The International Journal of Robotics Research
XX(X):1–21

DOI:10.1177/0278364920931948
journals.sagepub.com/home/ijr
SAGE

Robust and Efficient Forward, Differential, and Inverse Kinematics using Dual Quaternions

Neil T. Dantam

Abstract

Modern approaches for robot kinematics employ the product of exponentials formulation, represented using homogeneous transformation matrices. Quaternions over dual numbers are an established alternative representation; however, their use presents certain challenges: the dual quaternion exponential and logarithm contain a zero-angle singularity, and many common operations are less efficient using dual quaternions than with matrices. We present a new derivation of the dual quaternion exponential and logarithm that removes the singularity, we show an implicit representation of dual quaternions offers analytical and empirical efficiency advantages compared to both matrices and explicit dual quaternions, and we derive efficient dual quaternion forms of differential and inverse position kinematics. Analytically, implicit dual quaternions are more compact and require fewer arithmetic instructions for common operations, including chaining and exponentials. Empirically, we demonstrate a 30-40% speedup on forward kinematics and a 300-500% speedup on inverse position kinematics. This work relates dual quaternions with modern exponential coordinates and demonstrates that dual quaternions are a robust and efficient representation for robot kinematics.

Keywords

Kinematics, Manipulation, Control Architectures and Programming

1 Introduction

Efficient geometric computations are important for robot manipulation, 3D simulation, and other areas that must represent the physical world. The product of exponentials formulation, represented using homogeneous transformation matrices, has emerged as the conventional method for robot kinematics (Brockett 1984; Lynch and Park 2017; Murray 1994). For pure rotations, the unit quaternion has recently resurged in popularity, particularly for applications in graphics and estimation where the efficient interpolation and normalization of quaternions is especially useful. It is also possible to represent both rotation and translation by extending the ordinary unit quaternion to quaternions over dual numbers (Selig 2004; Study 1903). Such dual quaternions retain the unit quaternions' advantages of compactness and efficient normalization; however, they also present challenges. Common kinematics operations—constructing and chaining transforms—require more arithmetic instructions using dual quaternions than the equivalent transformation matrix computation. Critically, the dual quaternion exponential contains a small-angle singularity which we must handle for numerical robustness. We address these challenges and present a quaternion-based approach with advantages for robot kinematics.

We present a new derivation of the dual number quaternion exponential and logarithm that removes the small-angle singularity, we show that the implicit representation of a dual quaternion is more computationally-efficient for robot kinematics than homogeneous transformation matrices, and we apply dual quaternion analysis to improve efficiency of inverse position kinematics. The conventional representation of exponential coordinates using the homogeneous transformation matrix provides a baseline for comparison (see Sec. 3). We begin with the known forms of the ordinary quaternion exponential and logarithm (see Sec. 4.1). Using dual number arithmetic and quaternion trigonometry, we derive the

Corresponding author:
Neil T. Dantam, Department of Computer Science, Colorado School of Mines, Golden, CO, USA
Email: ndantam@mines.edu

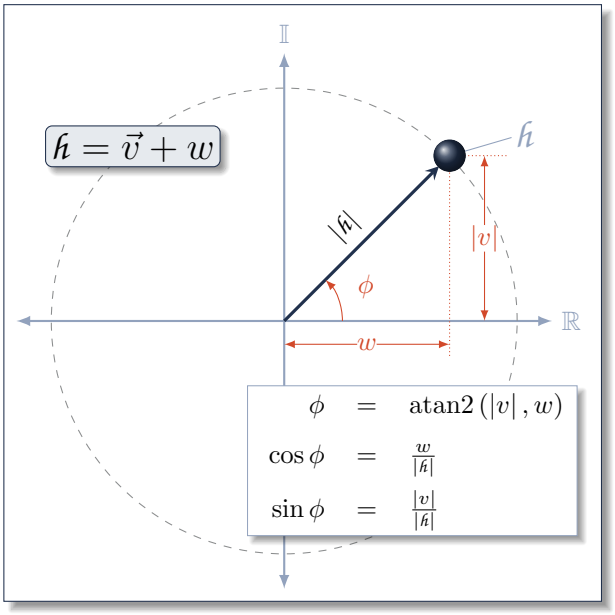


Figure 1. The quaternion-imaginary-plane, containing axes for the scalar w and vector (imaginary) magnitude $|v|$. The angle ϕ and trigonometric ratios between the scalar and vector parts guide our derivations.

exponential and logarithm for the dual quaternions and rewrite factors to identify Taylor series that remove the singularities (see Sec. 4.2). We extend this dual quaternion exponential and logarithm to the implicit representation of a dual quaternion as an ordinary (rotation) quaternion and a translation vector, which is more compact and computationally efficient than explicit dual quaternions (see Sec. 4.3). Then, we apply these quaternion forms to robot kinematics (see Sec. 5). We demonstrate a 30%-40% empirical performance gain over transformation matrices on forward kinematics, and we achieve a 300-500% speedup on inverse position kinematics formulated as sequential quadratic programming (SQP). Finally, we discuss issues of equivalence and efficiency between matrix and quaternion representations (see Sec. 6). The work presented in this paper is available as open source software.*

Quaternion-based forms present both challenges and advantages. A common challenge raised with quaternions is the difficulty of mentally visualizing the four-dimensional space of ordinary quaternions—or the eight-dimensional space of dual quaternions—whereas vector and matrix representations have a direct, 3-dimensional interpretation. Still, the planar projection of quaternions (see Fig. 1) offers insight into the relationship between quaternion components and angles. More importantly, a growing body of work continues to demonstrate that ordinary and dual

$\sin\theta$	$=$	$\theta - \frac{1}{6}\theta^3 + \frac{1}{120}\theta^5 + \dots$
$\cos\theta$	$=$	$1 - \frac{1}{2}\theta^2 + \frac{1}{24}\theta^4 + \dots$
$\frac{\sin\theta}{\theta}$	$=$	$1 - \frac{1}{6}\theta^2 + \frac{1}{120}\theta^4 + \dots$
$\frac{\theta}{\sin\theta}$	$=$	$1 + \frac{1}{6}\theta^2 + \frac{7}{360}\theta^4 + \dots$
$\frac{1-\cos\theta}{\theta^2}$	$=$	$\frac{1}{2} - \frac{1}{24}\theta^2 + \frac{1}{720}\theta^4 + \dots$

Table 1. Taylor Series for $\theta \to 0$

quaternions offer computational advantages in a variety of domains (Kavan et al. 2008; Markley and Mortari 2000; Shoemake 1985). The results of this paper are in the same vein. We demonstrate a dual-quaternion-based approach for kinematics that offers computational advantages over matrices. We mitigate the challenge of visualizing quaternions by using the relations of Fig. 1 in algebraic derivations.

A key technique in our derivations is to rewrite factors with singularities into forms with well-defined Taylor series which we evaluate near the singular point, i.e., in the limit. Grassia (1998) applies this idea to ordinary quaternions. For example, the ordinary quaternion exponential contains the factor $\frac{\sin\theta}{\theta}$, which has a singularity at $\theta = 0$. However, we use a Taylor series to remove the singularity:

$$\frac{\sin\theta}{\theta} = 1 - \frac{1}{6}\theta^2 + \frac{1}{120}\theta^4 - \frac{1}{5040}\theta^6 + \dots$$
$$\lim_{\theta\to 0} \frac{\sin\theta}{\theta} = 1\,. \tag{1}$$

Near the singularity, we need only the initial terms of the Taylor series to evaluate the factor to within machine precision because the final terms will be smaller than the precision of a floating point number. For example, using IEEE 754 (ISO/IEC JTC 1, Information Technology 2011) double precision numbers, $1 + 10^{-16} = 1$, so starting from a term of 1, we may discard terms smaller than 10^{-16} without affecting the final result. In (1), we have alternating positive and negative terms of decreasing magnitude, so the error after evaluating the first i terms is no greater than the magnitude of term $i + 1$. We need not evaluate any additional terms when this error is less than machine precision. Specifically, when θ^4 is less than machine precision, we may achieve minimum possible numerical error using only the first two terms $1 - \frac{1}{6}\theta^2$.

We extend this Taylor series construction to the dual quaternions, which have similar—though more complicated—factors containing singularities. We use quaternion trigonometry (see Fig. 1) to rewrite these

*Software available at http://amino.dyalab.org

Symbol	Description
$\mathring{h}$	Ordinary Quaternion
$\hat{\boldsymbol{i}},\hat{\boldsymbol{j}},\hat{\boldsymbol{k}}$	Quaternion basis elements
$\mathbf{J}$	Manipulator Jacobian
$\mathbf{R}$	Rotation Matrix
$\mathcal{S}$	Dual Quaternion
$\mathbf{T}$	Transformation Matrix
$\mathbf{v}, \vec{v}$	Translation Vector
s,c	Sine and Cosine
$\hat{\boldsymbol{u}}$	Joint / rotation axis
$\boldsymbol{\varepsilon}$	Dual number element
θ	Rotation angle / configuration
$\boldsymbol{\theta}$	Configuration Vector
ϕ	Quaternion scalar-vector angle
ω	Rotation vector/velocity
Ω	Twist

Table 2. Summary of Symbols

factors into forms that are defined in the limit via Taylor series. Table 1 lists several common Taylor series.

Note that the singularities in the logarithm and exponential are different from kinematic singularities that arise at certain manipulator configurations—e.g., with the arm fully outstretched—where the manipulator loses the ability to instantaneously move in one or more directions. Robust pseudoinverse methods address kinematic singularities (Buss 2004), whereas we identify factors and corresponding Taylor series to remove the logarithm and exponential singularities.

We use the following notation. Bold uppercase $\mathbf{R}$ denotes a matrix. Bold lowercase $\mathbf{v}$ denotes a vector. An over-arrow $\vec{v}$ denotes a length-three vector over the basis units $\hat{\boldsymbol{i}}$, $\hat{\boldsymbol{j}}$, $\hat{\boldsymbol{k}}$. An over-hat $\hat{\boldsymbol{u}}$ denotes a unit vector ($|u| = 1$). An over-tilde $\tilde{n}$ denotes a dual number ($\tilde{n} = n_{\text{real}} + n_{\text{dual}}\boldsymbol{\varepsilon}$). The lowercase script $\mathring{h}$ denotes an ordinary quaternion. The uppercase script $\mathcal{S}$ denotes a dual quaternion ($\mathcal{S} = s_{\text{real}} + s_{\text{dual}}\boldsymbol{\varepsilon}$). We abbreviate sin and cos with s and c. Table 2 summarizes the symbols that we use.

An initial version of this work appeared in Dantam (2018). This current paper extends the work to differential and inverse kinematics and includes additional mathematical details, performance tests, and discussion.

2 Related Work

Brockett (1984) connected robot kinematics with Lie groups expressed as matrix exponentials. This product of exponentials formulation has become the conventional approach for robot kinematics (Lynch and Park 2017; Murray 1994). Our work presents a practical connection between such exponential coordinates and quaternion-based representations, and we show that a quaternion-based approach offers efficiency advantages compared to matrices.

Quaternions provide an alternative to matrix-based geometric representations. Unit quaternions represent rotation (Hamilton 1866) with four elements: a 3-element vector and a scalar that together encode the rotational axis, and the sine and cosine of the rotational angle. Though vector analysis became the preferred notation in many areas (Altmann 1989; Gibbs 1884), quaternions have seen renewed use in recent years as a practical representation for rotation, interpolation, and estimation (Grassia 1998; LaViola 2003; Markley and Mortari 2000; Shoemake 1985). The computational advantages of quaternions in such applications suggest that a quaternion-based approach merits investigation in other areas typically addressed using vector or matrix representations.

Quaternions over dual numbers—the *dual quaternion*—can represent both rotation and translation (Study 1903, 1913). Selig (2004) presents a modern context for dual quaternions and more broadly Clifford algebras in relation to Lie algebras. Yang and Freudenstein (1964) applied dual quaternions to the analysis of spatial mechanisms (closed chains). Several recent authors have applied dual quaternions to robot kinematics (Chevallier 1991; Dantam et al. 2014a,b; Han et al. 2008; Kenwright 2012; Özgür and Mezouar 2016; Srivatsan et al. 2016; Valverde and Tsiotras 2018; Wang et al. 2012). Funda and Paul (1990) compare several representations for screw displacements, concluding that dual quaternions are the most efficient. Wang and Zhu (2014) compare dual quaternion and homogeneous matrix approaches, showing that dual quaternions are often more efficient. We continue this application of dual quaternions to robot kinematics by addressing issues of singularities and numerical robustness in the dual quaternion exponential and logarithm.

Though the form of the dual quaternion exponential is well established (Funda and Paul 1990; Selig 2010), there is, to our knowledge, no prior work that addresses the zero-angle singularity in the dual quaternion exponential and logarithm, which is necessary to practically use dual quaternions in the product of exponentials formulation. Han et al. (2008) observe, though do not address, the zero-angle discontinuity. Wang et al. (2012) provide an approximation of the logarithm. In this work, we present new, exact derivations of the dual quaternion exponential and logarithm that remove the zero-angle singularity, enabling the practical use of dual quaternions as exponential coordinates. Furthermore, we show that

implicitly representing a dual quaternion as an ordinary quaternion and a translation vector is both more compact and more computationally efficient for common kinematics operations than either explicit dual quaternions or homogeneous transformation matrices.

Kinematic singularities are a related issue where a manipulator loses the ability to instantaneously move in one or more dimensions due to the manipulator Jacobian at such configurations lacking linearly independent columns (Lynch and Park 2017; Murray 1994). Approaches to robustly compute manipulator velocities at kinematic singularities typically include damping terms in the pseudoinverse or apply the singular value decomposition (SVD) (Nakamura and Hanafusa 1986; Wampler 1986; Buss and Kim 2005). In this work we address a different type of singularity. Specifically, the exponential and logarithm of a dual quaternion become undefined at the zero angle. We remove these singularities by using quaternion trigonometry to identify factors with well-defined Taylor series. Additionally, the quaternion-based differential kinematics we present in Sec. 5.2 is complementary to approaches for kinematic singularities; robust pseudoinverses directly apply to quaternion forms of the manipulator Jacobian (63).

Many rececent works have applied constrained optimization to inverse kinematics (Beeson and Ames 2015; Fallon et al. 2015; Feng et al. 2015; Kingston et al. 2015; Rakita et al. 2018). Our dual quaternion analysis supports such optimization approaches by enabling the derivation of efficient, analytic gradients for optimization problems. In particular, we derive analytic gradients for SQP formulations of inverse position kinematics presented in TRAC-IK (Beeson and Ames 2015), resulting in a 300-500% speedup of the optimization.

3 Matrix Exponential and Logarithm

We briefly restate the rotation and transformation matrix exponential and logarithm to compare against the quaternion forms and demonstrate the Taylor series construction.

