Dual Quaternion Based Control

Akhil Sathuluri

ED14B037

1 Introduction

1.1 Why use quaternions?

The rotation matrices have inherent problems. The following are a few issues using the conventional methods of representation of a rigid body:

1. Well known reason of representation singularities using Euler Angles (Gimbal Lock).
2. The rotational part of the transformation matrix is orthogonal, but they drift and cause unwanted scaling and shearing the object of interest in a computer graphics scenario[1]. Re-normalisation of these matrices is not a simple task.
3. Quaternions are useful for another reason that they preserve the algebraic nature of the equations derived from them. They do not produce a complicated set of equations in sines and cosines.
4. Interpolation of matrices is not straightforward.

1.1.1 Quaternions

A short description of quaternions is given below.
Quaternion, introduced by Hamilton are four-dimensional extensions to the complex numbers we know. It is represented as,

$$\boldsymbol{q} = a_0 + a_1 i + a_2 j + a_3 k \tag{1}$$

where $a_0, a_1, a_2, a_3 \in \mathbb{R}$ and i, j, k are the imaginary components and are defined as,

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

$$ij = k, jk = i, ki = j, ji = -k, kj = -i, ik = -j \tag{3}$$

Another common representation of quaternions is as a scalar and a vector pair as shown,

$$\boldsymbol{q} = (s, \vec{v}) \tag{4}$$

The addition of two quaternions follows the normal element wise operation, where as multiplication is as follows,

$$\boldsymbol{q}_1\boldsymbol{q}_2 = (s_1 s_2 - \vec{v_1}\vec{v_2}, s_1\vec{v_2} + s_2\vec{v_1} + \vec{v_1} \times \vec{v_2}) \tag{5}$$

The conjugate of a quaternion and the norm are as defined,

$$\boldsymbol{q}^* = (s, -\vec{v}) \tag{6}$$

$$\|\boldsymbol{q}\| = \boldsymbol{q}\boldsymbol{q}^* \tag{7}$$

Unit quaternions, i.e. $\|\boldsymbol{q}\| = 1$ can be used to represent rotations. Given rotation of a rigid body by an angle $\boldsymbol{\theta}$ about an axis $\vec{k}$ it can be represented as,

$$\boldsymbol{q} = (cos(\frac{\boldsymbol{\theta}}{2}), \vec{k}sin(\frac{\boldsymbol{\theta}}{2})) \tag{8}$$

Other useful properties of quaternions are the Hamiltonian operators. For a given quaternion represented as (1), the Hamiltonian operators are as given [2],

$$\overset{+}{H}(\boldsymbol{q}) = \begin{pmatrix} a_0 & -a_1 & -a_2 & -a_3 \\ a_1 & a_0 & -a_3 & a_2 \\ a_2 & a_3 & a_0 & -a_1 \\ a_3 & -a_2 & a_1 & a_0 \end{pmatrix} \tag{9}$$

$$\overset{-}{H}(\boldsymbol{q}) = \begin{pmatrix} a_0 & -a_1 & -a_2 & -a_3 \\ a_1 & a_0 & a_3 & -a_2 \\ a_2 & -a_3 & a_0 & a_1 \\ a_3 & a_2 & -a_1 & a_0 \end{pmatrix} \tag{10}$$

These operators allow us to manipulate the quaternion multiplication as though they are commutative.

$$\boldsymbol{q}_1\boldsymbol{q}_2 = \overset{+}{H}(\boldsymbol{q}_1)\boldsymbol{q}_2 = \overset{-}{H}(\boldsymbol{q}_2)\boldsymbol{q}_1 \tag{11}$$

1.2 Why use dual numbers?

The reasons for using dual numbers for representation in robot kinematics are,

1. Allows us to do line transformations.
2. Gives a compact representation of both translation and rotation components, and their velocities, which are represented by a line in the 3D space.

1.2.1 Dual Numbers

Given the distance between two lines in 3D as, d and the angle their directions is α,

$$\hat{\alpha} = \alpha + \epsilon d \tag{12}$$

where ϵ^2 is defined to be 0

1.3 Why use dual quaternions?

This is a generalisation of a quaternion into a dual number. They carry the advantages of both using quaternions and dual numbers.
Unit dual quaternions can be used to represent general affine transformations. and are denoted as below,

$$\hat{\boldsymbol{q}} = \boldsymbol{q} + \frac{\epsilon}{2}\boldsymbol{t}\boldsymbol{q} \tag{13}$$

where $\boldsymbol{q}$ represents the quaternion corresponding to the rotation, $\boldsymbol{t}$ represents the quaternion representing translation i.e., $(0,\ \vec{t})$, where $\vec{t}$ represents the translation vector and $\hat{\boldsymbol{q}}$ represents the configuration of a rigid body translated by $\boldsymbol{t}$ and then rotated by $\boldsymbol{q}$.

