Quaternion Differentiation and Interpolation

Let us consider two quaternions

$$p = (p_0, p_1, p_2, p_3) = (\underset{\substack{\uparrow \\ \text{Scalar} \\ \text{Part}}}{p_0}, \underset{\substack{\uparrow \\ \text{Vector} \\ \text{Part}}}{\vec{p}})$$

$$\text{where } \vec{p} = \begin{pmatrix} p_1 \\ p_2 \\ p_3 \end{pmatrix},$$

$$\text{and } q = (q_0, q_1, q_2, q_3) = (q_0, \vec{q})$$

$$\text{where } \vec{q} = \begin{pmatrix} q_1 \\ q_2 \\ q_3 \end{pmatrix}$$

Then we know the following:

(1) Quaternion Multiplication

$$pq = \underbrace{(p_0 q_0 - \vec{p}\cdot\vec{q})}_{\text{Scalar Part}} + \underbrace{p_0\vec{q} + q_0\vec{p} + \vec{p}\times\vec{q}}_{\text{Vector Part}}$$

(2) Conjugate of a Quaternion is

$$q^* = (q_0, -q)$$

$$\text{and } (pq)^* = q^* p^*$$

(3) We call a quaternion q, a unit quaternion if $|q| = 1$, i.e., $q_0^2 + q_1^2 + q_2^2 + q_3^2 = 1$

For a unit quaternion $q^{-1} = q^*$

i.e. Inverse of a unit quaternion is its conjugate.

A unit quaternion can also be written as

$$q = \cos\theta + \hat{u}\sin\theta = q_0 + \vec{q}$$

where $q_0 = \cos\theta$; $\hat{u} = \dfrac{\vec{q}}{\|\vec{q}\|}$; $\sin\theta = \|\vec{q}\|$

Note: $|q|$ denotes magnitude of a quaternion q.
$\|\vec{q}\|$ denotes the norm of the vector part of the quaternion.

Power of a Quaternion:

For a general quaternion

$$q = |q|\, e^{\hat{u}\theta}$$

where $e^{\hat{u}\theta} = \cos\theta + \hat{u}\sin\theta$

$$\therefore q^{\rho} = |q|^{\rho} (e^{\hat{u}\theta})^{\rho} = |q|^{\rho} e^{\hat{u}\rho\theta}$$

$$= |q|^{\rho}(\cos\rho\theta + \hat{u}\sin\rho\theta)$$, for any $\rho \in \mathbb{R}$.

For a unit quaternion, $|q| = 1$

$$\therefore q^{\rho} = \cos(\rho\theta) + \hat{u}\sin(\rho\theta)$$

Quaternion Differentiation:

Let $q = q_0 + \vec{q}$ be a quaternion which is a function of time, t.

$$\therefore \dot{q} = \dot{q}_0 + \dot{\vec{q}} = \dot{q}_0 + \dot{q}_1\hat{i} + \dot{q}_2\hat{j} + \dot{q}_3\hat{k}$$

Similarly, for the product of two quaternions we can use the chain rule

$$\frac{d}{dt}(pq) = \dot{p}q + q\dot{p}$$

Note that the product on the right hand side of $\frac{d}{dt}(pq)$ is a quaternion multiplication.

When $q(t)$ is a unit quaternion, the differentiation is a bit more complicated. Without showing the derivation, we can write

$$\boxed{\dot{q}(t) = \frac{1}{2}\omega^s q = \frac{1}{2} q\,\omega^b}$$

where $\omega^s \leftarrow$ Spatial Angular velocity of a rigid body

$\omega^b \leftarrow$ Body Angular velocity of a rigid body.

or

$$\boxed{\begin{aligned} \omega^s &= 2\dot{q}\,q^* \\ \omega^b &= 2q^*\dot{q} \end{aligned}}$$

In matrix form we have

$$\omega^s = 2\begin{bmatrix} -q_1 & q_0 & -q_3 & q_2 \\ -q_2 & q_3 & q_0 & -q_1 \\ -q_3 & -q_2 & q_1 & q_0 \end{bmatrix}\begin{bmatrix} \dot{q}_0 \\ \dot{q}_1 \\ \dot{q}_2 \\ \dot{q}_3 \end{bmatrix}$$

$$= 2J_1\dot{q}$$

$$\omega^b = 2\begin{bmatrix} -q_1 & q_0 & q_3 & -q_2 \\ -q_2 & -q_3 & q_0 & q_1 \\ -q_3 & q_2 & -q_1 & q_0 \end{bmatrix}\begin{bmatrix} \dot{q}_0 \\ \dot{q}_1 \\ \dot{q}_2 \\ \dot{q}_3 \end{bmatrix}$$

