sciendo

Int. J. Appl. Math. Comput. Sci., 2023, Vol. 33, No. 2, 171–181
DOI: 10.34768/amcs-2023-0013

amcs

DUAL QUATERNIONS FOR THE KINEMATIC DESCRIPTION OF A FISH–LIKE PROPULSION SYSTEM

ZYGMUNT KITOWSKI [a], PAWEŁ PISKUR [a,*], MATEUSZ ORŁOWSKI [a]

[a]Faculty of Mechanical and Electrical Engineering
Polish Naval Academy
Śmidowicza 69, 81-127 Gdynia, Poland
e-mail: {z.kitowski,p.piskur,m.orlowski}@amw.gdynia.pl

This study discusses the use of quaternions and dual quaternions in the description of artificial fish kinematics. The investigation offered here illustrates quaternion and dual quaternion algebra, as well as its implementation in the software chosen. When it comes to numerical stability, quaternions are better than matrices because a normalised quaternion always shows the correct rotation, while a matrix more easily loses its orthogonality due to rounding errors and oversizing. Although quaternions are more compact than rotation matrices, using quaternions does not always provide less numerical computation and the amount of memory needed. In this paper, an algebraic form of quaternion representation is provided which is less memory-demanding than the matrix representation. All the functions that were used to prepare this work are presented, and they can be employed to conduct more research on how well quaternions work in a specific assignment.

Keywords: quaternions, dual quaternions, artificial fish, underwater vehicle, undulating propulsion system.

1. Introduction

Although quaternions were developed many years ago, they are still not used very often in robotics and autonomous vehicles due to the long tradition of employing homogeneous transformations. Quaternions and dual quaternions are utilised in quantum mechanics (Horwitz and Biedenharn, 1984), rigid body dynamics (Kiciński *et al*., 2021; Kluczyk and Grzadziela, 2015) and for navigation purposes (Naus *et al*., 2021; Felski *et al*., 2020; Jaskólski, 2017; Jaskólski *et al*., 2021) based on an inertial measurement unit (Sola, 2017; Jaskólski *et al*., 2019). In recent years quaternions have become popular in computer graphics (Leclercq *et al*., 2013) for body motion analysis (Pennestrì and Valentini, 2010), robot manipulators (Grzadziela *et al*., 2020) as well as for neuroscience (Ling *et al*., 2022). Numerical precision and efficiency of real-time calculations are critical for robot control systems (Chen *et al*., 2020), particularly for autonomous vehicles (Morawski *et al*., 2020; Hożyń and Zalewski, 2020; Wawrzyński *et al*., 2022).

One of the best known advantages in comparison with the Euler angle rotation matrix is the lack of ambiguous solutions. There is at least one point in every Euler angle sequence that loses a degree of freedom (Mouton, 2021). This is referred to as a gimbal lock (one of the rotation axes realigns with another axis).

According to the study by Radavelli *et al*. (2012), dual quaternions outperform the latter in terms of storage because homogeneous matrices require 12 numbers to express six degrees of freedom, whereas the former require only eight. Furthermore, unlike 3D stiff transformations controlled as a 4×4 homogeneous (overhead) matrix, they can be maintained using two independent components: a translation vector and a quaternion.

Dual quaternions are employed similarly to quaternions but have the added benefit of encapsulating both translation and rotation into a unitary state that can be easily concatenated and interpolated (Jarzebowska and Klak, 2020). Due to this redundancy, there are constraints on the eight values in a matrix to form a valid rotation matrix. The matrix must be orthogonal, which means that the row vectors must be orthonormal (each vector has length 1, and the scalar product of each pair is 0). Numerical errors are introduced while the rotation matrix is updated, often by concatenating it with incremental

*Corresponding author

rotation matrices. These inaccuracies build up with each update and are determined by the precision of numerical representations and the type of trigonometric function calculations used (Taylor series or CORDIC algorithm). The row vectors can go further and further away from being orthonormal. When the matrix deviates sufficiently from orthogonality, it can begin to visibly modify the geometry to which it is applied (skew, scaling, etc). When employing rotation matrices, these issues can be addressed by orthonormalizing the row vectors on a regular basis.

Rounding errors inevitably accumulate when assembling multiple rotations on a computer. When normalized, slightly offset quaternions still indicate rotation, while a slightly offset matrix may no longer be orthogonal and it is more difficult to convert it back to a correct orthogonal matrix. Despite the fact that quaternions are more compact than rotation matrices, using quaternions does not necessarily result in less numerical computation in the applications (Sarabandi and Thomas, 2019).

It can be tedious to reimplement all the basic functions required when working with quaternions, such as composition, conjugation, conversions to and from rotation matrices, and Euler angles. For a review of efficient floating-point algorithms for processing quaternions, see the work of Joldeş and Muller (2020). Because quaternions are a fairly common aspect of engineers' toolboxes, various libraries have been built for a number of scientific programming languages (Piórek and Jabłoński, 2020). However, it is not so simple to comprehend how they work.

The idea of quaternions and dual quaternions is presented in this paper, followed by an example of an analysis of an artificial fish with a hull designed from the connection of rigid components (Piskur *et al.*, 2020b). The artificial fish image captured during the test in a swimming pool is shown in Fig. 1. The same biomimetic underwater vehicle is shown in Fig. 2 for greater visualisation of each tail part connection. The trajectory of each tail fin part in an artificial fish movement control system must be changed as a function of velocity with regard to water. Therefore, constructing a control algorithm for the tail fin linkages requires efficient computation (Piskur *et al.*, 2020a; Piskur, 2022).

2. Fish-like propulsion system

The tail kinematic description for a fish-like movement can be described using the Lighthill equation (Lighthill, 1971):

$$y(x,t) = (c_1 x + c_2 x^2)\sin\Big(\frac{2\pi}{\lambda}x + 2\pi f t\Big), \qquad (1)$$

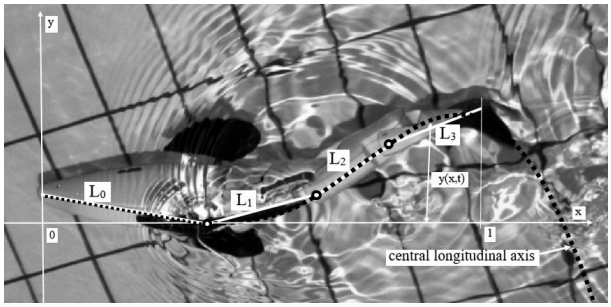

Fig. 1. Artificial fish in a swimming pool with a link description.

