 0]$ m (also arbitrary). Figure 19-subfigure A) shows the three-dimensional vector field created by the gradient descent algorithm to minimize the error function. Moreover, it converges to a solution in which the forward kinematics leads the end-effector to reach the desired position.

Figure 18 shows the change of the angles through the first 30 iterations on the left, the Euclidean error (between the end-effector and the desired position) and; how it converges to zero on the right (red line). In summary, if the angles $\theta = [-16.39, 56.51, 49.54]$ in (78) are evaluated; it can be seen that the end-effector reaches the desired coordinates ($p_f = [0, 0, 0]$).

6.3 PID control

In this last subsection, the quaternion-based algorithm developed to calculate the forward and inverse kinematics is applied to control the position of a Geomagic Touch haptic device (Fig. 16) in real-time to demonstrate that, although the proposed method to find inverse kinematics is an iterative method, it can be easily applied to a real-time control system (in this case, a PID controller).

The control scheme is shown in Fig. 20, where the desired reference (SP) is a point in the operational space, which is compared with the coordinates of the end-effector, and the resulting error is employed to obtain the required configurations through the inverse kinematics algorithm. The PID controller gains are as follows: 3.1 for the proportional, 0.01 for the integral,

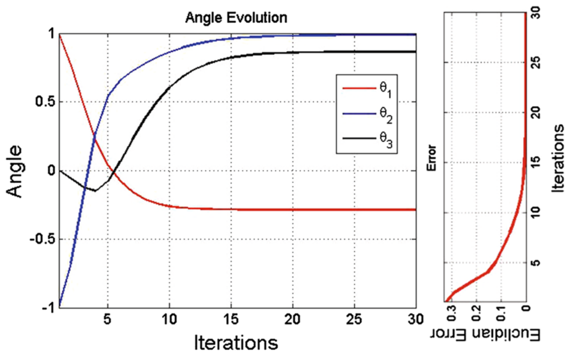

Fig. 18 The change of the three angles through the first 30 iterations is shown. On the right, it can be seen how the error converges to zero through these iterations

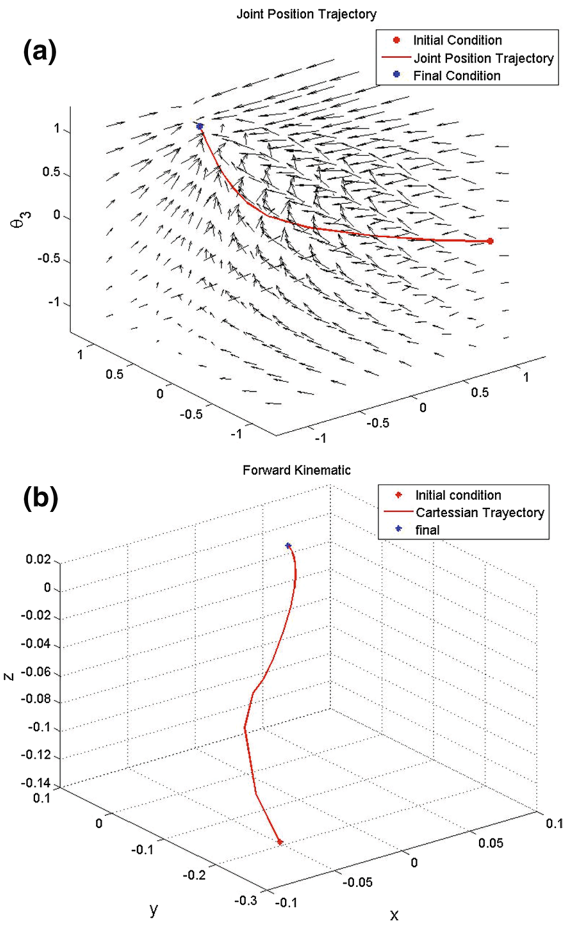

Fig. 19 End effector path (blue line) and powertrain link configuration (black lines)

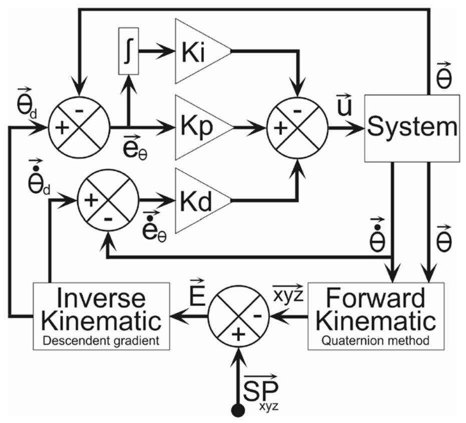

Fig. 20 Control scheme for the PID controller applied to the Geomagic Touch haptic device