3.1 Rotation Matrix

We define the rotation matrix exponential and logarithm using the *rotation vector*, i.e., the rotation axis scaled by the rotation angle, because separating the axis and angle results in an undefined axis when the angle is zero and poor numerical stability when attempting to construct the unit axis for small angles (Grassia 1998).

The rotation matrix exponential (Lynch and Park 2017) is:

$$\mathrm{e}^{[\omega]} = \mathbf{I} + \frac{\sin|\omega|}{|\omega|}[\omega] + \frac{1-\cos|\omega|}{|\omega|^2}[\omega]^2,$$
$$\text{where} \quad [\omega] = \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix}. \tag{2}$$

We remove the singularity at $|\omega| = 0$ via the Taylor series in Table 1 for $\frac{\sin|\omega|}{|\omega|}$ and $\frac{1-\cos|\omega|}{|\omega|^2}$.

The rotation matrix logarithm (Lynch and Park 2017) is:

$$\vec{\omega} = \frac{\theta}{2\sin\theta}\begin{bmatrix} r_{32}-r_{23} \\ r_{13}-r_{31} \\ r_{21}-r_{12} \end{bmatrix},$$
$$\text{where} \quad \theta = \cos^{-1}\left(\frac{r_{11}+r_{22}+r_{33}-1}{2}\right). \tag{3}$$

We remove the singularity at $\theta = 0$ via the Taylor series in Table 1 for $\frac{\theta}{\sin\theta}$.

Numerical stability is another distinguishing feature between matrices and quaternions. Repeated floating point operations result in an accumulation of floating point errors in the rotation matrix. General matrices may be ortho-normalized using the Gram-Schmidt procedure. For rotation matrices, a more efficient approximation is the sequence of cross products:

$$\text{orthonormalize}\left(\begin{bmatrix}\mathbf{r_x} & \mathbf{r_y} & \mathbf{r_z}\end{bmatrix}\right) = \begin{bmatrix}\mathbf{r'_x} & \mathbf{r'_y} & \mathbf{r'_z}\end{bmatrix},$$

where

$$\mathbf{r'_z} = \frac{\mathbf{r_x}\times\mathbf{r_y}}{|\mathbf{r_x}\times\mathbf{r_y}|},$$
$$\mathbf{r'_y} = \frac{\mathbf{r'_z}\times\mathbf{r_x}}{|\mathbf{r'_z}\times\mathbf{r_x}|},$$
$$\mathbf{r'_x} = \frac{\mathbf{r'_y}\times\mathbf{r'_z}}{|\mathbf{r'_y}\times\mathbf{r'_z}|}. \tag{4}$$

3.2 Transformation Matrix:

The transformation matrix exponential (Lynch and Park 2017) is:

$$\mathrm{e}^{\begin{pmatrix}\vec{\omega}\\ \vec{\nu}\end{pmatrix}} = \begin{bmatrix} \mathrm{e}^{[\omega]} & \left(\mathbf{I} + \frac{1-\cos|\omega|}{|\omega|}[\omega] + \frac{1-\frac{\sin|\omega|}{|\omega|}}{|\omega|^2}[\omega]^2\right)\vec{\nu} \\ \mathbf{0} & 1 \end{bmatrix}. \tag{5}$$

We remove the singularity at $|\omega| = 0$ via the Taylor series in Table 1 for $\frac{1-\cos|\omega|}{|\omega|^2}$ and the following:

$$\frac{1-\frac{\sin|\omega|}{|\omega|}}{|\omega|^2} = \frac{1}{6} - \frac{|\omega|^2}{120} + \frac{|\omega|^4}{5040} + \ldots \tag{6}$$

The transformation matrix logarithm (Lynch and Park 2017) is:

$$\ln \begin{bmatrix} \mathbf{R} & \mathbf{v} \\ \mathbf{0} & 1 \end{bmatrix} = \begin{pmatrix} \ln \mathbf{R} \\ \left(\mathbf{I} - \frac{[\omega]}{2} + \frac{2s - |\omega|(1+c)}{2s|\omega|^2} [\omega]^2 \right) \mathbf{v} \end{pmatrix} . \quad (7)$$

We remove the singularity at $|\omega| = 0$ via the following Taylor series:

$$\frac{2\sin|\omega| - |\omega|(1+\cos|\omega|)}{2(\sin|\omega|)|\omega|^2} = \frac{1}{12} + \frac{|\omega|^2}{720} + \frac{|\omega|^4}{30240} + \dots \quad (8)$$

4 Quaternion Exponential and Logarithm

Now, we present the key contribution of this work: new, singularity free forms of the dual quaternion exponential and logarithm and their corresponding forms for the implicit, quaternion-translation representation. Our derivation starts with the established ordinary quaterion exponential and logarithm (see Sec. 4.1). Then, we derive the dual quaternion forms (see Sec. 4.2) using quaternion trigonometry (see Fig. 1) to construct Taylor series that remove the singularities. Finally, we derive the equivalent exponential and logarithm for the more compact and efficient quaternion-translation representation (see Sec. 4.3).

4.1 Ordinary Quaternions

Ordinary quaternions extend complex numbers ($\hat{\boldsymbol{\imath}}^2 = -1$) to three units:

$$\hat{\boldsymbol{\imath}}^2 = \hat{\boldsymbol{\jmath}}^2 = \hat{\boldsymbol{k}}^2 = \hat{\boldsymbol{\imath}}\hat{\boldsymbol{\jmath}}\hat{\boldsymbol{k}} = -1 . \quad (9)$$

A quaternion, therefore, has four elements: the real term (scalar) and the coefficients of each quaternion unit $\hat{\boldsymbol{\imath}}$, $\hat{\boldsymbol{\jmath}}$, and $\hat{\boldsymbol{k}}$ (vector). We use the following notation for the quaternion elements:

$$h = \underbrace{x\hat{\boldsymbol{\imath}} + y\hat{\boldsymbol{\jmath}} + z\hat{\boldsymbol{k}}}_{\text{vector } \vec{v}} + \underbrace{w}_{\text{scalar}} = \vec{v} + w . \quad (10)$$

Quaternion Operations Multiplication of quaternions derives from (9). The dot ($\cdot$) and cross ($\times$) products, though actually introduced as an alternative to quaternions (Gibbs 1884), allow a compact representation of quaternion multiplication ($\otimes$):

$$q \otimes p = \underbrace{\vec{q}_v \times \vec{p}_v + q_w\vec{p}_v + p_w\vec{q}_v}_{\text{vector } \vec{v}} + \underbrace{q_w p_w - \vec{q}_v \cdot \vec{p}_v}_{\text{scalar}} . \quad (11)$$

Quaternion multiplication is associative and distributative, but not commutative. Table 3 summarizes some algebraic properties of quaternions. We may also restate quaternion multiplication as a matrix-vector product, which we use for differential and inverse kinematics (see Sec. 5.2 and Sec. 5.3). In the matrix form of quaternion multiplication, the vector is constructed from one quaternion and the matrix is constructed from the other:

Associative	$p \otimes (q \otimes r) = (p \otimes q) \otimes r$
Distributive	$p \otimes (q + r) = p \otimes q + p \otimes r$
NOT Commutative	$p \otimes q \neq q \otimes p$
Conjugate Mul.	$(p \otimes q)^* = q^* \otimes p^*$
Conjugate Add.	$(p + q)^* = q^* + p^*$

Table 3. Algebraic Properties of Quaternions

$$q \otimes p = \overbrace{\begin{bmatrix} q_w & -q_z & q_y & q_x \\ q_z & q_w & -q_x & q_y \\ -q_y & q_x & q_w & q_z \\ -q_x & -q_y & -q_z & q_w \end{bmatrix}}^{[q]_L} \begin{bmatrix} p_x \\ p_y \\ p_z \\ p_w \end{bmatrix} = \underbrace{\begin{bmatrix} p_w & p_z & -p_y & p_x \\ -p_z & p_w & p_x & p_y \\ p_y & -p_x & p_w & p_z \\ -p_x & -p_y & -p_z & p_w \end{bmatrix}}_{[p]_R} \begin{bmatrix} q_x \\ q_y \\ q_z \\ q_w \end{bmatrix} . \quad (12)$$

The quaternion conjugate negates the vector part; the product of a quaternion and its conjugate is a scalar.

$$h^* = -\vec{h}_v + h_w \quad \text{and} \quad h \otimes h^* = x^2 + y^2 + z^2 + w^2 . \quad (13)$$

The norm of a quaternion is the square root of the quaternion and its conjugate:

$$|h| = \sqrt{(h \otimes h^*)} = \sqrt{x^2 + y^2 + z^2 + w^2} . \quad (14)$$

Unit quaternions, $|h| = 1$, define three-dimensional rotations. Rotating a point by a unit quaternion is defined by:

$$\begin{aligned} {}^a p &= {}^a h_b \otimes {}^b p \otimes {}^a h_b^* \\ &= 2({}^a h_b)_v \times (({}^a h_b)_v \times {}^b p + ({}^a h_b)_w {}^b p) + {}^b p . \end{aligned} \quad (15)$$

The conjugate of a unit quaternion is the inverse. For unit quaternions representing rotation, the conjugate is the inverse rotation.

$$h \otimes h^* = 1, \quad \text{when } |h| = 1 . \quad (16)$$

Normalizing a quaternion is more efficient than the corresponding matrix operation in (4). To normalize a quaternion, we divide by its norm:

$$\text{normalize}(h) = \frac{h}{|h|} = \frac{x}{|h|}\hat{\boldsymbol{\imath}} + \frac{y}{|h|}\hat{\boldsymbol{\jmath}} + \frac{z}{|h|}\hat{\boldsymbol{k}} + \frac{w}{|h|} . \quad (17)$$

Quaternion Exponential and Logarithm The quaternion exponential (Grassia 1998) is:

$$e^{\vec{v}+w} = e^{w}\left(\left(\frac{\sin|v|}{|v|}\right)\vec{v} + \cos|v|\right) . \tag{18}$$

When $|v|$ approaches zero, we use the Taylor series for $\frac{\sin|v|}{|v|}$ in Table 1.

For unit quaternions representing rotation θ about unit axis $\hat{\boldsymbol{u}}$, the exponential simplifies to,

$$\exp(\theta, \hat{\boldsymbol{u}}) = \left(\sin\frac{\theta}{2}\right)\hat{\boldsymbol{u}} + \cos\frac{\theta}{2} . \tag{19}$$

To compute the logarithm, we first find the angle between the vector $\vec{v}$ and scalar w parts of the quaternion. Then the logarithm is as follows:

$$\phi = \operatorname{atan2}(|v|, w) \quad \text{and} \quad \ln \mathring{h} = \frac{\phi}{|v|}\vec{v} + \ln|\mathring{h}| . \tag{20}$$

When $|v|$ approaches zero, we handle the singularity in $\frac{\phi}{|v|}$ by rewriting as follows:

$$\frac{\phi}{|v|} = \frac{\frac{\phi}{\sin\phi}}{|\mathring{h}|} = \frac{1 + \frac{\phi^2}{6} + \frac{7\phi^4}{360} + \dots}{|\mathring{h}|} . \tag{21}$$

4.2 Dual Quaternions

Dual quaternions are a compact representation that offers useful analytic properties. A dual quaternion combines ordinary quaternions and dual numbers. We briefly review dual numbers and the use of dual quaternions for kinematics before introducing our new derivations of the exponential and logarithm to handle the small-angle singularity. For a more thorough overview of dual quaternions for kinematics, please see Selig (2004).

Dual Numbers A dual number $\tilde{n}$ contains the dual element ε:

$$\tilde{n} = n_r + n_d\varepsilon , \text{ where } \quad \varepsilon^2 = 0 \quad \text{and} \quad \varepsilon \neq 0 . \tag{22}$$

Multiplication of two dual numbers cancels the term containing $\varepsilon^2 = 0$:

$$\begin{aligned}\tilde{n}\tilde{m} &= (n_r + n_d\varepsilon)(m_r + m_d\varepsilon) \\ &= n_r m_r + n_r m_d\varepsilon + n_d m_r\varepsilon + n_d m_d\cancelto{0}{\varepsilon^2} \\ &= n_r m_r + (n_r m_d + n_d m_r)\varepsilon .\end{aligned} \tag{23}$$

The Taylor series for functions of dual numbers yields a useful property: all higher-order terms containing ε^2 cancel to zero.

$f(r + d\varepsilon)$	$=$	$f(r) + \varepsilon d(f'(r))$
$\cos(r + d\varepsilon)$	$=$	$\cos r - \varepsilon d \sin r$
$\sin(r + d\varepsilon)$	$=$	$\sin r + \varepsilon d \cos r$
$\tan^{-1}(r + d\varepsilon)$	$=$	$\tan^{-1} r + \frac{\varepsilon d}{r^2+1}$
$\exp(r + d\varepsilon)$	$=$	$e^r + \varepsilon e^r d$
$\ln(r + d\varepsilon)$	$=$	$\ln r + \frac{d}{r}\varepsilon$
$\sqrt{r + d\varepsilon}$	$=$	$\sqrt{r} + \varepsilon\frac{d}{2\sqrt{r}}$

Table 4. Dual numbers functions

$$\begin{aligned}f(r + d\varepsilon) &= f(r) + \frac{f'(r)}{1!}(d\varepsilon) + \frac{f''(r)}{2!}\cancelto{0}{(d\varepsilon)^2} + \cancelto{0}{\dots} \\ &= f(r) + \varepsilon d f'(r) .\end{aligned} \tag{24}$$

The dual number Taylor series (24) enables evaluation of dual number functions using only the value and derivative of the real function. We summarize several relevant dual functions in Table 4.