2 Dual Quaternion Control

The following method of formulating control law using dual quaternions is as given in [3]

2.1 Forward Kinematics

The robot used in the current study is the six-axis, PUMA 560 robot. The forward kinematic map in terms of the end-effector dual quaternion is obtained based on the DH parameters given in [4]

$$\hat{\boldsymbol{q}}_{ee} = \hat{\boldsymbol{q}}_1\hat{\boldsymbol{q}}_2\hat{\boldsymbol{q}}_3 \cdots \hat{\boldsymbol{q}}_n \tag{14}$$

where, $\hat{\boldsymbol{q}}_1$ represents the configuration of the end-effector and the other represent the individual quaternions corresponding to the DH parameters.

2.2 Formulating Jacobian

From here we assume the dual quaternions are mapped to the $\mathbb{R}^8$. So we differentiate the eight dimensional tuple with respect to the actuated variables, i.e. $\boldsymbol{\theta}_1$, $\boldsymbol{\theta}_2$, $\boldsymbol{\theta}_3$, $\boldsymbol{\theta}_4$, $\boldsymbol{\theta}_5$, $\boldsymbol{\theta}_6$. This gives us a Jacobian of size 8×6 which we shall call J_{dual}. This gives us,

$$\dot{\hat{\boldsymbol{q}}} = J_{dual}\dot{\vec{\boldsymbol{\theta}}} \tag{15}$$

2.3 Control Law

Here we only deal with a regulation problem using Jacobian based control. A regulation problem is nothing but a position control problem, i.e. given a position or a final configuration, we control the robot to reach that final position. In this case, this is done based on the Jacobian, i.e. the problem is simplified by considering only the kinematics, ignoring the dynamic effects of the robot.

2.3.1 Error Dynamics

Let the initial configuration of the end-effector be $\hat{\boldsymbol{q}}_1$ and the desired configuration to be $\hat{\boldsymbol{q}}_d$. Let us define the error to be,

$$\hat{\boldsymbol{q}}_e = \hat{\boldsymbol{q}}_d - \hat{\boldsymbol{q}} \tag{16}$$

where $\hat{\boldsymbol{q}}_e$ represents the error dual quaternion and $\hat{\boldsymbol{q}}$ represents the configuration at any given instant.

Differentiating equation (16), gives the following,

$$\dot{\hat{\boldsymbol{q}}}_e = \dot{\hat{\boldsymbol{q}}}_d - \dot{\hat{\boldsymbol{q}}} \tag{17}$$

Since, we are dealing with a regulation problem, we have a fixed $\dot{\hat{\boldsymbol{q}}}_d$ and hence its derivative is zero.

$$\dot{\hat{\boldsymbol{q}}}_e = -\dot{\hat{\boldsymbol{q}}} \tag{18}$$

Substituting (15) equation in (18) gives,

$$\dot{\hat{\boldsymbol{q}}}_e = -J_{dual}\dot{\vec{\boldsymbol{\theta}}} \tag{19}$$

Let us consider a proportional control of this system and propose the control law to be,

$$\dot{\vec{\boldsymbol{\theta}}} = J_{dual}^T K \hat{\boldsymbol{q}}_e \tag{20}$$

This choice is to ensure asymptotic stability of the system making $J_{dual}J_{dual}^T$ a positive definite matrix[3].

This gives the error dynamics to be,

$$\dot{\hat{\boldsymbol{q}}}_e + J_{dual}J_{dual}^T K \hat{\boldsymbol{q}}_e = 0 \tag{21}$$

Integrating the equations (21) and (20) together gives us the motion of the end-effector.

3 Discussion

The gain parameter K = 185 is tuned to for the performance parameters of settling time of 10 seconds.

