International Journal of Robotic Computing
Vol. 1, No. 1 (2019)

DOI: 10.35708/RC1868-126255

A Novel Dual Quaternion Based Cost Efficient Recursive Newton-Euler Inverse Dynamics Algorithm

Cristiana Miranda de Farias[1], Luis Felipe da Cruz Figueredo[2], and João Yoshiyuki Ishihara[1]

[1] Depart. Elect. Eng., University of Brasilia (UnB) – 70910-900, Brasilia,DF, Brazil
[2] School of Computing, University of Leeds – LS2 9JT, Leeds, UK
cristiana@lara.unb.br, l.figueredo@leeds.ac.uk, ishihara@ene.unb.br

Received (11/02/2018)
Revised (06/19/2019)
Accepted (07/20/2019)

Abstract. In this paper, the well known recursive Newton-Euler inverse dynamics algorithm for serial manipulators is reformulated into the context of the algebra of Dual Quaternions. Here we structure the forward kinematic description with screws and line displacements rather than the well established Denavit-Hartemberg parameters, thus accounting better efficiency, compactness and simpler dynamical models. We also present here the closed solution for the dqRNEA, and to do so we formalize some of the algebra for dual quaternion-vectors and dual quaternion-matrices. With a closed formulation of the dqRNEA we also create a dual quaternion based formulation for the computed torque control, a feedback linearization method for controlling a serial manipulator's torques in the joint space. Finally, a cost analysis of the main Dual Quaternions operations and of the Newton-Euler inverse dynamics algorithm as a whole is made and compared with other results in the literature.

Keywords: Dual Quaternion; Newton Euler Inverse Dynamic Algorithm; Robotic Manipulation

1 Introduction

Rigid body motion and control have been extensively studied in the past decades in a number of disciplines, including, mechanical systems, robotic manipulation, satellites, etc. These were usually investigated and designed by exploiting the

Homogeneous Transformation Matrices (HTM), however, recently, many new works have surfaced on the uses of dual quaternions (DQ) to describe such systems [7,11,12,14,27].

Particularly, regarding robotic manipulation, dual quaternion algebra proves itself to be advantageous compared to other representations as it allows us to represent the complete robot position and attitude—the pose—with a set of eight parameters. Moreover, it was argued that the unit dual quaternion (UDQ) in addition to being non-minimal and free of singularities, are also a more compact, efficient and less computationally demanding representation for rigid-body displacements compared to HTM [13,19]. Indeed, we can also list as further advantages of the dual quaternions the fact that they are very well suited to represent many geometric primitives, such as lines and planes, in a simple and intuitive way [26] and the fact that dual quaternions have been argued to be most efficient way to represent a screw motion [13]. The benefits of using dual quaternion algebra have been discussed also in many works with different applications comprising rigid body motion stabilization, tracking, multiple body coordination [14], kinematic control of manipulators with single and multiple arms and human-robot interaction [8,9,26], etc.

Although a lot has been published regarding the kinematic representation and control of robotic systems using dual quaternion algebra, there is still a lot to be done with respect to representing and controlling the dynamics of a robotic arm with DQ. Indeed, algorithms for rigid-body dynamics computation play a crucial role for simulation of motion, analysis of forces, torques and for the design of control techniques in robotics [17]. Rigid-body dynamics are at the core of many recent robotic applications in different fields, such as legged robot stabilization, forceful Human-Robot Interaction (HRI), computer animation and even biomechanics.

In Dooley and McCarthy's work [7], the authors propose one of the earliest formulations for the dynamic modeling for an unconstrained rigid body and for a serial manipulator using dual quaternion algebra in Kane and Levinson's formulation [16]. Although their work is pioneer, their equations are overly complicate and lack a clear and intuitive physical meaning. Other attempts to improve on [7] are presented in [14] in which it is proposed a decoupled formulation for both the rotational and translational dynamics using dual quaternions. Furthermore, they use this model in order to design a regulator for both the attitude and position of a body. Neither of these works exploit formulations for the dynamical problem that would couple together the translational and rotational forces acting on the body. Furthermore, a coupled formulation for the dynamic of an unconstrained rigid body can be found in [11, 12] in which the authors exploit the manner in which dual quaternion twists and wrenches can couple together the linear and angular variables.

We may further extend the dynamics of a rigid body to the dynamics of a whole articulated robotic manipulator. In the literature, there are a few algorithms that are usually used to describe such systems [15, 16, 18], with the most famous ones being the recursive Newton-Euler algorithm (RNEA) and the

Lagrange-Euler (LE) formulation. The LE equations and the RNEA are both used to describe the relations between the joint torques, forces at the end effector and kinematic variables, however, they differ in many other aspects. Concerning only computational complexity, the Newton–Euler formulation is considerably more efficient due to its inherently recursive formulation.

Regarding the dynamical representation of robotic manipulator in dual quaternion algebra, the literature is more lacking—especially when considering Newton-Euler's algorithm. Recent works on the subject include [28], in which the authors exploit the principle of virtual work and UDQ to describe an efficient solution for the dynamics of a parallel manipulator. In [3,7] both authors show an illustrative examples to how we can build the inverse dynamics of a manipulator, with [3] using the principle of virtual work and dual numbers and [7] the dual quaternion formulation and Kane's equations.

The only other works to focus on the dynamical description of a serial manipulator through a dual quaternion based on the recursive Newton-Euler algorithm have been published recently by [24,25] and [6]. In [24,25] the authors propose a formulation for describing the dynamics of a spacecraft-mounted robotic manipulator configured with different joint types. Although the equations in [24,25] have some similarities to the ones proposed in this manuscript and in our work in [6] there are also many differences between the algorithms. In [24,25] the authors propose framework, in which they take into account many different types of joints. However one big drawback in their formulation is that many equations presented in [24,25] are in the Euclidean vector-matrix formulation, and therefore have to be mapped and back to dual quaternions, creating the need for transformations in and out of the algebra, which leads to more costly cumbersome and, although correct, unintuitive equations.

Opposed to [24,25], the work in [6] proposed a more intuitive solution for both the recursive and closed formulation of the Newton-Euler algorithm following the dual quaternion notation in the [26]. In its recursive formulation our dqRNEA clearly presents a forward and a backward iteration, and the in the closed formulation of the algorithm we identify the mass, Coriolis and gravity terms. More so, in addition, we also contribute with an optimization of the adjoint transformation allowing our algorithm to be more cost efficient.

2 Mathematical Background

This section provides the reader background on many of the aspects pertaining the algebra of dual quaternions as well as some of the algebraic proprieties and operators required in the development of the algorithms presented in this work. Finally, we will also introduce in this section the formalism for describing dual quaternion based vectors and matrices.

2.1 Quaternion Algebra

Let $\hat{\imath}, \hat{\jmath}, \hat{k}$ be the three quaternionic units such that $\hat{\imath}^2 = \hat{\jmath}^2 = \hat{k}^2 = \hat{\imath}\hat{\jmath}\hat{k} = -1$. The algebra of quaternions is generated by the basis elements $1, \hat{\imath}, \hat{\jmath}$, and $\hat{k}$, which yields the set

$$\mathbb{H} \triangleq \left\{ \eta + \boldsymbol{\mu} \ : \ \boldsymbol{\mu}{=}\mu_1\hat{\imath} + \mu_2\hat{\jmath} + \mu_3\hat{k},\ \eta, \mu_1, \mu_2, \mu_3 {\in} \mathbb{R} \right\}. \tag{1}$$

A quaternion element $\boldsymbol{q} \in \mathbb{H}$ may be decomposed into a real component $\mathrm{Re}\,(\boldsymbol{q}) \triangleq \eta$ and an imaginary component $\mathrm{Im}\,(\boldsymbol{q}) \triangleq \boldsymbol{\mu}$, such that $\boldsymbol{q} = \mathrm{Re}\,(\boldsymbol{q}) + \mathrm{Im}\,(\boldsymbol{q})$. For each quaternion element $\boldsymbol{q}$, there is one correspondent quaternion conjugate given by $\boldsymbol{q}^* \triangleq \mathrm{Re}\,(\boldsymbol{q}) - \mathrm{Im}\,(\boldsymbol{q})$, which in turn defines the quaternion norm $\|\boldsymbol{q}\| \triangleq \sqrt{\boldsymbol{q}\boldsymbol{q}^*}$. The norm of quaternion elements coincides with the Euclidean norm of a vector $\mathrm{vec}\,\boldsymbol{q} \triangleq [\eta \ \ \mu_1 \ \ \mu_2 \ \ \mu_3]^T \in \mathbb{R}^4$, that is, $\|\boldsymbol{q}\| = \|\mathrm{vec}\,\boldsymbol{q}\|$. Quaternion elements with real component equal to zero belong to the set of *pure quaternions*

$$\mathbb{H}_0 \triangleq \{\boldsymbol{q}_0 : \boldsymbol{q}_0 \in \mathbb{H}\,,\, \mathrm{Re}\,(\boldsymbol{q}_0) = 0\} \tag{2}$$

whereby $\boldsymbol{q}_0^* = -\boldsymbol{q}_0$. Such elements are isomorphic to three-dimensional vectors and, similarly, they can be used to represent translation, angular and linear velocities, accelerations, momentum and wrenches. Hence, kinematics and dynamics can be compactly represented in a unified framework. Within the linear space of pure quaternions, we can define the inner and cross product operations similarly to their vectorial counterpart [26]. Given two pure quaternions $\boldsymbol{a} = a_1\hat{\imath} + a_2\hat{\jmath} + a_3\hat{k}$ and $\boldsymbol{b} = b_1\hat{\imath} + b_2\hat{\jmath} + b_3\hat{k}$ we have the inner and cross products[3]

$$\boldsymbol{a} \cdot \boldsymbol{b} \triangleq -\frac{\boldsymbol{a}\boldsymbol{b} + \boldsymbol{b}\boldsymbol{a}}{2} = a_1b_1 + a_2b_2 + a_3b_3, \tag{3}$$

$$\boldsymbol{a} \times \boldsymbol{b} \triangleq \frac{\boldsymbol{a}\boldsymbol{b} - \boldsymbol{b}\boldsymbol{a}}{2}. \tag{4}$$

In addition, a quaternion element can also represent arbitrary rotations when constrained to the set of unit quaternions $\mathcal{S}^3 \triangleq \{\boldsymbol{q} \in \mathbb{H} : \|\boldsymbol{q}\| = 1\}$. The set $\mathcal{S}^3$ together with the multiplication operation forms the Lie group of unit quaternions, Spin(3) [22]. An arbitrary rotation angle $\phi \in \mathbb{R}$ around the rotation axis $\boldsymbol{n} \in \mathbb{H}_0 \cap \mathcal{S}^3$, with $\boldsymbol{n} = n_x\hat{\imath} + n_y\hat{\jmath} + n_z\hat{k}$, is represented by the unit quaternion $\boldsymbol{r} = \cos(\phi/2) + \sin(\phi/2)\boldsymbol{n}$ [22, 26].

2.2 Dual Quaternions

Dual Quaternions are an extension of quaternions first introduced by Clifford to describe the complete and coupled rigid body motion [22]. The dual quaternion algebra is constituted by the set

$$\underline{\mathbb{H}} \triangleq \left\{ \underline{\boldsymbol{q}} = \boldsymbol{q_P} + \varepsilon \boldsymbol{q_D} \mid \boldsymbol{q_P}, \boldsymbol{q_D} {\in} \mathbb{H} \right\}, \tag{5}$$

where ε is called dual unit with $\varepsilon^2 = 0$, $\varepsilon \neq 0$.

A dual quaternion element can also be decomposed in primary and dual parts $\mathcal{P}(\underline{\boldsymbol{q}}) = \boldsymbol{q_P}$ and $\mathcal{D}(\underline{\boldsymbol{q}}) = \boldsymbol{q_D}$, respectively. And, for each dual quaternion element $\underline{\boldsymbol{q}}$, there exist one correspondent conjugate $\underline{\boldsymbol{q}}^* \triangleq \boldsymbol{q}_P^* + \varepsilon \boldsymbol{q}_D^*$ composed with the quaternion conjugate of the primary and dual parts. Under multiplication, the

[3] The operators $(\cdot\ ,\ \times)$ can similarly be defined for $\mathbb{H}$, see, e.g., [10, 26].

subset of *unit* dual quaternions $\underline{\boldsymbol{S}} \triangleq \left\{\underline{\boldsymbol{q}} \in \underline{\mathbb{H}} : \left\|\underline{\boldsymbol{q}}\right\| = 1\right\}$, forms the Lie group $\mathrm{Spin}(3) \ltimes \mathbb{R}^3$, whose identity element is 1 and the group inverse of $\underline{\boldsymbol{x}} \in \underline{\boldsymbol{S}}$ is $\underline{\boldsymbol{x}}^*$ [22]. The subset of dual quaternions constituted by pure quaternions belong to the set of *pure dual quaternions* $\underline{\mathbb{H}}_0 \triangleq \left\{\underline{\boldsymbol{q}}_0 = \boldsymbol{q_P} + \varepsilon \boldsymbol{q_D} \mid \boldsymbol{q_P}, \boldsymbol{q_D} \in \mathbb{H}_0\right\}$ where $\underline{\boldsymbol{q}}_0^* = -\underline{\boldsymbol{q}}_0$. Similar to pure quaternions, elements in $\underline{\mathbb{H}}_0$ can be deployed into the kinematics and dynamics analysis to compactly express the coupled angular and linear generalized rigid body twist, accelerations, momentum and wrenches. Moreover, in a similar fashion, the cross product may also be defined for pure dual quaternions

$$\underline{\boldsymbol{a}} \times \underline{\boldsymbol{b}} \triangleq \frac{\boldsymbol{ab} - \boldsymbol{ba}}{2}. \tag{6}$$