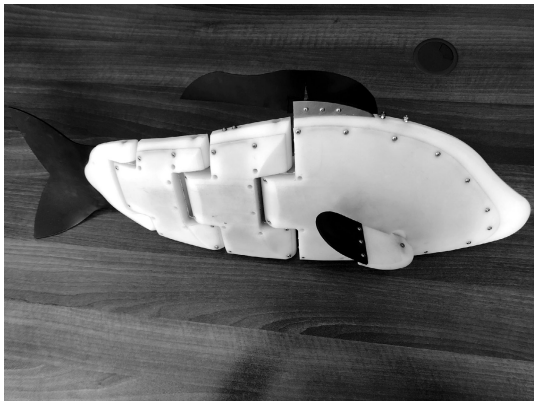

Fig. 2. Image of the artificial fish.

where $y(x,t)$ is the transverse displacement, x is the independent spatial variable, t is time, the second independent variable, c_1 is the primary coefficient of the wave envelope, c_2 is the quadratic term coefficient of the wave envelope, λ is the wavelength, f is the frequency the tail fin movement.

The geometrical representation of Eqn. (1) is depicted as a black curve in Fig. 1 with parameters adopted from Jurczyk *et al*. (2020).

The fishtail kinematics are described by means of three rigid links (L_0, L_1, L_2, L_3) analysed in the article. The difficulty of mathematical description is proportional to the number of links (Piskur *et al*., 2021). This is addressed in the current study using dual quaternions' theory.

3. Theory of quaternions and dual quaternions

Quaternions are an extension of complex number theory that allows the formulation of four-dimensional complex numbers known as hyper-complex ones. They can be used to describe the rotation of rigid bodies in a space around a precisely oriented axis. Dual numbers and dual vectors are special cases of a dual quaternion. A dual number is, in reality, a dual quaternion with a zero vector portion. A

dual vector is a dual quaternion with zero scalar part. The quaternion q can be defined as

$$q = q_0 + q_1\boldsymbol{i} + q_2\boldsymbol{j} + q_3\boldsymbol{k}, \tag{2}$$

where q_0, q_1, q_2, and q_3 are real numbers, $\boldsymbol{i}, \boldsymbol{j}, \boldsymbol{k}$ are imaginary units.

It follows from the above that the quaternion is composed of a scalar part and a vector part:

$$q = q_0 + \boldsymbol{v}_q = (q_0, \boldsymbol{v}_q), \tag{3}$$

where q_0 is a scalar part and $\boldsymbol{v}_q = q_1\boldsymbol{i} + q_2\boldsymbol{j} + q_3\boldsymbol{k}$ is a vector part. A vector $\boldsymbol{v}_q$ is called a pure quaternion in the form of $q = 0 + \boldsymbol{v}_q$. The module of the quaternion is defined as $|q| = \sqrt{q_0^2 + q_1^2 + q_2^2 + q_3^2}$. If quaternion $|q| = 1$ is a unit quaternion, it can be written as

$$q = \cos\left(\frac{\theta}{2}\right) + \frac{\mathbf{v}_q}{|\mathbf{v}_q|}\sin\left(\frac{\theta}{2}\right), \tag{4}$$

where θ is in range $[0, \pi]$.

Based on the normalisation process, the axis of rotation is specified as a unit quaternion. When the quaternion's scalar part is compared to 1, it remains unaffected. The quaternion conjugate is written as follows:

$$\bar{q} = q_0 - q_1\boldsymbol{i} - q_2\boldsymbol{j} - q_3\boldsymbol{k} = q_0 - \boldsymbol{v}_q = (q_0, -\boldsymbol{v}_q). \tag{5}$$

If two quaternions q and p are given as

$$\begin{aligned} q &= q_0 + q_1\boldsymbol{i} + q_2\boldsymbol{j} + q_3\boldsymbol{k} = (q_0, \boldsymbol{v}_q), \\ p &= p_0 + p_1\boldsymbol{i} + p_2\boldsymbol{j} + p_3\boldsymbol{k} = (p_0, \boldsymbol{v}_p), \end{aligned} \tag{6}$$

the following arithmetic operations can be performed:

- addition:

$$q + p = (q_0, \boldsymbol{v}_q) + (p_0, \boldsymbol{v}_p) = (q_0 + p_0, \boldsymbol{v}_q + \boldsymbol{v}_p) \tag{7}$$

or

$$q+p = (q_0+p_0)+(q_1+p_1)\boldsymbol{i}+(q_2+p_2)\boldsymbol{j}+(q_3+p_3)\boldsymbol{k}, \tag{8}$$

- subtraction:

$$q-p = (q_0-p_0)+(q_1-p_1)\boldsymbol{i}+(q_2-p_2)\boldsymbol{j}+(q_3-p_3)\boldsymbol{k} \tag{9}$$

or

$$q-p = (q_0, \boldsymbol{v}_q)-(p_0, \boldsymbol{v}_p) = (q_0-p_0, \boldsymbol{v}_q-\boldsymbol{v}_p), \tag{10}$$

- multiplication:

$$\begin{aligned} q \cdot p = {} & (q_0p_0 - q_1p_1 - q_2p_2 - q_3p_3) \\ & + (q_1p_0 + q_0p_1 + q_3p_2 - q_2p_3)\boldsymbol{i} \\ & + (q_2p_0 + q_3p_1 + q_0p_2 - q_1p_3)\boldsymbol{j} \\ & + (q_3p_0 - q_2p_1 + q_1p_2 + q_0p_3)\boldsymbol{k} \end{aligned} \tag{11}$$

or

$$q \cdot p = q_0 \cdot p_0 - \boldsymbol{v}_q \cdot \boldsymbol{v}_p + q_0 \cdot \boldsymbol{v}_q + p_0 \cdot \boldsymbol{v}_p + \boldsymbol{v}_q \times \boldsymbol{v}_p, \tag{12}$$

where

$$q_0 \cdot p_0 - \boldsymbol{v}_q \cdot \boldsymbol{v}_p$$

is a scalar part, and

$$q_0 \cdot \boldsymbol{v}_q + p_0 \cdot \boldsymbol{v}_p + \boldsymbol{v}_q \times \boldsymbol{v}_p$$

is a vector part of the new quaternion.

It is worth mentioning that quaternion multiplication is not commutative

$$q \cdot p \neq p \cdot q. \tag{13}$$

Taking Eqns. (11) and (12) into account, it can be seen that the manner of defining mathematical relations determines the memory required for quaternion definitions. As a result, while determining which type of description is more successful, the form of the mathematical presentation and the amount of memory required should be considered.