The asymptotic convergence of the error is as shown,

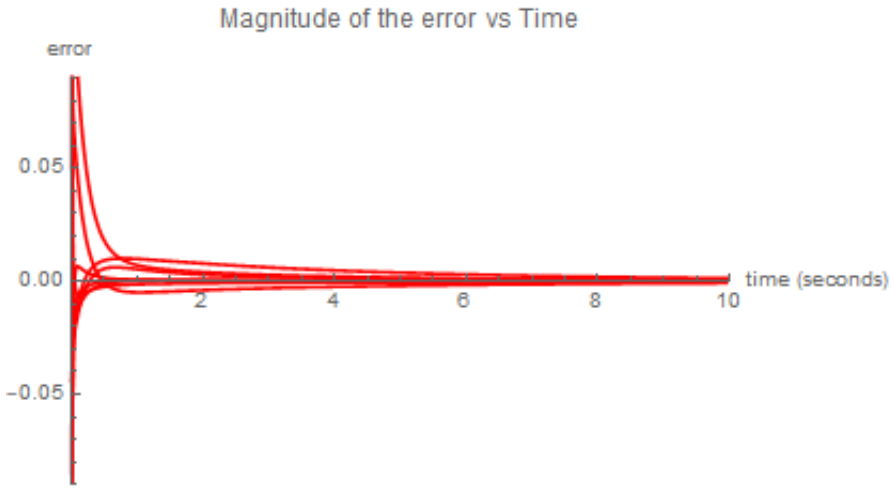

Figure 1: Asymptotic convergence of the quaternion error

The corresponding plot of the angles reaching the final position is as shown. With an increase

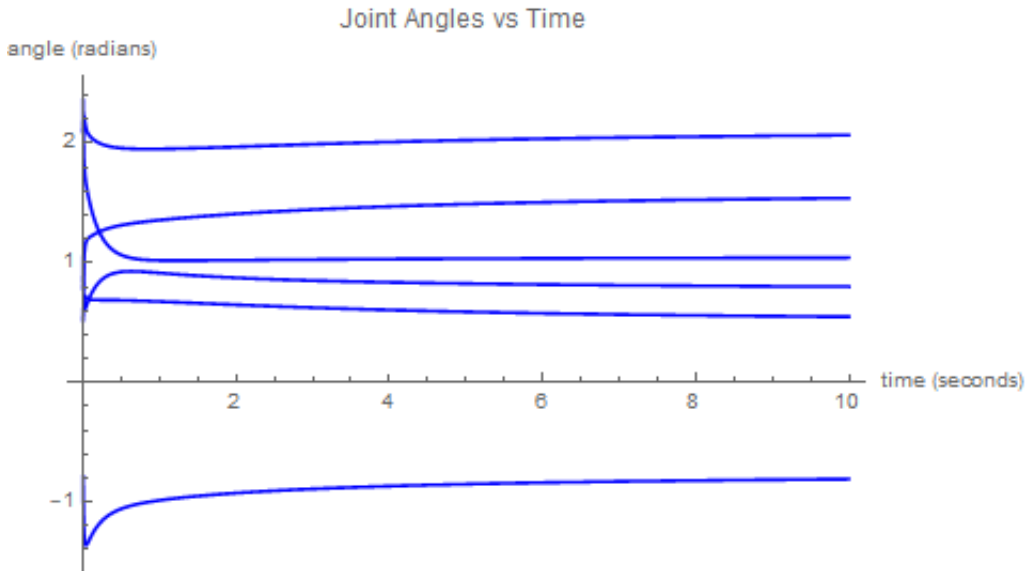

Figure 2: The trajectory of the six angles of PUMA 560 robot

in the gain as expected the manipulator tries to minimise the error more aggressively.

4 Utility

One of the applications for such dual quaternion based control could be in cooperative task space, where two robot arms need to move together to perform a task [5].

When two robots or a single robot with two hands do a task, using dual quaternions to represent the task would greatly simplify the task definition. The paper has suggested the use of various primitives for control. These primitives used in any sequence would give meaningful control to complete the task.

One of such primitive is the task definition of relative Cartesian position control which is demonstrated here. Consider the relative quaternion of the end effector to be,

$$\hat{\boldsymbol{q}}_r = \hat{\boldsymbol{q}}_2^* \hat{\boldsymbol{q}}_1 \tag{22}$$

$$\hat{\boldsymbol{q}}_r = \boldsymbol{q}_r + \boldsymbol{q}_r^{'} \tag{23}$$

$$\boldsymbol{q}_r^{'} = \frac{\epsilon}{2} \boldsymbol{t}_r \boldsymbol{q}_r \tag{24}$$