Also, we can also define the inner product for $\underline{\mathbb{H}}_0$ by taking the combined real value of the inner product from the primary part and dual parts, that is,

$$\underline{\boldsymbol{a}} \odot \underline{\boldsymbol{b}} \triangleq \boldsymbol{a}_P \cdot \boldsymbol{b}_P + \boldsymbol{a}_D \cdot \boldsymbol{b}_D, \tag{7}$$

which is similar to the Euclidean norm of corresponding vectors and is similar to the double-geodesic metric provided in Bullo and Murray [4] for $SE(3)$.[4]

2.3 Dual Quaternions Matrices and Vectors

Although not very common in the literature, it is also interesting to define how the structure of dual quaternion vectors and dual quaternion matrices work, as well as their main proprieties. Thus following the traditional vector formalism, we may define a dual quaternion vector as

$$\underline{\mathbb{H}}^n \triangleq \left\{\left[\,\underline{\boldsymbol{q}}_1 \;\; \underline{\boldsymbol{q}}_2 \;\; \underline{\boldsymbol{q}}_3 \;\cdots\; \underline{\boldsymbol{q}}_n\,\right]^T : \underline{\boldsymbol{q}}_i \in \underline{\mathbb{H}},\; i \in \{1, 2, \ldots, n\}\right\}, \tag{8}$$

which is endowed with the transpose and conjugate operations, that is, given the dual quaternion vector $\underline{\boldsymbol{Q}} = \in \underline{\mathbb{H}}^n$ we may also define its transpose and conjugate as

$$\underline{\boldsymbol{Q}}^T \triangleq \left[\,\underline{\boldsymbol{q}}_1 \;\; \underline{\boldsymbol{q}}_2 \;\; \underline{\boldsymbol{q}}_3 \cdots \underline{\boldsymbol{q}}_n\,\right], \quad \text{and} \quad \underline{\boldsymbol{Q}}^* = \left[\,\underline{\boldsymbol{q}}_1^* \;\; \underline{\boldsymbol{q}}_2^* \;\; \underline{\boldsymbol{q}}_3^* \;\; \cdots \underline{\boldsymbol{q}}_n^*\,\right]^T. \tag{9}$$

Given two dual quaternions vectors such as $\underline{\boldsymbol{Q}} = \left[\,\underline{\boldsymbol{q}}_1 \;\; \underline{\boldsymbol{q}}_2 \;\; \underline{\boldsymbol{q}}_3 \;\cdots\; \underline{\boldsymbol{q}}_n\,\right] \in \underline{\mathbb{H}}^n$ and $\underline{\boldsymbol{Q}}' = \left[\,\underline{\boldsymbol{q}}'_1 \;\; \underline{\boldsymbol{q}}'_2 \;\; \underline{\boldsymbol{q}}'_3 \;\cdots\; \underline{\boldsymbol{q}}'_n\,\right] \in \underline{\mathbb{H}}^n$ the addition operation ($\mathbb{H}^n \times \mathbb{H}^n \longrightarrow \mathbb{H}^n$) can be defined as

$$\underline{\boldsymbol{Q}} + \underline{\boldsymbol{Q}}' \triangleq \left[\,\underline{\boldsymbol{q}}_1 + \underline{\boldsymbol{q}}'_1 \;\; \underline{\boldsymbol{q}}_2 + \underline{\boldsymbol{q}}'_2 \;\; \underline{\boldsymbol{q}}_3 + \underline{\boldsymbol{q}}'_3 \;\cdots\; \underline{\boldsymbol{q}}_n + \underline{\boldsymbol{q}}'_n\,\right], \tag{10}$$

in which the individual elements of the vector are computed by the addition. Similarly the multiplication, ($\mathbb{H}^n \times (\mathbb{H}^n)^T \longrightarrow \mathbb{H}$), can be defined as

$$\underline{\boldsymbol{Q}}\left(\underline{\boldsymbol{Q}}'\right)^T = \underline{\boldsymbol{q}}_1\underline{\boldsymbol{q}}'_1 + \underline{\boldsymbol{q}}_2\underline{\boldsymbol{q}}'_2 + \underline{\boldsymbol{q}}_3\underline{\boldsymbol{q}}'_3 + \ldots + \underline{\boldsymbol{q}}_n\underline{\boldsymbol{q}}'_n, \tag{11}$$

[4] It is important to highlight that there is no well-defined Riemannian metric for the group Euclidean transformations nor for the (pure) dual quaternions, but the double-geodesic approach used in [4] and herein ensures positiveness and equal actions in the attitude and translation geodesics. The study on metrics for Euclidean displacements and the correspondent topological obstruction lies out of the scope of the current manuscript, but readers are refereed to excel works of [4, 20].

In a similar fashion, we may define the dual quaternion matrix as

$$\underline{\mathbb{H}}^{n\times m} \triangleq \left\{ \begin{bmatrix} \underline{\boldsymbol{q}}_{1,1} & \underline{\boldsymbol{q}}_{1,2} & \cdots & \underline{\boldsymbol{q}}_{1,m} \\ \underline{\boldsymbol{q}}_{2,1} & \underline{\boldsymbol{q}}_{2,2} & \cdots & \underline{\boldsymbol{q}}_{2,m} \\ \vdots & \vdots & \ddots & \vdots \\ \underline{\boldsymbol{q}}_{n,1} & \underline{\boldsymbol{q}}_{n,2} & \cdots & \underline{\boldsymbol{q}}_{n,m} \end{bmatrix} : \underline{\boldsymbol{q}}_{i,j} \in \underline{\mathbb{H}},\ i \in \{1,2,\ldots,n\} \text{ and } j \in \{1,2,\ldots,m\} \right\}. \tag{12}$$

Note that addition and multiplication operate similarly to how they do in the vector notation. Given the dual quaternion matrix $\underline{\boldsymbol{Q}} \in \underline{\mathbb{H}}^{n\times m}$, we similarly have the transpose and conjugate operations defined as

$$\underline{\boldsymbol{Q}}^T = \begin{bmatrix} \underline{\boldsymbol{q}}_{1,1} & \underline{\boldsymbol{q}}_{2,1} & \cdots & \underline{\boldsymbol{q}}_{n,1} \\ \underline{\boldsymbol{q}}_{1,2} & \underline{\boldsymbol{q}}_{2,2} & \cdots & \underline{\boldsymbol{q}}_{n,2} \\ \vdots & \vdots & \ddots & \vdots \\ \underline{\boldsymbol{q}}_{1,m} & \underline{\boldsymbol{q}}_{2,m} & \cdots & \underline{\boldsymbol{q}}_{n,m} \end{bmatrix} \quad \text{and} \quad \underline{\boldsymbol{Q}}^* = \begin{bmatrix} \underline{\boldsymbol{q}}^*_{1,1} & \underline{\boldsymbol{q}}^*_{1,2} & \cdots & \underline{\boldsymbol{q}}^*_{1,m} \\ \underline{\boldsymbol{q}}^*_{2,1} & \underline{\boldsymbol{q}}^*_{2,2} & \cdots & \underline{\boldsymbol{q}}^*_{2,m} \\ \vdots & \vdots & \ddots & \vdots \\ \underline{\boldsymbol{q}}^*_{n,1} & \underline{\boldsymbol{q}}^*_{n,2} & \cdots & \underline{\boldsymbol{q}}^*_{n,m} \end{bmatrix}. \tag{13}$$

Similar construction can be extended to pure quaternion and pure dual quaternions. Taking the latter, for instance, we have

$$\underline{\mathbb{H}}_{\mathbf{0}}{}^n \triangleq \left\{ \begin{bmatrix} \underline{\boldsymbol{q}}_1 & \underline{\boldsymbol{q}}_2 & \underline{\boldsymbol{q}}_3 & \cdots & \underline{\boldsymbol{q}}_n \end{bmatrix}^T : \underline{\boldsymbol{q}}_i \in \underline{\mathbb{H}}_0,\ i \in \{1,2,\ldots,n\} \right\}, \tag{14}$$

and the pure dual quaternion matrix $\underline{\mathbb{H}}_{\mathbf{0}}{}^{n\times m}$ defined in a similar fashion to (12)

Regarding the pure dual quaternion vectors we can also define three very important operations: the element-wise cross product, inner product and double geodesic products. All this operations take advantage of their equivalents in $\underline{\mathbb{H}}_0$ and perform these operations between every two pairs of pure dual quaternions in the vector.

Let $\underline{\boldsymbol{P}}_1 = \begin{bmatrix} \underline{\boldsymbol{p}}_{1,1} & \underline{\boldsymbol{p}}_{1,2} & \cdots & \underline{\boldsymbol{p}}_{1,n} \end{bmatrix}^T$ and $\underline{\boldsymbol{P}}_2 \begin{bmatrix} \underline{\boldsymbol{p}}_{2,1} & \underline{\boldsymbol{p}}_{2,2} & \cdots & \underline{\boldsymbol{p}}_{2,n} \end{bmatrix}^T \in \underline{\mathbb{H}}_{\mathbf{0}}{}^n$, then such operations are defined as follows

$$\underline{\boldsymbol{P}}_1 \otimes \underline{\boldsymbol{P}}_2 = \begin{bmatrix} \underline{\boldsymbol{p}}_{1,1} \times \underline{\boldsymbol{p}}_{2,1} & \underline{\boldsymbol{p}}_{1,2} \times \underline{\boldsymbol{p}}_{2,2} & \cdots & \underline{\boldsymbol{p}}_{1,n} \times \underline{\boldsymbol{p}}_{2,n} \end{bmatrix}^T. \tag{15}$$

$$\underline{\boldsymbol{P}}_1 \odot \underline{\boldsymbol{P}}_2 = \begin{bmatrix} \underline{\boldsymbol{p}}_{1,1} \cdot \underline{\boldsymbol{p}}_{2,1} & \underline{\boldsymbol{p}}_{1,2} \cdot \underline{\boldsymbol{p}}_{2,2} & \cdots & \underline{\boldsymbol{p}}_{1,n} \cdot \underline{\boldsymbol{p}}_{2,n} \end{bmatrix}^T. \tag{16}$$

$$\underline{\boldsymbol{P}}_1 \circledast \underline{\boldsymbol{P}}_2 = \begin{bmatrix} \underline{\boldsymbol{p}}_{1,1} \odot \underline{\boldsymbol{p}}_{2,1} & \underline{\boldsymbol{p}}_{1,2} \odot \underline{\boldsymbol{p}}_{2,2} & \cdots & \underline{\boldsymbol{p}}_{1,n} \odot \underline{\boldsymbol{p}}_{2,n} \end{bmatrix}^T. \tag{17}$$

3 Novel Transformations: Addressing Commutativity

As an associative division algebra over $\mathbb{R}$, the quaternion algebra is endowed with classic operations such as addition, scalar multiplication, quaternion multiplication—as it is dual quaternions algebra, although not being a division ring. Nonetheless, it is also well-known that $\mathbb{H}$ is a non-commutative group, and the same for the dual quaternion group—that is, if $\underline{\boldsymbol{a}}$ and $\underline{\boldsymbol{b}}$ are dual quaternions, then $\underline{\boldsymbol{ab}} \neq \underline{\boldsymbol{ba}}$.

To address possible constraints that may arise with its non-commutativity, e.g., additional computational load in frame transformations—as shown in the following section—a common strategy is to take the matrix algebra representation, e.g., $SU(2)$ which is isomorphic to Spin(3) can be commuted with proper sign modifications. This approach, however, has a further drawback which is the needlessly map from quaternion or dual quaternion to other Lie-matrix manifolds.

For instance, in existing literature, in order to perform multiplicative commutativity, the common strategy to take the matrix algebra representation of quaternions or dual quaternions is done by using the orthogonal matrix which is known as the Hamilton operator $\overset{\pm}{\boldsymbol{H}}(\boldsymbol{x}) \in \mathbb{R}^{4\times 4}$ of the quaternion element $\boldsymbol{x}$ [1]. In this manner, two quaternions $\boldsymbol{a}$ and $\boldsymbol{b}$ can commute as follows

$$\boldsymbol{y} = \boldsymbol{ab} = \underline{\text{vec}}\left(\overset{+}{\boldsymbol{H}}(\boldsymbol{a}) \operatorname{vec} \boldsymbol{b}\right) = \underline{\text{vec}}\left(\overset{-}{\boldsymbol{H}}(\boldsymbol{b}) \operatorname{vec} \boldsymbol{a}\right), \tag{18}$$

where $\operatorname{vec} : \mathbb{H} \to \mathbb{R}^4$ is the mapping from quaternion to $\mathbb{R}^4$ and $\underline{\text{vec}}$ is the inverse mapping. The Hamilton operators for the right and left multiplications can be computed from the matrix representation of orthogonal vectors from $\boldsymbol{x} = \eta + \nu_1 i + \nu_2 j + \nu_3 k$, that is,

$$\overset{+}{\boldsymbol{H}}(\boldsymbol{x}) \triangleq \begin{bmatrix} \eta & -\mu_1 & -\mu_2 & -\mu_3 \\ \mu_1 & \eta & -\mu_3 & \mu_2 \\ \mu_2 & \mu_3 & \eta & -\mu_1 \\ \mu_3 & -\mu_2 & \mu_1 & \eta \end{bmatrix} \quad \text{and} \quad \overset{-}{\boldsymbol{H}}(\boldsymbol{x}) \triangleq \begin{bmatrix} \eta & -\mu_1 & -\mu_2 & -\mu_3 \\ \mu_1 & \eta & \mu_3 & -\mu_2 \\ \mu_2 & -\mu_3 & \eta & \mu_1 \\ \mu_3 & \mu_2 & -\mu_1 & \eta \end{bmatrix}. \tag{19}$$

In order to address commutativity in a more intuitive and computationally efficient manner, it is important to devise new techniques to perform such transformations within the algebra of (dual) quaternions. In particular, here we define two operators, which allow us to describe the linear transformations (19) without the sequential mapping requirements: $\mathcal{T}_4$ and $\mathcal{T}_8$.

