Transformations for Dummies: A tutorial on commonly-used Lie groups in Robotics and Computer Vision

James Jackson

1 Introduction

This document motivates and derives many commonly-used formulae used when performing coordinate transformations in robotics and computer vision. There are a number of great resources on this topic [1, 3, 4, 5, 7], and I do not pretend to be adding anything particularly novel to the field of knowledge in this area. However, this has personally been a source of great confusion to me over the years and I wanted to derive these concepts from basics, highlighting the gotchas that I have encountered along the way. I hope that this ends up being useful to people new to the field of robotics or computer vision.

As many of my fellow graduate students and I used to say, "The two hardest problems in robotics are coordinate frames and naming things." While this won't help you much with the second problem, this will hopefully help dispel a lot of the confusion I encountered at first.

One difference with this document when compared with others is the use of extra notation throughout. This notation is much more verbose than a lot of other literature on robotics or computer vision, but the extra syntax clarifies what is going on, and makes it easier for someone like me (who likes to think of these ideas in terms of physical relationships, rather than abstract mathematical ideas) to visualize and understand what is going on. The extra notation has also allowed me to make distinctions between different fields of research that may have different conventions when it comes to transformations.

2 Vectors

Let us first describe the notation used in the rest of the document. Succinctly, this is described as follows:

$\mathbf{r}^c_{a/b}$	the vector $\mathbf{r}$, representing some quantity (e.g. position) of point a with respect to point b, expressed in frame c.
R^b_a	a transformation or rotation that executes a change of basis from frame a to frame b
$\lfloor \mathbf{v} \rfloor_\times$	The skew-symmetric matrix formed from $\mathbf{v}$

2.1 Vector Notation

If that's all you need, then great! You can carry on to the next section. However, for people like me, let's dig into this notation and (hopefully) let it sink in. Consider the illustration in Figure 2.1. We have some point a and another point b. Let $\mathbf{r}_{a/b}$ be the vector which describes the position of b with respect to a. Note that we could flip the vector around by negating the quantity, which means that for any arbitrary frame of reference

$$\mathbf{r}_{b/a} = -\mathbf{r}_{a/b}.$$

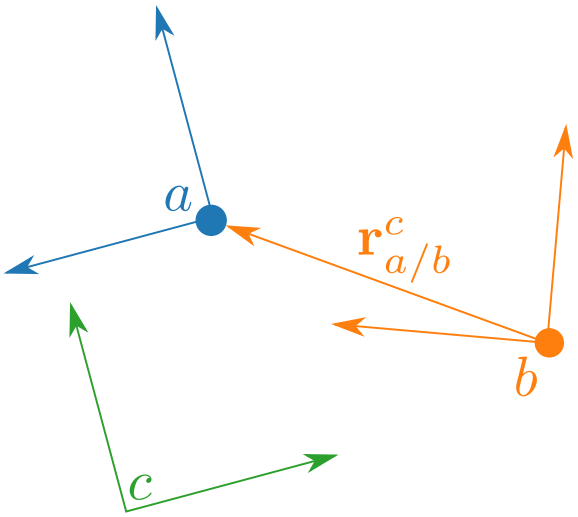

Fig. 2.1: Illustration of a vector with notation

Although it may be sometimes hard to actually draw some quantities clearly (like the angular rate of two coordinate frames), in engineering, we typically define vectors as some quantity with respect to some other reference. For example, the angular velocity of a rotating body with respect to the earth might be written as $\boldsymbol{\omega}_{b/E}$. While it may be difficult to visualize, we could also consider the opposite case, what the angular velocity of the earth was with respect to the rotating body, $\boldsymbol{\omega}_{E/b}$. (This can be a fun mind bender, turning frames of reference around in your head). It turns out that even in this case, the "earth-centric" representation is just the negative of the "body-centric" representation.

$$\boldsymbol{\omega}_{E/b} = -\boldsymbol{\omega}_{b/E}.$$

The next piece of information that we need is the frame of reference it is described in (its basis). For example, in Figure 2.1, we could describe $\mathbf{r}$ in any coordinate frame we want. We could use frame a, b or c, it doesn't really matter, so long as we always do all operations between vectors represented in the same frame.