Taking into consideration the multiplication of quaternion

$$q = \begin{bmatrix} q_0 & -q_3 & q_2 & q_1 \\ q_3 & q_0 & -q_1 & q_2 \\ -q_2 & q_1 & q_0 & q_3 \\ -q_1 & -q_2 & -q_3 & q_0 \end{bmatrix} \tag{14}$$

by the column vector p,

$$\begin{bmatrix} q_0 & -q_3 & q_2 & q_1 \\ q_3 & q_0 & -q_1 & q_2 \\ -q_2 & q_1 & q_0 & q_3 \\ -q_1 & -q_2 & -q_3 & q_0 \end{bmatrix} \cdot \begin{bmatrix} p_0 \\ p_1 \\ p_2 \\ p_3 \end{bmatrix}, \tag{15}$$

gives the same results as Eqns. (11) and (12).

The quaternion presented in the matrix form (14) requires more memory space than the quaternion supplied in the equation form (2). As a result, the quaternion representation as a matrix is only used in this study to better explain the notion of imaginary units: $\boldsymbol{i}, \boldsymbol{j}$, and $\boldsymbol{k}$.

If only one component is equal to 1 and the others are equal to 0, this yields the following quaternion represented as a matrix:

$$\boldsymbol{i} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0 \end{bmatrix}, \tag{16}$$

$$\boldsymbol{j} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \end{bmatrix}, \tag{17}$$

$$\boldsymbol{k} = \begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & -1 & 0 \end{bmatrix}. \tag{18}$$

It can be seen that the following conditions are true:

$$\boldsymbol{i}^2 = \boldsymbol{j}^2 = k^2 = \boldsymbol{i} \cdot \boldsymbol{j} \cdot \boldsymbol{k} = -1 \tag{19}$$

and

$$\begin{aligned} \boldsymbol{ij} &= \boldsymbol{k}, \\ \boldsymbol{jk} &= \boldsymbol{i}, \\ \boldsymbol{ki} &= \boldsymbol{j}, \\ \boldsymbol{ji} &= -\boldsymbol{k}, \\ \boldsymbol{kj} &= -\boldsymbol{i}, \\ \boldsymbol{ik} &= -\boldsymbol{j}. \end{aligned} \tag{20}$$

Also, it is worth mentioning that

$$\begin{aligned} \boldsymbol{i}^0 &= 1, \\ \boldsymbol{i}^1 &= \boldsymbol{i}, \\ \boldsymbol{i}^2 &= -1, \\ \boldsymbol{i}^3 &= -\boldsymbol{i}, \\ \boldsymbol{i}^4 &= 1, \\ &\vdots \end{aligned} \tag{21}$$

The aforementioned relation is utilised for quaternion multiplication and also helps to clarify the meaning of relations equal to -1 .

The multiplication table for quaternions and dual quaternions is shown in Table 1. Due to changes in multiplication order (see Eqn. (13)), the suffixes $a.\epsilon$ and $b.\epsilon$ are placed in the table.

3.1. Rotation point. In a three-dimensional space, a vector is expressed as a pure quaternion (a quaternion with no real part). A rotation is denoted by a quaternion n, with the additional constraint that its norm equals 1. Rotation point $P = (x_p, y_p, z_p)$ around vector $\vec{n} = [n_x, n_y, n_z]$ of length $|n|$ by angle θ, can be provided based on Algorithm 1. Also, the code function is suffixed in Appendix A. The following quaternion multiplication is provided to orient a vector p using a rotation:

$$p_R = q\, p\, \bar{q}, \tag{22}$$

where p is is the point before rotation, specified as a quaternion, q is the axis of rotation encapsulated as quaternion, $\bar{q}$ is the quaternion conjugate, p and $\bar{p}$ are represented as pure quaternions, i.e., quaternions whose real part are zero:

$$p = x_p\boldsymbol{i} + y_p\boldsymbol{j} + z_p\boldsymbol{k}. \tag{23}$$

The imaginary component describes the point coordinates.

3.2. Interpolation of the rotation. Quaternions are highly efficient for angle interpolation. Based on Algorithm 2 the point $P = (1, 0, 0)$ has been rotated 90 deg around the vector $v = (1, 1, 1)$ in 100 equal steps to demonstrate angle interpolation (see the results in Fig. 3). The code can be found in Appendix B. This algorithm takes only two quaternion multiplications, as shown in Eqn. (22); hence modest computing costs and memory utilisation can be predicted.

The fact that quaternions are not commutative also corresponds directly to the fact that rotations are not commutative (see Eqn. (13)).

This can be insufficient to represent the kinematics of a rigid body because motion is often a combination of rotation and translation. Only the end of the tail link in the artificial fish may be estimated using quaternions. The displacement of the remaining links must be determined using dual quaternions.

The theory of dual quaternions is explained in the following paragraph for further rotation and translation operations.

3.3. Dual quaternions. Dual quaternions are the concatenation of quaternion and dual-number theory, which means that quaternions represent the numbers in the dual-number equation. For rotation and translation, dual quaternions have the form

$$\sigma = p + \epsilon q, \tag{38}$$

where p and q are both quaternions and ϵ is the second order nilpotent dual factor, with relation $\epsilon^2 = 0$ and $\epsilon \neq 0$. They adhere to the quaternion algebra principles.

A dual quaternion can be represented as an 8-dimensional vector or as a juxtaposition of two 4-dimensional vectors representing the dual quaternion's two quaternion components (Leclercq *et al.*, 2013). Using the quaternions representation from Eqn. (6), the dual quaternion can be expressed as follows:

$$\sigma = (p_0, p_1, p_2, p_3, q_0, q_1, q_2, q_3). \tag{39}$$

By analogy with the operation on dual numbers, dual quaternions $\sigma_A = p_A + \epsilon q_A$ and $\sigma_B = p_B + \epsilon q_B$ can be added:

$$\sigma_A + \sigma_B = (p_0 + q_0) + \epsilon(\boldsymbol{v}_I + \boldsymbol{v}_2), \tag{40}$$

and multiplied,

$$\sigma_A \cdot \sigma_B = p_0 \cdot q_0 + \epsilon(\boldsymbol{v}_I \cdot \boldsymbol{v}_2 + \boldsymbol{v}_2 \cdot \boldsymbol{v}_I). \tag{41}$$

By setting the dual component to zero, the dual quaternion can express pure rotation in the same way as a quaternion. To represent a pure translation with no rotation, the real part can be set to the identity, and the dual part represents the translation.