The operator $\mathcal{T}_4$ can perform such linear transformations with quaternions without the need to map it to $\mathbb{R}^4$ and $\mathbb{R}^{4\times 4}$, that is, without leaving the quaternion algebra. Given a vector of quaternions $\boldsymbol{M} = \begin{bmatrix} \boldsymbol{m}_1 & \boldsymbol{m}_2 & \boldsymbol{m}_3 & \boldsymbol{m}_4 \end{bmatrix}$, with $\boldsymbol{m}_i \in \mathbb{H}$, $\mathcal{T}_4$ defines the following transformation

$$\mathcal{T}_4(\boldsymbol{M})\,\boldsymbol{q} = \boldsymbol{m_1} q_1 + \boldsymbol{m}_2 q_2 + \boldsymbol{m}_3 q_3 + \boldsymbol{m_4} q_4 \tag{20}$$

where $\boldsymbol{q} = q_1 + q_2\hat{\imath} + q_3\hat{\jmath} + q_4\hat{k} \in \mathbb{H}$. Similarly, for dual quaternions we have the operator $\mathcal{T}_8$, yielding a transformation such as

$$\mathcal{T}_8(\underline{\boldsymbol{N}})\,\underline{\boldsymbol{q}} = \underline{\boldsymbol{n}}_1 q_1 + \underline{\boldsymbol{n}}_2 q_2 + \underline{\boldsymbol{n}}_3 q_3 + \underline{\boldsymbol{n}}_4 q_4 + \underline{\boldsymbol{n}}_5 q_5 + \underline{\boldsymbol{n}}_6 q_6 + \underline{\boldsymbol{n}}_7 q_7 + \underline{\boldsymbol{n}}_8 q_8 \tag{21}$$

in which $\underline{\boldsymbol{N}}$ is a vector of dual quaternions, $\boldsymbol{n}_i \in \underline{\mathbb{H}}$ and $\underline{\boldsymbol{q}} \in \underline{\mathbb{H}}$.

Now, note that $\underline{\boldsymbol{M}}$ and $\underline{\boldsymbol{N}}$ can represent transformations as (20)-(21) in a much more intuitive manner. Take, for instance, classic Hamiltonian matrix, it

can be better represented by a quaternion vector

$$\begin{aligned}\overset{+}{\boldsymbol{h}}(\boldsymbol{x}) &= [\operatorname{vec}\boldsymbol{x}\ \ \operatorname{vec}(\boldsymbol{x}\hat{\imath})\ \ \operatorname{vec}(\boldsymbol{x}\hat{\jmath})\ \ \operatorname{vec}\left(\boldsymbol{x}\hat{k}\right)];\\ \overset{-}{\boldsymbol{h}}(\boldsymbol{x}) &= [\operatorname{vec}\boldsymbol{x}\ \ \operatorname{vec}(\hat{\imath}\boldsymbol{x})\ \ \operatorname{vec}(\hat{\jmath}\boldsymbol{x})\ \ \operatorname{vec}\left(\hat{k}\boldsymbol{x}\right)].\end{aligned} \tag{22}$$

From (19)-(20), we can rewrite the Hamiltonian in order to stay within the algebra of (dual) quaternions. That is, the same commutative property can be obtained as

$$\begin{aligned}\boldsymbol{y} &= \underline{\operatorname{vec}}(\overset{+}{\boldsymbol{H}}(\boldsymbol{a})\operatorname{vec}\boldsymbol{b}) = \overset{+}{\boldsymbol{h}}(\boldsymbol{a})\,b = \boldsymbol{a}b_0 + \boldsymbol{a}\hat{\imath}b_1 + \boldsymbol{a}\hat{\jmath}b_2 + \boldsymbol{a}\hat{k}b_3,\\ &= \underline{\operatorname{vec}}(\overset{-}{\boldsymbol{H}}(\boldsymbol{b})\operatorname{vec}\boldsymbol{a}) = \overset{-}{\boldsymbol{h}}(\boldsymbol{b})\,a = \boldsymbol{b}a_0 + \hat{\imath}\boldsymbol{b}a_1 + \hat{\jmath}\boldsymbol{b}a_2 + \hat{k}\boldsymbol{b}a_3,\end{aligned} \tag{23}$$

where $\boldsymbol{a} = a_1 + a_2\hat{\imath} + a_3\hat{\jmath} + a_4\hat{k} \in \mathbb{H}$ and $\boldsymbol{b} = b_1 + b_2\hat{\imath} + b_3\hat{\jmath} + b_4\hat{k} \in \mathbb{H}$. The resulting operation has the advantage of being defined only over quaternion multiplication by scalars whilst avoiding the tedious and costly operation of mapping quaternions to $\mathbb{R}^4$ and $\mathbb{R}^{4\times4}$, and back to $\mathbb{H}$.

For dual quaternions, the same matrix algebra operator can be obtained based on a $\mathbb{R}^{8\times8}$ multiplication by the vector representation of a dual quaternion element—and, then mapped back to $\underline{\mathbb{H}}$—as in [10,26]. In contrast, [2] introduces the concept of orthogonal dual matrix combining the Hamilton operator matrix for quaternions with dual unit ε with $\varepsilon^2 = 0, \varepsilon \neq 0$. Herein, to commute between dual quaternions we take a similar approach based on the direct product of dual numbers with quaternion Hamilton operators (18)-(23), that is, the multiplication between dual quaternions $\underline{\boldsymbol{y}} = \underline{\boldsymbol{a}}\underline{\boldsymbol{b}}$ can be rewritten as

$$\begin{aligned}\underline{\boldsymbol{y}} &= \overset{+}{\underline{\boldsymbol{h}}}(\underline{\boldsymbol{a}})\,\underline{\boldsymbol{b}} = \overset{+}{\boldsymbol{h}}(\boldsymbol{a})\boldsymbol{b} + \varepsilon(\overset{+}{\boldsymbol{h}}(\boldsymbol{a})\boldsymbol{b}' + \overset{+}{\boldsymbol{h}}(\boldsymbol{a}')\boldsymbol{b}),\\ &= \overset{-}{\underline{\boldsymbol{h}}}(\underline{\boldsymbol{b}})\,\underline{\boldsymbol{a}} = \overset{-}{\boldsymbol{h}}(\boldsymbol{b})\boldsymbol{a} + \varepsilon(\overset{-}{\boldsymbol{h}}(\boldsymbol{b})\boldsymbol{a}' + \overset{-}{\boldsymbol{h}}(\boldsymbol{b}')\boldsymbol{a}),\end{aligned} \tag{24}$$

where $\underline{\boldsymbol{a}} = \boldsymbol{a} + \varepsilon\boldsymbol{a}'$ and $\underline{\boldsymbol{b}} = \boldsymbol{b} + \varepsilon\boldsymbol{b}'$, and $\overset{+}{\underline{\boldsymbol{h}}}$ and $\overset{-}{\underline{\boldsymbol{h}}}$ yields a orthogonal dual matrix as described in [2].

4 Unconstrained rigid body motion: kinematics and dynamics

The advantages of using dual quaternion algebra for rigid body pose representation and kinematics description are well justified in the literature [13, 19, 23, 26]. In this section, we will exploit these advantages amid unconstrained rigid bodies. The results herein will then built the basis for the modeling over serial manipulators that follows.

The scope of operations with UDQs used for three dimensional motions are particularly interesting when addressing representations of screws. The screw axis is nothing but a tridimensional line in space, and thus can be represented with a Plücker line through a set of three parameters: a unit direction vector

($\boldsymbol{l}$), a point that passes through the line ($\boldsymbol{p}$) and the line's moment $\boldsymbol{m} = \boldsymbol{p} \times \boldsymbol{l}$. This set of parameters are denoted Plücker coordinates. A Plücker line can be defined by the dual quaternion, $\underline{\boldsymbol{l}} = \boldsymbol{l} + \varepsilon \boldsymbol{m}$, with $\boldsymbol{m} = \boldsymbol{p} \times \boldsymbol{l}$ being the moment around the origin and $\boldsymbol{m}$,$\boldsymbol{p}$,$\boldsymbol{l} \in \mathbb{H}_0$ being pure quaternions [19, 26].

More so, when attached to a fixed point the line may rotate and perform a motion with an arc-like trajectory. Much like with imaginary numbers, the dual quaternions exponential is suited to represent a curve in space. Thus, as shown in [26], for the dual quaternion $\underline{\boldsymbol{q}}$, the exponential can be defined as

$$exp(\underline{\boldsymbol{q}}) = \mathcal{P}\left(exp(\underline{\boldsymbol{q}})\right) + \varepsilon \mathcal{D}\left(\underline{\boldsymbol{q}}\right) \mathcal{P}\left(exp(\underline{\boldsymbol{q}})\right), \tag{25}$$

with $\mathcal{P}\left(exp(\underline{\boldsymbol{q}})\right)$ being defined as

$$\begin{cases} cos \left\| \mathcal{P}\left(\underline{\boldsymbol{q}}\right) \right\| + \frac{sin \left\| \mathcal{P}(\underline{\boldsymbol{q}}) \right\|}{\left\| \mathcal{P}(\underline{\boldsymbol{q}}) \right\|} \mathcal{P}\left(\underline{\boldsymbol{q}}\right) & \text{if } \left\| \mathcal{P}\left(\underline{\boldsymbol{q}}\right) \right\| \neq 0 \\ 1 & \text{o.w.} \end{cases} \tag{26}$$

We highlight here the manner in which the exponential and Plucker line relate to representing a screw axis displacement. That is, given the Plucker line $\underline{\boldsymbol{s}} = \boldsymbol{l} + \varepsilon \boldsymbol{m}$ and the dual angle $\hat{\theta} = \theta + \varepsilon d$, with $\theta, d \in \mathbb{R}$, we can represent the screw axis' position as the UDQ $\underline{\boldsymbol{x}} \in \text{Spin}(3) \ltimes \mathbb{R}^3$, that is,

$$\underline{\boldsymbol{x}} = exp\left(\frac{\hat{\theta}}{2}\underline{\boldsymbol{s}}\right) = cos\left(\frac{\hat{\theta}}{2}\right) + \underline{\boldsymbol{s}} sin\left(\frac{\hat{\theta}}{2}\right). \tag{27}$$

The above expression is a generalization which exploits the dual propriety to couple the rotation of the screw (θ) and its translation (d). For brevity and to simplify the proposed method description, we will only focus on revolute joints manipulators—which in turn yields $d = 0$ and $\hat{\theta} = \theta$ to the remainder of this text. Note, nonetheless, that extension to prismatic joints should be trivial as stressed in [19].

Furthermore, the unit dual quaternion described in (27) can also be rewritten in terms of a quaternion pair as follows

$$\underline{\boldsymbol{x}} = \boldsymbol{r} + \varepsilon(1/2)\boldsymbol{r}\boldsymbol{p}^b. \tag{28}$$

In this case, $\boldsymbol{r} \in \text{Spin}(3)$ denotes the body attitude w.r.t. a inertial frame, whilst $\boldsymbol{p}^b = p_x\hat{\imath} + p_y\hat{\jmath} + p_z\hat{k}$, is the rigid-body position expressed in the body frame. The expression (28) is also useful to describe rigid displacements through a rotation $\boldsymbol{r}$ followed by a translation $\boldsymbol{p}^b$.

We can then derive the first order kinematic equation to obtain the velocity, $\dot{\underline{\boldsymbol{x}}}$ (at the tangent space at $\underline{\boldsymbol{x}}$, as is done in [26][5]

$$\dot{\underline{\boldsymbol{x}}} = \tfrac{1}{2}\underline{\boldsymbol{x}}\underline{\boldsymbol{\omega}}^b, \tag{29}$$

[5] For further information on dual quaternion based kinematics please refer to [10, 26], whilst the readers are refereed to [12, 27] for the dynamics—limited to unconstrained rigid bodies.

with $\underline{\boldsymbol{\omega}}^b$ being the generalized twist in body frame with $\underline{\boldsymbol{\omega}}^b = \boldsymbol{\omega}^b + \varepsilon \boldsymbol{v}^b$, where $\boldsymbol{\omega}^b \in \mathbb{H}_0$ is the angular velocity and $\boldsymbol{v}^b \in \mathbb{H}_0$ is the linear velocity.

Although not much has been published on the formulation of the rigid body dynamics with dual quaternion, there are some interesting results in the literature. In the formulation first proposed by Dooley in [7], UDQs are directly applied to solve the general dynamics problem of an unconstrained rigid body (e.g., a holonomic flying robot). However, as argued in [27] the equations of motion are overly complicated, lack an intuitive meaning and are hard to implement. In this context, the proposition in [27] then describes a single rigid body dynamics from wrench/twist formulations, but instead of taking full advantage of the UDQ representation, the operations are made element-wise rather than an operation of a dual object. A more thorough solution is presented in [12], and in this work, we take advantage of such a formulation. To describe the system dynamics, the authors explicitly describe the angular and linear momentums, as well as the resulting kinematic wrenches, stemming from the body motion by introducing an augmented linear matrix representation of the body mass and inertia tensor, the dual inertia matrix (a real 8×8 symmetric matrix). In other words, in [12], the body mass $m \in \mathbb{R}$ is described as a diagonal matrix within a block diagonal matrix structure together the inertia tensor of the body about its center of mass, that is, $\boldsymbol{I}^b \in \mathbb{R}^{3\times 3}$, which in turn relied on a manifold mapping from $\mathbb{H}_0 \otimes \mathbb{D}$ to $\mathbb{R}^8$ and its inverse mapping. The result proposed in [12] also required a complicated switch operator to properly describe the resulting angular and linear momentums (and wrenches) in the dual quaternion primary and dual parts.