We will use the superscript notation to describe the frame of reference, as in

$$\mathbf{r}^c_{a/b}.$$

Which in words means "the position of point a with respect to point b, expressed in frame c." All vectors have this concept of frame of reference, even if they are hard to visualize, like angular rate, acceleration, etc...

2.2 Rotating Vectors

I'm going to jump ahead a little bit and introduce the concept of a rotation as a change of basis. Let us say that we have some version of $\mathbf{r}$ expressed in the c frame, but we want to express it in the b frame. How do we do that? Easy, we simply change the basis.

$$\mathbf{r}^b_{a/b} = R^b_c \mathbf{r}^c_{a/b}$$

I will get into the details of how this actually works a little later, but for now, just believe me that we can change this basis. The notation is such that you can "cancel out" the frames, as in

$$\mathbf{r}^b_{a/b} = R^b_{\not c} \mathbf{r}^{\not c}_{a/b}.$$

So what if I want to go all the way to frame a but I only have rotations from $c \to b$ and $b \to a$? Easy peasy, just compose the rotations, and everything cancels out.

$$\mathbf{r}^a_{a/b} = R^a_{\not b} R^{\not b}_{\not c} \mathbf{r}^{\not c}_{a/b}.$$

Again, I'm deliberately skimming over most of the actual math. We'll get into this a lot more later, but we first need to get the mechanics of working with this notation.

2.3 Composing Vectors

Let's now take a minute to remember what defines a vector space. From Wikipedia:

> A vector space is a collection of objects called vectors, which may be added together and multiplied ("scaled") by numbers, called scalars.

All vector spaces abide by the following rules:

Tab. 1: Table of the rules of a Vector Space

Axiom	Meaning
Associativity	$\mathbf{a} + (\mathbf{b} + \mathbf{c}) = (\mathbf{a} + \mathbf{b}) + \mathbf{c}$
Commutativity	$\mathbf{a} + \mathbf{b} = \mathbf{b} + \mathbf{a}$
Identity of addition	There exists an element $\mathbf{0}$ such that $\mathbf{a} + \mathbf{0} = \mathbf{a}$ for every vector in the space
Inverse element of addition	for every vector $\mathbf{a}$ in the space, there is one and only one vector $-\mathbf{a}$ such that $\mathbf{a} + (-\mathbf{a}) = \mathbf{0}$.
distributivity of scalar multiplication	$(v + u)\,\mathbf{a} = v\mathbf{a} + u\mathbf{a}$ $v\,(\mathbf{a} + \mathbf{b}) = v\mathbf{a} + v\mathbf{b}$
Identity element of scalar multiplication	$1\mathbf{a} = \mathbf{a}$

If you've taken a course in linear algebra, these will not be new to you. However, for me, up to this point, I had never worked with objects that were *not* in a vector space, so the rules seemed obvious and redundant.

We like vector spaces because computers are well-suited for solving linear algebra problems. There are high-performance BLAS libraries (basic linear algebra subprograms) and methods for performing matrix decompositions (SVD, LU, QR, Cholesky etc...) that allow us to solve complicated problems quickly and accurately.

It turns out that rotations and transformations do *not* lie in a vector space. This means that we end up performing manipulations to get our problems into a vector space where we can use powerful linear algebra techniques, and the distinction between vector spaces and non-vector spaces will become very important.

Let us say we have three points, each with an associated coordinate frame a, b, and c, as shown in Figure 2.2. Let's also say that we have only the orange vectors. (the vector from $b \to a$ and the vector from $c \to b$) but we want the green vector (the one that goes all the way from $c \to a$).

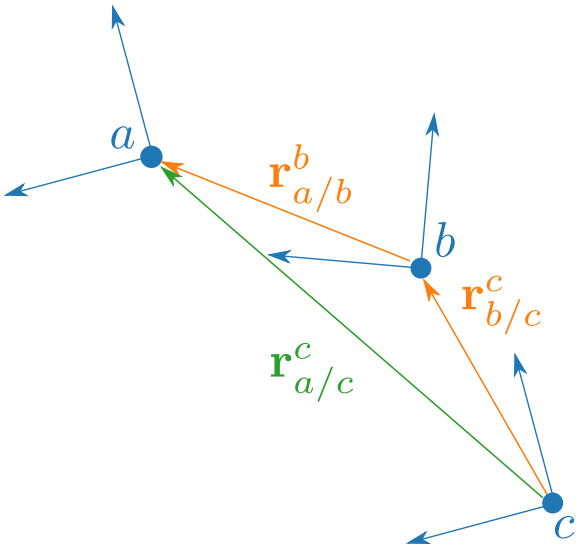

Fig. 2.2: Illustration of a vector triangle

Well, this would be easy if they were all represented in the same coordinate frame