There are three conjugates applied for calculation:

Table 1. Multiplication table for dual quaternions.

a· b	b.1	b.i	b.j	b.k	b.ϵ	b.ϵ i	b.ϵ j	b.ϵ k
a.1	1	i	j	k	ϵ	ϵi	ϵj	ϵk
a.i	i	-1	k	$-$j	ϵi	$-\epsilon$	$-\epsilon$k	ϵj
a.j	j	$-$k	-1	i	ϵj	ϵk	$-\epsilon$	$-\epsilon$i
a.k	k	j	$-$i	-1	ϵk	$-\epsilon$j	ϵi	$-\epsilon$
a.ϵ	ϵ	$-\epsilon$i	$-\epsilon$j	$-\epsilon$k	0	0	0	0
a.ϵ i	ϵi	ϵ	$-\epsilon$k	ϵj	0	0	0	0
a.ϵ j	ϵj	ϵk	ϵ	$-\epsilon$i	0	0	0	0
a.ϵ k	ϵk	$-\epsilon$j	ϵi	ϵ	0	0	0	0

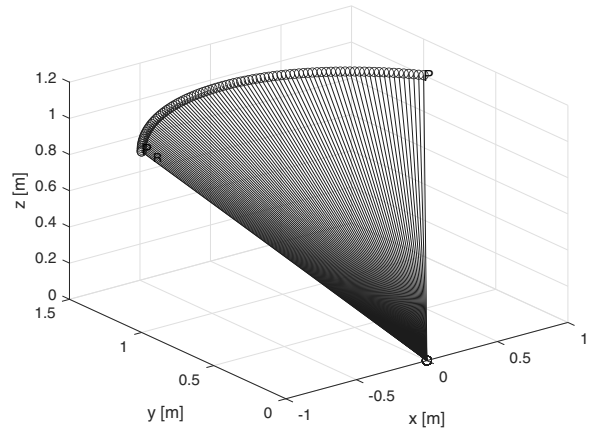

Fig. 3. Interpolation of point P, where point P_R indicates the final position.

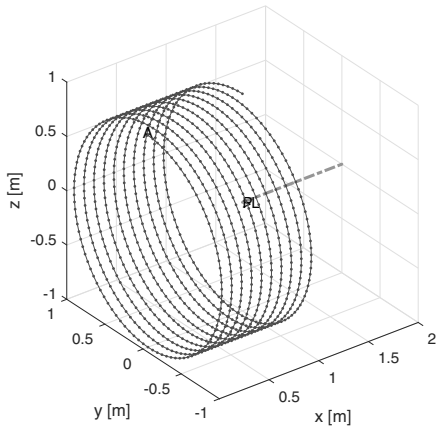

Fig. 4. Rotation and translation based on dual quaternions (A: rotational point, L: axis of rotation, P: point on the surface of the rotation and translation).

- the conjugate of the dual number:

$$\sigma^{\bullet} = (p_0, p_1, p_2, p_3, -q_0, -q_1, -q_2, -q_3), \quad (42)$$

- the conjugate adopted from the classical quaternion conjugation:

$$\sigma^{*} = (p_0, -p_1, -p_2, -p_3, q_0, -q_1, -q_2, -q_3), \quad (43)$$

- a combination of the first two conjugations:

$$\sigma^{\circ} = (p_0, -p_1, -p_2, -p_3, -q_0, q_1, q_2, q_3). \quad (44)$$

For each of the three conjugates, the conjugate of the conjugate of σ is σ itself. The conjugate of the product of dual quaternions equals the product of the individual conjugates of these dual quaternions but in the reverse form. In analogy to the quaternion, dual quaternions can be used for calculation of coordinates by multiplication of dual quaternions:

$$P_{R,T} = \widehat{q}_R \widehat{q}_T P \widehat{q}_T^{\circ} \widehat{q}_R^{\circ}, \quad (45)$$

where $\widehat{q}_R$ and $\widehat{q}_R^{\circ}$ stand respectively for the dual quaternion and its conjunction that describe the rotational movement, $\widehat{q}_T$ and $\widehat{q}_T^{\circ}$ mean the dual quaternion and its conjunction that describe the translational movement.

An example of the rotation and translation of the point A made in 1000 steps is shown in Fig. 4. Just like a 4×4 matrix stores rotation and translation information about an object, a dual quaternion stores the same type of information but in two different quaternions.

The multiplication shown in Eqn. (45) can be applied as many times as the number of translations and rotations required for kinematic description. The number is determined by the number of frames that must be rotated and the number of translations. In the following paragraph, Eqn. (45) is used to describe the kinematics of a fish-like propulsion system.

3.4. Fishtail kinematic description based on dual quaternions. Rotation and translation in terms of the description of the artificial fishtail kinematics is a special case in which the displacement takes place

Algorithm 1. Rotation of the point.

Rotation of the point $P = [x_p, y_p, z_p]^T$ around the vector: $n = [n_x, n_y, n_z]$ of the length $|n|$ by the angle 2θ.

Step 1. Compute the next values:
q_0, q_1, q_2, q_3, unit quaternion coefficients:

$$\begin{aligned} q_0 &= \cos\theta, \\ q_1 &= \frac{n_x}{|n|}\sin\theta, \\ q_2 &= \frac{n_y}{|n|}\sin\theta, \\ q_3 &= \frac{n_z}{|n|}\sin\theta, \end{aligned} \tag{24}$$

where $|n| = \sqrt{n_x^2 + n_y^2 + n_z^2}$.

Step 2a. Calculate Q, the rotation matrix:

$$Q = \begin{bmatrix} 1-2(q_2^2+q_3^2) & 2(q_1q_2 - q_3q_0) & 2(q_1q_3 + q_2q_0) \\ 2(q_1q_2 + q_3q_0) & 1-2(q_1^2+q_3^2) & 2(q_2q_3 - q_1q_0) \\ 2(q_1q_3 - q_2q_0) & 2(q_2q_3 + q_1q_0) & 1-2(q_1^2+q_2^2) \end{bmatrix}. \tag{25}$$

Step 3a. According to Eqn. (11) calculate the new coordinates:

$$P_R = Q \cdot P. \tag{26}$$

Step 2b. Define the next quaternions:
q, quaternion of rotation defined as

$$q = q_0 + q_1\boldsymbol{i} + q_2\boldsymbol{j} + q_3\boldsymbol{k}, \tag{27}$$

$\bar{q}$, conjunction of quaternion q, equal to

$$\bar{q} = q_0 - q_1\boldsymbol{i} - q_2\boldsymbol{j} - q_3\boldsymbol{k}, \tag{28}$$

p, rotation point encapsulated as

$$p = 0 + x_P\boldsymbol{i} + y_P\boldsymbol{j} + z_P\boldsymbol{k}. \tag{29}$$

Step 3b. According to Eqn. (22) calculate the new quaternion:

$$p_R = q \cdot p \cdot \bar{q}, \tag{30}$$

where $p_R = 0 + P_{Rx}\boldsymbol{i} + P_{Ry}\boldsymbol{j} + P_{Rz}\boldsymbol{k}$

Step 4b. After the quaternion encapsulation, the vector parts of p_R give the point coordinates $P_R = (P_{Rx}, P_{Ry}, P_{Rz})$ after rotation.

in the two-dimensional space. The forward kinematics approach to the concatenating transform is the same for dual a quaternions and matrices, and it uses simple multiplication to propagate transforms between

Algorithm 2. Interpolation of the rotation.