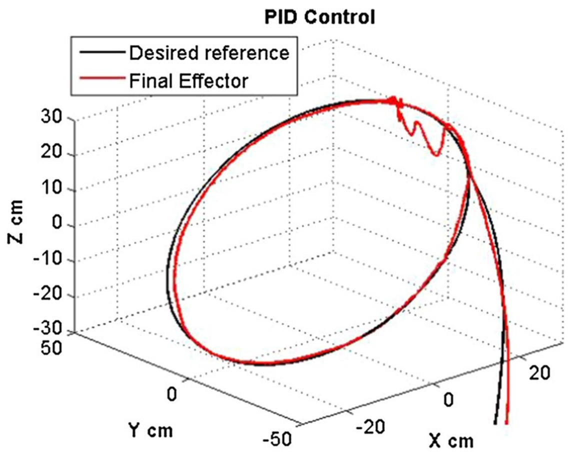

Fig. 21 Response of the Geomagic Touch controlled by the PID control

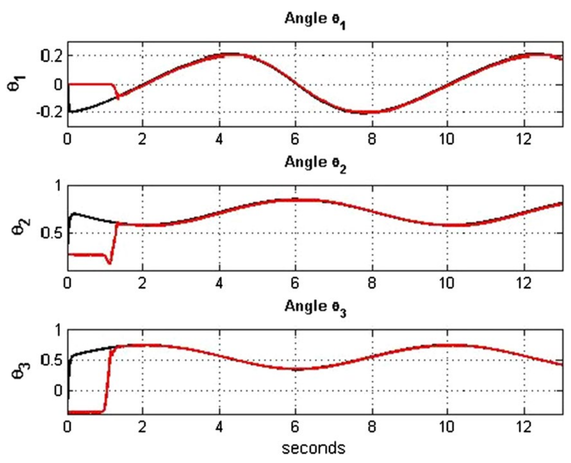

Fig. 22 Evolution of each angle (in radians) of the Geomagic Touch device (red line) and inverse kinematics proposed by the gradient descent algorithm (black line)

and 0.3 for the derivative gain, respectively; these gains were calculated previously with the Ziegler-Nichols tuning technique [26]. Figure 21 also shows the desired path in operational space (black line) as well as the path obtained from the Geomagic Touch haptic device (red line).

As can be seen, the proposed inverse kinematics algorithm works correctly, finding the required joint positions. In Fig. 22, the evolution of each angle through the duration of the experiment is shown separately, whereas in the same case for Fig. 21, the black lines represent the desired path created by the gradient descent technique, and the red lines are the direct reading from the Geomagic Touch device according the end-effector position. As can bee seen, the control scheme allows the haptic to follow the generated trajectory.

7 Conclusions

In this work, a set of algorithms based on gradient descent was proposed to solve the inverse position kinematics for n-degree-of-freedom kinematic chains with revolute joints. In a first approach employing Quaternion Algebra, the algorithm was capable of solving the inverse position and velocity kinematics for any degree-of-freedom model, allowing the end-effector to reach any desired point. Additionally, the algorithm was implemented in real-time, employing a Geomatic Touch Haptic device with a PID controller, proving to be robust and fast enough to be employed in real-time applications. Finally, an algorithm extension using Conformal Geometric Algebra was also proposed, developing update rules that can be easily applied to any n-degree-of-freedom kinematic chains with revolute joints for the inverse position kinematics solution.

In future work, the authors will extend the current algorithms to consider kinematic chains with prismatic joints, as well as other joint configurations, employing Dual Quaternions and CGA Motors.

Acknowledgements The authors would like to thank CONACYT and CINVESTAV-IPN for the scholarship and the economic and technological support for the realization of this work.

Author Contributions All authors contributed to the study conception and design of this work. L. Lechuga-Gutierrez: Paper writing, main contributions in Sects. 2, 3 and 6. E. Macias-Garcia: Paper writing, document review, main contributions in Sects. 2, 4 and 5. G. Martínez-Terán: Main contributions in Sect. 6. J. Zamora-Esquivel: Document review, main contributions in Sects. 4 and 5. E. Bayro-Corrochano: Paper writing, document review, project supervision.

Funding This work was supported by CONACYT institute and CINVESTAV-IPN.

Declarations

Conflict of interest The authors have no conflicts of interest to declare that are relevant to the content of this article.