Dual Quaternion Operations A dual quaternion contains eight coefficients covering all combinations of the quaternion elements, dual element, and scalars. We write a dual quaternion as:

$$\begin{aligned}\mathcal{S} = \mathring{h} + \mathring{d}\varepsilon = &\overbrace{\left(\mathring{h}_x\hat{\boldsymbol{i}} + \mathring{h}_y\hat{\boldsymbol{j}} + \mathring{h}_z\hat{\boldsymbol{k}} + \mathring{h}_w\right)}^{\text{real part } \mathring{h}} \\ &+ \underbrace{\left(\mathring{d}_x\hat{\boldsymbol{i}} + \mathring{d}_y\hat{\boldsymbol{j}} + \mathring{d}_z\hat{\boldsymbol{k}} + \mathring{d}_w\right)}_{\text{dual part } \mathring{d}}\varepsilon .\end{aligned} \tag{25}$$

The Euclidean transformation consisting of unit quaternion rotation $\mathring{h}$ and translation vector $\vec{v}$ corresponds to the following dual quaternion.

$$\mathcal{S} = \mathring{h} + \mathring{d}\varepsilon = \mathring{h} + \frac{1}{2}\vec{v}\otimes\mathring{h}\varepsilon \quad \text{and} \quad \vec{v} = 2\mathring{d}\otimes\mathring{h}^* . \tag{26}$$

Multiplication of dual quaternions chains successive transforms.

$$\begin{aligned}{}^a\mathcal{S}_c = {}^a\mathcal{S}_b \otimes {}^b\mathcal{S}_c &= ({}^a\mathring{h}_b + {}^a\mathring{d}_b\varepsilon)\otimes({}^b\mathring{h}_c + {}^b\mathring{d}_c\varepsilon) \\ &= {}^a\mathring{h}_b \otimes {}^b\mathring{h}_c + ({}^a\mathring{h}_b \otimes {}^b\mathring{d}_c + {}^a\mathring{d}_b \otimes {}^b\mathring{h}_c)\varepsilon .\end{aligned} \tag{27}$$

Dual quaternion multiplication also has a corresponding matrix-vector product form:

$$\mathring{h} + \mathring{d}\varepsilon \overset{\text{as vector}}{\rightsquigarrow} \begin{bmatrix}\mathring{h}_x & \mathring{h}_y & \mathring{h}_z & \mathring{h}_w & \mathring{d}_x & \mathring{d}_y & \mathring{d}_z & \mathring{d}_w\end{bmatrix}^T$$

$$\begin{aligned}\mathcal{C}\otimes\mathcal{S} = \begin{bmatrix}\mathcal{C}_r\otimes\mathcal{S}_r \\ \mathcal{C}_d\otimes\mathcal{S}_r + \mathcal{C}_r\otimes\mathcal{S}_d\end{bmatrix} &= \overbrace{\begin{bmatrix}([\mathcal{C}_r]_L) & \mathbf{0}_{4\times4} \\ ([\mathcal{C}_d]_L) & ([\mathcal{C}_r]_L)\end{bmatrix}}^{[\mathcal{C}]_L}\mathcal{S} \\ &= \underbrace{\begin{bmatrix}([\mathcal{S}_r]_R) & \mathbf{0}_{4\times4} \\ ([\mathcal{S}_d]_R) & ([\mathcal{S}_r]_R)\end{bmatrix}}_{[\mathcal{S}]_R}\mathcal{C} ,\end{aligned} \tag{28}$$

where blocks of the form $[\mathcal{S}_r]_R$ are the ordinary quaternion multiply matrices from (12) constructed for the real (r) or dual (d) parts of $\mathcal{S}$ and $\mathcal{C}$.

Rewriting (27) in terms of a transformation and a point yields the dual quaternion form to transform a point. An equivalent derivation extends (15) to the dual numbers.

$$\begin{aligned} {}^a\mathcal{S}_c &= {}^a\mathcal{S}_b \otimes \left(1 + \frac{1}{2}{}^b p\varepsilon\right) \\ \rightsquigarrow \quad {}^a p &= \left(2d + h \otimes {}^b p\right) \otimes h^* \,. \end{aligned} \tag{29}$$

The quaternion-conjugate of a dual quaternion[†] is the conjugate of both real and dual parts.

$$(h + d\varepsilon)^* = h^* + d^*\varepsilon = -\vec{h}_v + h_w + \left(-\vec{d}_v + d_w\right)\varepsilon \,. \tag{30}$$

The conjugate of a unit dual quaternion produces the inverse transformation.

$$(h + d\varepsilon) \otimes (h + d\varepsilon)^* = 1, \quad \text{when } |h| = 1 \,. \tag{31}$$

Singularity-Free Dual Quaternion Exponential To derive a suitable form of the dual quaternion exponential, we begin by rewriting the ordinary quaternion exponential (18) over dual numbers.

$$\begin{aligned} \tilde{\phi} &= \sqrt{\tilde{x}^2 + \tilde{y}^2 + \tilde{z}^2} \\ \mathrm{e}^{\mathcal{S}} &= \mathrm{e}^{\tilde{w}} \left(\frac{\sin\tilde{\phi}}{\tilde{\phi}} \left(\tilde{x}\hat{\boldsymbol{i}} + \tilde{y}\hat{\boldsymbol{j}} + \tilde{z}\hat{\boldsymbol{k}}\right) + \cos\tilde{\phi} \right) \end{aligned} \tag{32}$$

Direct evaluation of (32) must contend with the singularity (zero denominator) in the factor $\frac{\sin\tilde{\phi}}{\tilde{\phi}}$. To handle the singularity, we algebraically expand the dual arithmetic and rewrite factors based on quaternion trigonometry into forms where we find suitable Taylor series.

First, we expand the dual quaternion angle $\tilde{\phi}$.

$$\begin{aligned} \tilde{\phi} &= \sqrt{(h_x + d_x\varepsilon)^2 + (h_y + d_y\varepsilon)^2 + (h_z + d_z\varepsilon)^2} \\ &= \sqrt{h_x^2 + h_y^2 + h_z^2} + \frac{h_x d_x + h_y d_y + h_z d_z}{\sqrt{h_x^2 + h_y^2 + h_z^2}}\varepsilon \\ &= \phi + \frac{\gamma}{\phi}\varepsilon \,, \end{aligned} \tag{33}$$

where ϕ is the same as the ordinary quaternion angle and $\gamma = \vec{h}_v \cdot \vec{d}_v$.

The dual sin and cos are then

$$\cos\tilde{\phi} = c - \frac{\gamma}{\phi}s\varepsilon \qquad \text{and} \qquad \sin\tilde{\phi} = s + \frac{\gamma}{\phi}c\varepsilon \,. \tag{34}$$

where $s = \sin\phi$ and $c = \cos\phi$.

Next, we expand the dual sinc function $\frac{\sin\tilde{\phi}}{\tilde{\phi}}$ and rearrange terms to find a suitable Taylor series to handle the singularity at $\phi = 0$:

$$\begin{aligned} \frac{\sin\tilde{\phi}}{\tilde{\phi}} &= \frac{\sin(\phi) + \frac{\gamma}{\phi}\cos(\phi)\varepsilon}{\phi + \frac{\gamma}{\phi}\varepsilon} \\ &= \frac{\sin(\phi)}{\phi} + \gamma\left(\frac{\cos(\phi) - \frac{\sin(\phi)}{\phi}}{\phi^2}\right)\varepsilon \\ &= \underbrace{\left(1 - \frac{\phi^2}{6} + \ldots\right)}_{(\sin\phi)/\phi} + \gamma\underbrace{\left(-\frac{1}{3} + \frac{\phi^2}{30} + \ldots\right)}_{(\cos\phi - (\sin\phi)/\phi)/\phi^2}\varepsilon \,. \end{aligned} \tag{35}$$

Finally, we expand the original form of the exponential in (32):

$$\begin{aligned} &\mathrm{e}^{\mathcal{S}} = \mathrm{e}^{\tilde{w}}\,(a_r + a_d\varepsilon) \\ &\text{where} \\ &a_r = \kappa_r\vec{h}_v + c \\ &a_d = \kappa_r\vec{d}_v + \kappa_d\vec{h}_v - \kappa_r\gamma \\ &\phi = |h_v| \,, \quad s = \sin\phi \,, \quad c = \cos\phi \\ &\gamma = \vec{h}_v \cdot \vec{d}_v \\ &\mathrm{e}^{\tilde{w}} = \mathrm{e}^{h_w} + d_w\mathrm{e}^{h_w}\varepsilon \\ &\kappa_r = \frac{s}{\phi} = 1 - \frac{\phi^2}{6} + \frac{\phi^4}{120} + \ldots \\ &\kappa_d = \gamma\frac{c - \kappa_r}{\phi^2} = \\ &\qquad = \gamma\left(-\frac{1}{3} + \frac{\phi^2}{30} - \frac{\phi^4}{840} + \ldots\right) \,. \end{aligned} \tag{36}$$

By applying the Taylor series in (35), we stably evaluate (36) in the neighborhood of $\phi = 0$.

Singularity-Free Dual Quaternion Logarithm We derive the dual quaternion logarithm by expanding the ordinary form (20) with dual arithmetic.

$$\ln\mathcal{S} = \frac{\tilde{\phi}}{\tilde{n}}\left(\vec{h}_v + \vec{d}_v\varepsilon\right) + \ln\tilde{m} \,, \tag{37}$$

where $\tilde{\phi}$, $\tilde{n}$, and $\tilde{m}$ are the dual number forms of ϕ, $|h_v|$, and $|h|$, (respectively) from (20). The dual arithmetic

[†]Dual quaternions also have a dual-conjugate $(r + d\varepsilon)^\bullet = r - d\varepsilon$ and a joint-conjugate $(\mathcal{S}^*)^\bullet$

expands as follows:

$$\begin{aligned}\tilde{n} &= \sqrt{(\hbar_x + \partial_x\varepsilon)^2 + (\hbar_y + \partial_y\varepsilon)^2 + (\hbar_z + \partial_z\varepsilon)^2} \\ &= |\hbar_v| + \frac{\vec{\hbar}_v \cdot \vec{\partial}_v}{|\hbar_v|}\varepsilon = |\hbar_v| + n_d\varepsilon, \\ \tilde{m} &= |\hbar| + \frac{\hbar \cdot \partial}{|\hbar|}\varepsilon = |\hbar| + m_d\varepsilon, \\ \tilde{\phi} &= \tan^{-1}\frac{|\hbar_v| + n_d\varepsilon}{\hbar_w + \partial_w\varepsilon}\,. \end{aligned} \tag{38}$$

Further expanding $\tilde{\phi}$ via the dual Taylor series for $\tan^{-1}$ (see Table 4):

$$\begin{aligned}\tilde{\phi} &= \operatorname{atan2}(|\hbar_v|, \hbar_w) + \left(\frac{\hbar_w n_d - |\hbar_v|\,\partial_w}{|\hbar|^2}\right)\varepsilon \\ &= \phi + \left(\frac{\hbar_w n_d - |\hbar_v|\,\partial_w}{|\hbar|^2}\right)\varepsilon\,. \end{aligned} \tag{39}$$

Next, we consider the dual $\frac{\tilde{\phi}}{\tilde{n}}$ using quaternion trigonometry (see Fig. 1) to rewrite factors as trigonometric functions for which we find well-defined Taylor series. We expand the dual arithmetic and reorder $\frac{\tilde{\phi}}{\tilde{n}}$:

$$\begin{aligned}\frac{\tilde{\phi}}{\tilde{n}} &= \frac{\phi + \left(\frac{\hbar_w n_d - |\hbar_v|\partial_w}{|\hbar|^2}\right)\varepsilon}{|\hbar_v| + n_d\varepsilon} \\ &= \frac{\phi}{|\hbar_v|} + \left(\frac{\hbar_w n_d}{|\hbar_v|\,|\hbar|^2} - \frac{\phi n_d}{|\hbar_v|^2} - \frac{\partial_w}{|\hbar|^2}\right)\varepsilon\,. \end{aligned} \tag{40}$$

Equation (40) contains a singularity where $|\hbar| = 0$. We evaluate the term $\frac{\phi}{|\hbar_v|}$ as in (21). We rewrite the larger term in the dual coefficient as follows:

$$\frac{\hbar_w n_d}{|\hbar_v|\,|\hbar|^2} - \frac{\phi n_d}{|\hbar_v|^2} = \vec{\hbar}_v \cdot \vec{\partial}_v \left(\frac{\hbar_w}{|\hbar_v|^2\,|\hbar|^2} - \frac{\phi}{|\hbar_v|^3}\right)\,. \tag{41}$$

Next, we substitute the trigonometric functions (see Fig. 1) for $|\hbar_v|$ and $\hbar_w$ and produce the corresponding Taylor series:

$$\begin{aligned}\frac{\hbar_w}{|\hbar_v|^2\,|\hbar|^2} - \frac{\phi}{|\hbar_v|^3} &= \frac{1}{|\hbar|^3}\left(\frac{\hbar_w}{|\hbar|}\frac{|\hbar|^2}{|\hbar_v|^2} - \frac{\phi\,|\hbar|^3}{|\hbar_v|^3}\right) \\ &= \frac{1}{|\hbar|^3}\left(\frac{\cos\phi}{\sin^2\phi} - \frac{\phi}{\sin^3\phi}\right) \\ &= \frac{1}{|\hbar|^3}\left(-\frac{2}{3} - \frac{1}{5}\phi^2 + \ldots\right)\,. \end{aligned} \tag{42}$$

Now that we have identified Taylor series to handle the singularities, we have the full dual quaternion logarithm:

$$\begin{aligned}\ln\mathcal{S} &= \ln\hbar + \left(\kappa_d\vec{\hbar}_v + \kappa_r\vec{\partial}_v + \frac{\hbar\cdot\partial}{|\hbar|^2}\right)\varepsilon \\ \text{where}\quad \phi &= \operatorname{atan2}(\hbar_w, |\hbar_v|) \\ \ln\hbar &= \kappa_r\vec{\hbar}_v + \ln|\hbar| \\ \kappa_r &= \frac{\phi}{|\hbar_v|} = \frac{1 + \frac{\phi^2}{6} + \frac{7\phi^4}{360} + \ldots}{|\hbar|} \\ \kappa_d &= \left(\vec{\hbar}_v \cdot \vec{\partial}_v\right)\zeta - \frac{\partial_w}{|\hbar|^2} \\ \zeta &= \frac{\frac{\hbar_w}{|\hbar|^2} - \kappa_r}{|\hbar_v|^2} = \frac{\frac{c}{s^2} - \frac{\phi}{s^3}}{|\hbar|^3} \\ &= \frac{-\frac{2}{3} - \frac{\phi^2}{5} - \frac{17\phi^4}{420} + \ldots}{|\hbar|^3} \end{aligned} \tag{43}$$

4.3 Implicit Dual Quaternions

Just as we may represent transformations with a rotation matrix and translation vector—i.e., the homogeneous transformation matrix—we can also represent transformations with a rotation quaternion and translation vector. The quaternion-translation form offers computational advantages: it consists of only seven elements and chaining requires fewer operations than both the dual quaternion and matrix forms. However, because chaining is no longer a multiplication, as with dual quaternions or matrices, analysis of quaternion-translation kinematics is more complicated, particularly for differential cases involving finding derivatives or integrating transforms. We address the analytic challenge of the quaternion-translation form by reinterpreting quaternion-translations as *implicit dual quaternions*, or alternately stated, by adopting an in-memory representation of dual quaternions as a quaternion-translation. The implicit dual quaternion combines the analytic convenience of dual quaternions and the computational efficiency the quaternion-translation representation.