$$\mathbf{r}_{a/c} = \mathbf{r}_{a/b} + \mathbf{r}_{b/c},$$

but they aren't. However, we just have to remember that vectors can only be added or subtracted if they are in the same frame, so we just rotate them into a common frame before doing the additions, like this:

$$\mathbf{r}^c_{a/c} = R^c_b \left(\mathbf{r}^b_{b/c} + R^b_a \mathbf{r}^a_{a/b} \right).$$

From a theoretical perspective, the choice of coordinate frame doesn't matter, however in practice there is often a choice of frame that makes things easier.

2.4 Skew-symmetric matrices

Before going much further, I also need to introduce skew-symmetric matrices. This[1] is the skew-symmetric matrix operator $\lfloor \mathbf{v} \rfloor_\times$. It is defined as

$$\lfloor \mathbf{v} \rfloor_\times = \begin{bmatrix} 0 & -\mathbf{v}_z & \mathbf{v}_y \\ \mathbf{v}_z & 0 & -\mathbf{v}_x \\ -\mathbf{v}_y & \mathbf{v}_x & 0 \end{bmatrix},$$

and is related to taking the cross-product between two vectors as

$$\mathbf{a} \times \mathbf{b} = \lfloor \mathbf{a} \rfloor_\times \mathbf{b}.$$

[1] There are a variety of symbols used to communicate this operation. $\mathbf{v}_\times$ and $(\mathbf{v})^\times$ are also commonly used.

The skew-symmetric matrix has some really interesting and helpful properties. The first is that the operator is anti-commutable, that is

$$\lfloor \mathbf{a} \rfloor_\times \mathbf{b} = - \lfloor \mathbf{b} \rfloor_\times \mathbf{a}. \tag{2.1}$$

The second is that rotation matrices can be moved in and out of the skew-symmetric operator with

$$\lfloor R\mathbf{v} \rfloor_\times = R \lfloor \mathbf{v} \rfloor_\times R^{-1}. \tag{2.2}$$

Finally, you could probably see from inspection that the transpose of a skew-symmetric matrix is its negative

$$\lfloor \mathbf{v} \rfloor_\times^\top = - \lfloor \mathbf{v} \rfloor_\times .$$

These are all super useful tricks that show up all the time. I'll try to point out when I'm using one of these tricks, but I might miss some.

3 A Look at the Matrix Exponential

The matrix exponential is central to the rest of this document. In this section, we are going to take a deeper look at the matrix exponential to gain some intuition about what it's doing. If we can understand what the exponential really means, then the reason we use it so often in Lie Group theory will become obvious.

Let's start with the scalar, first-order differential equation

$$\dot{x} = ax. \tag{3.1}$$

This equation has the simple solution,

$$x(t) = x(t_0)\, e^{at},$$

the derivation of which is a central part of any undergraduate differential equations class. However, let us for a moment pretend that we don't know about the natural number e, but we were tasked with solving Eq. 3.1. To make things more concrete, we could stand in the shoes of Jacob Bernoulli as he studies the following problem:

> An account starts with \$1.00 and pays 100 percent interest per year. If the interest is credited once, at the end of the year, the value of the account at year-end will be \$2.00. What happens if the interest is computed and credited more frequently during the year?

This problem is formulated as

$$\$1.00 \times \left(1 + \frac{1}{n}\right)^n,$$

where n is the number of intervals.

Jacob Bernoulli noticed that this series converges to what we now know as e, the natural number, as $n \to \infty$.

So what is going on here? The main factor at play is that the amount of interest that the banker pays to the account depends on how much money is *already in* the account. It just so happens that

this form of problem, where the derivative of some state is directly proportional to its current value has this really nice form that depends on the limit to this infinite series.