Code availability The Conformal Geometric Algebra library developed in this work is available at: github.com/iqedgarmg/conformal_library while additional multimedia resources at: drive.google.com/drive/folders/11DfFiQ8wZsfY31VDIK2i-Mg0rnHeucCM.

Ethical approval Not applicable

Consents to participate Not applicable

Consent for publication Not applicable

References

1. Hamilton W (1866) Elements of quaternions. Green & Co, London
2. Hart C, Francis K, Kauffman H (1994) Visualizing quaternion rotation. ACM Trans Graph 13(3):256–276
3. Shoemake K (1985) Animating rotation with quaternion curves. In: Proceedings of the 12th annual conference on computer graphics and interactive techniques, pp 245–254
4. Featherstone R (1983) Position and velocity transformations between robot end-effector coordinates and joint angles. Int J Robot Res 2(2):35–45
5. Bayro-Corrochano E, Daniilidis K, Sommer G (2000) Motor algebra for 3D kinematics: The case of the hand-eye calibration. J Math Imag Vis 13(2):79–99
6. Bayro-Corrochano E, Reyes-Lozano L, Zamora-Esquivel J (2006) Conformal geometric algebra for robotic vision. J Math Imag Vis 24(1):55–81
7. Macias-Garcia E, Zamora-Esquivel J, Bayro-Corrochano E (2020) Conformal geometric algebra library for MATLAB. https://github.com/iqedgarmg/conformal_library. Accessed 12 Dec 2020
8. Hamilton W (1853) Lectures on quaternions. Hodges and Smith, Dublin
9. Bayro-Corrochano E (2020) Geometric algebra application, vol II: Robot modelling and control. Springer, London
10. Goldman R (2010) Rethinking quaternions. Morgan Claypool 4(1):157–157
11. Lechuga-Gutierrez L, Medrano-Hermosillo J, Bayro-Corrochano E (2018) Quaternion spiking neural networks control for robotics. In: 2018 IEEE Latin American conference on computational intelligence, pp 1–6
12. Peijun Y, Keqiang X, Jiancheng L (2011) A design of reconfigurable satellite control system with reaction wheels based on error quaternion model. In: 2011 International conference on internet computing and information services, pp 215–218

13. Vince J (2016) Mathematics for computer graphics. Springer, London
14. XiaoLong Y, HongTao W, Yao L, et al (2019) Computationally efficient inverse dynamics of a class of six-DoF parallel robots: Dual quaternion approach. J Intell Robot Sys 94(1):101–113
15. Cuevas-Jimenez E, Osuna-Enciso J, Oliva-Navarro D (2016) Optimización, algoritmos programados con MATLAB. Alfaomega, México
16. Curry H (1944) The method of steepest descent for non-linear minimization problems. Quarter Appl Math 2(3):258–261
17. Jones M (2005) AI application programming. Charles River Media, Massachusetts
18. Hamming R (1973) Numerical methods for scientists and engineers, 2nd edn. Dover Publications Inc, New York
19. Fuente J, Santiago J, Román A et al (2014) Handbook on robotics, vol 25. Springer, Berling, pp 1682–1690
20. Radavelli L, Martins D, De Pieri E, Simoni R (2015) Cinemática posicional de robôs via iteração e quatérnios. Proc Ser Brazilian Soc Comput Appl Math 3(1):1
21. Bayro-Corrochano E, Zamora-Esquivel J (2007) Differential and inverse kinematics of robot devices using conformal geometric algebra. Robotica 25(1):43–61
22. Lechuga-Gutierrez L, Macias-Garcia E, Martinez-Terán G, Zamora-Esquivel J, Bayro-Corrochano E (2021) Iterative inverse kinematics for robot manipulators using quaternion algebra and conformal geometric algebra: supplementary material. drive.google.com/drive/folders/11DfFiQ8wZsfY31VDIK2i-Mg0rnHeucCM. Accessed 25 Jun 2021
23. Kingma D, Ba J (2014) Adam: a method for stochastic optimization. arXiv:1412.6980
24. Moré J (1978) The Levenberg–Marquardt algorithm: implementation and theory. In: Numerical analysis. Springer, Berlin, pp 105–116
25. Jarillo-Silva A, Domínguez-Ramírez O, Parra-Vega V, Ordaz-Oliver J (2009) Phantom Omni Haptic device: kinematics and manipulability. In: IEEE electronics, robotics and automotive mechanics conference, pp 193–198
26. Ogata K (2010) Ingeniería de control moderna. Pearson Education, S.A. Madrid

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.