Now form this the translation quaternion can be extracted out as,

$$\boldsymbol{t}_r = 2\boldsymbol{q}_r^{'} \boldsymbol{q}_r^* \tag{25}$$

Again consider the quaternion to be $\boldsymbol{t}_r$ to be an element in $\mathbb{R}^8$. Differentiating this equation gives and using (11),

$$\dot{\boldsymbol{t}_r} = 2\dot{\boldsymbol{q}^{'}}_r \boldsymbol{q}_r^* + 2\boldsymbol{q}_r^{'} \dot{\boldsymbol{q}}_r^* \tag{26}$$

$$\dot{\boldsymbol{t}_r} = 2(\bar{H}\ (\boldsymbol{q}_r^*) J_{\boldsymbol{q}_r^{'}} + \overset{+}{H}\ (\boldsymbol{q}^{'}) J_{\boldsymbol{q}_r^*}) \dot{\vec{\boldsymbol{\theta}}} \tag{27}$$

where $\dot{\vec{\boldsymbol{\theta}}}$ here represents a 12 dimensional vector. So controlling the relative translation quaternion which is 4 dimensional gives us the definition of control for all the joint angles.

The similar strategy was used in [5] with four different primitives to perform simple tasks like grasping a balloon and pouring water in a cup from a bottle with a two armed robot. Using this new definition of Jacobian for the given task, we implement the control strategy as mentioned in section (2.3). The following is the asymptotic error convergence plot and the corresponding change in the angles of both the manipulators.

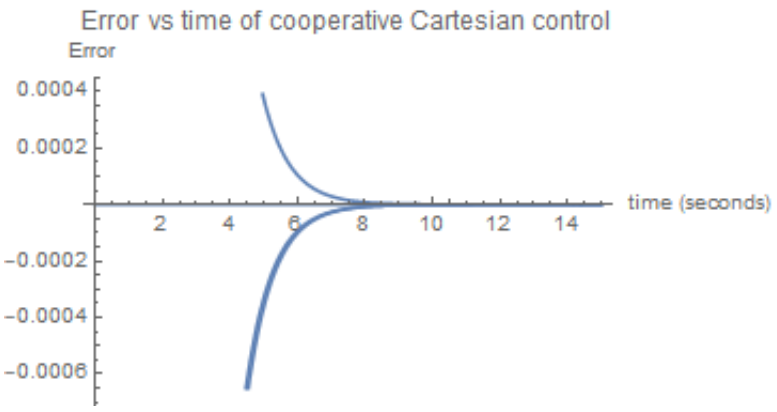

Figure 3: Asymptotic convergence of the error for relative Cartesian control

5 Error in error

Based on the (brief) literature survey of usage of dual quaternions done, it is observed that in practical purposes it is comfortable to implement control assuming that these dual quaternions are embedded in $\mathbb{R}^8$ as in [3] and [5] to mention a few. But we know that these entities represent elements from $\mathbb{SE}(3)$, and hence the difference between the two values does not yield a distance measure. It would no longer represent a physically realisable robot configuration as the error is no longer a unit dual quaternion.

5.1 What do we do?

For the next part of the work, we follow the path of [6], which extends the classical PD controllers to $\mathbb{SO}(3)$ and $\mathbb{SE}(3)$.

Following the formulation in [6], we choose to use the logarithmic feedback controller. The setting given in the paper is generic to any system in $\mathbb{SE}(3)$ and an example in $\mathbb{SE}(2)$ of a differential mobile robot is given. In the current work, we extend this formulation to dual quaternions and show its utility in control of a six-axis serial robot.

6 Error Dynamics

Knowing the fact that dual quaternions represent elements in $\mathbb{SE}(3)$, a more reasonable choice of error would be,

$$\hat{\boldsymbol{q}}_e = \hat{\boldsymbol{q}^*}_d \hat{\boldsymbol{q}} \tag{28}$$

Differentiating the above term and setting $\dot{\hat{\boldsymbol{q}}}_d^*$ to zero(regulation problem), we get,

$$\dot{\hat{\boldsymbol{q}}}_e = \hat{\boldsymbol{q}^*}_d \dot{\hat{\boldsymbol{q}}} \tag{29}$$

The logarithmic error feedback in our case would be of the form,

$$\dot{\hat{\boldsymbol{q}}} = -K \log(\hat{\boldsymbol{q}}_e) \tag{30}$$

where K is a matrix positive definite matrix. Unlike the previous case, in this case, we can tune the gains for linear and angular velocities separately by internally having different coefficients in the matrix.

There are some interesting properties of a dual quaternion. Similar to any complex number which can be represented using the Euler's form, dual quaternions can be represented in a similar way [7].

$$\hat{\boldsymbol{q}} = \boldsymbol{q} \exp(\epsilon \frac{\vec{t}}{2}) \tag{31}$$

So applying a logarithm would give us the following,

$$\log(\hat{\boldsymbol{q}}) = \frac{\boldsymbol{\theta}\vec{k}}{2} + \epsilon\frac{\vec{t}}{2} \tag{32}$$

Using the above equations, we reformulate the error dynamics in terms of the elements in $se(3)$ and implement control over these elements.