Spherical Linear Interpolation:

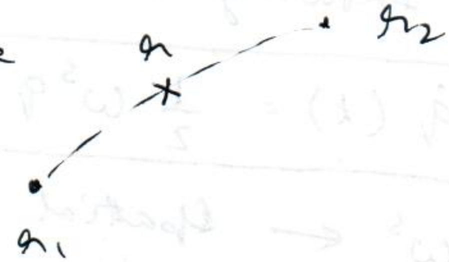
(or Interpolation between two quaternions)

Let $q_1 = \cos\theta_1 + \hat{u}_1 \sin\theta_1$

and $q_2 = \cos\theta_2 + \hat{u}_2 \sin\theta_2$

be two quaternions

Let $\tau \in [0, 1]$ be the interpolation parameter.



∴ An interpolant

$$q(\tau) = q_1 (q_1^* q_2)^\tau$$

Note: All multiplications on the right hand side are quaternion multiplications.

Dual Numbers:

$$\hat{D} = a + \epsilon b , \quad a, b \in \mathbb{R} ,$$

$\underset{\text{Real Part}}{a}$, $\underset{\text{Dual Part}}{b}$, $\quad \epsilon \neq 0, \quad \epsilon^2 = 0.$

More generally, a, b are elements of an algebraic field.

Let $\hat{D}_i = a_i + \epsilon b_i$

Addition:

$$D_1 + D_2 = (a_1 + a_2) + \epsilon (b_1 + b_2)$$

Multiplication:

$$D_1 \otimes D_2 = (a_1 + \epsilon b_1)(a_2 + \epsilon b_2)$$

$$= a_1 a_2 + \epsilon^2 \cancel{b_1 b_2}^{0} + \epsilon . a_1 b_2 + \epsilon b_1 a_2$$

$$= a_1 a_2 + \epsilon . (a_1 b_2 + a_2 b_1)$$

$$=$$

Inverse:

$$D^{-1} = \frac{1}{a}\left(1 - \epsilon \frac{b}{a}\right) \quad \text{assuming } a \neq 0$$

If $a = 0$, $\hat{D} = \epsilon b$ has no inverse.

Dual Vector:

$\vec{D} = (\hat{D}_1, \hat{D}_2, \hat{D}_3)$ ← Each component is a dual number

$$= \begin{pmatrix} a_1 + \epsilon b_1 \\ a_2 + \epsilon b_2 \\ a_3 + \epsilon b_3 \end{pmatrix} = \begin{pmatrix} a_1 \\ a_2 \\ a_3 \end{pmatrix} + \epsilon \begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix}$$

$$= \vec{a} + \epsilon \vec{b}$$

where $\vec{a} = \begin{pmatrix} a_1 \\ a_2 \\ a_3 \end{pmatrix}$ is a ~~vector~~, $\vec{a}, \vec{b}$ are real vectors.

Product of a dual number with a dual vector is

$$D \vec{D} = (D \otimes D_1, D \otimes D_2, D \otimes D_3)$$

$$\vec{D} . \vec{E} = D_1 \otimes E_1 + D_2 \otimes E_2 + D_3 \otimes E_3$$

$$\vec{D} \times \vec{E} = \begin{pmatrix} D_2 \otimes E_3 - D_3 \otimes E_2 \\ D_3 \otimes E_1 - D_1 \otimes E_3 \\ D_1 \otimes E_2 - D_2 \otimes E_1 \end{pmatrix}$$

Conjugate of Dual Numbers: $D = a + \epsilon b$, $D^* = a - \epsilon b$

Dual Quaternion:

$$A = p + \epsilon q \quad , \quad A = (p_0 + \epsilon q_0) + (p_1 + \epsilon q_1)\hat{i} + (p_2 + \epsilon q_2)\hat{j} + (p_3 + \epsilon q_3)\hat{k}$$

where p, q are quaternions.

$$= \hat{D} + \vec{\hat{D}} \leftarrow \text{Dual vector.}$$

$\hat{D}$: Dual Number

$$p = p_0 + \vec{p}$$

$$q = q_0 + \vec{q}$$

$\therefore$ A dual quaternion has 8 numbers.