The quaternion-translation form stores separately the rotation quaterion $\hbar$ and translation vector $\vec{v}$, eliminating the coupling of rotation and translation in the dual part of the dual quaternion:

$$\underbrace{\hbar + \frac{1}{2}\vec{v}\otimes\hbar\varepsilon}_{\text{explicit dual quaternion}} \overset{\text{rewrite}}{\rightsquigarrow} \underbrace{\begin{pmatrix}\hbar\\ \vec{v}\end{pmatrix}}_{\text{implicit dual quaternion}}\,. \tag{44}$$

To transform a point, we first apply the rotation, then add the translation—the same operations performed by

the homogenous transformation matrix and which we may derive from (26) and (29):

$$ {}^{a}p = {}^{a}\hbar_b \otimes {}^{b}p \otimes ({}^{a}\hbar_b)^* + {}^{a}\vec{v}_b \; . \tag{45} $$

Chaining of transforms consists of the following operations, again analogous to the transformation matrix and derivable from (26) and (27):

$$ \begin{pmatrix} {}^{a}\hbar_c \\ {}^{a}\vec{v}_c \end{pmatrix} = \begin{pmatrix} {}^{a}\hbar_b \otimes {}^{b}\hbar_c \\ {}^{a}\hbar_b \otimes {}^{b}v_c \otimes ({}^{a}\hbar_b)^* + {}^{a}v_b \end{pmatrix} \; . \tag{46} $$

Implicit Exponential We derive the exponential for the implicit dual quaternion starting with (36), extracting the translation, and finally identifying Taylor series.

First, we simplify (36) to the pure case, i.e., zero scalar part:

$$ \begin{aligned} &\mathrm{e}^{\vec{\omega}+\vec{\nu}\boldsymbol{\varepsilon}} = \left(\frac{s}{\phi}\vec{\omega} + c\right) + \left(\frac{s}{\phi}\vec{\nu} + \frac{c - \frac{s}{\phi}}{\phi^2}\gamma\vec{\omega} - \frac{s}{\phi}\gamma\right)\boldsymbol{\varepsilon} \\ &\text{where} \quad \gamma = \vec{\omega}\cdot\vec{\nu} \quad \text{and} \quad \phi = |\vec{\omega}| \; . \end{aligned} \tag{47} $$

Next, we extract the translation from the dual part.

$$ \begin{aligned} &\breve{\exp}\left(\vec{\omega} + \vec{\nu}\boldsymbol{\varepsilon}\right) = \begin{pmatrix} \hbar \\ \vec{v} \end{pmatrix} \\ &= \begin{pmatrix} \left(\frac{s}{\phi}\vec{\omega} + c\right) \\ 2\left(\frac{s}{\phi}\vec{\nu} + \frac{c-\frac{s}{\phi}}{\phi^2}\gamma\vec{\omega} - \frac{s}{\phi}\gamma\right) \otimes \left(\frac{s}{\phi}\vec{\omega} + c\right)^* \end{pmatrix} \; . \end{aligned} \tag{48} $$

In (48), we may evaluate the rotation part $\hbar$ as in the ordinary quaternion case. For the translation part $\vec{v}$, we first algebraically simplify:

$$ \vec{v} = 2\left(\frac{s}{\phi}\vec{\nu} + \frac{c - \frac{s}{\phi}}{\phi^2}\gamma\vec{\omega} - \frac{s}{\phi}\gamma\right) \otimes \left(\frac{s}{\phi}\vec{\omega} + c\right)^* \tag{49} $$

$$ = 2\left(-\frac{s^2}{\phi^2}\vec{\nu}\times\vec{\omega} + \frac{cs}{\phi}\vec{\nu} + \frac{c(c-\frac{s}{\phi}) + s^2}{\phi^2}\gamma\vec{\omega}\right) \; . \tag{50} $$

Then, we simplify trigonometric factors and identify the common subexpressions.

$$ \vec{v} = \frac{2s}{\phi}\left(\left(\frac{s}{\phi}\vec{\omega}\right)\times\vec{\nu}\right) + c\frac{2s}{\phi}\vec{\nu} + \left(\frac{2 - c\frac{2s}{\phi}}{\phi^2}\right)\gamma\vec{\omega} \; . \tag{51} $$

Using the Taylor series from Table 1 and for the new factor, we obtain:

$$ \begin{aligned} &\breve{\exp}\left(\vec{\omega} + \vec{\nu}\boldsymbol{\varepsilon}\right) = \begin{pmatrix} \hbar \\ \vec{v} \end{pmatrix} \\ &\text{where} \\ &\hbar = \mu_r\vec{\omega} + c \\ &\vec{v} = 2\mu_r\left(\vec{\hbar}_v \times \vec{\nu}\right) + c\left(2\mu_r\right)\vec{\nu} + \mu_d\gamma\vec{\omega} \\ &\gamma = \vec{\omega}\cdot\vec{\nu} \\ &\phi = |\vec{\omega}| \;, \quad s = \sin\phi \;, \quad c = \cos\phi \\ &\mu_r = \frac{s}{\phi} = 1 - \frac{\phi^2}{6} + \frac{\phi^4}{120} + \ldots \\ &\mu_d = \frac{2 - c(2\mu_r)}{\phi^2} = \frac{4}{3} - \frac{4\phi^2}{15} + \frac{8\phi^4}{315} + \ldots \end{aligned} \tag{52} $$

Implicit Logarithm We derive the implicit logarithm starting with (43), substituting the translation vector, and finally identifying suitable Taylor series.

We begin with the dual quaternion logarithm (43):

$$ \breve{\ln}\begin{pmatrix} \hbar \\ \vec{v} \end{pmatrix} = \vec{\omega} + \vec{\nu}\boldsymbol{\varepsilon} = \ln\left(\hbar + \frac{1}{2}\vec{v}\otimes\hbar\boldsymbol{\varepsilon}\right) \; . \tag{53} $$

The real part $\vec{\omega}$ of the implicit logarithm is identical to the dual quaternion case (43). We assume a unit quaternion $|\hbar| = 1$, so the scalar part of the logarithm is zero.

$$ \left(\breve{\ln}\begin{pmatrix} \hbar \\ \vec{v} \end{pmatrix}\right)_{\text{real}} = \vec{\omega} = \frac{\phi}{|\hbar_v|}\vec{\hbar}_v = \frac{\phi}{\sin\phi}\vec{\hbar}_v \; . \tag{54} $$

For the dual part $\vec{\nu}$, we expand (43), simplifying for the unit case $|\hbar| = 1$:

$$ \begin{aligned} \left(\breve{\ln}\begin{pmatrix} \hbar \\ \vec{v} \end{pmatrix}\right)_{\text{dual}} &= \vec{\nu} = \kappa_d\vec{\hbar}_v + \kappa_r\vec{d}_v + \vec{\hbar}_v\cdot\vec{d}_v + \hbar_w d_w \\ \text{where} \quad \vec{d}_v &= \frac{1}{2}\vec{v}\times\vec{\hbar}_v + \frac{1}{2}\hbar_w\vec{v} \\ d_w &= -\frac{1}{2}\vec{v}\cdot\vec{\hbar}_v \; . \end{aligned} \tag{55} $$

Substituting for κ_r and κ_d from (43) and for dual part d in terms of translation $\vec{v}$, we simplify to:

$$ \vec{\nu} = -\frac{1}{2}\vec{v}\cdot\vec{\hbar}_v\left(\frac{\frac{\hbar_w\phi}{s} - 1}{s^2}\right)\vec{\hbar}_v + \frac{\hbar_w\phi}{2s}\vec{v} + \frac{\phi}{2s}\left(\vec{v}\times\vec{\hbar}_v\right) \; . \tag{56} $$

Noting that $\hbar_w = \cos\phi$ and $\vec{\omega} = \frac{\phi}{\sin\phi}\vec{\hbar}_v$, we further simplify to:

$$ \vec{\nu} = \left(\frac{\vec{v}}{2}\right)\cdot\vec{\omega}\left(\frac{1 - c\frac{\phi}{s}}{\phi^2}\right)\vec{\omega} + c\frac{\phi}{s}\left(\frac{\vec{v}}{2}\right) + \left(\left(\frac{\vec{v}}{2}\right)\times\omega\right) \tag{57} $$

Finally, we identify the Taylor series to obtain the implicit logarithm as follows:

$$
\begin{aligned}
&\check{\ln}\begin{pmatrix}\hbar\\ \vec{v}\end{pmatrix} = \vec{\omega} + \vec{\nu}\boldsymbol{\varepsilon}\\
&\text{where } s = |\hbar_v| \ , \ c = \hbar_w \ , \ \phi = \operatorname{atan2}(s, c)\\
&\qquad \vec{\omega} = \frac{\phi}{s}\vec{\hbar}_v\\
&\qquad \vec{\nu} = \mu_d\left(\frac{\vec{v}}{2}\cdot\vec{\omega}\right)\vec{\omega} + \mu_r\frac{\vec{v}}{2} + \frac{\vec{v}}{2}\times\omega\\
&\qquad \mu_r = \frac{c\phi}{s} = 1 - \frac{\phi^2}{3} - \frac{\phi^4}{45} - \dots\\
&\qquad \mu_d = \frac{1-\mu_r}{\phi^2} = \frac{1}{3} + \frac{\phi^2}{45} + \frac{2\phi^4}{945} + \dots
\end{aligned}
\tag{58}
$$

5 Application to Kinematics

We present an application and performance analysis of quaternion-based kinematics. First, we compare quaternion forms with matrices for forward kinematics. Then, we derive dual quaternion forms for differential kinematics. Finally, we apply dual quaternion analysis to inverse position kinematics formulated as sequential quadratic programming (SQP). The results show that quaternion forms and analysis offer improved performance for robot kinematics.

The test platform for empirical performance evaluations was an Intel® Xeon E3-1275 v6. We used the kinematics implementations in Amino (`http://amino.dyalab.org`), and we solved the SQP using NLopt (Johnson 2019; Kraft 1988, 1994).

5.1 Forward Kinematics

Both matrix and quaterion representations may be used to compute the forward kinematics of robot manipulators. We compare the different representations and show that the quaternion-translation offers the best forward kinematics performance. Analytically, quaternion-translations require the fewest arithmetic instructions, and in our empirical evaluation, quaternion-translations require the shortest execution time.

Table 5 and Table 6 compare operations for quaternion and matrix forms.

Table 7 summarizes the construction of relative transformations for single degree-of-freedom joints using matrix and quaternion forms. We use the known axis of joints to simplify construction over the general-case exponential. The result shows that quaternion-translations require the fewest arithmetic instructions.

We empirically compare the computation time for the forward kinematics of several common manipulators using matrices, dual quaternions, and quaternion-translations. For each manipulator, we generated random configurations using the `rand()` function (IEEE and The Open Group 2018; Loosemore et al. 2020) to sample each configuration variable θ_i as follows:

$$
\theta_i = \theta_{i,\min} + (\theta_{i,\max} - \theta_{i,\min})\frac{\texttt{rand()}}{\texttt{RAND\_MAX}} \ , \tag{59}
$$

where $\theta_{i,\min}$ and $\theta_{i,\max}$ are the minimum and maximum values of the configuration variable, respectively. We tested 10,000 random configurations and then evaluated the forward kinematics 10,000 times for each configuration. Fig. 2 presents the results in terms of speedup over the baseline matrix representation. The timing variation between configurations was minor (standard deviation less than 10% of the mean) since computing the forward kinematics requires the same sequence of floating point operations for any configuration; the observed variation may arise due CPU or operating system issues such as caching or context switching, which are effectively nondeterministic. The quaternion-translation shows the best empirical performance, consistent with the instruction counts in Table 5, Table 6, and Table 7. Additionally, the explicit dual quaternion also offers slightly better performance than matrices in our tests. Even though matrices require fewer arithmetic instructions to construct and chain, several other advantages of the dual quaternions lead to the improved performance. Dual quaternions are more compact than matrices, which reduces necessary data shuffling, and quaternions require fewer operations for the exponential and rotation chaining, which are heavily used in robots with many revolute frames.

5.2 Differential Kinematics

Next, we derive dual quaternion forms for differential kinematics. We first introduce the dual quaternion derivative. Then, we restate the conventional construction of the geometric manipulator Jacobian and compare construction using matrices, explicit, and implicit dual quaternions. Finally, we derive the dual quaternion form of the manipulator Jacobian. For a more thorough overview of Jacobian kinematics methods, please see Buss (2004).

Quaternion Derivatives First, we relate velocities and the quaternion derivatives. The ordinary quaternion derivative $\dot{\hbar}$ relates to angular velocity ω as follows (Grassia 1998):

$$
\dot{\hbar} = \frac{1}{2}\omega\otimes\hbar \ . \tag{60}
$$

	Representation	Storage	Chain Mul.	Chain Add	Rot./Tf. Mul.	Rot./Tf. Add	Normalize Mul.	Normalize Add	Normalize Other
Rot.	*Rotation Matrix*	9	27	18	9	6	27	15	sqrt(3)
	Quaternion	4	16	12	15	15	8	3	sqrt
Tf.	*Transformation Matrix*	12	36	27	9	9	27	15	sqrt(3)
	Dual Quaternion	8	48	40	28	28	12	3	sqrt
	Quaternion-Translation	7	31	30	15	18	8	3	sqrt

Table 5. Requirements for storage, chaining, and point transformation. Quaternion-based representations are more compact than matrices. Ordinary quaternions and quaternion-translations are most efficient for chaining rotations and transformations, respectively. Matrices are most efficient for rotating and transforming points.

	Representation	Exponential Mul.	Exponential Add	Exponential Other	Logarithm Mul.	Logarithm Add	Logarithm Other
Rot.	*Rot. Matrix*	17	15	sqrt, sincos	5	7	sqrt, atan2
	Quaternion	9	2	sqrt, sincos, exp	8	3	sqrt(2), atan2, ln
	Unit Q.	7	2	sqrt, sincos	7	2	sqrt, atan2
Tf.	*Tf. Matrix*	39	34	sqrt, sincos	31	32	sqrt, atan2
	Dual Quat.	31	12	sqrt, sincos, exp	22	11	sqrt(2), atan2, ln
	Unit Dual Q.	19	8	sqrt, sincos	18	9	sqrt, atan2
	Quat.-Trans.	28	15	sqrt, sincos	28	16	sqrt, atan2

Table 6. Exponential and Logarithm Operation Counts. Ordinary and dual quaternions are more efficient than their matrix equivalents. The quaternion-translation costs are between the matrix and dual-quaternion.