Note: Please understand that,

$$\log(\hat{\boldsymbol{q}}) = \log(\hat{\boldsymbol{q}}_1\hat{\boldsymbol{q}}_2) \tag{33}$$

satisfies but,

$$\log(\hat{\boldsymbol{q}}) \neq \log(\hat{\boldsymbol{q}}_1) + \log(\hat{\boldsymbol{q}}_2) \tag{34}$$

$$\log(\hat{\boldsymbol{q}}) - \log(\hat{\boldsymbol{q}}_1) \neq \log(\hat{\boldsymbol{q}}_2) \tag{35}$$

This is from the fact that dual quaternions multiplication is not same as multiplication in $\mathbb{R}^n$ Instead we have,

$$\log(\hat{\boldsymbol{q}}_1^*\hat{\boldsymbol{q}}) = \log(\hat{\boldsymbol{q}}_2) \tag{36}$$

7 Discussion

The error in the controlled entities i.e., the error in end effector orientation and the position asymptotically go to zero.

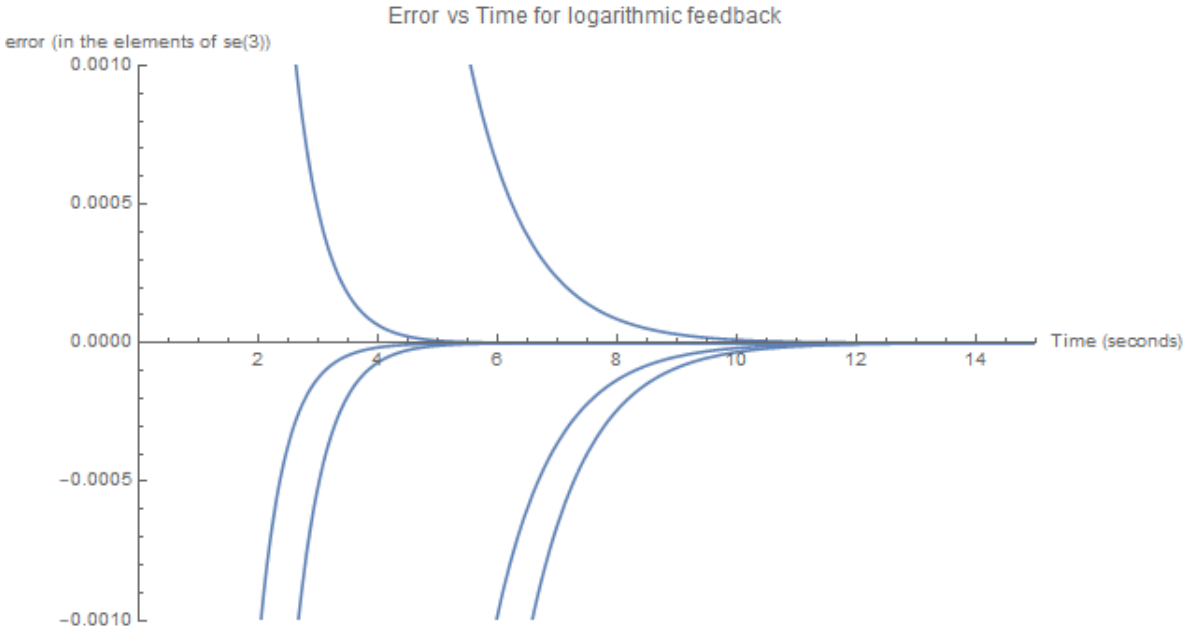

Figure 4: Asymptotic convergence of the error for logarithmic feedback

The proportional gain of both the linear and angular velocity components was kept equal to 1 for the above plot.

A quick observation is that the variables that converge quickly are the ones corresponding to the errors in the translation of the end-effector. It is also to be noted that the control scheme for position is dependent on the angle at every instant.

8 Comparison of the two controllers

Both the controllers were given the same desired, and final quaternion configurations and the resulting control laws are observed. A same gain value is given to both the controllers. We shall call the logarithmic controller as logc and difference controller as diffc from now on.