In contrast, herein, we take advantage of the pure dual quaternion representation for velocities and accelerations to derive a more direct description of angular and linear momentums and wrenches. Since the linear part of the generalized twist and body acceleration are described by a pure quaternion, as seen in (29), the resulting momentum and force is obtained by a simple scalar multiplication. Whilst the angular momentum and torque can be extracted directly from the orthonormal vectors of the inertia tensor about its center of mass w.r.t. the body axis multiplied by the angular velocities and accelerations. The resulting dual quaternion momentum and wrenches stemming, respectively, from body velocities and accelerations are therefore given by a *Dual Quaternion Inertia Transformation*, $G : \mathbb{H}_0 \otimes \mathbb{D} \to \mathbb{H}_0 \otimes \mathbb{D}$, as

$$\begin{aligned} G(\underline{\boldsymbol{\omega}}_b) \triangleq & m\mathcal{D}(\underline{\boldsymbol{\omega}}_b) + \varepsilon \left(\boldsymbol{I}_x^b \omega_x + \boldsymbol{I}_y^b \omega_y + \boldsymbol{I}_z^b \omega_z \right), \\ G(\underline{\dot{\boldsymbol{\omega}}}_b) \triangleq & m\mathcal{D}(\underline{\dot{\boldsymbol{\omega}}}_b) + \varepsilon \left(\boldsymbol{I}_x^b \dot{\omega}_x + \boldsymbol{I}_y^b \dot{\omega}_y + \boldsymbol{I}_z^b \dot{\omega}_z \right) \end{aligned} \tag{30}$$

where $\underline{\boldsymbol{\omega}}_b$ and $\underline{\dot{\boldsymbol{\omega}}}_b$ are the body generalized twist and accelerations, w.r.t. center of mass, with angular component being given by $\mathcal{P}(\underline{\boldsymbol{\omega}}_b) = \omega_x \hat{\imath} + \omega_y \hat{\jmath} + \omega_z \hat{k}$ and $\mathcal{P}(\underline{\dot{\boldsymbol{\omega}}}_b) = \dot{\omega}_x \hat{\imath} + \dot{\omega}_y \hat{\jmath} + \dot{\omega}_z \hat{k}$. And, the inertia tensor of the rigid body characterized by $\boldsymbol{I}_x^b = I_{xx}\hat{\imath} + I_{xy}\hat{\jmath} + I_{xz}\hat{k}$, $\boldsymbol{I}_y^b = I_{xy}\hat{\imath} + I_{yy}\hat{\jmath} + I_{yz}\hat{k}$, and $\boldsymbol{I}_z^b = I_{xz}\hat{\imath} + I_{yz}\hat{\jmath} + I_{yz}\hat{k}$,w.r.t to the coordinate axis of the center of mass—note they can be viewed as pure quaternion representations of column vectors of the tensor matrix described in

[12]. Finally, from (30), we can compute the rigid body dynamics as

$$G(\dot{\underline{\boldsymbol{\omega}}}_b) + \underline{\boldsymbol{\omega}}_b \times G(\underline{\boldsymbol{\omega}}_b) - \underline{\boldsymbol{f}}_b = 0, \tag{31}$$

in which $\underline{\boldsymbol{f}}_b = \boldsymbol{f_b} + \boldsymbol{\varepsilon m_b}$ is the body wrench ($\boldsymbol{f_b}$ being the forces and $\boldsymbol{m}_b$ the moments applied about the body's center of mass).

Moreover, the dual inertia operator can also be extended as an operator over (8), that is, given $\underline{\boldsymbol{P}} = \left[\, \underline{\boldsymbol{p}}_1 \;\; \underline{\boldsymbol{p}}_2 \;\; \cdots \;\; \underline{\boldsymbol{p}}_n \,\right]^T \in \underline{\mathbb{H}}_0^n$, $\boldsymbol{G} : \underline{\mathbb{H}}_0^n \longrightarrow \underline{\mathbb{H}}_0^n$ yields

$$\boldsymbol{G}(\underline{\boldsymbol{P}}) = \left[\, G(\underline{\boldsymbol{p}}_1) \;\; G(\underline{\boldsymbol{p}}_2) \;\; \ldots \;\; G(\underline{\boldsymbol{p}}_n) \,\right]^T. \tag{32}$$

In which $G(\bullet)$ is the Dual Quaternion Inertia Transformation from (30).

The dual quaternion inertia transformation can also be defined for matrices in a similar manner to what was done in (32). In this case our operator shall be defined as $\boldsymbol{G} : \underline{\mathbb{H}}_0^{n\times m} \longrightarrow \underline{\mathbb{H}}_0^{n\times m}$ and we apply $G(\bullet)$ in every element of the matrix.

Following the definitions in Subsection 2.3, we may also derive some proprieties for the operator (32) w.r.t. the inertia transformation in (32). Given the dual quaternion-vectors $\underline{\boldsymbol{P}}_1 = \left[\, \underline{\boldsymbol{p}}_{1,1} \;\; \underline{\boldsymbol{p}}_{1,2} \;\; \cdots \;\; \underline{\boldsymbol{p}}_{1,n} \,\right]^T$ and $\underline{\boldsymbol{P}}_2 \left[\, \underline{\boldsymbol{p}}_{2,1} \;\; \underline{\boldsymbol{p}}_{2,2} \;\; \cdots \;\; \underline{\boldsymbol{p}}_{2,n} \,\right]^T \in \underline{\mathbb{H}}_0^n$ and the constant $\boldsymbol{K} \in \mathbb{R}^n$ we have

1. $\boldsymbol{G}\left(\underline{\boldsymbol{P}}_1 + \underline{\boldsymbol{P}}_2\right) = \boldsymbol{G}\left(\underline{\boldsymbol{P}}_1\right) + \boldsymbol{G}\left(\underline{\boldsymbol{P}}_2\right)$;
2. $\boldsymbol{G}\left(\underline{\boldsymbol{P}}_1 \boldsymbol{K}\right) = \boldsymbol{G}\left(\underline{\boldsymbol{P}}_1\right) \boldsymbol{K}$.

Both proprieties can be proved by inspection.

Finally, it is also useful to describe the congruence transformation that takes velocities, accelerations and wrenches from one frame to the other, that is an adjoint mapping [26]. Given the frame transformation $\underline{\boldsymbol{x}} \in \mathrm{Spin}(3) \ltimes \mathbb{R}^3$ from $\mathcal{F}_0$ to $\mathcal{F}_1$, the adjoint operation that takes $\underline{\boldsymbol{\mu}} \in \mathbb{H}_0 \otimes \mathbb{D}$ from $\mathcal{F}_0$ to be expressed in $\mathcal{F}_1$ yields

$$\underline{\boldsymbol{\mu}}^1 = Ad(\underline{\boldsymbol{x}})\underline{\boldsymbol{\mu}} = \underline{\boldsymbol{x}}\underline{\boldsymbol{\mu}}\underline{\boldsymbol{x}}^*, \tag{33}$$

In [6] an alternative and more efficient manner for representing the adjoint transformations has been presented, in which (33) can be expressed in terms of Hamilton operators as described in Section 3. Here, however, we present yet another alternative for calculating this transformation, which will be used throughout the remainder of this work. That is, we define the operators $\boldsymbol{\mathcal{A}_P}$ and $\boldsymbol{\mathcal{A}_D}$.

$$\mathcal{A}_{\mathcal{P}}(\underline{\boldsymbol{x}})\boldsymbol{\mu} = \boldsymbol{\mathcal{A}}_{P_i}\mu_1 + \boldsymbol{\mathcal{A}}_{P_j}\mu_2 + \boldsymbol{\mathcal{A}}_{P_k}\mu_3. \tag{34}$$

with $\boldsymbol{\mathcal{A}}_{P_i} = (\boldsymbol{x_P}\hat{\imath}\boldsymbol{x}_P^*)$, $\boldsymbol{\mathcal{A}}_{P_j} = (\boldsymbol{x_P}\hat{\jmath}\boldsymbol{x}_P^*)$ and $\boldsymbol{\mathcal{A}}_{P_k} = \left(\boldsymbol{x_P}\hat{k}\boldsymbol{x}_P^*\right)$ being the individual base transformation. And

$$\mathcal{A}_{\mathcal{D}}(\underline{\boldsymbol{x}})\boldsymbol{\mu} = \boldsymbol{\mathcal{A}}_{D_i}\mu_1 + \boldsymbol{\mathcal{A}}_{D_j}\mu_2 + \boldsymbol{\mathcal{A}}_{D_k}\mu_3, \tag{35}$$

with $\boldsymbol{\mathcal{A}}_{D_i} = 2\,\mathrm{Im}\left(\boldsymbol{x_P}\hat{\imath}\boldsymbol{x_D}^*\right)$, $\boldsymbol{\mathcal{A}}_{D_i} = 2\,\mathrm{Im}\left(\boldsymbol{x_P}\hat{\jmath}\boldsymbol{x_D}^*\right)$ and $\boldsymbol{\mathcal{A}}_{D_i} = 2\,\mathrm{Im}\left(\boldsymbol{x_P}\hat{k}\boldsymbol{x_D}^*\right)$. And, furthermore, we calculate the adjoint as

$$Ad(\underline{\boldsymbol{x}})\underline{\boldsymbol{p}} = \underline{\boldsymbol{x}}\underline{\boldsymbol{p}}\underline{\boldsymbol{x}}^* = \mathcal{A}_{\mathcal{P}}(\underline{\boldsymbol{x}})\boldsymbol{p_P} + \varepsilon\left(\mathcal{A}_{\mathcal{P}}(\underline{\boldsymbol{x}})\boldsymbol{p_D} + \mathcal{A}_{\mathcal{D}}(\underline{\boldsymbol{x}})\boldsymbol{p_P}\right). \tag{36}$$

5 Inverse dynamics with unit dual quaternions

In this section, we propose a novel formalism for the recursive Newton-Euler inverse dynamics algorithm that combines the advantages of screw theory formalized over the dual quaternion algebra in an unified framework representation. As summarized in Algorithm 1, given known joint positions, velocities and accelerations, we compute and propagate the pose, velocities and accelerations of the links based on Newton's law of motion using the same algebraic structure without the need of D-H parameters computations—similarly to [19]. Then, we compute the wrenches (forces and moments) acting on each link in a recursive fashion, starting from the wrenches applied to the end-effector. Both forward and backward iteration are described herein with the same dual quaternion framework exploiting the manifold capabilities to represent rigid body pose, velocities, accelerations, momentum, and wrenches.

Algorithm 1 Dual-quaternion based recursive Newton-Euler Inverse Dynamics for $\boldsymbol{n}$ DoF Manipulator

Initialization:

At the home config., set the frames $\mathcal{F}_0$ to the base, $\mathcal{F}_1$ to $\mathcal{F}_n$ to the n-links' center of mass, and $\mathcal{F}_{n+1}$ to the end-pose.

Set $\underline{\boldsymbol{\delta}}^i_{i-1}$, $\underline{\boldsymbol{a}}_i$ as the pose of $\mathcal{F}_{i-1}$ and the screw axis of joint i expressed in $\mathcal{F}_i$, with null vel. $\underline{\boldsymbol{\omega}}_0$=0 and gravity accel. $\dot{\underline{\boldsymbol{\omega}}}_0 = -\varepsilon\boldsymbol{g}$ at base and end-pose wrench $\underline{\boldsymbol{f}}_{EF} = \boldsymbol{f}_{EF} + \varepsilon\boldsymbol{m}_{EF}$.

Forward iteration

$\underline{\boldsymbol{x}}^{i-1}_i(\theta_i(t)) = \underline{\boldsymbol{\delta}}^i_{i-1} exp\left(\underline{\boldsymbol{a}}_i \frac{\Delta\theta_i}{2}\right)$

Compute $\mathcal{A}_P(\underline{\boldsymbol{x}}^{i-1}_i)$ and $\mathcal{A}_D(\underline{\boldsymbol{x}}^{i-1}_i)$ from (34) and (35)

$\underline{\boldsymbol{\omega}}_i = \underline{\boldsymbol{a}}_i\dot{\theta}_i(t) + Ad(\underline{\boldsymbol{x}}^{i-1}_i)\underline{\boldsymbol{\omega}}_{i-1}$

$\dot{\underline{\boldsymbol{\omega}}}_i = \underline{\boldsymbol{a}}_i\ddot{\theta}_i(t) + Ad(\underline{\boldsymbol{x}}^{i-1}_i)\dot{\underline{\boldsymbol{\omega}}}_{i-1} + \left(\underline{\boldsymbol{\omega}}_i \times \underline{\boldsymbol{a}}_i\right)\dot{\theta}_i(t)$

Backward iteration

$\underline{\boldsymbol{f}}_{R,i} = \underline{\boldsymbol{\omega}}_i \times \left(G_i\left(\underline{\boldsymbol{\omega}}_i\right)\right) + G_i\left(\dot{\underline{\boldsymbol{\omega}}}_i\right)$

$\underline{\boldsymbol{f}}_i = \underline{\boldsymbol{f}}_{R,i} + Ad(\underline{\boldsymbol{x}}^{i+1}_i)\underline{\boldsymbol{f}}_{i+1}$

$\tau_i = \underline{\boldsymbol{f}}_i \odot \underline{\boldsymbol{a}}_i$

5.1 Forward Pose, velocity and acceleration kinematics

The forward iteration of the algorithm, much like the well established FKM for open chains, aims at obtaining the pose and velocity for each of the robot's links. For the dynamic analysis, we also compute the twist's first derivative, that is, the acceleration. All operations are performed using dual quaterion algebra.