		Form	Mul.	Add	Other
Revolute	*Tf. Matrix*	$\begin{bmatrix} e^{[\theta\hat{u}]} & \mathbf{v} \\ 0 & 1 \end{bmatrix}$	12	13	sincos
	Dual Quat.	$e^{\frac{\theta}{2}\hat{u}} + \frac{\vec{v}}{2} \otimes e^{\frac{\theta}{2}\hat{u}}\boldsymbol{\varepsilon}$	19	12	sincos
	Quat.-Trans.	$\begin{pmatrix} e^{\frac{\theta}{2}\hat{u}} \\ \vec{v} \end{pmatrix}$	3	0	sincos
Prismatic	*Tf. Matrix*	$\begin{bmatrix} \mathbf{R} & \ell\hat{\mathbf{u}} \\ 0 & 1 \end{bmatrix}$	3	0	-
	Dual Quat.	$\hbar + \ell\left(\frac{\hat{u}}{2} \otimes \hbar\right)\boldsymbol{\varepsilon}$	4	0	-
	Quat.-Trans.	$\begin{pmatrix} \hbar \\ \ell\hat{u} \end{pmatrix}$	3	0	-
Helical	*Tf. Matrix*	$\begin{bmatrix} e^{[\theta\hat{u}]} & (k\hat{\mathbf{u}})\,\theta \\ 0 & 1 \end{bmatrix}$	15	13	sincos
	Dual Quat.	$e^{\frac{\theta}{2}\hat{u}} + \theta\frac{k\hat{u}}{2} \otimes e^{\frac{\theta}{2}\hat{u}}\boldsymbol{\varepsilon}$	23	14	sincos
	Quat.-Trans.	$\begin{pmatrix} e^{\frac{\theta}{2}\hat{u}} \\ (k\hat{u})\theta \end{pmatrix}$	6	0	sincos

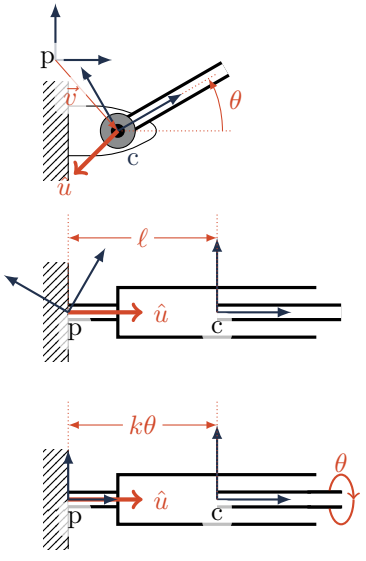
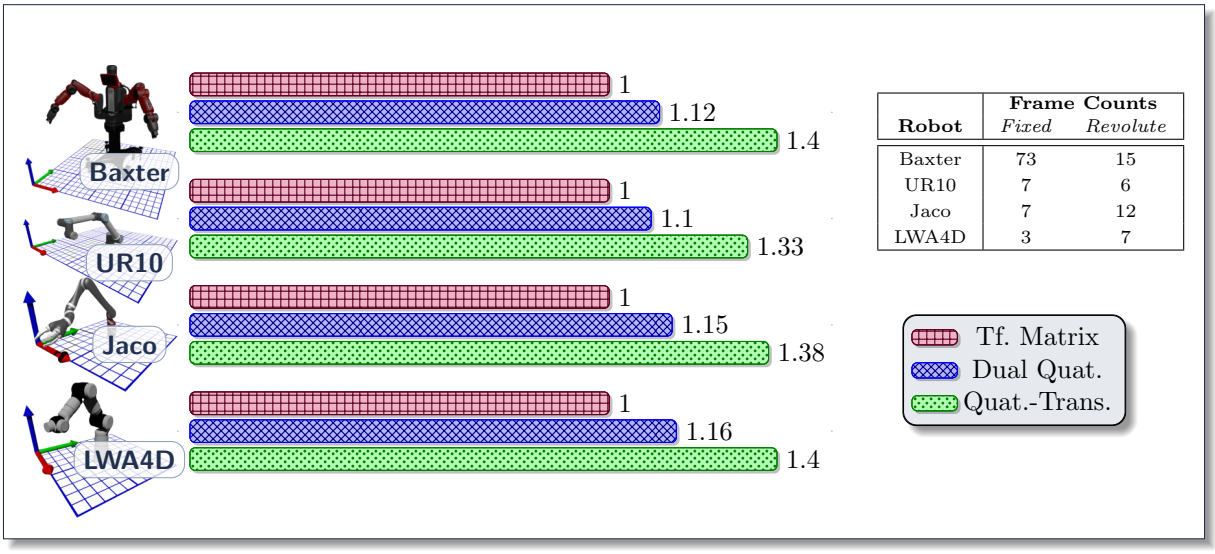


Table 7. Single degree-of-freedom joint transforms and operation counts. The quaternion-translation representation is most efficient to construct.

We relate the dual quaternion derivative to rotational and translational velocities by differentiating (26) and substituting for $\dot{\hbar}$ using (60):

$$\dot{\mathcal{S}} = \frac{d}{dt}\left(\hbar + \frac{1}{2}\vec{v} \otimes \hbar\boldsymbol{\varepsilon}\right) = \dot{\hbar} + \frac{1}{2}\left(\dot{v} \otimes \hbar + \vec{v} \otimes \dot{\hbar}\right)\boldsymbol{\varepsilon}$$
$$= \frac{1}{2}\left(\omega \otimes \hbar + \frac{1}{2}\left(\dot{v} \otimes \hbar + \frac{1}{2}\vec{v} \otimes \omega \otimes \hbar\right)\boldsymbol{\varepsilon}\right). \quad (61)$$



Robot	Frame Counts Fixed	Revolute
Baxter	73	15
UR10	7	6
Jaco	7	12
LWA4D	3	7

Figure 2. Forward kinematics speedup (higher is better), demonstrating a 30%-40% performance improvement using quaternion forms. We compare the execution time to compute the forward kinematics for the Rethink Baxter, Universal Robots UR10, Kinova Jaco, and Schunk LWA4D manipulators using transformation matrices, dual quaternions, and quaternion-translations. The results are shown as speedup ($\frac{t_{\text{baseline}}}{t_{\text{new}}}$) over the transformation matrix case.

Then, we factor (61) to separate the spatial twist Ω as a dual quaternion,

$$\dot{\mathcal{S}} = \frac{1}{2}\Omega \otimes \mathcal{S}\,, \quad \text{where} \quad \Omega = \omega + (\dot{v} + \vec{v} \times \omega)\,\boldsymbol{\varepsilon}\,. \tag{62}$$

Note that twist Ω in (62) is a pure dual quaternion, i.e., zero scalar in the real and dual parts.

Geometric Manipulator Jacobian The manipulator Jacobian relates changes in workspace and configuration space. There are multiple ways we might parameterize workspace change, e.g., rotational (ω) and translational ($\dot{v}$) velocities, twist (Ω), or the dual quaternion derivative ($\dot{\mathcal{S}}$):

$$\begin{bmatrix}\omega \\ \dot{v}\end{bmatrix} = \mathbf{J}_x\dot{\boldsymbol{\theta}}\,, \quad \Omega = \mathbf{J}_\Omega\dot{\boldsymbol{\theta}}\,, \quad \text{and} \quad \dot{\mathcal{S}} = \frac{\partial \mathcal{S}}{\partial \boldsymbol{\theta}}\dot{\boldsymbol{\theta}}\,. \tag{63}$$

We compare the construction of velocity ($\mathbf{J}_x$) and twist ($\mathbf{J}_\Omega$) Jacobians across different representations. Then, we derive the dual quaternion Jacobian $\frac{\partial \mathcal{S}}{\partial \boldsymbol{\theta}}$.

Each column i of the Jacobian $\mathbf{J}$ corresponds to the i^{th} configuration variable and contains a rotational part ($\mathbf{j}_r$) and a translational part ($\mathbf{j}_v$ or $\mathbf{j}_p$):

$$\mathbf{J}_x = \begin{bmatrix}\begin{pmatrix}\mathbf{j}_r \\ \mathbf{j}_v\end{pmatrix}_1 \cdots \begin{pmatrix}\mathbf{j}_r \\ \mathbf{j}_v\end{pmatrix}_n\end{bmatrix}$$
$$\mathbf{J}_\Omega = \begin{bmatrix}\begin{pmatrix}\mathbf{j}_r \\ \mathbf{j}_p\end{pmatrix}_1 \cdots \begin{pmatrix}\mathbf{j}_r \\ \mathbf{j}_p\end{pmatrix}_n\end{bmatrix}. \tag{64}$$

For the common cases of revolute and prismatic joints, we may construct the corresponding Jacobian column based on the joint axis and origin. We construct the velocity Jacobian $\mathbf{J}_x$ as follows (Buss 2004):

$$\begin{pmatrix}\mathbf{j}_r \\ \mathbf{j}_v\end{pmatrix}_i\Big|_{\text{revolute}} = \begin{pmatrix}{}^G\hat{\boldsymbol{u}}_i \\ {}^G\hat{\boldsymbol{u}}_i \times ({}^G\vec{v}_e - {}^G\vec{v}_i)\end{pmatrix}$$
$$\begin{pmatrix}\mathbf{j}_r \\ \mathbf{j}_v\end{pmatrix}_i\Big|_{\text{prismatic}} = \begin{pmatrix}\mathbf{0} \\ {}^G\hat{\boldsymbol{u}}_i\end{pmatrix}, \tag{65}$$

where ${}^G\hat{\boldsymbol{u}}_i$ is the joint axis, ${}^G\vec{v}_e$ is the end-effector translation, and ${}^G\vec{v}_i$ is the origin of joint i, all in global (fixed) frame G. We construct twist Jacobian $\mathbf{J}_\Omega$ as follows (Lynch and Park 2017):

$$\begin{pmatrix}\mathbf{j}_r \\ \mathbf{j}_p\end{pmatrix}_i\Big|_{\text{revolute}} = \begin{pmatrix}{}^G\hat{\boldsymbol{u}}_i \\ {}^G\vec{v}_i \times {}^G\hat{\boldsymbol{u}}_i\end{pmatrix}$$
$$\begin{pmatrix}\mathbf{j}_r \\ \mathbf{j}_p\end{pmatrix}_i\Big|_{\text{prismatic}} = \begin{pmatrix}\mathbf{0} \\ {}^G\hat{\boldsymbol{u}}_i\end{pmatrix}. \tag{66}$$

Thus, the main operation to construct each column of $\mathbf{J}_x$ or $\mathbf{J}_\Omega$ is to rotate the joint's axis into the global frame. Table 8 compares the construction of twist Jacobian $\mathbf{J}_\Omega$ for matrices, explicit, and implicit dual quaternions; construction of $\mathbf{J}_x$ is similar, requiring only an additional vector subtraction for each revolute joint. To directly construct the Jacobian column, matrices require the fewest operations due to their efficiency at applying a rotation. However, when we also consider the cost to construct and chain the transformation,

implicit dual quaternions require fewer operations in the revolute case, which typically make up the majority of robot joints.

Quaternion Manipulator Jacobian We now combine the geometric Jacobian and quaternion derivatives to construct the quaternion manipulator Jacobian, i.e., the relationship between configuration velocity $\dot{\boldsymbol{\theta}}$ and dual quaternion derivative $\dot{\mathcal{S}}$. First, we rearrange (62) in matrix form (see (28)) to separate twist Ω to one side,

$$\dot{\mathcal{S}} = \frac{1}{2}\Omega \otimes \mathcal{S} = \frac{1}{2}\left[\mathcal{S}\right]_R \Omega \,. \tag{67}$$

Then, we substitute for twist Ω in (67) using the twist Jacobian $\mathbf{J}_\Omega$ from (63),

$$\dot{\mathcal{S}} = \frac{1}{2}\left[\mathcal{S}\right]_R \Omega \qquad \rightsquigarrow \qquad \dot{\mathcal{S}} = \underbrace{\left(\frac{1}{2}\left[\mathcal{S}\right]_R \mathbf{J}_\Omega\right)}_{\partial \mathcal{S}/\partial \boldsymbol{\theta}} \dot{\boldsymbol{\theta}} \,. \tag{68}$$

We now have a form for the dual quaternion Jacobian in terms of the twist Jacobian $\mathbf{J}_\Omega$ obtained from Table 8:

$$\boxed{\frac{\partial \mathcal{S}}{\partial \boldsymbol{\theta}} = \frac{1}{2}\left[\mathcal{S}\right]_R \mathbf{J}_\Omega \,.} \tag{69}$$

If we need the Jacobian only for the rotation quaterion, we take the upper-half (rotation part) of (69) and simplify to,

$$\frac{\partial \hbar}{\partial \boldsymbol{\theta}} = \frac{1}{2}\left[\hbar\right]_R \begin{bmatrix} \mathbf{J}_r \\ \mathbf{0}_{1\times n} \end{bmatrix} \,, \tag{70}$$

where $\mathbf{J}_r$ is the upper-half (rotation part) of $\mathbf{J}_\Omega$ or equivalently $\mathbf{J}_x$.

We may solve inverse velocity kinematics—finding a suitable joint velocity to achieve a desired workspace change—by solving the system of linear equations in any of the forms of (63). To address kinematic singularities, robust methods to compute the pseudoinverse may be applied to any of these Jacobian forms (Nakamura and Hanafusa 1986; Wampler 1986; Buss 2004; Buss and Kim 2005). The velocity Jacobian $\mathbf{J}_x$ is a $6 \times n$ matrix. The twist Jacobian $\mathbf{J}_\Omega$ contains all zeroes in its real and dual scalar rows and thus simplifies to a $6 \times n$ matrix. In contrast, the dual quaternion Jacobian $\frac{\partial \mathcal{S}}{\partial \boldsymbol{\theta}}$ is an $8 \times n$ matrix. Thus $\mathbf{J}_x$ and $\mathbf{J}_\Omega$, being smaller matrices, are more efficient for inverse velocity kinematics. However, we use $\frac{\partial \mathcal{S}}{\partial \boldsymbol{\theta}}$ to achieve an efficiency advantage for inverse *position* kinematics.

5.3 Inverse Position Kinematics

We extend our dual quaternion analysis to improve the efficiency and robustness of inverse position kinematics. Inverse position kinematics has a direct interpretation as constrained optimization to minimize error f between actual ($\mathcal{S}_{\text{act}}$) and reference ($\mathcal{S}_{\text{ref}}$) poses, subject to joint constraints:

$$\begin{aligned} \underset{\boldsymbol{\theta}}{\text{minimize}} \quad & f\left(\mathcal{S}_{\text{act}}(\boldsymbol{\theta}),\, \mathcal{S}_{\text{ref}}\right) \\ \text{subject to} \quad & \boldsymbol{\theta}_{\min} \le \boldsymbol{\theta} \le \boldsymbol{\theta}_{\max} \,. \end{aligned} \tag{71}$$

Beeson and Ames (2015) formulate inverse position kinematics as sequential quadratic programming (SQP) in "TRAC-IK" and present several objective functions for pose error. The SQP offers improved robustness over a benchmark Newton-Raphson solver. However, the TRAC-IK implementation computes gradients via finite difference, requiring repeated evaluation of the robot's forward kinematics. Consequently, the SQP is often slower than the Newton-Raphson benchmark, leading to an overall approach that runs multiple solvers across different threads.

We derive the analytic gradients for objective functions in Beeson and Ames (2015). As with the TRAK-IK implementation, we use a quasi-Newton SQP solver—specifically NLopt (Johnson 2019; Kraft 1988, 1994)—which approximates the Hessian, so only the gradient is necessary. Avoiding the repeated forward kinematics computations required by the finite difference gradient results in a 300-500% speedup of inverse kinematics (see Table 9).