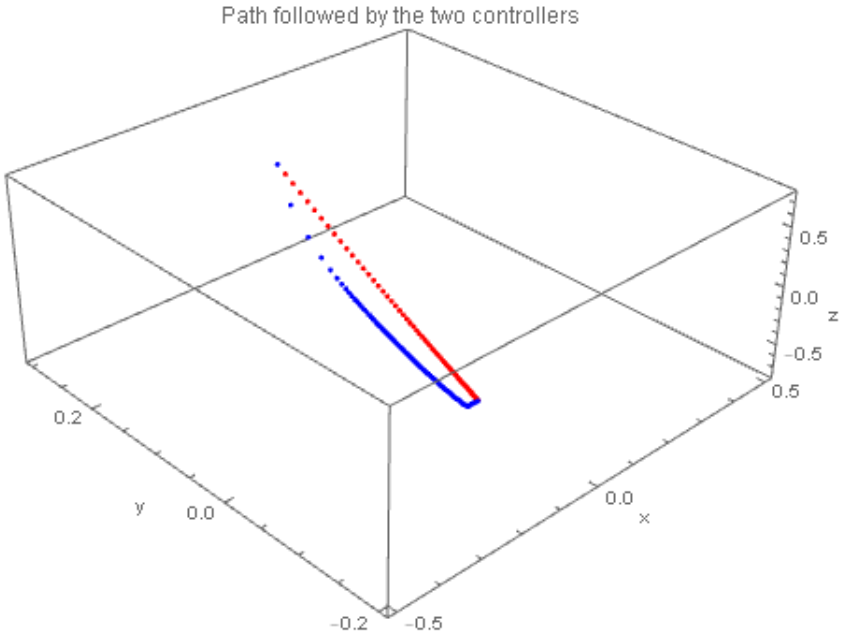

Figure 5: Path traced by the logarithmic controller (red) and the difference controller(blue)

where the three axes represent $\mathbb{R}^3$.

Note that diffc travels extra distance than the logc. Also that the initial motion of the manipulatior is quite rapid as the dots are farther from each other as compared to the red dots. Investigating the path followed by diffc alone would be as shown below.