It is notable that Algorithm 1 exploits the screw theory formulation from Section 4 for computing the FKM, rather than the more traditional alternative, the Denavit Hartemberg (D-H) notation. In [19, 26], we see different recursive techniques being presented to solve the forward kinematics model (FKM) of a serial manipulator based on dual quaternions. In particular, [19] makes a compelling case for the screw theory alternative. As it is argued, this approach based on line transformations is considerably more efficient than the other techniques such as the computation of the FKM based on D-H approach. More so, the screw representation is much more intuitive and the position of the frames can be chosen without the many restrictions that D-H parameters impose, for instance, each link's frame of reference can be at the center of mass of the body, which simplifies the dynamical equations.

From dual-quaternion algebra, we can easily represent the end-effector pose of an n-joint serial manipulator by means of successive rigid transformations between its links, i.e., taking an initial joint configuration, we can compute the end-effector's pose by means of the relative displacements of each links from such initial configuration. Let us define $\boldsymbol{\theta}^h \in \mathbb{R}^n$ as initial (home) joint configuration. At this configuration, let $\underline{\boldsymbol{\delta}}_{i-1}^{i} \in \mathrm{Spin}(3) \ltimes \mathbb{R}^3$ denote the configuration of $\mathcal{F}_{i-1}$ expressed in $\mathcal{F}_i$, where $\mathcal{F}_0$ is the frame coinciding with the base. Frames $\mathcal{F}_1$ to $\mathcal{F}_n$ should be at the center of mass of each link and $\mathcal{F}_{n+1}$ is the reference frame attached to the end-effector. We then represent the transformation from one frame to another as $\underline{\boldsymbol{\delta}}_{i}^{i-1} = \left(\underline{\boldsymbol{\delta}}_{i-1}^{0}\right)^* \underline{\boldsymbol{\delta}}_{i}^{0}$ and $\left(\underline{\boldsymbol{\delta}}_{i}^{i-1}\right)^* = \underline{\boldsymbol{\delta}}_{i-1}^{i} = \left(\underline{\boldsymbol{\delta}}_{i}^{0}\right)^* \underline{\boldsymbol{\delta}}_{i-1}^{0}$.

Hence, the rigid transformation from base to end-effector is given by

$$\underline{\boldsymbol{\delta}}_{EF} = \underline{\boldsymbol{\delta}}_{1}^{0} \underline{\boldsymbol{\delta}}_{2}^{1} ... \underline{\boldsymbol{\delta}}_{n+1}^{n}. \tag{37}$$

Taking the screw axis of the link at home position as the Plucker line $\underline{\boldsymbol{s}}_i = \boldsymbol{\omega} + \varepsilon \boldsymbol{v}$, where $\boldsymbol{\omega}$ and $\boldsymbol{v}$ are the pure quaternions of the angular and linear velocity, respectively and, given a easily defined point $\boldsymbol{p}$ in the screw and we have $\boldsymbol{v} = \boldsymbol{p} \times \boldsymbol{\omega}$. This representation is in $\mathcal{F}_0$. To transform it to the link's frame of reference, we apply a transformation $\underline{\boldsymbol{a}}_i = Ad\left(\underline{\boldsymbol{\delta}}_{0}^{i}\right) \underline{\boldsymbol{s}}_i$ from frame $\mathcal{F}_0$ to $\mathcal{F}_i$.

Now, from the current joint position $\theta_i(t)$, the pose transformation of the link i expressed in frame $\mathcal{F}_{i-1}$ is given by

$$\underline{\boldsymbol{x}}_{i}^{i-1}(\theta_i(t)) = \underline{\boldsymbol{\delta}}_{i-1}^{i} exp\left(\underline{\boldsymbol{a}}_i \frac{\Delta\theta_i}{2}\right), \tag{38}$$

with $\Delta\theta_i = \theta_i(t) - \theta^h$ being the position deviation from the home configuration. Then, the end-pose can be calculated by

$$\underline{\boldsymbol{x}}_{EF} = \underline{\boldsymbol{x}}_{1}^{0} \underline{\boldsymbol{x}}_{2}^{1} ... \underline{\boldsymbol{x}}_{n+1}^{n}. \tag{39}$$

Considering the twist of a body defined as in (29), and that for serial chains the twist of a particular link is the sum of the twist at previous links (propagated from base to end-effector) added to the twist generated by the velocities of that link's joint, we can write the twist equation of joint i as

$$\underline{\boldsymbol{\omega}}_i = \underline{\boldsymbol{a}}_i \dot{\Delta\theta}_i(t) + Ad(\underline{\boldsymbol{x}}_{i}^{i-1}) \underline{\boldsymbol{\omega}}_{i-1}, \tag{40}$$

where the adjoint transformation is required to properly propagate velocities to the new frame of reference. Note similar strategy to this point has been proposed in [19] showing considerable computational complexity costs improvements over the use of D-H parameters. Herein, we extend such result taking individual links and end-effector pose acceleration into account. Taking the derivative of (40), we have

$$\begin{aligned}\dot{\underline{\boldsymbol{\omega}}}_i =& \underline{\boldsymbol{a}}_i \Delta\ddot{\theta}_i(t) + \frac{d}{dt}\left(\underline{\boldsymbol{x}}_i^{i-1}\underline{\boldsymbol{\omega}}_{i-1}\underline{\boldsymbol{x}}_{i-1}^{i}\right) \\ =& \underline{\boldsymbol{a}}_i \Delta\ddot{\theta}_i(t) + \underline{\boldsymbol{x}}_i^{i-1}\dot{\underline{\boldsymbol{\omega}}}_{i-1}\underline{\boldsymbol{x}}_{i-1}^{i} + \dot{\underline{\boldsymbol{x}}}_i^{i-1}\underline{\boldsymbol{\omega}}_{i-1}\underline{\boldsymbol{x}}_{i-1}^{i} + \underline{\boldsymbol{x}}_i^{i-1}\underline{\boldsymbol{\omega}}_{i-1}\dot{\underline{\boldsymbol{x}}}_{i-1}^{i}\end{aligned} \tag{41}$$

whilst the the derivative of (38) give us

$$\dot{\underline{\boldsymbol{x}}}_{i-1}^{i}(\theta_i(t)) = \frac{1}{2}\underline{\boldsymbol{x}}_{i-1}^{i}\underline{\boldsymbol{a}}_i \Delta\dot{\theta}_i(t). \tag{42}$$

Now, combining (41) and (42), we have

$$\dot{\underline{\boldsymbol{\omega}}}_i = \underline{\boldsymbol{a}}_i \Delta\ddot{\theta}_i(t) + Ad(\underline{\boldsymbol{x}}_i^{i-1})\dot{\underline{\boldsymbol{\omega}}}_{i-1} + \left(\underline{\boldsymbol{x}}_i^{i-1}\underline{\boldsymbol{\omega}}_{i-1}\underline{\boldsymbol{x}}_{i-1}^{i}\underline{\boldsymbol{a}}_i - \underline{\boldsymbol{a}}_i\underline{\boldsymbol{x}}_i^{i-1}\underline{\boldsymbol{\omega}}_{i-1}\underline{\boldsymbol{x}}_{i-1}^{i}\right)\frac{\Delta\dot{\theta}_i(t)}{2}, \tag{43}$$

which can further be simplified by taking the cross product definition (4) as

$$\dot{\underline{\boldsymbol{\omega}}}_i = \underline{\boldsymbol{a}}_i \Delta\ddot{\theta}_i(t) + Ad(\underline{\boldsymbol{x}}_i^{i-1})\dot{\underline{\boldsymbol{\omega}}}_{i-1} + (\underline{\boldsymbol{x}}_i^{i-1}\underline{\boldsymbol{\omega}}_{i-1}\underline{\boldsymbol{x}}_{i-1}^{i} \times \underline{\boldsymbol{a}}_i)\Delta\dot{\theta}_i(t)$$

From (40), we obtain a final expression for the recursive forward model for the accelerations derived solely from screw theory based on dual quaternion algebra,

$$\dot{\underline{\boldsymbol{\omega}}}_i = \underline{\boldsymbol{a}}_i \Delta\ddot{\theta}_i(t) + Ad(\underline{\boldsymbol{x}}_i^{i-1})\dot{\underline{\boldsymbol{\omega}}}_{i-1} + (\underline{\boldsymbol{\omega}}_i \times \underline{\boldsymbol{a}}_i)\,\Delta\dot{\theta}_i(t). \tag{44}$$

5.2 Backward Iteration

In the backwards iteration, given provided dual quaternion wrenches acting at the end-effector and the resulting velocities and accelerations of the link's center of mass from the forward kinematics, we can compute the required torques to be applied at the joints to obtain the prescribed motion.

From individual rigid body dynamics stemming from each link's velocities and accelerations (31), the resulting wrench acting on the link i can be computed as

$$\underline{\boldsymbol{f}}_{R,i} = \underline{\boldsymbol{\omega}}_i \times (G_i(\underline{\boldsymbol{\omega}}_i)) + G_i(\dot{\underline{\boldsymbol{\omega}}}_i), \tag{45}$$

where $G_i(*)$ is the *Dual Quaternion Inertia Transformation* as in (30) for the *ith* link. Now, as the resulting forces must be equal to the sum of the forces being applied at attached joints, and given the serial kinematic chain propagation between links, the resulting wrench $\underline{\boldsymbol{f}}_{R,i}$ is given

$$\underline{\boldsymbol{f}}_{R,i} = \underline{\boldsymbol{f}}_i + Ad\left(\underline{\boldsymbol{x}}_i^{i+1}\right)\underline{\boldsymbol{f}}_{i+1}. \tag{46}$$

By combining (45) and (46), we obtain the full dynamic equation of the required dual quaternion wrench at link i,that is,

$$\underline{\boldsymbol{f}}_i = \underline{\boldsymbol{\omega}}_i \times (G_i(\underline{\boldsymbol{\omega}}_i)) + G_i(\dot{\underline{\boldsymbol{\omega}}}_i) - Ad\left(\underline{\boldsymbol{x}}_i^{i+1}\right) \underline{\boldsymbol{f}}_{i+1}, \tag{47}$$

which, thereafter, can be used to find the torque at the joint in the axis direction,

$$\tau_i = \underline{\boldsymbol{f}}_i \odot \left[\mathcal{D}(\underline{\boldsymbol{a}}_i) + \varepsilon \mathcal{P}(\underline{\boldsymbol{a}}_i)\right]. \tag{48}$$

Both the dual quaternion forward iteration and recursive propagation of the wrenches required to compute the resulting joint torques are summarized in Algorithm 1.

5.3 Closed form algorithm

To describe the inverse dynamics in a closed form, we exploit Algorithm 1 and take advantage of the dual quaternion based matrices and vector formulation from Section 2. We may define the joint, torque, wrenches, twist vectors, respectively as

$$\boldsymbol{\theta} = \left[\, \theta_1(t)\ \theta_2(t)\ \ldots\ \theta_n(t) \,\right]^T \in \mathbb{R}^n, \tag{49}$$

$$\boldsymbol{\tau} = \left[\, \tau_1\ \tau_2\ \ldots\ \tau_n \,\right]^T \in \mathbb{R}^n, \tag{50}$$

$$\underline{\boldsymbol{F}}_R = \left[\, \underline{\boldsymbol{f}}_{R,1}\ \underline{\boldsymbol{f}}_{R,2}\ \cdots\ \underline{\boldsymbol{f}}_{R,n} \,\right]^T \in \underline{\mathbb{H}}_0^n, \tag{51}$$

$$\underline{\boldsymbol{F}} = \left[\, \underline{\boldsymbol{f}}_1\ \underline{\boldsymbol{f}}_2\ \cdots\ \underline{\boldsymbol{f}}_n \,\right]^T \in \underline{\mathbb{H}}_0^n \tag{52}$$

$$\underline{\boldsymbol{W}} = \left[\, \underline{\boldsymbol{\omega}}_1\ \underline{\boldsymbol{\omega}}_2\ \ldots\ \underline{\boldsymbol{\omega}}_n \,\right]^T \in \underline{\mathbb{H}}_0^n \tag{53}$$

and the derivatives of (49) and (53) as

$$\dot{\boldsymbol{\theta}}\left[\, \dot{\theta}_1(t)\ \dot{\theta}_2(t)\ \ldots\ \dot{\theta}_n(t) \,\right]^T \in \mathbb{R}^n, \tag{54}$$

$$\ddot{\boldsymbol{\theta}} = \left[\, \dot{\theta}_1(t)\ \dot{\theta}_2(t)\ \ldots\ \dot{\theta}_n(t) \,\right]^T \in \mathbb{R}^n, \tag{55}$$

$$\dot{\underline{\boldsymbol{W}}} = \left[\, \dot{\underline{\boldsymbol{\omega}}}_1\ \dot{\underline{\boldsymbol{\omega}}}_2\ \ldots\ \dot{\underline{\boldsymbol{\omega}}}_n \,\right]^T \in \underline{\mathbb{H}}_0^n. \tag{56}$$