Rotation of the point $P = [x_p, y_p, z_p]^T$ around the vector: $n = [n_x, n_y, n_z]$ of the length $|n|$ by the angle divided into N equal angles.

Step 1. Compute the next values:

$$|q| = \sqrt{q_0^2 + q_1^2 + q_2^2 + q_3^2}, \quad |v| = \sqrt{q_1^2 + q_2^2 + q_3^2} \tag{31}$$

$$\theta = \arccos\frac{q_0}{|q|}, \qquad u_{q_1} = \frac{q_1}{|v|}, \qquad u_{q_2} = \frac{q_2}{|v|}, \qquad u_{q_3} = \frac{q_3}{|v|}, \tag{32}$$

and then

$$\begin{aligned} \theta_N &= \frac{\theta}{N}, \\ q_{0_N} &= \cos(\theta_N), \\ q_{1_N} &= u_{q_1}\sin\theta_N, \\ q_{2_N} &= u_{q_2}\sin\theta_N, \\ q_{3_N} &= u_{q_3}\sin\theta_N. \end{aligned} \tag{33}$$

Step 2a. Calculate Q_N, rotation matrices:

$$\begin{bmatrix} 1-2(q_{2_N}^2+q_{3_N}^2) & 2(q_{1_N}q_{2_N} - q_{3_N}q_{0_N}) & 2(q_{1_N}q_{3_N} + q_{2_N}q_{0_N}) \\ 2(q_{1_N}q_{2_N} + q_{3_N}q_{0_N}) & 1-2(q_{1_N}^2+q_{3_N}^2) & 2(q_{2_N}q_{3_N} - q_{1_N}q_{0_N}) \\ 1-2(q_{1_N}^2+q_{2_N}^2) & 2(q_{2_N}q_{3_N} + q_{1_N}q_{0_N}) & 1-2(q_{1_N}^2+q_{2_N}^2) \end{bmatrix}. \tag{34}$$

Step 3a. Calculate N times P^I, the coordinates of rotation points:

$$P_N^I = Q_N \cdot P. \tag{35}$$

Step 2b. Define the quaternion:

$$q_N = q_{0_N} + q_{1_N}\boldsymbol{i} + q_{2_N}\boldsymbol{j} + q_{3_N}\boldsymbol{k},$$

its conjunction:

$$\bar{q} = q_{0_N} - q_{1_N}\boldsymbol{i} - q_{2_N}\boldsymbol{j} - q_{3_N}\boldsymbol{k},$$

and calculate N times

$$p_N^I = q \cdot P \cdot \bar{q}, \tag{36}$$

where $p_N^I = 0 + P_{Nx}^I\boldsymbol{i} + P_{Ny}^I\boldsymbol{j} + P_{Nz}^I\boldsymbol{k}$.

Step 3b. Encapsulate quaternions to obtain the coordinates of rotation points:

$$P_N^I = P_{Nx}^I + P_{Ny}^I + P_{Nz}^I. \tag{37}$$

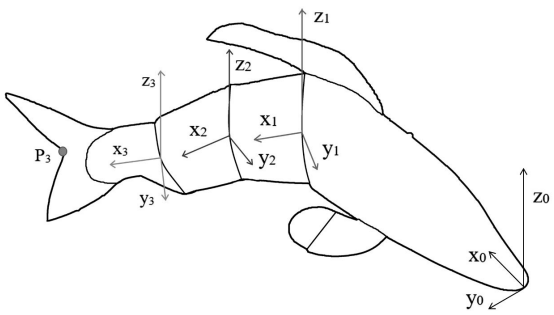

Fig. 5. Artificial fish graph with coordinate systems

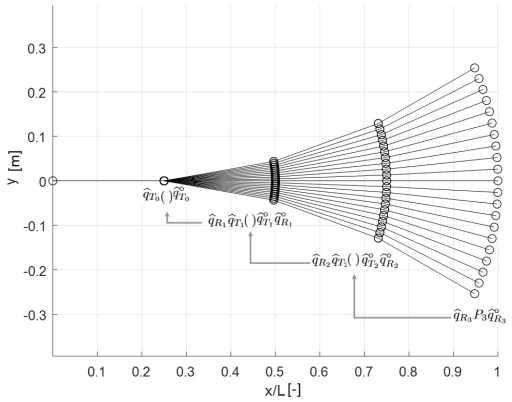

Fig. 6. Final representation of point coordinates (L: the total length of the fish body).

the connected links. Each coordinate is calculated in accordance with a specific convention for attaching reference frames to each stiff section of the artificial fish hull, as shown in Fig. 5. The fish-like description kinematic is offered as a collection of transformations connected with rotation and translations, similar to D–H parameters (Denavit and Hartenberg, 1955). A coordinate system for each joint, a coordinate system to the end-effector, and a reference coordinate system are required by the D–H approach. The coordinate system is associated with each stiff part of the fishtail in this paper's analysis so that the x axis runs along each section and all z axes are parallel to each other (see Fig. 5). The point P_3 is put in the centre of the flexible fin, and the movement description is assumed to be the same as for the rigid body. It rotates by θ_3 degrees in reference to the coordinate system x_3, y_3, z_3. As a result, the following mathematical relation based on dual quaternion algebra can be used to define the coordinates P_3 with reference to the coordinate system x_0, y_0, z_0:

$$P_{03} = \widehat{q}_{T_0}\widehat{q}_{R_1}\widehat{q}_{T_1}\widehat{q}_{R_2}\widehat{q}_{T_2}\widehat{q}_{R_3}P_3\widehat{q}^{\circ}_{R_3}\widehat{q}^{\circ}_{T_2}\widehat{q}^{\circ}_{R_2}\widehat{q}^{\circ}_{T_1}\widehat{q}^{\circ}_{R_1}\widehat{q}^{\circ}_{T_0}, \tag{46}$$

where point P_3 is defined as a dual quaternion

$$P_3 = [1, 0, 0, 0, 0, L_3, 0, 0]$$

and all rotation and translation operations are described as dual quaternions:

$$\begin{aligned}
\widehat{q}_{R_3} &= [\cos(\theta_3/2),\ 0,\ 0,\ \sin(\theta_3/2),\ 0,\ 0,\ 0,\ 0],\\
\widehat{q}_{T_2} &= [1,\ 0,\ 0,\ 0,\ 0,\ L_2/2,\ 0,\ 0],\\
\widehat{q}_{R_2} &= [\cos(\theta_2/2),\ 0,\ 0,\ \sin(\theta_2/2),\ 0,\ 0,\ 0,\ 0],\\
\widehat{q}_{T_1} &= [1,\ 0,\ 0,\ 0,\ 0,\ L_1/2,\ 0,\ 0],\\
\widehat{q}_{R_1} &= [\cos(\theta_1/2),\ 0,\ 0,\ \sin(\theta_1/2),\ 0,\ 0,\ 0,\ 0],\\
\widehat{q}_{T_0} &= [1,\ 0,\ 0,\ 0,\ 0,\ L_0/2,\ 0,\ 0],
\end{aligned}$$