To bring this concept to a robotic application, let's consider two-dimensional transforms. Let's say we have a unicycle robot[2] with inputs linear velocity v and angular velocity ω. The dynamics of this system are given as

$$\begin{pmatrix}\dot{x}\\ \dot{y}\\ \dot{\theta}\end{pmatrix} = \begin{pmatrix}v\sin(\theta)\\ v\cos(\theta)\\ \omega\end{pmatrix}. \tag{3.2}$$

Let's say that given a constant velocity and angular rate command, we want to know where this robot is after one second. How should we solve this? Well, one straight-forward way is with good ol' Euler integration

$$\mathbf{x}[t+\delta t] = \mathbf{x}[t] + \delta t\dot{\mathbf{x}}[t].$$

If we do this, we should get something like what is shown in Figure 3.1. In each trajectory, I have integrated the same amount of time, but with smaller and smaller time steps. If we think about what is happening from the point of view of the robot, we could imagine that for each δt, we drive forward at some velocity, and then turn. We then repeat this, over and over until we reach the end of the interval. In Figure 3.1 we can see that as our δt's get smaller, we seem to be approaching some ideal situation, where we are no longer separating the "drive forward" step from the "turn" step. They are happening simultaneously, and we are getting that natural curving motion that we intuitively expect.

This is exactly the same situation as the compounding interest problem discussed earlier, and we can set up our problem in such a way that the matrix exponential solves it for us. The matrix exponential, similar to the scalar exponential, is defined by the infinite series

$$\exp(A) = \sum_{k=0}^{\infty}\frac{1}{k!}A^k. \tag{3.3}$$

We will see in Section 7 that for many useful situations, this expression has a closed form. To solve Eq. 3.2, with the matrix exponential we first need to convert the system into the matrix differential equation[3]

$$\dot{\begin{bmatrix}\cos(\theta) & \sin(\theta) & x\\ -\sin(\theta) & \cos(\theta) & y\\ 0 & 0 & 1\end{bmatrix}} = \begin{bmatrix}\cos(\theta) & \sin(\theta) & x_0\\ -\sin(\theta) & \cos(\theta) & y_0\\ 0 & 0 & 1\end{bmatrix}\cdot\begin{bmatrix}0 & -\omega & v\\ \omega & 0 & 0\\ 0 & 0 & 0\end{bmatrix},$$

Don't be too worried about how this works. We will talk about this a lot more later. Just know that this is an equivalent expression to 3.2. As soon as we do, the solution is given as

$$\begin{bmatrix}\cos(\theta) & \sin(\theta) & x\\ -\sin(\theta) & \cos(\theta) & y\\ 0 & 0 & 1\end{bmatrix}(t) = \begin{bmatrix}\cos(\theta_0) & \sin(\theta_0) & x_0\\ -\sin(\theta_0) & \cos(\theta_0) & y_0\\ 0 & 0 & 1\end{bmatrix}\cdot\exp\begin{pmatrix}0 & -\omega t & vt\\ \omega t & 0 & 0\\ 0 & 0 & 0\end{pmatrix}.$$

[2] This is a standard model in robotics for studying non-holonomic controls and consists of a planar robot that can only drive forward and turn. This model is also known as Dubin's car.

[3] This is an example of $SE(2)$, the special euclidean group with two dimensions. The top left 2×2 block is the planar rotation matrix

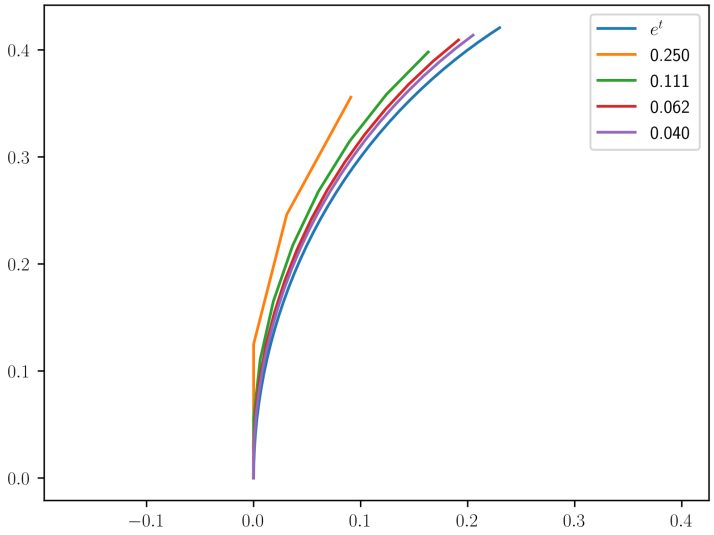

Fig. 3.1: Illustration of simple forward-mode numerical integration of a unicyle robot with different sized steps.