From the quaternion-vector based formulation, we may group expressions in Algorithm 1 together and rewrite the equations such as in Table 1, in which, for an n-DoF robot the matrix $\underline{\boldsymbol{A}} \in \underline{\mathbb{H}}^{n\times n}$ and its counterpart $\underline{\boldsymbol{A}}_s \in \underline{\mathbb{H}}^{n\times n}$ are defined as

$$\underline{\boldsymbol{A}} = \begin{bmatrix} \underline{\boldsymbol{a}}_1 & 0 & 0 & \ldots & 0 \\ 0 & \underline{\boldsymbol{a}}_2 & 0 & \ldots & 0 \\ 0 & 0 & \underline{\boldsymbol{a}}_3 & \ldots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & \underline{\boldsymbol{a}}_n \end{bmatrix} \quad \text{and} \quad \underline{\boldsymbol{A}}_S = \begin{bmatrix} \mathcal{D}(\underline{\boldsymbol{a}}_1) + \varepsilon\mathcal{P}(\underline{\boldsymbol{a}}_1) \\ \mathcal{D}(\underline{\boldsymbol{a}}_2) + \varepsilon\mathcal{P}(\underline{\boldsymbol{a}}_2) \\ \mathcal{D}(\underline{\boldsymbol{a}}_3) + \varepsilon\mathcal{P}(\underline{\boldsymbol{a}}_3) \\ \vdots \\ \mathcal{D}(\underline{\boldsymbol{a}}_n) + \varepsilon\mathcal{P}(\underline{\boldsymbol{a}}_n) \end{bmatrix}, \tag{57}$$

Equation
1 $\underline{\boldsymbol{W}} = \underline{\boldsymbol{L}}\left(\underline{\boldsymbol{A}}\dot{\boldsymbol{\theta}} + \underline{\boldsymbol{W}}_{\mathrm{BASE}}\right)$
2 $\dot{\underline{\boldsymbol{W}}} = \underline{\boldsymbol{L}}\left(\underline{\boldsymbol{A}}\ddot{\boldsymbol{\theta}} + \underline{\boldsymbol{C}}\ddot{\boldsymbol{\theta}} + \dot{\underline{\boldsymbol{W}}}_{\mathrm{BASE}}\right)$
3 $\underline{\boldsymbol{F}}_R = \underline{\boldsymbol{W}} \otimes \boldsymbol{G}\left(\underline{\boldsymbol{W}}\right) + \boldsymbol{G}\left(\dot{\underline{\boldsymbol{W}}}\right)$
4 $\underline{\boldsymbol{F}} = \underline{\boldsymbol{L}}^{*}\left(\underline{\boldsymbol{F}}_R + \underline{\boldsymbol{F}}_{\mathrm{BASE}}\right).$
5 $\boldsymbol{\tau} = \underline{\boldsymbol{F}} \circledast \underline{\boldsymbol{A}}_S$

Table 1: List of main operations in Dual Quaternionic-Matrix formulation

and the matrix $\underline{\boldsymbol{C}} \in \underline{\mathbb{H}}^{n\times n}$ as

$$\underline{\boldsymbol{C}} = \begin{bmatrix} (\underline{\boldsymbol{\omega}}_1 \times \underline{\boldsymbol{a}}_1) & 0 & 0 & \dots & 0 \\ 0 & (\underline{\boldsymbol{\omega}}_2 \times \underline{\boldsymbol{a}}_2) & 0 & 0 & 0 \\ 0 & 0 & (\underline{\boldsymbol{\omega}}_3 \times \underline{\boldsymbol{a}}_3) & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & (\underline{\boldsymbol{\omega}}_n \times \underline{\boldsymbol{a}}_n) \end{bmatrix}, \tag{58}$$

and the vectors $\underline{\boldsymbol{W}}_{\mathrm{BASE}} \in \underline{\mathbb{H}}^n$, $\dot{\underline{\boldsymbol{W}}}_{\mathrm{BASE}} \in \underline{\mathbb{H}}^n$, $\underline{\boldsymbol{F}}_{\mathrm{BASE}} \in \underline{\mathbb{H}}^n$ contain the respective initialization values.

In addition to the elements presented above, we also have the variable $\underline{\boldsymbol{L}} \in \underline{\mathbb{H}}^{n\times n}$. This stems from the adjoint term on the algorithm, that is, considering

$$Ad(\underline{\boldsymbol{x}})\underline{\boldsymbol{p}} = \underline{\boldsymbol{x}}\underline{\boldsymbol{p}}\underline{\boldsymbol{x}}^{*} \tag{59}$$

we take advantage of the Hamilton operator to isolate the adjoint such as

$$A_{\mathrm{H}}(\underline{\boldsymbol{x}})\underline{\boldsymbol{p}} = \overset{+}{\underline{\boldsymbol{h}}}\left(\underline{\boldsymbol{x}}\right)\overset{-}{\underline{\boldsymbol{h}}}\left(\underline{\boldsymbol{x}}^{*}\right)\underline{\boldsymbol{p}}. \tag{60}$$

in which $A_{\mathrm{H}}(\underline{\boldsymbol{x}})$ will be a dual quaternion operator such that $A_{\mathrm{H}}(\underline{\boldsymbol{x}}) = \overset{+}{\underline{\boldsymbol{h}}}\left(\underline{\boldsymbol{x}}\right)\overset{-}{\underline{\boldsymbol{h}}}\left(\underline{\boldsymbol{x}}^{*}\right)$. Furthermore, we may extend this operator to the matrix formulation to build

$$\underline{\boldsymbol{Q}} = \begin{bmatrix} 0 & 0 & 0 & \dots & 0 \\ A_{\mathrm{H}}(\underline{\boldsymbol{x}}_2^1) & 0 & 0 & \dots & 0 \\ & A_{\mathrm{H}}(\underline{\boldsymbol{x}}_3^2) & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & A_{\mathrm{H}}(\underline{\boldsymbol{x}}_n^{n-1}) & 0 \end{bmatrix}. \tag{61}$$

Moreover, from (61) we can define $\underline{\boldsymbol{L}}_{\mathrm{tmp}} = \underline{\boldsymbol{I}}_n - \underline{\boldsymbol{Q}}$ in which $\underline{\boldsymbol{I}}_n$ is the identity $n \times n$ dual quaternion matrix and $\underline{\boldsymbol{L}}^{-1} = \underline{\boldsymbol{L}}_{\mathrm{tmp}}$. We also notice that there exists an inverse such that $\underline{\boldsymbol{L}}_{\mathrm{tmp}}\underline{\boldsymbol{L}}_{\mathrm{tmp}}^{-1} = \underline{\boldsymbol{I}}_n$ which is given by

$$\underline{\boldsymbol{L}}_{\mathrm{tmp}}^{-1} = \begin{bmatrix} \underline{\mathbf{1}} & 0 & 0 & \dots & 0 & 0 \\ A_{\mathrm{H}}(\underline{\boldsymbol{x}}_2^1) & \underline{\mathbf{1}} & 0 & \dots & 0 & 0 \\ A_{\mathrm{H}}(\underline{\boldsymbol{x}}_3^1) & A_{\mathrm{H}}(\underline{\boldsymbol{x}}_3^2) & \underline{\mathbf{1}} & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ A_{\mathrm{H}}(\underline{\boldsymbol{x}}_n^1) & A_{\mathrm{H}}(\underline{\boldsymbol{x}}_n^2) & A_{\mathrm{H}}(\underline{\boldsymbol{x}}_n^3) & \dots & A_{\mathrm{H}}(\underline{\boldsymbol{x}}_n^{n-1}) & \underline{\mathbf{1}} \end{bmatrix}. \tag{62}$$

In order to obtain the closed form equation for the dqRNEA, such that,

$$\tau = \underline{\boldsymbol{M}}(\theta)\ddot{\theta} + \underline{\boldsymbol{c}}(\theta,\dot{\theta}) + \underline{\boldsymbol{g}}(\theta) + \underline{\boldsymbol{J}}^{T}(\theta)\underline{\boldsymbol{F}}_{\mathrm{EF}}, \tag{63}$$

we take advantage of the expressions in Table 1. Starting with the torque, note that

$$\boldsymbol{\tau} = \underline{\boldsymbol{F}} \circledast \underline{\boldsymbol{A}}_S = \underline{\boldsymbol{A}}_S \circledast \underline{\boldsymbol{F}}. \tag{64}$$

as the double geodesic product is commutative, and because the operation defined in (17) is done element-wise throughout the vectors, then it stands that the vectors may commute as well. Furthermore, we substitute the fourth expression in Table 1 in the right-hand side of (64), resulting in

$$\boldsymbol{\tau} = \underline{\boldsymbol{A}}_S \circledast [\underline{\boldsymbol{L}}^{*}\underline{\boldsymbol{F}}_R] + \underline{\boldsymbol{A}}_S \circledast [\underline{\boldsymbol{L}}^{*}\underline{\boldsymbol{F}}_{\mathrm{EF}}]\,. \tag{65}$$

Lastly we combine (65) with the remainder of the expressions resulting in

$$\boldsymbol{\tau} = \underline{\boldsymbol{A}}_S \circledast [\underline{\boldsymbol{L}}^{*}\,(\underline{\boldsymbol{W}} \otimes \boldsymbol{G}\,(\underline{\boldsymbol{W}}))] + \underline{\boldsymbol{A}}_S \circledast \left[\underline{\boldsymbol{L}}^{*}\boldsymbol{G}\left(\underline{\dot{\boldsymbol{W}}}\right)\right] + \underline{\boldsymbol{A}}_S \circledast [\underline{\boldsymbol{L}}^{*}\underline{\boldsymbol{F}}_{\mathrm{EF}}]\,, \tag{66}$$

$$\begin{aligned}\boldsymbol{\tau} =& \underline{\boldsymbol{A}}_S \circledast \left[\underline{\boldsymbol{L}}^{*}\left(\underline{\boldsymbol{W}} \otimes \boldsymbol{G}\left(\underline{\boldsymbol{L}}\underline{\boldsymbol{A}}\dot{\boldsymbol{\theta}} + \underline{\boldsymbol{L}}\underline{\boldsymbol{W}}_{\mathrm{BASE}}\right)\right)\right] \\ &+ \underline{\boldsymbol{A}}_S \circledast \left[\underline{\boldsymbol{L}}^{*}\boldsymbol{G}\left(\underline{\boldsymbol{L}}\underline{\boldsymbol{A}}\ddot{\boldsymbol{\theta}} + \underline{\boldsymbol{L}}\underline{\boldsymbol{C}}\dot{\boldsymbol{\theta}} + \underline{\boldsymbol{L}}\underline{\dot{\boldsymbol{W}}}_{\mathrm{BASE}}\right)\right] + \underline{\boldsymbol{A}}_S \circledast [\underline{\boldsymbol{L}}^{*}\underline{\boldsymbol{F}}_{\mathrm{EF}}]\,.\end{aligned} \tag{67}$$

Here we use the proprieties linked to (32) in order to break down the terms in $\boldsymbol{G}(\bullet)$. Thus

$$\begin{aligned}\boldsymbol{\tau} =& \underline{\boldsymbol{A}}_S \circledast \left[\underline{\boldsymbol{L}}^{*}\underline{\boldsymbol{W}} \otimes \left(\boldsymbol{G}\left(\underline{\boldsymbol{L}}\underline{\boldsymbol{A}}\right)\dot{\boldsymbol{\theta}}\right)\right] + \underline{\boldsymbol{A}}_S \circledast [\underline{\boldsymbol{L}}^{*}\underline{\boldsymbol{W}} \otimes \boldsymbol{G}\left(\underline{\boldsymbol{L}}\underline{\boldsymbol{W}}_{\mathrm{BASE}}\right)] \\ &+ \underline{\boldsymbol{A}}_S \circledast [\underline{\boldsymbol{L}}^{*}\boldsymbol{G}\left(\underline{\boldsymbol{L}}\underline{\boldsymbol{A}}\right)]\,\ddot{\boldsymbol{\theta}} + \underline{\boldsymbol{A}}_S \circledast [\underline{\boldsymbol{L}}^{*}\boldsymbol{G}\left(\underline{\boldsymbol{L}}\underline{\boldsymbol{C}}\right)]\,\dot{\boldsymbol{\theta}} \\ &+ \underline{\boldsymbol{A}}_S \circledast \left[\underline{\boldsymbol{L}}^{*}\boldsymbol{G}\left(\underline{\boldsymbol{L}}\underline{\dot{\boldsymbol{W}}}_{\mathrm{BASE}}\right)\right] + [\underline{\boldsymbol{A}}_S \circledast \underline{\boldsymbol{L}}^{*}]\,\underline{\boldsymbol{F}}_{\mathrm{EF}}.\end{aligned} \tag{68}$$

The expression obtained in (68) is the closed form version of the dqRNEA, however, to obtain a more intuitive representation as in (63), we take

$$\underline{\boldsymbol{M}}(\theta) = \underline{\boldsymbol{A}}_S \circledast [\underline{\boldsymbol{L}}^{*}\boldsymbol{G}\left(\underline{\boldsymbol{L}}\underline{\boldsymbol{A}}\right)]\,; \tag{69}$$

$$\begin{aligned}\underline{\boldsymbol{c}}(\theta,\dot{\theta}) =& \underline{\boldsymbol{A}}_S \circledast \left[\underline{\boldsymbol{L}}^{*}\underline{\boldsymbol{W}} \otimes \left(\boldsymbol{G}\left(\underline{\boldsymbol{L}}\underline{\boldsymbol{A}}\right)\dot{\boldsymbol{\theta}}\right)\right] + \underline{\boldsymbol{A}}_S \circledast [\underline{\boldsymbol{L}}^{*}\underline{\boldsymbol{W}} \otimes \boldsymbol{G}\left(\underline{\boldsymbol{L}}\underline{\boldsymbol{W}}_{\mathrm{BASE}}\right)] \\ &+ \underline{\boldsymbol{A}}_S \circledast [\underline{\boldsymbol{L}}^{*}\boldsymbol{G}\left(\underline{\boldsymbol{L}}\underline{\boldsymbol{C}}\right)]\,\dot{\boldsymbol{\theta}};\end{aligned} \tag{70}$$

$$\underline{\boldsymbol{g}}(\theta) = \underline{\boldsymbol{A}}_S \circledast \left[\underline{\boldsymbol{L}}^{*}\boldsymbol{G}\left(\underline{\boldsymbol{L}}\underline{\dot{\boldsymbol{W}}}_{\mathrm{BASE}}\right)\right]; \tag{71}$$

$$\underline{\boldsymbol{J}}^{T}(\theta) = \underline{\boldsymbol{A}}_S \circledast \underline{\boldsymbol{L}}^{*}. \tag{72}$$

We add here that our formulation for the dqRNEA is one attempt to close a gap in the literature, both with the recursive formulation and the closed one. Indeed, there not many works tackling the use of dual quaternions for describing the RNEA for serial manipulators (or any manipulator for that matter). One exception however are the recent works in [24,25], which describe the quaternion-based inverse dynamics of a spacecraft-mounted robotic manipulator configured

with different joint types. We reiterate here, however, that although some of the equations in [24, 25] are similar to our own, their approach is vastly different. While they propose a framework for all types of joint, their approach has many equations presented the Euclidean vector-matrix formulation. This is solution, although valid, is more costly and many times also unintuitive and cumbersome, as there is the need for mapping the dual quaternions to real vectors and matrices.