$\widehat{q}^{\circ}_{R}$, $\widehat{q}^{\circ}_{T}$ are the conjunctions of dual quaternions (see Eqn. (44)).

It is worth mentioning that in the coordinate system x_3, y_3, z_3 the point P_3 is only rotated, and the coordinate system x_1, y_1, z_1 in reference to x_0, y_0, z_0 is only translated. The ready-to-use code is attached in Appendix D, and the result of the provided analysis is depicted in Fig. 6 for 20 angles changed by 1 degree in each step.

4. Summary

This paper provided a concise explanation of quaternions and dual quaternions, as well as an examination of the kinematic description of a fish-like propulsion system. It can be observed that the four-element rotation construct is more compact than the nine-element rotation matrix. Using efficient methods, any rotation represented by quaternions can be linearly interpolated. In that situation, quaternion algebra is more computationally efficient than Euler angles since it does not employ trigonometric functions but only basic operations on integers.

Quaternions are particularly efficient at describing rotation but not translations. Therefore, dual quaternions were used to describe the kinematics of the artificial fishtail. The developed algorithm was shown on an artificial fishtail modelled as a three-degree-of-freedom manipulator with rotational joints. All calculation functions are supplied in Appendices, allowing readers to compare the efficiency of quaternion algebra with other methods and with a more degree of freedom object of analysis. The subject of the analysis can be studied by assuming a greater number of degrees of freedom. There are no built-in functions, only multiplication and additions. Thus, the analysis can be used in environments such as Matlab, Octave, Python with NumPy/SciPy packages, SciLab, or others. All internal functions are included in Appendices.

There are also some disadvantages when working with quaternions, such as pre-processing required to

begin with this representation and certain visualisation challenges due to the four-dimension representation.

Acknowledgment

The authors of this paper would like to thank Mateusz Kowalski for a series of lectures on quaternions accessible at https://www.kowalskimateusz.pl/.

References

Chen, L., Zielinska, T., Wang, J. and Ge, W. (2020). Solution of an inverse kinematics problem using dual quaternions, *International Journal of Applied Mathematics and Computer Science* **30**(2): 351–361, DOI: 10.34768/amcs-2020-0027.

Denavit, J. and Hartenberg, R.S. (1955). A kinematic notation for lower-pair mechanisms based on matrices, *American Society of Mechanical Engineers* **22**(2): 215–221, DOI: 10.1115/1.4011045.

Felski, A., Jaskólski, K., Zwolak, K. and Piskur, P. (2020). Analysis of satellite compass error's spectrum, *Sensors* **20**(15), Paper ID: 4067, DOI: 10.3390/s20154067.

Grzadziela, A., Szymak, P. and Piskur, P. (2020). Method for assessing the dynamics and efficiency of diving fins, *Acta of Bioengineering & Biomechanics* **22**(4): 139–150, DOI: 10.37190/ABB-01589-2020-06.

Horwitz, L.P. and Biedenharn, L.C. (1984). Quaternion quantum mechanics: Second quantization and gauge fields, *Annals of Physics* **157**(2): 432–488.

Hożyń, S. and Zalewski, J. (2020). Shoreline detection and land segmentation for autonomous surface vehicle navigation with the use of an optical system, *Sensors* **20**(10), Paper ID: 2799, DOI: 10.3390/s20102799.

Jarzebowska, E. and Klak, M. (2020). Quaternion-based spacecraft dynamic modeling and reorientation control using the dynamically equivalent manipulator approach, *in* T. Sands (Ed.), *Advances in Spacecraft Attitude Control*, InfoTech Open, Rijeka, Chapter 35, DOI: 10.5772/intechopen.88080.

Jaskólski, K. (2017). Two-dimensional coordinate estimation for missing automatic identification system (AIS) signals based on the discrete Kalman filter algorithm and universal transverse mercator (UTM) projection, *Scientific Journals of the Maritime University of Szczecin* **52** (124): 82–89.

Jaskólski, K., Felski, A. and Piskur, P. (2019). The compass error comparison of an onboard standard gyrocompass, fiber-optic gyrocompass (FOG) and satellite compass, *Sensors* **19**(8), Paper ID: 1942, DOI: 10.3390/s19081942.

Jaskólski, K., Marchel, Ł., Felski, A., Jaskólski, M. and Specht, M. (2021). Automatic identification system (AIS) dynamic data integrity monitoring and trajectory tracking based on the simultaneous localization and mapping (SLAM) process model, *Sensors* **21**(24), Paper ID: 8430.

Joldeş, M. and Muller, J.-M. (2020). Algorithms for manipulating quaternions in floating-point arithmetic, *2020 IEEE 27th Symposium on Computer Arithmetic (ARITH), Portland, USA*, pp. 48–55, DOI: 10.1109/ARITH48897.2020.00016.

Jurczyk, K., Piskur, P. and Szymak, P. (2020). Parameters identification of the flexible fin kinematics model using vision and genetic algorithms, *Polish Maritime Research* **27**(2): 39–47, DOI: 10.2478/pomr-2020-0025.

Kiciński, R., Szturomski, B. and Marchel, Ł. (2021). A more reasonable model for submarines rescues seat strength analysis, *Ocean Engineering* **237**, Paper ID: 109580.

Kluczyk, M. and Grzadziela, A. (2015). Simulation model of four stroke, six cylinder marine diesel engine, *Solid State Phenomena* **236**: 113–118, DOI: 10.4028/www.scientific.net/SSP.236.113.

Leclercq, G., Lefèvre, P. and Blohm, G. (2013). 3D kinematics using dual quaternions: Theory and applications in neuroscience, *Frontiers in Behavioral Neuroscience* **7**(7): 1–25, DOI: 10.3389/fnbeh.2013.00007.