This is precisely what Lie theory is all about. We can use the matrix exponential to map to curvy structures (rotation matrices or transformations) from linear systems (vectors) in an efficient and principled way. Consequently, we can consider concepts of rotation and translation simultaneously and how they interact.

4 Rotations

There are a couple of common conventions found in different bodies of modern literature when dealing with rotations. This section enumerates a number of these conventions the implications, of using them and serves as a backdrop for much of the later work. It is important to realize that not all bodies of literature consider rotations in same way, and even within a single field, different conventions may apply depending on the source. It's also important to realize that none of these conventions is necessarily "right." (with the exception of Euler angles, which are usually wrong). So long as we effectively communicate what we are talking about and apply proper methods mathematically, each of these conventions have different strengths, which is why they have been adopted differently in different fields.

We will deal with both rotation matrices and unit quaternions, although I will start with a brief explanation of Euler angles, mainly to argue for why you shouldn't be using them.

4.1 Euler Angles

Euler angles are often the first method that people are introduced to when talking about rotations. Euler angles describe rotations about subsequent axes, and there are actually several variations on the order of these axes. The most common are the 3-2-1 Euler angles, typically referred to yaw, pitch and roll angles. Despite Euler angles being historically significant, when compared with modern methods, they are usually a poor choice for describing a rotation. Euler angles do not form a vector space[4], they are numerically unstable in many real-life situations, and they are computationally inefficient. However, Euler angles are often the most intuitive for human consumption, so a common workflow is to convert from rotation matrices or quaternions to Euler angles for debugging problems or plotting.

For reference, I have included the algorithms for converting rotation matrices and unit quaternions to Euler angles with ψ, θ ϕ (referring to yaw, pitch, and roll, respectively). Note that in the case that $\theta = \pi/2$, we get infinitely many solutions. This is known as gimbal lock, and is a consequence of the fact that for increasingly large values of pitch, roll and yaw look more and more alike until they are indistinguishable. While hitting gimbal lock would surely be catastrophic in an Euler-angle based system, this problem has real consequences even far from actual lock. This is because the representation becomes increasingly non-linear as pitch angle grows. So even if a system never actually hit gimbal lock, the risk of incurring significant linearization error is a quite high for even moderate pitch angles.

Rotation matrices and unit quaternions, while perhaps less intuitive, have none of these weaknesses. They can be easily cast into a vector space (using their associated Lie algebra, which I will talk about later), they are computationally efficient, and they are numerically stable... everywhere. The downside is the intellectual hurdle of jumping into matrix kinematics. I hope that this document is able to demystify rotation matrices and unit quaternions to the point that they are just as easy to use as Euler angles, and that future students don't wade through the problems of Euler angles any longer than they have to.

4.2 Rotation Matrices

A rotation matrix is a member of the special orthogonal group $SO(3)$. This is the set of real 3×3 matrices with determinant 1. Another way of saying this, which may be a bit more intuitive, is that the vectors making up each row and column are orthogonal to one another, have unit length, and follow the right-hand rule[5]. These vectors form the primary x, y, z axis of a Cartesian coordinate frame.

The columns of R_a^b contain the x, y, z vectors of the b frame, expressed in the a frame, while the rows of the matrix contain the basis vectors of the a frame, expressed in the b frame. This makes this matrix the perfect translator between a space and b space because it directly encodes the change of basis from one coordinate frame to another. This gives rise to the definition given earlier:

$$\mathbf{r}^b = R_a^b \mathbf{r}^a, \tag{4.1}$$

[4] Euler angles do not follow the rules of associativity, commutativity (if you roll first, then pitch, you get a different rotation than if you first pitched, then rolled) or scalar multiplication (what would it mean to just "double" all my euler angles?)