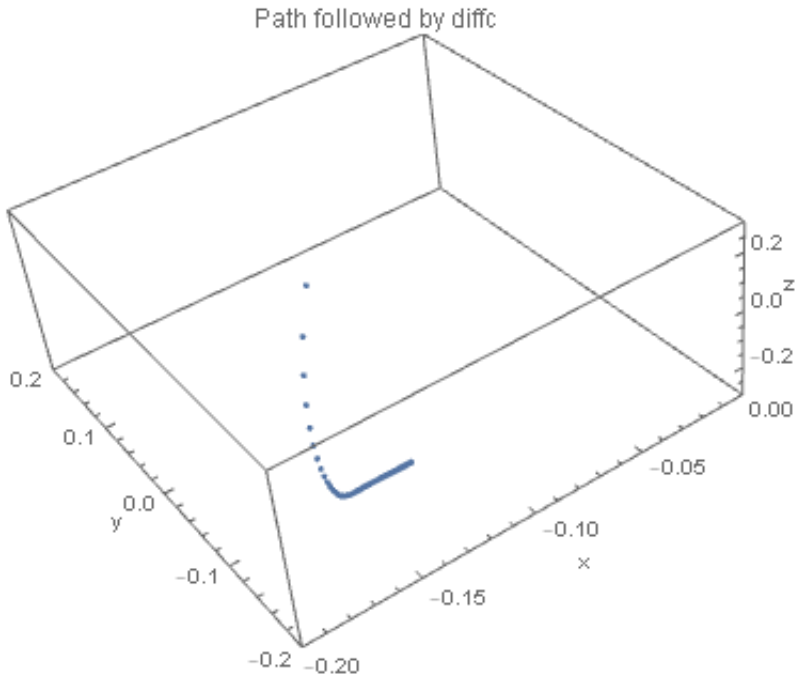

Figure 6: Curved path traced by diffc

The $\mathbb{SE}(3)$ visualisation of the diffc controller is,

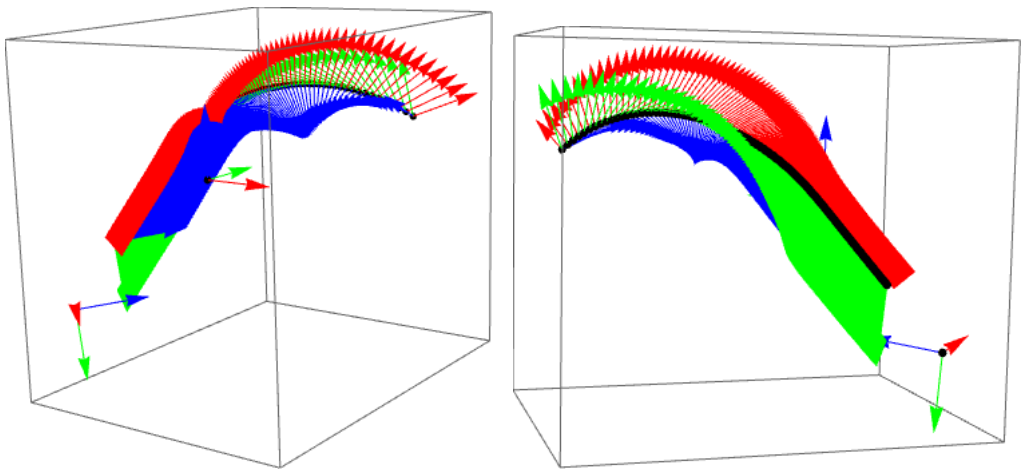

Figure 7: The $\mathbb{SE}(3)$ visualisation of diffc in two viewing angles

The box represents the $\mathbb{R}^3$. The tri-colour axes represent the $\mathbb{SO}(3)$ part and the black dots represent the $\mathbb{R}^3$ part of the motion of the manipulator end-effector.

It can be seen that the end-effector does not reach its final configuration in the given time. So increasing the gain to 185 which matches the settling time with logc is as shown,

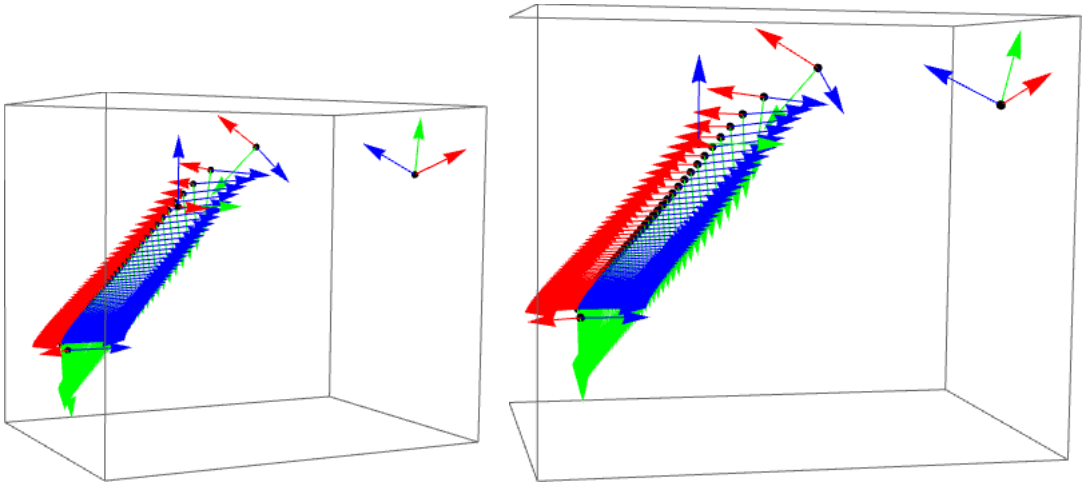

Figure 8: The $\mathbb{SE}(3)$ visualisation of logc in two viewing angles for the controller gain of $Kp = 185$

The initial part of the trajectory is missing in the above plot as the controller was too aggressive in the initial part and the motion took place in a very small time.

The $\mathbb{SE}(3)$ visualisation of the logc controller is,

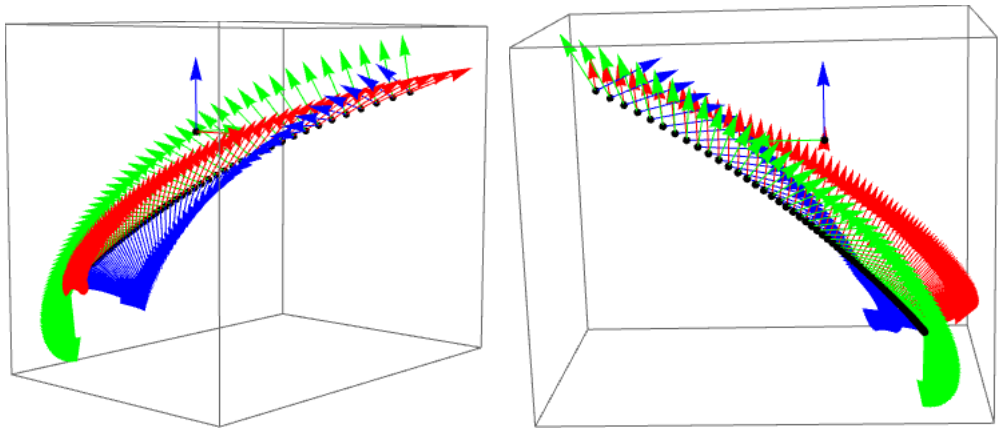

Figure 9: The $\mathbb{SE}(3)$ visualisation of logc in two viewing angles

So for a given final configuration logc takes a much smaller Kp value 1 compared to the diffc controller which requires a 185, to achieve the same settling time (within 2% of the final value) of 10s.