Lighthill, M.J. (1971). Large-amplitude elongated-body theory of fish locomotion, *Proceedings of the Royal Society of London B: Biological Sciences* **179**(1055): 125–138.

Ling, C., Qi, L. and Yan, H. (2022). Minimax principle for right eigenvalues of dual quaternion matrices and their generalized inverses, *arXiv* 2203.03161.

Morawski, M., Slota, A., Jerzy, Z. and Malec, M. (2020). Fish-like shaped robot for underwater surveillance and reconnaissance—Hull design and study of drag and noise, *Ocean Engineering* **217**, Paper ID: 107889.

Mouton, H.D. (2021). Comparison of body rotations using Euler angles and quaternions, http://hdl.handle.net/11427/33222.

Naus, K., Szymak, P., Piskur, P., Niedziela, M. and Nowak, A. (2021). Methodology for the correction of the spatial orientation angles of the unmanned aerial vehicle using real time GNSS, a shoreline image and an electronic navigational chart, *Energies* **14**(10), Paper ID: 2810, DOI: 10.3390/en14102810.

Pennestrì, E. and Valentini, P.P. (2010). Dual quaternions as a tool for rigid body motion analysis: A tutorial with an application to biomechanics, *Archive of Mechanical Engineering* **57**(2): 187–205, DOI: 10.2478/v10180-010-0010-2.

Piórek, M. and Jabłoński, B. (2020). A quaternion clustering framework, *International Journal of Applied Mathematics and Computer Science* **30**(1): 133–147, DOI: 10.34768/amcs-2020-0011.

Piskur, P. (2022). Strouhal number measurement for novel biomimetic folding fins using an image processing method, *Journal of Marine Science and Engineering* **10**(4), Paper ID: 484, DOI: 10.3390/jmse10040484.

Piskur, P., Szymak, P., Flis, L. and Sznajder, J. (2020a). Analysis of a fin drag force in a biomimetic underwater vehicle, *Naše more* **67**(3): 192–198.

Piskur, P., Szymak, P., Kitowski, Z. and Flis, L. (2020b). Influence of fin's material capabilities on the propulsion system of biomimetic underwater vehicle, *Polish Maritime Research* **27**(4): 179–185, DOI: 10.2478/pomr-2020-0078.

Piskur, P., Szymak, P., Przybylski, M., Naus, K., Jaskólski, K. and Żokowski, M. (2021). Innovative energy-saving propulsion system for low-speed biomimetic underwater vehicles, *Energies* **14**(24), Paper ID: 8418, DOI: 10.3390/en14248418.

Radavelli, L., Simoni, R., De Pieri, E. and Martins, D. (2012). A comparative study of the kinematics of robots manipulators by Denavit–Hartenberg and dual quaternion, *Mecánica Computacional* **31**(15): 2833–2848.

Sarabandi, S. and Thomas, F. (2019). A survey on the computation of quaternions from rotation matrices, *Journal of Mechanisms and Robotics* **11**(2), Paper ID: 021006, DOI: 10.1115/1.4041889.

Sola, J. (2017). Quaternion kinematics for the error-state Kalman filter, *arXiv* 1711.02508, DOI: 10.48550/arXiv.1711.02508.

Wawrzyński, W., Zieja, M., Żokowski, M. and Sigiel, N. (2022). Optimization of autonomous underwater vehicle mission planning process, *Bulletin of the Polish Academy of Sciences: Technical Sciences* **70**(2), Paper ID: e140371.

Zygmunt Kitowski is a graduate of the Polish Naval Academy (PNA) in Gdynia in the field of electrical engineering (1969). Then, after a short service on a submarine as an electromechanic officer, he undertook scientific and teaching work at the PNA. In 1973 he obtained his MS degree in electrical machinery from the Gdańsk University of Technology. In 1978 he obtained his PhD at the Naval Academy in Leningrad in the field of automation and robotics. In 1989, the Council of the Faculty of Power Engineering and Aviation of the Warsaw University of Technology awarded him the DSc degree in the field of automation and robotics. In 1997, he received the professorial title from the President of the Republic of Poland. Since 1970 he has been working in various positions at the PNA. His scientific and teaching activity is related to computer control systems of manned and unmanned surface ships and underwater vehicles.

Paweł Piskur, Lt Cdr, PhD, Eng, graduated from the Military University of Technology in Warsaw with a degree in airplane studies in 2004. He then worked for 13 years in the Marine Aviation Base in the Polish Army. In 2010 he received a PhD degree at the Koszalin University of Technology in mechanical engineering. Since 2017 he has been working at the Polish Naval Academy. His research area is strictly connected with unmanned underwater vehicles, especially biomimetic propulsion systems.

Mateusz Orłowski, Lt, MSc, Eng, graduated from the Military University of Technology in the Faculty of Mechatronics and Aerospace in Warsaw with a specialization in avionics in 2019. He currently serves at the Naval Air Base in Gdynia. He is also an employee of the Faculty of Mechanical and Electrical Engineering of the Polish Naval Academy.

Appendices

All the scripts included in Appendices can be obtained upon sending a request to `p.piskur@amw.gdynia.pl`.

Appendix A

Rotation of a point

```
clc; clear; close all;
% Point to rotate P = [x, y, z]
P1 = [1,1,1];
% Point P encapsulated into quaternion :
pq = [0 P1(1) P1(2) P1(3)];
% axis of rotation :
v = [0, 0, 1];
% rotation angle :
θ = 90 * (pi/180);
% half the angle of rotation :
θ2 = θ/2;
% the scalar part of the quaternion
% describing the axis of rotation :
so = cos(θ2);
% module of the vector part of the quaternion
% describing the axis of rotation :
vm = sqrt(v(1)^2 + v(2)^2 + v(3)^2);
xo = (v(1)/vm) * sin(θ2);
yo = (v(2)/vm) * sin(θ2);
zo = (v(3)/vm) * sin(θ2);
% quaternion :
q = [so xo yo zo];
% quaternion conjugated :
cq = [so -xo -yo -zo];
qt = Qmultiplication(q, pq);
p2 = Qmultiplication(qt, cq);
PR = p2(2:4);
% visualisation :
figure(1)
P0 = [0,0,0];
plot3([0P1(1)], [0P1(2)], [0P1(3)],'g-o')
text(P1(1), P1(2), P1(3),'P')
hold on
plot3([0PR(1)], [0PR(2)], [0PR(3)],'b-o')
text(PR(1), PR(2), PR(3),'PR')
axis equal
grid on
hold off
```