Logarithm Objective First, we derive the analytic gradient for the objective function in equation (3) of Beeson and Ames (2015). This objective function computes error as the sum of squares of the logarithm of the error dual quaternion:

$$f(\boldsymbol{\theta}) = \left|\Omega_{\text{rel}}\right|^2 = \left|\ln\left(\mathcal{S}^*_{\text{act}}(\boldsymbol{\theta}) \otimes \mathcal{S}_{\text{ref}}\right)\right|^2 \,. \tag{72}$$

To find the gradient of (72), we first modify our notation by rewriting operations as functions,

$$f(\boldsymbol{\theta}) = \text{ssq}\left(\ln\left(\text{conj}\left(\mathcal{S}_{\text{act}}(\boldsymbol{\theta})\right) \otimes \mathcal{S}_{\text{ref}}\right)\right) \,, \tag{73}$$

where $\text{ssq}\left(\mathbf{x}\right) = \mathbf{x}^T\mathbf{x}$ and $\text{conj}\left(\mathcal{S}\right) = \mathcal{S}^*$.

Then, we differentiate by repeatedly applying the chain rule:

1. $\nabla f(\boldsymbol{\theta}) = \nabla \text{ssq}\left(\ln\left(\mathcal{S}^*_{\text{act}}(\boldsymbol{\theta}) \otimes \mathcal{S}_{\text{ref}}\right)\right) * \frac{\partial}{\partial \boldsymbol{\theta}} \ln\left(\mathcal{S}^*_{\text{act}}(\boldsymbol{\theta}) \otimes \mathcal{S}_{\text{ref}}\right)$
2. $\frac{\partial}{\partial \boldsymbol{\theta}} \ln\left(\mathcal{S}^*_{\text{act}}(\boldsymbol{\theta}) \otimes \mathcal{S}_{\text{ref}}\right) = \frac{\partial \ln}{\partial \mathcal{S}}\left(\mathcal{S}^*_{\text{act}}(\boldsymbol{\theta}) \otimes \mathcal{S}_{\text{ref}}\right) * \frac{\partial}{\partial \boldsymbol{\theta}} \text{conj}\left(\mathcal{S}_{\text{act}}(\boldsymbol{\theta})\right) \otimes \mathcal{S}_{\text{ref}}$
3. $\frac{\partial}{\partial \boldsymbol{\theta}} \text{conj}\left(\mathcal{S}_{\text{act}}(\boldsymbol{\theta})\right) \otimes \mathcal{S}_{\text{ref}} = \frac{\partial \text{conj}}{\partial \boldsymbol{\theta}}\left(\mathcal{S}_{\text{act}}(\boldsymbol{\theta})\right) \otimes \mathcal{S}_{\text{ref}} + \text{conj}\left(\mathcal{S}_{\text{act}}(\boldsymbol{\theta})\right) \otimes \cancelto{0}{\frac{\partial \mathcal{S}_{\text{ref}}}{\partial \boldsymbol{\theta}}}$
4. $\frac{\partial \text{conj}}{\partial \boldsymbol{\theta}}\left(\mathcal{S}_{\text{act}}(\boldsymbol{\theta})\right) = \frac{\partial \text{conj}}{\partial \mathcal{S}}\left(\mathcal{S}_{\text{act}}(\boldsymbol{\theta})\right) \frac{\partial \mathcal{S}_{\text{act}}}{\partial \boldsymbol{\theta}}$

Representation		Jacobian Column	Direct *Mul.*	Direct *Add*	Total *Mul.*	Total *Add*	Total *Other*
Revolute	*Tf. Matrix*	$\begin{pmatrix} {}^G\mathbf{R}_i\, {}^i\hat{\mathbf{u}} \\ {}^G\vec{v}_i \times \mathbf{j}_r \end{pmatrix}$	15	9	63	49	sincos
	Dual Quat.	$\begin{pmatrix} {}^G\hbar_i \otimes {}^i\hat{\mathbf{u}} \otimes {}^G\hbar_i^* \\ \left(2\,{}^G d_i \otimes {}^G\hbar_i^*\right) \times \mathbf{j}_r \end{pmatrix}$	37	34	104	86	sincos
	Quat.-Trans.	$\begin{pmatrix} {}^G\hbar_i \otimes {}^i\hat{\mathbf{u}} \otimes {}^G\hbar_i^* \\ {}^G\vec{v}_i \times \mathbf{j}_r \end{pmatrix}$	21	18	55	48	sincos
Prismatic	*Tf. Matrix*	$\begin{pmatrix} \mathbf{0} \\ {}^G\mathbf{R}_i\, {}^i\hat{\mathbf{u}} \end{pmatrix}$	9	6	48	33	-
	Dual Quat.	$\begin{pmatrix} \mathbf{0} \\ {}^G\hbar_i \otimes {}^i\hat{\mathbf{u}} \otimes {}^G\hbar_i^* \end{pmatrix}$	15	15	67	55	-
	Quat.-Trans.	$\begin{pmatrix} \mathbf{0} \\ {}^G\hbar_i \otimes {}^i\hat{\mathbf{u}} \otimes {}^G\hbar_i^* \end{pmatrix}$	15	15	49	45	-

Table 8. Manipulator Twist Jacobian ($\mathbf{J}_\Omega$) construction and operation counts. Transformation matrices require the least operations to directly construct the corresponding Jacobian column. However, if we include constructing and chaining the transform, Quaternion-translations require the least operations for the common-case of revolute joints.

The resulting gradient is as follows. We omit arguments for each partial derivative to simplify notation and convert the quaternion multiplication to matrix multiplication via (28).

$$\begin{aligned} \nabla f &= (\nabla \text{ssq}) \left(\frac{\partial \ln}{\partial \mathcal{S}}\right) \left(\left(\frac{\partial \text{conj}}{\partial \mathcal{S}}\right) \left(\frac{\partial \mathcal{S}_{\text{act}}}{\partial \boldsymbol{\theta}}\right)\right) \otimes \mathcal{S}_{\text{ref}} \\ &= (\nabla \text{ssq}) \left(\frac{\partial \ln}{\partial \mathcal{S}}\right) [\mathcal{S}_{\text{ref}}]_R \left(\frac{\partial \text{conj}}{\partial \mathcal{S}}\right) \left(\frac{\partial \mathcal{S}_{\text{act}}}{\partial \boldsymbol{\theta}}\right) . \end{aligned} \tag{74}$$

Next, we find each partial derivative in (74). We have $\frac{\partial \mathcal{S}_{\text{act}}}{\partial \boldsymbol{\theta}}$ from (69), and we derive ∇ ssq and $\frac{\partial \text{conj}}{\partial \mathcal{S}}$ directly as follows:

$$\begin{aligned} \nabla \text{ssq}(\mathbf{x}) &= \begin{bmatrix} 2x_1 & 2x_2 & \dots & 2x_n \end{bmatrix} \\ \frac{\partial \text{conj}}{\partial \mathcal{S}} &= \begin{bmatrix} -\mathbf{I}_{3\times 3} & 0 & \mathbf{0}_{3\times 3} & 0 \\ \mathbf{0}_{1\times 3} & 1 & \mathbf{0}_{1\times 3} & 0 \\ \mathbf{0}_{3\times 3} & 0 & -\mathbf{I}_{3\times 3} & 0 \\ \mathbf{0}_{1\times 3} & 0 & \mathbf{0}_{1\times 3} & 1 \end{bmatrix} . \end{aligned} \tag{75}$$

Deriving the Jacobian of the quaternion logarithms is more involved. Direct symbolic partial differentiation of (20) and (43) yields large expressions that contain singularities. However, by carefully inspecting the elements, we again apply the quaternion trigonometry of Fig. 1 and identify well-defined Taylor series to remove the singularities. We summarize the resulting Jacobians for the ordinary and dual quaternion logarithms.

The Jacobian of the ordinary quaternion logarithm is,

$$\begin{aligned} \frac{\partial \ln}{\partial \hbar} &= \\ &\begin{bmatrix} \zeta x^2 + \eta & \zeta xy & \zeta xz & -\frac{x}{|\hbar|^2} \\ \zeta xy & \zeta y^2 + \eta & \zeta yz & -\frac{y}{|\hbar|^2} \\ \zeta xz & \zeta yz & \zeta z^2 + \eta & -\frac{z}{|\hbar|^2} \\ \frac{x}{|\hbar|^2} & \frac{y}{|\hbar|^2} & \frac{z}{|\hbar|^2} & \frac{w}{|\hbar|^2} \end{bmatrix} \\ &\text{where} \\ \eta &= \frac{\phi}{|\hbar_v|} = \frac{1}{|\hbar|}\left(\frac{\phi}{\sin\phi}\right) \\ &= \frac{1}{|\hbar|}\left(1 + \frac{\phi^2}{6} + 7\frac{\phi^4}{360} + \dots\right) \\ \zeta &= \frac{\frac{\hbar_w}{|\hbar|^2} - \eta}{|\hbar_v|^2} = \frac{\frac{\cos\phi}{\sin^2\phi} - \frac{\phi}{\sin^3\phi}}{|\hbar|^3} \\ &= \frac{-\frac{2}{3} - \frac{\phi^2}{5} - \frac{17\phi^4}{420} + \dots}{|\hbar|^3} . \end{aligned} \tag{76}$$

We note that factor ζ also appears in the dual quaternion logarithm (43). Derivatives and dual number functions are related through the dual number Taylor series (24).

The Jacobian of the dual quaternion logarithm is,

$$\frac{\partial \ln}{\partial \mathcal{S}} = \begin{bmatrix} \left(\frac{\partial \ln}{\partial \mathcal{S}}\Big|_{\text{real}}\right) & \mathbf{0}_{4\times 4} \\ \left(\frac{\partial \ln}{\partial \mathcal{S}}\Big|_{\text{dual}}\right) & \left(\frac{\partial \ln}{\partial \mathcal{S}}\Big|_{\text{real}}\right) \end{bmatrix} \tag{77}$$

where the real block $\frac{\partial \ln}{\partial \mathcal{S}}\big|_{\text{real}}$ is the is the ordinary Jacobian from (76) and the dual block $\frac{\partial \ln}{\partial \mathcal{S}}\big|_{\text{dual}}$ is as follows:

$$\frac{\partial \ln}{\partial \mathcal{S}}\bigg|_{\text{dual}} = \begin{bmatrix} f_{x,x}+\tau & f_{x,y} & f_{x,z} & -g_x \\ f_{y,x} & f_{y,y}+\tau & f_{y,z} & -g_y \\ f_{z,x} & f_{z,y} & f_{z,z}+\tau & -g_z \\ g_x & g_y & g_z & g_w \end{bmatrix}$$

where

$$\begin{aligned} f_{a,b} &= (\hbar_a d_b + \hbar_b d_a)\,\zeta \\ &\quad + \hbar_a \hbar_b \left(\gamma\mu + \frac{2 d_w}{|\hbar|^4}\right) \\ g_a &= \frac{d_a}{|\hbar|^2} - \hbar_a \frac{2(\gamma + \hbar_w d_w)}{|\hbar|^4} \\ \gamma &= \vec{\hbar}_v \cdot \vec{d}_v \\ \tau &= \gamma\zeta - \frac{d_w}{|\hbar|^2} \\ \mu &= \frac{\frac{-2\hbar_w}{|\hbar|^4} - 3\zeta}{|\hbar_v|^2} = \frac{-2cs^3 - 3cs + 3\phi}{s^5\,|\hbar|^5} \\ &= \frac{\frac{8}{5} + \frac{4\phi^2}{7} + \frac{\phi^4}{7} + \dots}{|\hbar|^5}\,. \end{aligned} \tag{78}$$

We note that the direct symbolic partial differentiation of (43) yields elements in the w column of $\frac{\partial \ln}{\partial \mathcal{S}}\big|_{\text{dual}}$ of the form,

$$\ell_a = -\frac{d_a}{|\hbar|^2}\xi_1 + \hbar_a \frac{2\gamma}{|\hbar|^4}\xi_2 + \hbar_a \frac{2\hbar_w d_w}{|\hbar|^4} = -g_a \tag{79}$$

where

$$\begin{aligned} \xi_1 &= \frac{|\hbar|^2 - \hbar_w^2}{|\hbar_v|^2} = \frac{1-c^2}{s^2} = 1 \\ \xi_2 &= -\frac{\hbar_w^2}{|\hbar_v|^2} - \frac{|\hbar|^2\,\hbar_w^2}{2\,|\hbar_v|^4} + \frac{|\hbar|^2}{2\,|\hbar_v|^2} + \frac{|\hbar|^4}{2\,|\hbar_v|^4} \\ &= -\frac{2c^2s^2 - s^2 + c^2 - 1}{2s^4} = 1 \end{aligned} \tag{80}$$

Since the factors ξ_1 and ξ_2 thus simplify to 1, the entries in this column are equal to $-g_a$.

Now that we have found all the partial derivatives in (74), we simplify to evaluate more efficiently. First, we substitute for $\nabla \operatorname{ssq}$ and $\partial\mathcal{S}/\partial\boldsymbol{\theta}$. Then, we reorder the multiplication to a sequence of matrix-vector products from right-to-left, and finally simplify the multiplication by $\frac{\partial \text{conj}}{\partial \mathcal{S}}$ to a conjugation:

$$\begin{aligned} \nabla f &= 2\Omega_{\text{rel}}^T \left(\frac{\partial \ln}{\partial \mathcal{S}}\right) [\mathcal{S}_{\text{ref}}]_R \left(\frac{\partial \text{conj}}{\partial \mathcal{S}}\right) \left(\frac{1}{2}\,[\mathcal{S}_{\text{act}}]_R\,\mathbf{J}_\Omega\right) \\ &= \left(\mathbf{J}_\Omega^T\,[\mathcal{S}_{\text{act}}]_R^T \left(\frac{\partial \text{conj}}{\partial \mathcal{S}}\right)^T [\mathcal{S}_{\text{ref}}]_R^T \left(\frac{\partial \ln}{\partial \mathcal{S}}\right)^T \Omega_{\text{rel}}\right)^T \\ &= \left(\mathbf{J}_\Omega^T\,[\mathcal{S}_{\text{act}}]_R^T \left([\mathcal{S}_{\text{ref}}]_R^T \left(\frac{\partial \ln}{\partial \mathcal{S}}\right)^T \Omega_{\text{rel}}\right)^*\right)^T . \end{aligned} \tag{81}$$

where $\Omega_{\text{rel}} = \ln(\mathcal{S}_{\text{act}}^* \otimes \mathcal{S}_{\text{ref}})$.