The expressions presented in (63) is similar to the general formulation presented in existing results in the literature for closed-form inverse dynamics. Notwithstanding, the results derived herein are all expressed in unit-dual quaternions and its corresponding tangent-space. Such approach should facilitate the direct usage of inverse dynamics within applications derived within dual quaternion algebra.

5.4 DQ Based Inverse Dynamics Controller

Following the modelling results presented in this Section, this paper proposes a dual-quaternion based computed torque controller (CTC). Although conceptually simple and well known, this MIMO controller based on feedback linearization is ideal to illustrate the validity of the approach and the usage of the dqRNEA in practical applications. In the following, we present, as a contribution, the formulation for a PD computed torque controller based on (63) to (72).[6]

To design our controller we take (63) as a starting point. The intuition for this controller is to find a feedback law $\tau_{\text{input}} = f(\theta, \dot{\theta}, \ddot{\theta})$ such that the nonlinearities stemming from the inverse dynamics model are removed yielding a simpler linear system. To this aim, let us choose a control law such

$$\tau_{\text{input}} = \underline{\boldsymbol{M}}(\theta)\boldsymbol{a}_c + \underline{\boldsymbol{c}}(\theta, \dot{\theta}) + \underline{\boldsymbol{g}}(\theta) + \underline{\boldsymbol{J}}^T(\theta)\underline{\boldsymbol{F}}_{\text{EF}}, \tag{73}$$

in which $\boldsymbol{a}_c$ is an input we are still to choose. Furthermore, as $\underline{\boldsymbol{M}}(\theta)$ is invertible, then we may combine (63) and (73) to obtain

$$\ddot{\boldsymbol{\theta}} = \boldsymbol{a}_c. \tag{74}$$

The system in (74) is a double integrator and it stands out as a linear and decoupled system, enabling us to use each individual element of $\boldsymbol{a}_c$ to control each output.

We now design the input $\boldsymbol{a}_c$ with a control law of our choice, in this case a PD controller scheme to drive the input as a function of the position and velocity,

$$\boldsymbol{a}_c = -\boldsymbol{K}_{\text{P}}\boldsymbol{\theta} - \boldsymbol{K}_{\text{D}}\dot{\boldsymbol{\theta}} + \boldsymbol{r}(t) \tag{75}$$

with $\boldsymbol{K}_{\text{P}}$ and $\boldsymbol{K}_{\text{D}}$ being respectively the proportional and derivative gains and $\boldsymbol{r}(t)$ is the desired feed-forward trajectory, which we may define as

$$\boldsymbol{r}(t) = \ddot{\boldsymbol{\theta}}_{\text{desired}} + \boldsymbol{K}_{\text{P}}\boldsymbol{\theta}_{\text{desired}} + \boldsymbol{K}_{\text{D}}\dot{\boldsymbol{\theta}}_{\text{desired}}. \tag{76}$$

[6] Although in this work, we are only presenting the basic formulation for the CTC, we highlight there is a great variety of torque control based approaches that can better account for uncertainties in the models being used [17].

Substituting (76) in (75) we have

$$\boldsymbol{a}_c = \boldsymbol{K}_{\mathrm{P}} \left(\boldsymbol{\theta}_{\mathrm{desired}} - \boldsymbol{\theta}\right) + \boldsymbol{K}_{\mathrm{D}} \left(\dot{\boldsymbol{\theta}}_{\mathrm{desired}} - \dot{\boldsymbol{\theta}}\right) + \ddot{\boldsymbol{\theta}}_{\mathrm{desired}}, \tag{77}$$

in which we may define $\left(\boldsymbol{\theta}_{\mathrm{desired}} - \boldsymbol{\theta}\right) = \boldsymbol{e}$ and $\left(\dot{\boldsymbol{\theta}}_{\mathrm{desired}} - \dot{\boldsymbol{\theta}}\right) = \dot{\boldsymbol{e}}$ as the error functions, thus creating our control law,

$$\boldsymbol{a}_c = \boldsymbol{K}_{\mathrm{P}}\boldsymbol{e} + \boldsymbol{K}_{\mathrm{D}}\dot{\boldsymbol{e}} + \ddot{\boldsymbol{\theta}}_{\mathrm{desired}}. \tag{78}$$

6 Computational Complexity Cost Analysis

This section aims at scrutinizing the computational costs involved in our algorithm and to compare it with basic results in RNEA literature. The methodology to assess such costs are similar to the ones in [13, 19, 21, 23, 26]. Particularly, [19] and [23] advocates over the performance of the UDQ $(8f, 48\times, 40+)$ over the HTM representation $(12f, 64\times, 48+)$, where $f, \times, +$ stands for the storage, scalar addition and multiplications costs related to a single group operation—which in turn relates to a single rigid-body transformation [19]. Here, as in our previous work [6], we add as a contribution the analysis of individual costs for operations with dual quaternions, as well as the costs for our dqRNEA algorithm.

The computational cost is computed in terms of the number of elementary operations (multiplications/division and addition/subtraction of floating units), and of storage [21]. Thus, the computational cost for an operation or for an equation will be given by

$$\mathrm{cost}_{\mathrm{Total}}(\mathrm{Operation}) = \left([\mathrm{storage}]\, f, [\mathrm{n^o\ Mult.}]\, \times, [\mathrm{n^o\ Add.}]\, +\right). \tag{79}$$

Table 2 lists the costs related to quaternion algebra, whilst Table 3 shows a detailed list of dual quaternion operations with respective storage and costs. Note that, although we reckon the same computational complexity could be directly extracted from basic operations, to explicitly state the compound operations is valid for future reference and computational cost analysis within the dual quaternion algebra.

From Tables 2 and 3, we can compute the total cost for all operations of the proposed algorithm for a n-joints serial manipulator. Such results are shown in Table 4.

7 Quantitative Analysis

In this section, we present a simple example to validate the proposed algorithm implementation yet, most importantly, we provide a quantitative analysis of the proposed algorithm given different conditions and compare with the HTM based solution.

First, to illustrate the framework, we take a simple scenario, i.e., a two-link robot arm with similar links with length of 1.0 m, weight of 1 kg and centre of mass given in the link's centroid. Fig. 1 compares the result with different

Table 2: Cost requirements of quaternion algebra operations

Operation	Storage	$\times/\div$	$+/-$
Addition ($\boldsymbol{x}_1 + \boldsymbol{x}_2$)	$8f$	$0\times$	$4+$
Multiplication by Scalar ($\alpha\boldsymbol{x}$)	$5f$	$4\times$	$0+$
Quat. Multiplication $\boldsymbol{x}_1\boldsymbol{x}_2$	$8f$	$16\times$	$12+$
Quat. Cross Product (4)	$6f$	$9\times$	$5+$
Quat. Inner Product (3)	$6f$	$3\times$	$2+$
Matrix Linear Transf. $\mathcal{T}_4$ (20)	$20f$	$16\times$	$12+$
Quat. Hamilton Operator $\overset{+}{\boldsymbol{h}}(\bullet)$ or $\overset{-}{\boldsymbol{h}}(\bullet)$ (23)	$8f$	$16\times$	$12+$
Quat. Adjoint Transformation	$7f$	$20\times$	$28+$

* where $\alpha \in \mathbb{R}$ and $\boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathbb{H}$

Table 3: Cost requirements for dual quaternion operations

Operation	Storage	$\times/\div$	$+/-$
Addition ($\underline{\boldsymbol{x}}_1 + \underline{\boldsymbol{x}}_2$)	$16f$	$0\times$	$8+$
Multiplication by Scalar ($\alpha\underline{\boldsymbol{x}}_1$)	$9f$	$8\times$	$0+$
Dual Quat. Multiplication ($\underline{\boldsymbol{x}}_1\underline{\boldsymbol{x}}_2$)	$16f$	$48\times$	$40+$
Cross Product (6)	$12f$	$18\times$	$12+$
Inner Product	$12f$	$6\times$	$5+$
Double Geodesic Product (7)	$12f$	$6\times$	$5+$
Dual Quat Linear Transformation	$72f$	$64\times$	$56+$
Dual Quat. Hamilton Operator (24)	$28f$	$48\times$	$48+$
Adjoint Construction of $\boldsymbol{\mathcal{A}_P}$ and $\boldsymbol{\mathcal{A}_D}$	$12f$	$108\times$	$72+$
Adjoint computation from (36)	$30f$	$27\times$	$30+$

* where $\alpha \in \mathbb{R}$ and $\underline{\boldsymbol{x}}_1, \underline{\boldsymbol{x}}_2 \in \underline{\mathbb{H}}$

versions of the proposed algorithm and with the HTM based RNEA using a third party software (Peter Corke's Robotics Toolbox, [5]). All computations were performed using the DQ Robotics toolbox[7].

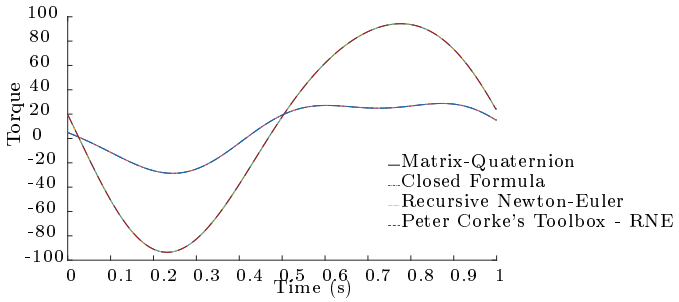


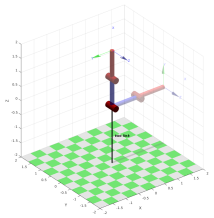
Fig. 1: Torque plot for each of the joints in different versions of the algorithm, that is, matrix-quaternion formulation (solid lines), closed equation (dash-dotted) and recursive algorithm (dotted) compared to a third-party HTM-based RNEA (dashed).

[7] http://dqrobotics.sourceforge.net

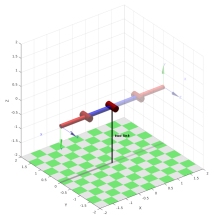
Table 4: Cost of dqRNEA algorithm operations for UDQ

Oper.	DQ - PoE			DQ - DH		
	Storage(f)	$\times/\div$	$+/-$	Storage(f)	$\times/\div$	$+/-$
Eq. (39)	$8n$	$48n$	$40n$	$24n$	$144n$	$120n$
Eqs. (34, 35)	$12n$	$108n$	$72n$	$12n$	$108n$	$72n$
Eq. (40)	$37n$	$33n$	$36n$	$37n$	$33n$	$36n$
Eq. (44)	$44n$	$57n$	$54n$	$44n$	$57n$	$54n$
Eq. (45)	$22n$	$42n$	$30n$	$22n$	$42n$	$30n$
Eq. (46)	$36n$	$27n$	$36n$	$36n$	$27n$	$36n$
Eq. (48)	$12n$	$6n$	$5n$	$12n$	$6n$	$5n$
TOTAL	$171n$	$321n$	$273n$	$187n$	$417n$	$353n$

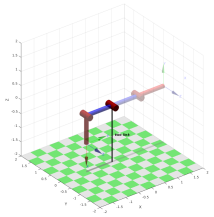
Furthermore, to further validate the results we also implemented the proposed computed toque controller to analyze the stability and convergence of the closed-loop system. Some results for the controller can be seen in Figs. 2 and 3.



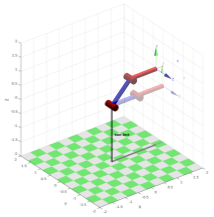
(a) $\boldsymbol{\theta}_0 = \{0, 0\}$, $\boldsymbol{\theta}_{end} = \{0, \frac{\pi}{2}\}$.