Addition:

Let $A = p + \epsilon q$, $B = \mu + \epsilon v$

where p, q, μ, v are quaternions.

$$A + B = (p + \mu) + \epsilon(q + v)$$

Multiplication:

$$A \otimes B = (p + \epsilon q)(\mu + \epsilon v)$$

$$= p\mu + \cancel{\epsilon^2 q v}^{0} + \epsilon q \mu + \epsilon p v$$

$$= p\mu + \epsilon(q\mu + pv)$$

The multiplication on the RHS is quaternion multiplication. So you have to be careful about the order of multiplication

$$A \otimes B = (D_1 + \vec{D}_1)(D_2 + \vec{D}_2)$$

$$= \left(D_1 \otimes D_2 - \vec{D}_1 \cdot \vec{D}_2\right) + \left(D_1 \vec{D}_2 + D_2 \vec{D}_1 + \vec{D}_1 \times \vec{D}_2\right)$$

Conjugate of A:

$$A^* = p^* - \epsilon q^*$$

~~Conjugate of A^*~~

Unit Dual Quaternion:

Let $A = p + \epsilon q$ be a dual quaternion.

A is a unit dual quaternion if

$$A \otimes A^* = 1$$

$$\Rightarrow \quad p_0^2 + p_1^2 + p_2^2 + p_3^2 = 1$$

$$p_0 q_0 + p_1 q_1 + p_2 q_2 + p_3 q_3 = 0$$

i.e. (a) Real part p must be a unit quaternion.

(b) Real & Dual part must be orthogonal considering ~~them as~~ p & q as elements of $\mathbb{R}^4$.

Representation of Rigid Displacement with unit dual quaternions:

Given a transformation matrix $g = \begin{bmatrix} R & p \\ 0 & 1 \end{bmatrix}$

what is its ~~representation as~~ dual quaternion representation? Let $A = \alpha + \epsilon \beta$

(1) Convert R to a quaternion representation. Let p be the corresponding quaternion.

Then the real part of the dual quaternion becomes p.

$$\alpha = \cos\frac{\theta}{2} + \hat{u}\sin\frac{\theta}{2}, \qquad \hat{u} \leftarrow \text{Unit vector along axis of rotation.} \quad \theta \leftarrow \text{Angle of rotation.}$$

(2) $\beta = \frac{1}{2} p \alpha$

$\therefore$ The dual quaternion corresponding to g is

$$A = \alpha + \frac{\epsilon}{2} p\alpha \quad \text{Quaternion Product.}$$

where α is the unit quaternion representing rotation

Conversely, Given a ~~unit~~ dual quaternion representing a rigid body motion.

$$A = \alpha + \epsilon \beta$$

~~The Rota~~ $\alpha \leftarrow$ ~~Quater~~ Unit Quaternion representation of rotation.

$$\cancel{p = 2\alpha\beta}$$

$$\beta = \frac{1}{2} p \alpha$$

$$p = 2 \beta \alpha^*$$

Differentiation of a dual quaternion:

$$A = \alpha + \frac{\epsilon}{2} p \alpha$$

$$\therefore \dot{A} = \dot{\alpha} + \frac{\epsilon}{2} \left(\dot{p} \alpha + p \dot{\alpha} \right)$$

$$= \frac{1}{2} \omega^s \alpha + \frac{\epsilon}{2} \dot{p} \alpha + \frac{\epsilon}{4} p \omega^s \alpha$$

Interpolation:

$$A = \alpha + \frac{\epsilon}{2} p \alpha$$

$$B = \gamma + \frac{\epsilon}{2} p \gamma$$

To interpolate between A & B.

$$C(\tau) = A \otimes \left(A^* \otimes B \right)^{\tau}, \quad \tau \in [0,1]$$

This formula is analogous to the formula for ~~interpola~~ quaternion interpolation that we had seen earlier.

Motion Planning (Kinematic) for Redundant Manipulators

Let p denote the position of the end effector and α be the quaternion representing the orientation of the end effector.

Then we know that

$$\omega^s = 2 J_1 \dot{\alpha} \quad \text{(from Page 3 of lecture notes)}$$

where ω^s is the spatial angular velocity.