Rotation Logarithm and Translation Objective Next, we derive the analytic gradient for an objective function that separates the rotational and translational parts. We restate (4) from Beeson and Ames (2015) in terms of rotation quaternions and translation vectors:

$$\begin{aligned} f(\boldsymbol{\theta}) &= f_r(\boldsymbol{\theta}) + f_v(\boldsymbol{\theta}) = |\omega_{\text{rel}}|^2 + |\vec{v}_{\text{rel}}|^2 \\ &= |\ln(\hbar_{\text{act}}^*(\boldsymbol{\theta}) \otimes \hbar_{\text{ref}})|^2 + |\vec{v}_{\text{act}}(\boldsymbol{\theta}) - \vec{v}_{\text{ref}}|^2\,. \end{aligned} \tag{82}$$

The gradient of (82) is the sum of gradients of the rotational part f_r and translational part f_v:

$$\nabla f(\boldsymbol{\theta}) = \nabla f_r(\boldsymbol{\theta}) + \nabla f_v(\boldsymbol{\theta})\,. \tag{83}$$

The derivation of rotational gradient ∇f_r is equivalent to our gradient derivation for the previous logarithmic objective function (81), but now we need only ordinary quaternions for rotation instead of dual quaternions. The result is equivalent in structure to (81):

$$\nabla f_r(\boldsymbol{\theta}) = \left(\mathbf{J}_r^T\,[\hbar_{\text{act}}]_R^T \left([\hbar_{\text{ref}}]_R^T \left(\frac{\partial \ln}{\partial \hbar}\right)^T \omega_{\text{rel}}\right)^*\right)^T , \tag{84}$$

where $\mathbf{J}_r$ is the rotational block of the manipulator Jacobian and $\omega_{\text{rel}} = \ln(\hbar_{\text{act}}^* \otimes \hbar_{\text{ref}})$.

To find the translational gradient ∇f_v, we first modify our notation and then differentiate:

$$f_v(\boldsymbol{\theta}) = \operatorname{ssd}(\vec{v}_{\text{act}}(\boldsymbol{\theta})) \tag{85}$$

$$\nabla f_v(\boldsymbol{\theta}) = (\nabla \operatorname{ssd}) \left(\frac{\partial \vec{v}_{\text{act}}}{\partial \boldsymbol{\theta}}\right)\,. \tag{86}$$

The partial derivatives in (86) are then,

$$\begin{aligned} \nabla \operatorname{ssd} &= 2\,(\vec{v}_{\text{act}} - \vec{v}_{\text{ref}})^T \\ \frac{\partial \vec{v}_{\text{act}}}{\partial \boldsymbol{\theta}} &= \mathbf{J}_v\,, \end{aligned} \tag{87}$$

where $\mathbf{J}_v$ is the translational part of the velocity Jacobian $\mathbf{J}_x$ in (63), which may be computed from (65).

We again reorder the gradient to a the matrix-vector product,

$$\begin{aligned}\nabla f_v(\boldsymbol{\theta}) &= 2\left(\vec{v}_{\text{act}} - \vec{v}_{\text{ref}}\right)^T \mathbf{J}_v \\ &= 2\left(\mathbf{J}_v^T\left(\vec{v}_{\text{act}} - \vec{v}_{\text{ref}}\right)\right)^T . \end{aligned} \tag{88}$$

We now have the full gradient of the objective function (82),

$$\begin{aligned}\nabla f(\boldsymbol{\theta}) = &\overbrace{\left(\mathbf{J}_r^T\left[\hbar_{\text{act}}\right]_R^T\left(\left[\hbar_{\text{ref}}\right]_R^T\left(\frac{\partial \ln}{\partial \hbar}\right)^T \omega_{\text{rel}}\right)^*\right)^T}^{\text{Rotational: } \nabla f_r} \\ &+ \underbrace{2\left(\mathbf{J}_v^T\left(\vec{v}_{\text{act}} - \vec{v}_{\text{ref}}\right)\right)^T}_{\text{Translational: } \nabla f_v} . \end{aligned} \tag{89}$$

Inverse Kinematics Results We evaluate the analytic gradients (81) and (89) within an SQP and compare against our implementation of the benchmark finite difference gradient. For this evaluation, we produced a set of feasible inverse kinematics problems by sampling random configurations using (59) and then computing the forward kinematics for each configuration to give the reference pose for the inverse kinematics problem. Table 9 shows the results for 20,000 such inverse kinematics problems. Avoiding the repeated computation of forward kinematics required by the finite difference gradient improves the efficiency of the SQP for inverse position kinematics and yields both an empirical 300-500% speedup and improved robustness under fixed time budgets. The separated rotation and translation objective (82) performs better, which is consistent with results in Beeson and Ames (2015). The performance difference is most pronounced for time-limited random restarts using the analytic gradient (89). The analytic logarithm gradient (81) requires 8×8 matrix-vector products while the analytic rotation and translation gradient (89) requires only 4×4 matrix-vector products.

The variation in time to solve inverse kinematics for difference reference poses is significant as shown by the standard deviations in Table 9. Different reference poses or initial seeds may change the number of required iterations to solve the SQP. Moreover, the optimization may reach a local minima that is not at the reference pose, thus requiring additional random restarts to solve. The presented analytic gradients partially mitigate this issue by solving more instances within a given time limit, but optimization-based approaches in general vary in required iterations and need restarts and timeouts to address local minima (Goldenberg et al. 1985; Kumar et al. 2010; Beeson and Ames 2015).

6 Discussion

We often have the choice of a matrix or quaternion form for any particular application; both produce a mathematically-equivalent result, but the computational efficiency differs. For example, interpolation is commonly regarded as a key application area for quaternions; however, we can achieve the same result—at greater computational cost—using rotation matrices. Spherical linear interpolation (SLERP) (Shoemake 1985) interpolates from an initial to final orientation with constant rotational axis and linearly-varying angle. The algebraic form of SLERP (Dam et al. 1998) has a direct matrix equivalent:

$$\hbar(\tau) = \hbar(0) \otimes \exp\left(\tau \ln\left(\left(\hbar(0)\right)^* \otimes \hbar(1)\right)\right) \tag{90}$$

$$\mathbf{R}(\tau) = \mathbf{R}(0)\ \exp\left(\tau \ln\left(\left(\mathbf{R}(0)\right)^{-1}\mathbf{R}(1)\right)\right) , \tag{91}$$

where $\hbar(0), \mathbf{R}(0)$ is the initial orientation and $\hbar(1), \mathbf{R}(1)$ is the final orientation. Both (90) and (91) equivalently interpolate orientation. However, the quaternion form (90) is more efficient to compute than the matrix form, and the more commonly used geometric simplification of (90) is even more efficient (Shoemake 1985).

Similarly, the logarithm objective function (72) for inverse position kinematics has a corresponding matrix form:

$$\begin{aligned} f_{\text{quat}}(\boldsymbol{\theta}) &= \left(\ln\left(\mathcal{S}_{\text{act}}^*(\boldsymbol{\theta}) \otimes \mathcal{S}_{\text{ref}}\right)\right)^T \left(\ln\left(\mathcal{S}_{\text{act}}^*(\boldsymbol{\theta}) \otimes \mathcal{S}_{\text{ref}}\right)\right) \\ f_{\text{mat}}(\boldsymbol{\theta}) &= \left(\ln\left(\mathbf{T}_{\text{act}}^{-1}(\boldsymbol{\theta})\mathbf{T}_{\text{ref}}\right)\right)^T \left(\ln\left(\mathbf{T}_{\text{act}}^{-1}(\boldsymbol{\theta})\mathbf{T}_{\text{ref}}\right)\right) . \end{aligned} \tag{92}$$

We define the separated rotation and translation objective function (82) in terms of quaternions whereas the implementation in TRAK-IK uses matrices.

Ordinary and dual quaternions also provide computational advantages for the blending or averaging of rotations and transformations (Kavan et al. 2008; Markley and Mortari 2000), which was described by Wahba (1965) as optimal rotation based on a set of weighted observations.

Floating point operations for both matrices and quaternions may benefit from parallelism though *Single Instruction, Multiple Data* (SIMD) operations such as Intel AVX (Firasta et al. 2008) or Arm Neon. SIMD uses multiple processing elements in parallel—for example, to simultaneously add the elements of two vectors. Matrix-matrix and matrix-vector products both contain parallelizable multiplications and additions. Quaternion products are similarly paralelizable, as shown in the matrix form of the quaternion multiplication (12).

The efficiency gains from quaternion-based kinematics may generally support real-time computation in robot systems to the extent of reducing the required

		Robot	Finite Difference			Analytic Gradient			speedup
			mean(ms)	*std(ms)*	*solved(%)*	*mean(ms)*	*std(ms)*	*solved(%)*	
$\lvert\ln(\mathcal{S}^*_{\text{act}} \otimes \mathcal{S}_{\text{ref}})^\vee\rvert^2$	**Single**	baxter	0.686	0.350	82.75	0.153	0.072	82.77	4.472
		ur10	0.490	0.238	86.92	0.115	0.045	86.92	4.269
		jaco	0.411	0.204	74.05	0.125	0.060	74.05	3.294
		lwa4d	0.433	0.200	96.91	0.133	0.051	96.91	3.249
	5ms	baxter	0.974	0.957	98.72	0.236	0.318	99.93	4.133
		ur10	0.658	0.664	99.58	0.160	0.184	100.00	4.127
		jaco	0.754	0.931	98.38	0.259	0.480	99.69	2.912
		lwa4d	0.457	0.265	100.00	0.141	0.075	100.00	3.235
	50ms	baxter	1.040	1.506	99.98	0.236	0.343	100.00	4.401
		ur10	0.678	0.868	100.00	0.159	0.179	100.00	4.266
		jaco	0.871	1.732	100.00	0.271	0.614	100.00	3.208
		lwa4d	0.457	0.264	100.00	0.142	0.076	100.00	3.226
$\lvert\ln(\mathcal{H}^*_{\text{act}} \otimes \mathcal{H}_{\text{ref}})^\vee\rvert^2 + \lvert\vec{v}_{\text{act}} - \vec{v}_{\text{ref}}\rvert^2$	**Single**	baxter	0.622	0.338	84.58	0.115	0.058	84.58	5.389
		ur10	0.478	0.215	88.10	0.090	0.035	88.09	5.336
		jaco	0.397	0.158	79.19	0.091	0.038	79.19	4.376
		lwa4d	0.422	0.176	98.07	0.097	0.036	98.07	4.362
	5ms	baxter	0.868	0.870	99.08	0.172	0.225	99.98	5.063
		ur10	0.626	0.616	99.59	0.120	0.135	100.00	5.192
		jaco	0.652	0.767	99.11	0.163	0.281	99.94	4.013
		lwa4d	0.434	0.233	100.00	0.100	0.050	100.00	4.314
	50ms	baxter	0.901	1.158	100.00	0.173	0.256	100.00	5.223
		ur10	0.636	0.707	100.00	0.122	0.135	100.00	5.229
		jaco	0.698	1.308	100.00	0.164	0.317	100.00	4.263
		lwa4d	0.433	0.233	100.00	0.101	0.050	100.00	4.310

Table 9. Inverse kinematics results demonstrating a 300-500% speedup and improved robustness for a fixed time budget using the analytic gradients. The upper rows use objective function (72), and the lower rows use objective function (82). We test for 20,000 random, valid poses obtained by sampling configurations and computing the forward kinematics. The "Single" entries attempt each inverse position kinematics problem once from a seed at the joint center configuration; the "5ms" and "50ms" entries restart each inverse position kinematics problem from a random seed for up to the listed timeout.

amount of computation. However, the key issue in real-time computing is not overall throughput but *latency and predictability* (Lee 2009; Dantam et al. 2015, 2016). Thus, it is important to observe that optimization-based inverse position kinematics, both as presented in this work and as developed by others (Smits et al. 2011; Beeson and Ames 2015), is not *complete* in that it does not indicate when a desired pose is infeasible. Instead, such solvers run until reaching a local minima or with random restarts until a timeout. To bound computation for real-time motion, one may precompute a path outside the real-time loop—such as by using optimization-based inverse kinematics to sample goal configurations for a motion planner (Kuffner and LaValle 2000; Şucan et al. 2012)—and then track that path in real-time. The quaternion-based SQP formulation would improve the efficiency of such goal sampling; however, the overall requirement remains to bound computation in real-time systems.

Generally, the matrix and quaternion representations of rotation and Euclidean transformation share group structure. Just as the rotation and transformation matrices form Lie groups with associated Lie algebras based on the exponential, so too do the ordinary and dual quaternions form Lie groups and associated algebras. We can map every quaternion representation to matrix equivalent. Specifically, there is a surjective homomorphism (double-cover) from the ordinary unit quaternions to the special orthogonal group $\mathcal{SO}(3)$ of rotation matrices. Similarly, we have a surjective homomorphism from the dual unit quaternions to the special Euclidean group $\mathcal{SE}(3)$ of homogeneous transformation matrices.

The results we have presented continue the broader developments of methods based on ordinary and dual quaternions which offer computational advantages over their matrix counterparts. The quaternion methods we have presented achieve mathematically-equivalent results, but are more compact and efficient, than the matrices.

7 Conclusion

We have presented new derivations of the dual quaternion exponential, logarithm, and derivatives which handle the small-angle singularity and enable robust use of dual quaternions for robot kinematics.

By extending our singularity-robust exponential and logarithm to the implicit representation of dual quaternions as an ordinary quaternion and translation vector, we demonstrate a 30%-40% performance improvement in forward kinematics over the conventional homogeneous transformation matrices. Applying our quaternion analysis to inverse position kinematics formulated as sequential quadratic programming yields a 300-500% performance improvement. Our implementation is available as open source code[‡]. These results show that dual quaternions—though equivalent in representational capability to transformation matrices—offer distinct computational advantages.

While matrices are a widely-used representation for Euclidean transformations, the quaternion forms are both more compact and—for most cases—require fewer arithmetic instructions. In the one case where matrices have an efficiency advantage—transforming large numbers of points—it may still be more efficient to chain transformations via quaternions and then convert the final transform to a matrix to apply to the point set. We hope these derivations of singularity-free exponentials and logarithms for the quaternion forms of transformations, along with our demonstrations of forward and inverse kinematics, will enable widespread use of these more efficient representations.

Funding

This work was supported in part by the Army Research Laboratory Distributed and Collaborative Intelligent Systems and Technology Collaborative Research Alliance [W911NF-17-2-0181]; and the National Science Foundation [CNS-1823245].

A Construction from Denavit-Hartenberg Parameters

Though recent works have favored the product of exponentials formulation (Lynch and Park 2017), Denavit-Hartenberg (DH) parameters (Hartenberg and Denavit 1964) continue as a potential approach to specify robot kinematics. Radavelli, et al. present a construction of dual quaternions from DH parameters in the distal convention (Radavelli et al. 2014). We relate DH parameters in both distal and proximal conventions to implicit dual quaternions, using the double angle identities to remove extra computations of sin and cos.

The proximal DH convention (Lynch and Park 2017) defines a rotation and translation about $\hat{\boldsymbol{\imath}}$ followed by a rotation and translation about $\hat{\boldsymbol{k}}$,

$$\{\alpha, a, \phi, d\}_{\text{prox}} \rightsquigarrow \text{Rot}(\hat{\boldsymbol{\imath}}, \alpha) * \text{Trans}(\hat{\boldsymbol{\imath}}, a) * \text{Rot}(\hat{\boldsymbol{k}}, \phi) * \text{Trans}(\hat{\boldsymbol{k}}, d) \ .$$

The distal DH convention (Murray 1994) defines a rotation and translation about $\hat{\boldsymbol{k}}$ followed by a rotation and translation about $\hat{\boldsymbol{\imath}}$,

$$\{\phi, d, \alpha, a\}_{\text{dist}} \rightsquigarrow \text{Rot}(\hat{\boldsymbol{k}}, \phi) * \text{Trans}(\hat{\boldsymbol{k}}, d) * \text{Rot}(\hat{\boldsymbol{\imath}}, \alpha) * \text{Trans}(\hat{\boldsymbol{\imath}}, a) \ .$$

Table 10 compares the construction from DH parameters for matrices, explicit, and implicit dual quaternions.