(b) $\boldsymbol{\theta}_0 = \{0, 0\}$, $\boldsymbol{\theta}_{end} = \{0, \pi\}$



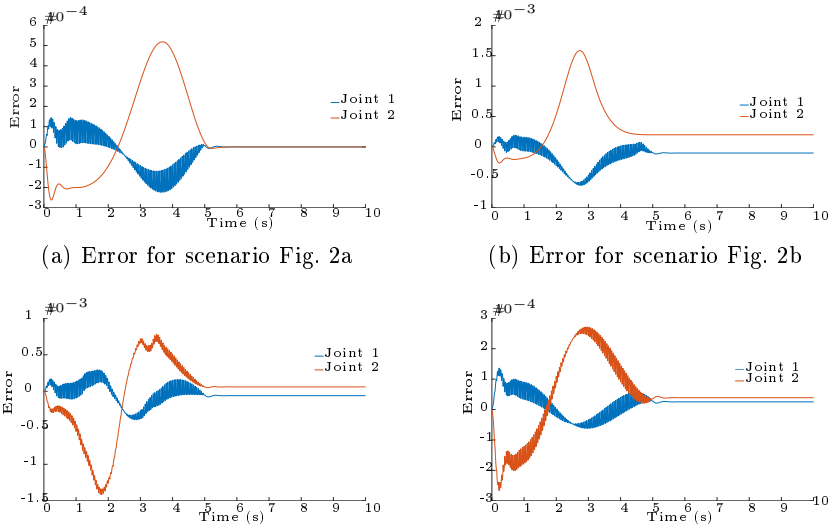
(c) $\boldsymbol{\theta}_0 = \{0, 0\}$, $\boldsymbol{\theta}_{end} = \{\pi, \frac{\pi}{2}\}$



(d) $\boldsymbol{\theta}_0 = \{0, 0\}$, $\boldsymbol{\theta}_{end} = \{\pi, \frac{\pi}{2}\}$

Fig. 2: Robot plot at different configurations. Here we merged the images for the plot at the initial joint configuration $\boldsymbol{\theta}_0$ and the final joint configuration $\boldsymbol{\theta}_{end}$.

To quantitatively assess the computational complexity performance of the proposed algorithm, we devised and computed the cost for different scenarios in terms of degrees of freedom (DoFs). Fig. 4 shows the respective costs results concerning solely the FKM transformation. As stressed in in [26], [19], the dual

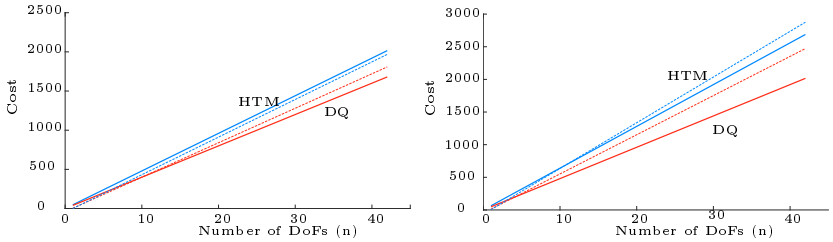


(a) Error for scenario Fig. 2a (b) Error for scenario Fig. 2b

(c) Error for scenario Fig. 2c (d) Error for scenario Fig. 2d

Fig. 3: Error convergene for the experiments in Fig. 2

quaternion based solution performs better than HTM and PoE formulation has an advantage over the DH parameters. This is particularly more relevant for larger DoFs.



(a) Costs for Addition (b) Costs for Multiplication

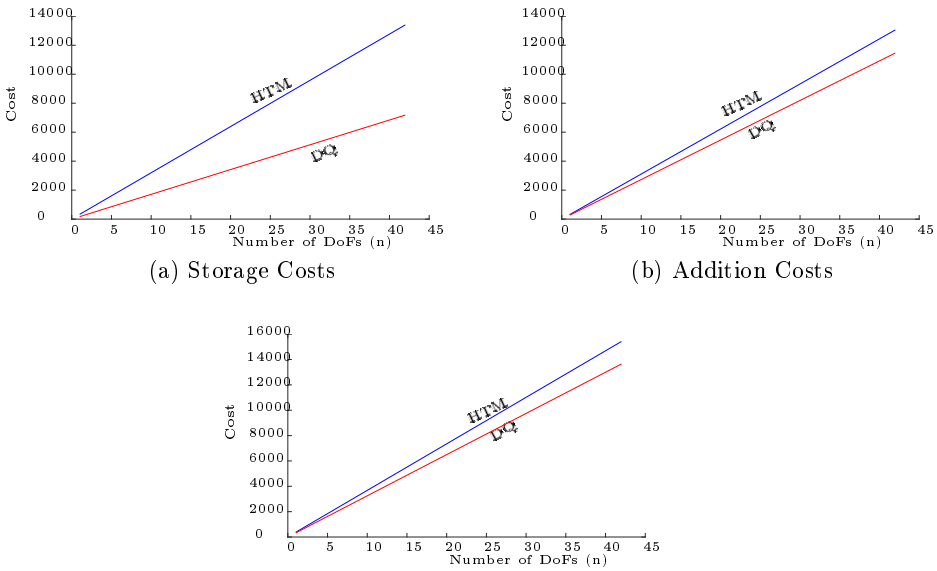
Fig. 4: Costs for FKM computation using: HTM (blue) and UDQ (red) with D-H representation (dotted) or screw theory (solid) for n ranging from 1-42.

Fig. 5 shows the cost comparison between the dqRNEA with PoE and the new proposed adjoint in comparison to the HTM recursive Newton-Euler[8]. We notice that for all three variables we have measured the dual quaternion representation performs significantly better. This is a relevant result as the computational efficiency of the proposed algorithm is one of questions we aimed to answer in this work. Yet, such result is not sufficient to claim the superiorty of the

[8] This algorithm can be found on chapter 8 of [17].

algorithm over other versions of RNEA which exist in the literature, as many works have been focused solely in optimizing the performance of RNEA as herein we have not aimed to perform any optimization in none of the HTM or DQ based results.

It is worth to mention that we compared our result with classic HTM based RNEA as both UDQ and HTM are singularity free representations and for having the translation and rotation coupled together in only one structure, which by itself is one great advantage for applications in path planning and control.



(a) Storage Costs

(b) Addition Costs

(c) Multiplication Costs

Fig. 5: Total costs for RNEA of both HTM and DQ. In the plots, the HTM is represented in blue and the DQ is represented in Red. Also n ranges from 1-42.

8 Conclusion

The main focus of this paper was to provide a novel and cost effective formalization for the RNEA with dual quaternion algebra and thus extend results found in the literature of robotic manipulators. Thus, we have also addressed some of the issues regarding the development of the dqRNEA, analysis in terms of computational complexity, development of a closed form controller and further implementation of a computed torque controller. Furthermore, to achieve improved performance, we have opted for a less traditional method of describing a serial manipulator. In order to avoid the extra costs of the D-H parameters and in order to design simpler dynamical equations, we have chosen to base our formulation on screw theory. Moreover, we have also contributed

to the literature by defining a alternative methods for describing the adjoint transformation, linear transformation and to address commutativity, all within the algebra of dual quaternions. The resulting algorithm, in addition to provide suitable results within the algebra of dual quaternions, has also shown improved performance over the classic HTM solution. For future works we want to extend our algorithm to cooperative manipulation taking advantage of the cooperative kinematic description within DQ algebra with the efficient computation of the manipulator dynamics based on screw theory with UDQ.

References

1. Agrawal, O.P.: Hamilton operators and dual-number-quaternions in spatial kinematics. Mechanism and Machine Theory **22**(6), 569–575 (1987)
2. Akyar, B.: Dual quaternions in spatial kinematics in an algebraic sense. Turk. J. Math **32**, 373–391 (2008)
3. Brodsky, V., Shoham, M.: The Dual Inertia Operator and Its Application to Robot Dynamics. Journal of Mechanical Design **116**(4), 1089 (1994). https://doi.org/10.1115/1.2919491
4. Bullo, F., Murray, R.: Proportional derivative (PD) control on the euclidean group. Technical Report Caltech/CDS 95–010, Caltech (1995)
5. Corke, P.: Robotics, Vision and Control: Fundamental Algorithms In MATLAB. Springer International Publishing (2017)
6. de Farias, C.M., Figueredo, L.F.C., Ishihara, J.Y.: Performance study on dqrnea - a novel dual quaternion based recursive newton-euler inverse dynamics algorithms. In: 3rd IEEE International Conference on Robotic Computing (IRC). pp. 94–101 (2019). https://doi.org/10.1109/IRC.2019.00022
7. Dooley, J.R., McCarthy, J.M.: Spatial rigid body dynamics using dual quaternion components. In: Proceedings of 1991 IEEE International Conference on Robotics and Automation. pp. 90–95 (1991)
8. de Farias, C.M., Rocha, Y.G., Figueredo, L.F.C., Bernardes, M.C.: Design of singularity-robust and task-priority primitive controllers for cooperative manipulation using dual quaternion representation. In: IEEE Conference on Control Technology and Applications (CCTA). pp. 740–745 (2017). https://doi.org/10.1109/CCTA.2017.8062550
9. Figueredo, L.F.C., Adorno, B.V., Ishihara, J.Y., Borges, G.A.: Robust kinematic control of manipulator robots using dual quaternion representation. In: 2013 IEEE International Conference on Robotics and Automation. pp. 1949–1955 (2013). https://doi.org/10.1109/ICRA.2013.6630836
10. Figueredo, L.F.C.: Kinematic control based on dual quaternion algebra and its application to robot manipulators. Ph.D. thesis, University of Brasilia (2016)
11. Filipe, N., Tsiotras, P.: Adaptive Model-Independent Tracking of Rigid Body Position and Attitude Motion with Mass and Inertia Matrix Identification using Dual Quaternions. AIAA Guidance, Navigation, and Control (GNC) Conference (January 2015), 1–15 (2013). https://doi.org/10.2514/6.2013-5173
12. Filipe, N., Tsiotras, P.: Simultaneous position and attitude control without linear and angular velocity feedback using dual quaternions. Proceedings of 2013 American Control Conference (September 2014), 4808–4813 (2013). https://doi.org/10.1109/ACC.2013.6580582
13. Funda, J., Paul, R.: A computational analysis of screw transformations in robotics. IEEE Transactions on Robotics and Automation **6**(3), 348–356 (1990)

14. Han, D., Wei, Q., Li, Z.: A dual-quaternion method for control of spatial rigid body. In: 2008 IEEE International Conference on Networking, Sensing and Control. pp. 1–6 (April 2008). https://doi.org/10.1109/ICNSC.2008.4525172
15. Hollerbach, J.M.: A recursive lagrangian formulation of maniputator dynamics and a comparative study of dynamics formulation complexity. IEEE Transactions on Systems, Man, and Cybernetics **10**, 730–736 (1980)
16. Kane, T.R., Levinson, D.A.: The use of kane's dynamical equations in robotics. The International Journal of Robotics Research **2**(3), 3–21 (1983). https://doi.org/10.1177/027836498300200301
17. Lynch, K., Park, F.: Modern robotics. chap. 8. Cambridge University Press (2017)
18. Nasser, M.: Recursive newton-euler formulation of manipulator dynamics. In: NASA Conference on Space Telerobotics. p. 309 (1989)
19. Özgür, E., Mezouar, Y.: Kinematic modeling and control of a robot arm using unit dual quaternions. Robotics and Autonomous Systems **77**, 66–73 (2016). https://doi.org/10.1016/j.robot.2015.12.005
20. Park, F.C.: Distance Metrics on the Rigid-Body Motions with Applications to Mechanism Design. ASME. Journal of Mechanical Design **117**(1), 48–54 (1995). https://doi.org/10.1115/1.2826116
21. Sariyildiz, E., Cakiray, E., Temeltas, H.: A comparative study of three inverse kinematic methods of serial industrial robot manipulators in the screw theory framework. International Journal of Advanced Robotic Systems **8**(5), 9–24 (2011). https://doi.org/10.5772/45696
22. Selig, J.M.: Geometric Fundamentals of Robotics. Springer-Verlag New York Inc., 2nd edn. (2005)
23. Sharkawy, A.N., Aspragathos, N.: A comparative study of two methods for forward kinematics and Jacobian matrix determination. In: 2017 International Conference on Mechanical, System and Control Engineering, ICMSC 2017 (2017). https://doi.org/10.1109/ICMSC.2017.7959467
24. Valverde, A., Tsiotras, P.: Dual quaternion framework for modeling of spacecraft-mounted multibody robotic systems. Frontiers in Robotics and AI **5** (Nov, 2018). https://doi.org/10.3389/frobt.2018.00128
25. Valverde, A., Tsiotras, P.: Modeling of spacecraft-mounted robot dynamics and control using dual quaternions. pp. 670–675 (June 2018). https://doi.org/10.23919/ACC.2018.8431054
26. Vilhena Adorno, B.: Robot Kinematic Modeling and Control Based on Dual Quaternion Algebra — Part I: Fundamentals. (Feb 2017)
27. Wang, X., Yu, C.: Feedback linearization regulator with coupled attitude and translation dynamics based on unit dual quaternion. In: IEEE International Symposium on Intelligent Control - Proceedings. pp. 2380–2384 (2010). https://doi.org/10.1109/ISIC.2010.5612894
28. Yang, X.L., Wu, H.T., Li, Y., Kang, S.Z., Chen, B.: Computationally Efficient Inverse Dynamics of a Class of Six-DOF Parallel Robots: Dual Quaternion Approach. Journal of Intelligent and Robotic Systems: Theory and Applications pp. 1–13 (2018). https://doi.org/10.1007/s10846-018-0800-1