It should also be noted that the logc produces a much less curved solution in $\mathbb{R}^3$ where as the diffc controller has a comparatively larger curvature. The path of the diffc controller overshoots the final position and comes back to the final configuration, which is a serious trouble in applications like welding and bolting.

The logarithmic controller is arguably in some sense more "natural".

9 Concluding question

Taking this discussion a step further ignoring the internal actuation, dynamics, structure of the hand, I ask the following question,

Our eyes cannot perceive $\mathbb{SE}(3)$ but can our brains do?

Or does it do even better?

10 Miscellaneous

The scope of the current work is as below,

1. There is yet another way to describe the error. That is by splitting the $\mathbb{SO}(3)$ and $\mathbb{R}^3$ parts of the dual quaternion and follow geodesics in each of these space. This approach is called the Double Geodesic Control, which we have not described here.

2. We have limited ourselves here to a very small subset of the problem. Firstly we deal only with regulation and secondly we are using only a P controller. One could use PD as well, as shown in [6].

3. There is no sense of optimality in the formulation, i.e. we do not extremise any objective here.

4. Both the controllers behave abruptly as the gain is increased, but the from the error plots and animations, we see that logc closes the gap in Cartesian space and then $\mathbb{SO}(3)$ whereas diffc quickly reaches the orientation and then covers the gap in Cartesian space.

References

[1] B. Kenwright, "A beginners guide to dual-quaternions: What they are, how they work, and how to use them for 3D character hierarchies," *The 20th International Conference on Computer Graphics, Visualization and Computer Vision*, pp. 1–13, June 2012.

[2] B. Akyar, "Dual quaternions in spatial kinematics in an algebraic sense," *Turkish Journal of Mathematics*, vol. 32, no. 4, pp. 373–391, 2008.

[3] H.-L. Pham, V. Perdereau, B. V. Adorno, and P. Fraisse, "Position and orientation control of robot manipulators using dual quaternion feedback," in *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pp. 658–663, IEEE, 2010.

[4] A. Ghosal, *Robotics: fundamental concepts and analysis*. Oxford university press, 2006.

[5] B. V. Adorno, P. Fraisse, and S. Druon, "Dual position control strategies using the cooperative dual task-space framework," in *IROS'10: International Conference on Intelligent Robots and Systems*, pp. 3955–3960, IEEE, 2010.

[6] F. Bullo and R. M. Murray, "Proportional Derivative (PD) control on the Euclidean group," in *European Control Conference*, vol. 2, pp. 1091–1097, 1995.

[7] O. Bottema and B. Roth, "Theoretical kinematics," *NORTH-HOLLAND PUBL. CO., N. Y., 1979, 558*, 1979.

A Modules

The following functions are written and used for the completion of the above work.

A.1 Quaternions

1. quat: Generates a unit quaternion
2. qmult: Quaternion multiplication
3. cquat: Conjugate of a given quaternion
4. Hp: Generates the $\overset{+}{H}$ of a given quaternion
5. Hn: Generates the $\overset{-}{H}$ of a given quaternion
6. decq: Decomposes a quaternion into axis angle form

A.2 Dual Quaternions

1. unitdq: Generates a unit dual quaternion
2. cdualq: Gives the conjugate of a given dual quaternion
3. dqmult: Dual quaternion multiplication
4. dHp: Generates $\overset{+}{H}$ for the corresponding dual quaternion
5. dHp: Generates $\overset{-}{H}$ for the corresponding dual quaternion
6. logdq: Return the logarithm of a dual quaternion
7. explogdq: Given a logarithm of a dual quaternion it gives the exponent of it

A.3 Miscellaneous

1. R82dual: Converts an element from $\mathbb{R}^8$ into a dual quaternion
2. R62dual: Converts an element from $\mathbb{R}^6$ into a dual quaternion
3. frameplotter: Given an element in $se(3)$, the module plots the corresponding frame with respect to an origin

A.4 Initial Conditions

The non-singular initial and final poses of the manipulator are taken from [4]

All the modules and the codes can be found in my github page