Also, the spatial velocity, v_s is

$$v^s = \dot{p} - \hat{\omega}^s p$$

$$\text{or } v^s = \dot{p} + \hat{p}\,\omega^s$$

$$\text{or } v^s = \dot{p} + 2\hat{p} J_1 \dot{\alpha}$$

$\therefore$ Writing the equations for v^s and ω^s in vector- ~~form~~ matrix form

$$\begin{bmatrix} v^s \\ \omega^s \end{bmatrix} = \begin{bmatrix} I_{3\times3} & 2\hat{p} J_1 \\ 0_{3\times3} & 2J_1 \end{bmatrix} \begin{bmatrix} \dot{p} \\ \dot{\alpha} \end{bmatrix}$$

$$\therefore \text{ let } \begin{bmatrix} I_{3\times3} & 2\hat{p} J_1 \\ 0_{3\times3} & 2J_1 \end{bmatrix} = \underbrace{J_2}_{6\times7}$$

Now $\begin{bmatrix} v^s \\ \omega^s \end{bmatrix} = J^s \dot{\theta}$

where $\dot{\theta} \leftarrow$ Vektor of Joint angle ~~values~~ rates

$J^s \leftarrow$ Spatial Jacobian

$$\therefore \dot{\theta} = (J^s)^T \left(J^s (J^s)^T\right)^{-1} \begin{bmatrix} v^s \\ \omega^s \end{bmatrix}$$

$$\text{or} \quad \underset{7\times 1}{\dot{\theta}} = \underbrace{(J^s)^T}_{7\times 6} \left(\underset{6\times 7}{J^s} \underbrace{(J^s)^T}_{7\times 6}\right)^{-1} \underset{6\times 7}{J_2} \underset{7\times 1}{\begin{bmatrix} \dot{p} \\ \dot{\alpha} \end{bmatrix}}$$

Let $\begin{bmatrix} p \\ \alpha \end{bmatrix} = \gamma$, $\therefore \begin{bmatrix} \dot{p} \\ \dot{\alpha} \end{bmatrix} = \dot{\gamma}$

$$\therefore \boxed{\dot{\theta} = B \dot{\gamma}} \qquad \text{—— (1)}$$

where $B = (J^s)^T \left(J^s (J^s)^T\right)^{-1} J_2$

Equation (1) can be used for both kinematics based motion planning and inverse kinematics (or redundancy resolution) for redundant manipulators.

Using a Euler time-step to discretize Equation (1), where h is a small time-step.

$$\frac{\theta(t+h) - \theta(t)}{\cancel{h}} = B(\theta(t)) \frac{\gamma(t+h) - \gamma(t)}{\cancel{h}}$$

$$\boxed{\begin{array}{l} \text{or} \quad \theta(t+h) = B(\theta(t)) \cdot \left(\gamma(t+h) - \gamma(t)\right) + \theta(t) \\ \text{where} \quad B = (J^s)^T \left(J^s (J^s)^T\right)^{-1} J_2 \end{array}}$$

Simple Motion Planning Algorithm

Input: Initial Configuration g_0 and final configuration g_f

~~1) Convert g_0 and g_f~~

Output: A sequence of joint angles $\theta(0), \theta(1), \ldots$, [Each $\theta(\tau)$ is a $n \times 1$ vector where n is the number of joints]

Step 1: Convert g_0 and g_f to a dual quaternion representation A_0 and A_f.

Step 2: Start with $A_s = A_0$ and A_f. Perform a dual quaternion interpolation between A_s and A_f and obtain the next configuration to A_s. Label it A_n.

Step 3: ~~Find cA Use~~ $\cancel{\gamma(\tau+h) = A_n, \ \gamma(\tau) = A_s}$ Find $\gamma(\tau+h)$ from A_n, $\gamma(\tau)$ from A_s, and use so as to approximate $\gamma(\tau)$

$$\theta(\tau+h) = \beta B\left(\gamma(\tau+h) - \gamma(\tau)\right) + \theta(\tau)$$

where $B = (J^s)^T (J^s J^{sT})^{-1} J_2$

~~and~~

[Note: Assume $\theta(\tau)$ is known for the initial configuration A_0 or it can be found ~~by simple~~ from Inverse Kinematics.]

~~Forward kinematics to find~~

~~Step 4: Use~~

β is a step size parameter, which is a constant. ~~These~~ You need to set this by trial and error.

Step 4: Use forward kinematics to form g_n corresponding to $\theta(\tau+h)$.
Convert g_n to a dual quaternion say $\bar{A}_n$.
~~and set $A_s = \bar{A}_n$.~~

Step 5: ~~Go to step~~ Check whether g_n is near enough to g_f. If not go back to step 2 with $A_s = \bar{A}_n$.
If converged then output the solution.

<u>Note 1</u>: If you are following a path, you can actually discretize the path and remove/modify the interpolation step.

<u>Note 2</u>: For inverse kinematics, you can choose any initial configuration where you assume the joint angles are known and apply the above algorithm. The IK solution is the final set of joint angles that you obtain.

<u>Note 3</u>: The above algorithm does not check for collision, so this is applicable in a carefully engineered environment. However collision avoidance and joint limit avoidance are possible using a variation of this algorithm.