Appendix B

Interpolation of a point rotation

```
clc; clear; close all;
% point to be rotated P = [x, y, z]
P1 = [1, 0, 0];
pq = [0, P1(1), P1(2), P1(3)]; v = [1, 1, 1];
% angle of rotation
θ = 90 * (pi/180);
N = 100; % angle division
θN = (1/N) * θ;
```

```
so = cos(θ_N);
v_m = sqrt(v(1)^2 + v(2)^2 + v(3)^2);
xo = (v(1)/v_m) * sin(θ_N);
yo = (v(2)/v_m) * sin(θ_N);
zo = (v(3)/v_m) * sin(θ_N);
% quaternion
q = [so xo yo zo];
% quaternion conjugated
q_sp = [so, -xo, -yo, -zo];
% loop
for i = 1 : N
  pq = Qmultiplication(q, pq);
  pq = Qmultiplication(pq, q_sp);
%point coordinates after rotation
  P2(i,:) = pq(2 : 4);
end
```

Appendix C

Translation and rotation

```
clc; clear; close all;
% point
A = [0, 0, 2];
% point A as a dual quaternion
dqA = [1, 0, 0, 0, 0, A];
% axis of rotation
v1 = [0, 1, 0];
% angle of rotation
theta_1 = 90 * (pi/180);
% translation
d1 = 2;
P01 = [0, 0, 0];
dq1 = pdq(v1, theta_1, d1, P01);
dqB = conversion(dq1, dqA);
P_RT = dqB(6 : 8)
```

Appendix D

Fish tail kinematics

```
clc; clear; close all;
% link length :
l_1 = 0.1;
l_2 = 0.1;
l_3 = 0.12;
% rotation angles :
θ_1 = 1 * (pi/180);
θ_2 = 1 * (pi/180);
θ_3 = 1 * (pi/180);
data as a dual quaternions :
dq3r = [cos(θ_3/2), 0, 0, sin(θ_3/2), 0, 0, 0, 0];
dq3t = [1, 0, 0, 0, 0, l_3/2, 0, 0];
dq2r = [cos(θ_2/2), 0, 0, sin(θ_2/2), 0, 0, 0, 0];
dq2t = [1, 0, 0, 0, 0, l_2/2, 0, 0];
dq1r = [cos(θ_1/2), 0, 0, sin(θ_1/2), 0, 0, 0, 0];
dq1t = [1, 0, 0, 0, 0, l_1/2, 0, 0];
N = 10;
for i = 1 : N
  dq3r = [cos(i*θ_3/2), 0, 0, sin(i*θ_3/2), 0, 0, 0, 0];
  dq3t = [1, 0, 0, 0, 0, L2/2, 0, 0];
  dq2r = [cos(i*θ_2/2), 0, 0, sin(i*θ_2/2), 0, 0, 0, 0];
  dq2t = [1, 0, 0, 0, 0, L1/2, 0, 0];
  dq1r = [cos(i*θ_1/2), 0, 0, sin(i*θ_1/2), 0, 0, 0, 0];
  dq1t = [1, 0, 0, 0, 0, L0/2, 0, 0];
  dq1 = DQmultiplication(dq1t, dq1r);
  dq1temp = DQmultiplication(dq1, dq2t);
  dq63 = DQmultiplication(dq1temp, dq2r);
  dq62 = DQmultiplication(dq63, dq3t);
  dq3 = DQmultiplication(dq62, dq3r);
  resultDQ61 = conversion(dq3, dqP3);
  result61(i,:) = resultDQ61(:, 6 : 8);
  resultDQ63 = conversion(dq63, dqP2);
  result63(i,:) = resultDQ63(:, 6 : 8);
  resulsDQ65 = conversion(dq1, dqP1);
  result65(i,:) = resulsDQ65(:, 6 : 8);
end
%plotting the results
plot3(result63(:, 1), result63(:, 2), result63(:, 3), 'bo')
hold on
plot3(result61(:, 1), result61(:, 2), result61(:, 3), 'rd')
plot3(result65(:, 1), result65(:, 2), result65(:, 3), 'g*')
grid on
axis equal
```

Appendix E

Internal functions used

```
function q3 = Qmultiplication(q1, q2)
   % Description :
   % Two quaternions (q1 and q2) multiplication.
   % The function gives the third quaternion q3.
   % The both quaternions :
   % q1 = [q1(1) q1(2) q1(3) q1(4)];
    % and
    % q2 = [q2(1) q2(2) q2(3) q2(4)];
   % have a scalar part :
   % s1 = q1(1); s2 = q2(1);
   % and vector parts :
   % v1 = q1(2 : end); v2 = q2(2 : end);
   % As a result q3 is calculated :
   % q3 = [s v];
   % where :
   % s = s1 * s2 - dot(v1, v2);
   % v = s1 * v2 + s2 * v1 + cross(v1, v2);
       s1 = q1(1);
       v1 = q1(2 : end);
```

```
    s2 = q2(1);
    v2 = q2(2 : end);
    s = s1 * s2 - dot(v1, v2);
    v = s1 * v2 + s2 * v1 + cross(v1, v2);
    q3 = [s v];
end
function [dq] = pdq(v, theta, d, P0)
 % parameters for dual quaternions
  qr0 = cos(0.5 * theta);
  norm_v = norm(v);
  L = (v/norm_v);
  qr13 = L * sin(0.5 * theta);
  qr = [qr0 qr13]; %rotation
  qP0 = [0 P0];
  qL = [0 L];
 %dq = qr + 0.5 * (d * qL * qr + P0 * qr - qr * P0)
  dq(1, 1 : 4) = qr;
   dq(1, 5 : 8) = 0.5 * (Qmultiplication(qL * d +
qP0, qr) - Qmultiplication(qr, qP0));
end

function cdq = cDQ(dq)
 %conjuction of qual quaternions
  cdq(1, 1) = dq(1, 1);
  cdq(1, 2 : 4) = - dq(1, 2 : 4);
  cdq(1, 5) = - dq(1, 5);
  cdq(1, 6 : 8) = dq(1, 6 : 8);
end

    function dq3 = DQmultiplication(dq1, dq2)
 % dual quaternion multiplication;
  q1 = dq1(1 : 4);
  w1 = dq1(5 : 8);
  q2 = dq2(1 : 4);
  w2 = dq2(5 : 8);
  dq3(1, 1 : 4) = Qmultiplication(q1, q2);
  dq3(1, 5 : 8) = Qmultiplication(q1, w2) +
                + Qmultiplication(w1, q2);
end

function dqB = conversion(dq, dqA)
  DQromb = cDQ(dq);
  temp = DQmultiplication(dq, dqA);
  dqB = DQmultiplication(temp, DQromb);
end
```

Received: 1 August 2022
Revised: 5 November 2022
Accepted: 1 February 2023