[5] This just means that if you take the x vector cross the y vector, you get the z vector. This comes from the constraint that $\det(A) = 1$. A matrix whose columns followed the left-hand rule would have a determinant -1

Algorithm 1 Computing Euler angles from a Rotation Matrix[8]

if $R_{31} \neq \pm 1$ **then**
  $\theta \leftarrow -\arcsin(R_{31})$
  $\psi = \arctan 2\left(\frac{R_{32}}{\cos\theta}, \frac{R_{33}}{\cos\theta}\right)$
  $\phi = \arctan 2\left(\frac{R_{21}}{\cos\theta}, \frac{R_{11}}{\cos\theta}\right)$
else
  $\phi =$ anything; set to 0
  if $R_{31} = -1$ **then**
    $\theta \leftarrow \frac{\pi}{2}$
    $\psi = \phi + \arctan 2(R_{12}, R_{13})$
  else
    $\phi = -\frac{\pi}{2}$
    $\psi = -\phi + \arctan 2(-R_{12}, -R_{13})$
  end if
end if

Algorithm 2 Computing Euler angles from a Unit Quaternion

$s \leftarrow 2(q_0 q_y - q_x q_z)$
if $|s| > 1.0$ **then**
  $\theta \leftarrow \text{sign}(s)\frac{\pi}{2}$
  $\phi \leftarrow$ anything; set to 0
else
  $\theta \leftarrow \arcsin s$
  $\phi \leftarrow \arctan 2\left(2(q_0 q_z + q_x q_y), 1 - 2\left(q_y^2 + q_z^2\right)\right)$
end if
$\psi \leftarrow \arctan 2\left(2(q_0 q_z + q_x q_y), 1 - 2\left(q_y^2 + q_z^2\right)\right)$

and the fact that rotation matrices compound from right to left, as in

$$R_a^c = R_b^c R_a^b.$$

4.3 Passive Versus Active Rotations

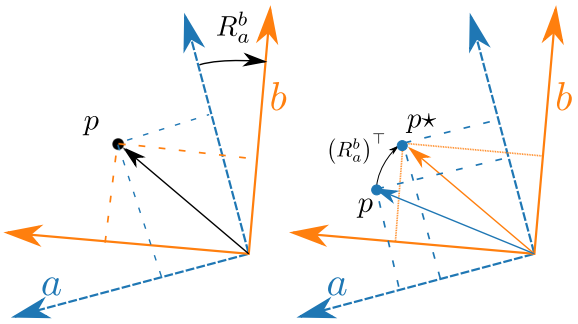

Fig. 4.1: Illustration of a passive rotation (left) and an active rotation (right)

We are using a convention here, where we define R as a *passive* rotation. Consider Equation 4.1. The vector quantity $\mathbf{r}$ did not change during this rotation, it is simply being expressed from another point of view. Some literature defines rotations as *active*. Figure 4.1 illustrates the difference between active and passive rotations. In the passive case (left), the point p does not change when rotating from frame a to b. It remains fixed while the perspective changes. In contrast to the active case (right), point p is moved to become a new object, point $p^\star$. This interpretation is much more common in the computer graphics literature, where a common operation is to generate some asset at an arbitrary origin, and then move that object to be rendered at some other location. Engineering and robotics literature typically deal with passive rotations, where the goal is often to reason about physical entities that are not always directly controlled.

Confusingly, both kinds of rotations are typically denoted simply as R, and readers are expected to infer passive or active convention. However, active and passive rotations are actually the transpose of one another, as

$$R_{active} = R_{passive}^\top.$$

This fact makes literature which may use one or the other convention quickly accessible to any and all readers who know the rules and make note of the choice in convention.

Readers should also note that our notation is not well-suited for active transformations. If you try to perform an active rotation in our notation, the coordinate frames will not line up, which can be very confusing. This is a potential shortcoming of this notation if the desire is to perform a lot of active transformations.

While rotation matrices are probably the most straight-forward method to represent rotations, they have some shortcomings. The first is that they require 9 parameters to describe 3 degrees of freedom. This causes larger memory requirements than strictly necessary, and requires more operations than necessary when concatenating matrices With modern computing resources, this is less of a problem, however the biggest disadvantage is that numerical errors can accumulate in rotation

matrices and cause unwanted scaling and shearing as the matrix loses orthonormality. This error can be corrected by Gram-Schmidt orthogonalization, but doing so can be computationally expensive. As a result, rotation matrices are still much more efficient than an Euler angle representation, but they are not as efficient as unit quaternions.