References

Altmann SL (1989) Hamilton, Rodrigues, and the quaternion scandal. *Mathematics Magazine* 62(5): 291–308.

Beeson P and Ames B (2015) TRAC-IK: An open-source library for improved solving of generic inverse kinematics. In: *International Conference on Humanoid Robots (Humanoids)*. IEEE, pp. 928–935.

Brockett RW (1984) Robotic manipulators and the product of exponentials formula. In: *Mathematical theory of networks and systems*. Springer, pp. 120–129.

Buss SR (2004) Introduction to inverse kinematics with Jacobian transpose, pseudoinverse and damped least squares methods. Technical report, Department of Mathematics, University of California, San Diego. http://www.math.ucsd.edu/~sbuss/ResearchWeb/ikmethods/iksurvey.pdf.

Buss SR and Kim JS (2005) Selectively damped least squares for inverse kinematics. *Journal of Graphics tools* 10(3): 37–49.

Chevallier D (1991) Lie algebras, modules, dual quaternions and algebraic methods in kinematics. *Mechanism and Machine Theory* 26(6): 613–627.

Şucan IA, Moll M and Kavraki LE (2012) The open motion planning library. *Robotics & Automation Magazine (RAM)* 19(4): 72–82.

Dam EB, Koch M and Lillholm M (1998) Quaternions, interpolation and animation. Technical Report DIKU-TR-98/5, Datalogisk Institut, Københavns Universitet Copenhagen.

Dantam NT (2018) Practical exponential coordinates using implicit dual quaternions. In: *Workshop on the Algorithmic Foundations of Robotics*.

[‡]Software available at http://amino.dyalab.org

Representation		Form	*Mul.*	*Add*	*Other*
Proximal	*Tf. Matrix*	$\begin{bmatrix} c_\phi & -s_\phi & 0 & a \\ s_\phi c_\alpha & c_\phi c_\alpha & -s_\alpha & -d s_\alpha \\ s_\phi s_\alpha & c_\phi s_\alpha & c_\alpha & d c_\alpha \\ 0 & 0 & 0 & 1 \end{bmatrix}$	6	0	sincos(2)
	Dual Quat.	$\begin{array}{l} s_{\frac{\alpha}{2}} c_{\frac{\phi}{2}} \hat{\boldsymbol{i}} - s_{\frac{\alpha}{2}} s_{\frac{\phi}{2}} \hat{\boldsymbol{j}} + c_{\frac{\alpha}{2}} s_{\frac{\phi}{2}} \hat{\boldsymbol{k}} + c_{\frac{\alpha}{2}} c_{\frac{\phi}{2}} \\ + \quad ((\frac{a}{2} h_w + \frac{d}{2} h_y)\hat{\boldsymbol{i}} + (-\frac{a}{2} h_z - \frac{d}{2} h_x)\hat{\boldsymbol{j}} \\ +(\frac{a}{2} h_y + \frac{d}{2} h_w)\hat{\boldsymbol{k}} + (-\frac{a}{2} h_x - \frac{d}{2} h_z))\boldsymbol{\varepsilon} \end{array}$	12	4	sincos(2)
	Quat.-Trans.	$\begin{pmatrix} s_{\frac{\alpha}{2}} c_{\frac{\phi}{2}} \hat{\boldsymbol{i}} - s_{\frac{\alpha}{2}} s_{\frac{\phi}{2}} \hat{\boldsymbol{j}} + c_{\frac{\alpha}{2}} s_{\frac{\phi}{2}} \hat{\boldsymbol{k}} + c_{\frac{\alpha}{2}} c_{\frac{\phi}{2}} \\ a\hat{\boldsymbol{i}} - 2d s_{\frac{\alpha}{2}} c_{\frac{\alpha}{2}} \hat{\boldsymbol{j}} + d\left(c_{\frac{\alpha}{2}}^2 - s_{\frac{\alpha}{2}}^2\right)\hat{\boldsymbol{k}} \end{pmatrix}$	8	2	sincos(2)
Distal	*Tf. Matrix*	$\begin{bmatrix} c_\phi & -s_\phi c_\alpha & s_\phi s_\alpha & a c_\phi \\ s_\phi & c_\phi c_\alpha & -c_\phi s_\alpha & a s_\phi \\ 0 & s_\alpha & c_\alpha & d \\ 0 & 0 & 0 & 1 \end{bmatrix}$	6	0	sincos(2)
	Dual Quat.	$\begin{array}{l} s_{\frac{\alpha}{2}} c_{\frac{\phi}{2}} \hat{\boldsymbol{i}} + s_{\frac{\alpha}{2}} s_{\frac{\phi}{2}} \hat{\boldsymbol{j}} + c_{\frac{\alpha}{2}} s_{\frac{\phi}{2}} \hat{\boldsymbol{k}} + c_{\frac{\alpha}{2}} c_{\frac{\phi}{2}} \\ + \quad ((\frac{a}{2} h_w - \frac{d}{2} h_y)\hat{\boldsymbol{i}} + (\frac{a}{2} h_z + \frac{d}{2} h_x)\hat{\boldsymbol{j}} \\ +(-\frac{a}{2} h_y + \frac{d}{2} h_w)\hat{\boldsymbol{k}} + (-\frac{a}{2} h_x - \frac{d}{2} h_z))\boldsymbol{\varepsilon} \end{array}$	12	4	sincos(2)
	Quat.-Trans.	$\begin{pmatrix} s_{\frac{\alpha}{2}} c_{\frac{\phi}{2}} \hat{\boldsymbol{i}} + s_{\frac{\alpha}{2}} s_{\frac{\phi}{2}} \hat{\boldsymbol{j}} + c_{\frac{\alpha}{2}} s_{\frac{\phi}{2}} \hat{\boldsymbol{k}} + c_{\frac{\alpha}{2}} c_{\frac{\phi}{2}} \\ a\left(c_{\frac{\phi}{2}}^2 - s_{\frac{\phi}{2}}^2\right)\hat{\boldsymbol{i}} + 2a s_{\frac{\phi}{2}} c_{\frac{\phi}{2}} \hat{\boldsymbol{j}} + d\hat{\boldsymbol{k}} \end{pmatrix}$	8	2	sincos(2)

Table 10. Construction from Denavit-Hartenberg Parameters

Dantam NT, Amor HB, Christensen H and Stilman M (2014a) Online camera registration for robot manipulation. In: *International Symposium on Experimental Robotics*. Springer, pp. 179–194.

Dantam NT, Ben Amor H, Christensen H and Stilman M (2014b) Online multi-camera registration for bimanual workspace trajectories. In: *International Conference on Humanoid Robots*. IEEE, pp. 588–593.

Dantam NT, Bøndergaard K, Johansson MA, Furuholm T and Kavraki LE (2016) Unix philosophy and the real world: Control software for humanoid robots. *Frontiers in Robotics and Artificial Intelligence, Research Topic on Software Architectures for Humanoid Robotics* 3.

Dantam NT, Lofaro DM, Hereid A, Oh P, Ames A and Stilman M (2015) The ach ipc library. *Robotics and Automation Magazine* 22(1): 76–85.

Fallon M, Kuindersma S, Karumanchi S, Antone M, Schneider T, Dai H, D'Arpino CP, Deits R, DiCicco M, Fourie D, Koolen T, Marion P, Posa M, Valenzuela A, Yu KT, Shah J, Iagnemma K, Tedrake R and Teller S (2015) An architecture for online affordance-based perception and whole-body planning. *Journal of Field Robotics* 32(2): 229–254.

Feng S, Whitman E, Xinjilefu X and Atkeson CG (2015) Optimization-based full body control for the DARPA robotics challenge. *Journal of Field Robotics* 32(2): 293–312.

Firasta N, Buxton M, Jinbo P, Nasri K and Kuo S (2008) Intel avx: New frontiers in performance improvements and energy efficiency. Technical report, Intel.

Funda J and Paul RP (1990) A computational analysis of screw transformations in robotics. *Transactions on Robotics and Automation* 6(3): 348–356.

Gibbs JW (1884) *Elements of vector analysis: arranged for the use of students in physics*. Tuttle, Morehouse & Taylor.

Goldenberg A, Benhabib B and Fenton R (1985) A complete generalized solution to the inverse kinematics of robots. *IEEE Journal on Robotics and Automation* 1(1): 14–20.

Grassia FS (1998) Practical parameterization of rotations using the exponential map. *Journal of graphics tools* 3(3): 29–48.

Hamilton WR (1866) *Elements of quaternions*. Longmans, Green, & Company.

Han DP, Wei Q and Li ZX (2008) Kinematic control of free rigid bodies using dual quaternions. *International Journal of Automation and Computing* 5(3): 319–324.

Hartenberg RS and Denavit J (1964) *Kinematic synthesis of linkages*. McGraw-Hill.

IEEE and The Open Group (2018) *IEEE Std 1003.1-2017, Standard for Information Technology – Portable Operating System Interface (POSIX)*. https://pubs.

opengroup.org/onlinepubs/9699919799/.

ISO/IEC JTC 1, Information Technology (2011) *Information technology – Microprocessor Systems – Floating-Point arithmetic*. IEEE. https://www.iso.org/standard/57469.html.

Johnson SG (2019) The NLopt nonlinear-optimization package. http://github.com/stevengj/nlopt.

Kavan L, Collins S, Žára J and O'Sullivan C (2008) Geometric skinning with approximate dual quaternion blending. *ACM Transactions on Graphics (TOG)* 27(4): 105.

Kenwright B (2012) A beginners guide to dual-quaternions: What they are, how they work, and how to use them for 3D character hierarchies. In: *International Conference on Computer Graphics, Visualization and Computer Vision*. WSCG, pp. 1–10.

Kingston ZK, Dantam NT and Kavraki LE (2015) Kinematically constrained workspace control via linear optimization. In: *International Conference on Humanoid Robots*. IEEE, pp. 758–764.

Kraft D (1988) A software package for sequential quadratic programming. Technical Report DFVLR-FB 88-28, Institut für Dynamik der Flugsysteme, Oberpfaffenhofen.

Kraft D (1994) Algorithm 733: TOMP–fortran modules for optimal control calculations. *Transactions on Mathematical Software (TOMS)* 20(3): 262–281.

Kuffner JJ and LaValle SM (2000) RRT-connect: An efficient approach to single-query path planning. In: *International Conference on Robotics and Automation*, volume 2. IEEE, pp. 995–1001.

Kumar S, Sukavanam N and Balasubramanian R (2010) An optimization approach to solve the inverse kinematics of redundant manipulator. *International Journal of Information and System Sciences (Institute for Scientific Computing and Information)* 6(4): 414–423.

LaViola JJ (2003) A comparison of unscented and extended Kalman filtering for estimating quaternion motion. In: *American Control Conference*, volume 3. IEEE, pp. 2435–2440.

Lee EA (2009) Computing needs time. *Commununications of the ACM* 52(5): 70–79. DOI:10.1145/1506409.1506426. URL http://doi.acm.org/10.1145/1506409.1506426.

Loosemore S, Stallman RM, McGrath R, Oram A, and Drepper U (2020) *The GNU C Library Reference Manual*. The GNU Project, 2.31 edition. https://www.gnu.org/software/libc/manual/.

Lynch KM and Park FC (2017) *Modern Robotics: Mechanics, Planning, and Control*. Cambridge University Press.

Markley FL and Mortari D (2000) Quaternion attitude estimation using vector observations. *Journal of the Astronautical Sciences* 48(2): 359–380.

Murray RM (1994) *A mathematical introduction to robotic manipulation*. CRC press.

Nakamura Y and Hanafusa H (1986) Inverse kinematic solutions with singularity robustness for robot manipulator control. *Journal of Dynamic Systems, Measurement, and Control* 108: 163–171.

Özgür E and Mezouar Y (2016) Kinematic modeling and control of a robot arm using unit dual quaternions. *Robotics and Autonomous Systems* 77: 66–73.

Radavelli LA, De Pieri ER, Martins D and Simoni R (2014) A screw dual quaternion operator for serial robot kinematics.

Rakita D, Mutlu B and Gleicher M (2018) RelaxedIK: Real-time synthesis of accurate and feasible robot arm motion. In: *Robotics: Science and Systems*. DOI: 10.15607/RSS.2018.XIV.043.

Selig JM (2004) *Geometric fundamentals of robotics*. Springer Science & Business Media.

Selig JM (2010) Exponential and Cayley maps for dual quaternions. *Advances in applied Clifford algebras* 20(3-4): 923–936.

Shoemake K (1985) Animating rotation with quaternion curves. In: *SIGGRAPH computer graphics*, volume 19. ACM, pp. 245–254.

Smits R, Bruyninckx H and Aertbeliën E (2011) KDL: Kinematics and dynamics library. http://www.orocos.org/kdl.

Srivatsan RA, Rosen GT, Mohamed DFN and Choset H (2016) Estimating SE(3) elements using a dual quaternion based linear Kalman filter. In: *Robotics: Science and Systems*.

Study E (1903) *Geometrie der Dynamen*. Druck und Verlag von B. G. Teubner.

Study E (1913) Foundations and goals of analytical kinematics. In: *Berlin Mathematical Society*. Translated by D. H. Delphenich.

Valverde A and Tsiotras P (2018) Spacecraft robot kinematics using dual quaternions. *Robotics* 7(4): 64.

Wahba G (1965) A least squares estimate of satellite attitude. *SIAM review* 7(3): 409–409.

Wampler CW (1986) Manipulator inverse kinematic solutions based on vector formulations and damped least-squares methods. *Transactions on Systems, Man, and Cybernetics* 16(1): 93–101.

Wang X, Han Dp, Yu C and Zheng Z (2012) The geometric structure of unit dual quaternion with application in kinematic control. *Journal of Mathematical Analysis and Applications* 389(2): 1352–1364.

Wang X and Zhu H (2014) On the comparisons of unit dual quaternion and homogeneous transformation matrix. *Advances in Applied Clifford Algebras* 24(1): 213–229.

Yang AT and Freudenstein F (1964) Application of dual-number quaternion algebra to the analysis of spatial mechanisms. *Journal of Applied Mechanics* 31(2): 300–308.

Errata

- Equation 3: Corrected second element from $r_{13} - r_{30}$ to $r_{13} - r_{31}$.
- Equation 27: corrected real part from ${}^a\mathcal{H}_b \otimes {}^a\mathcal{H}_b$ to ${}^a\mathcal{H}_b \otimes {}^b\mathcal{H}_c$.