4.4 Unit Quaternions

A quaternion, is a hyper-complex number with four elements, commonly written[6]

$$\mathbf{q} = q_0 + q_x i + q_y j + q_z k.$$

Here you can see two additional imaginary numbers, j and k. These new imaginary numbers follow the following rules

$$i^2 = j^2 = k^2 = ijk = -1,$$

as well as the right-hand rule

$$ij = k, \quad ji = -k \quad ...$$

Which gives rise to the definition of quaternion multiplication as[7]

$$\mathbf{q}^a \cdot \mathbf{q}^b = \begin{array}{c} q_0^a q_0^b - q_x^a q_x^b - q_y^a q_y^b - q_z^a q_z^b \\ + \left(q_0^a q_x^b + q_x^a q_0^b + q_y^a q_z^b - q_z^a q_y^b\right) i \\ + \left(q_0^a q_y^b - q_x^a q_z^b + q_y^a q_0^b + q_z^a q_x^b\right) j \\ + \left(q_0^a q_z^b + q_x^a q_y^b - q_y^a q_x^b + q_z^a q_0^b\right) k. \end{array}$$

When representing rotations, quaternions must also satisfy the additional constraint that the l_2 norm of all four elements is equal to 1, hence the specification *unit* quaternions, and the use of the group $\mathcal{S}^3$, the unit sphere in four dimensions.

Unit quaternions, unlike rotation matrices, multiply from left to right, as in[8]

$$\mathbf{q}_a^c = \mathbf{q}_a^b \cdot \mathbf{q}_b^c.$$

For convenience, we will sometimes refer to the complex portion of the quaternion as the vector

$$\vec{\mathbf{q}} = \begin{bmatrix} q_x & q_y & q_z \end{bmatrix}$$

and write quaternions as the tuple of the real and complex portion, as in

$$\mathbf{q} = \begin{pmatrix} q_0 \\ \vec{\mathbf{q}} \end{pmatrix}.$$

[6] Sometimes q_w is used instead of q_o.

[7] To derive this yourself, all you have to do is distribute out the product and apply the rules above.

[8] Because of the reverse order of concatenation, there is actually another convention for quaternions, first used by NASA JPL, where the quaternion imaginary numbers follow the *left-hand* rule instead of the right hand rule. This change makes unit quaternions concatenate right-to-left (to match rotation matrices). The left-handed definition is known as JPL notation, whereas the right-handed definition presented here is known as Hamilton notation. While this change may seem innocent on the surface, it has huge implications when we start talking about Lie Groups. Modern scientific literature in robotics and computer vision that use quaternions seem to almost always use Hamilton notation, although JPL is used in some early quaternion papers relating to spacecraft attitude observers.

Algorithm 3 Algorithm for converting Rotation matrix to quaternion

$\delta \leftarrow \mathrm{tr}\,(R)$
if $\delta > 0$ **then**
  $s \leftarrow 2\sqrt{\delta + 1}$
  $\mathbf{q} \leftarrow \frac{s}{4} + \frac{1}{s}\left(R_{23} - R_{32}\right) i + \frac{1}{s}\left(R_{31} - R_{13}\right) j + \frac{1}{s}\left(R_{12} - R_{21}\right) k$
else if $R_{11} > R_{22}$ **and** $R_{11} > R33$ **then**
  $s \leftarrow 2\sqrt{1 + R_{11} - R_{22} - R_{33}}$
  $\mathbf{q} \leftarrow \frac{1}{s}\left(R_{23} - R_{32}\right) + \frac{s}{4} i + \frac{1}{s}\left(R_{21} + R_{12}\right) j + \frac{1}{s}\left(R_{31} + R_{13}\right) k$
else if $R_{22} > R_3 3$ **then**
  $s \leftarrow 2\sqrt{1 + R_{22} - R_{11} - R_{33}}$
  $\mathbf{q} \leftarrow \frac{1}{2}\left(R_{31} - R_{13}\right) + \frac{1}{s}\left(R_{21} + R_{12}\right) i + \frac{s}{4} j + \frac{1}{s}\left(R_{32} + R_{23}\right) k$
else
  $s \leftarrow 2\sqrt{1 + R_{33} - R_{11} - R_{22}}$
  $\mathbf{q} \leftarrow \frac{1}{s}\left(R_{12} - R_{21}\right) + \frac{1}{s}\left(R_{31} + R_{13}\right) i + \frac{1}{s}\left(R_{32} + R_{13}\right) j + \frac{s}{4} k$
end if

This means that the somewhat unwieldy quaternion multiplication expression above can be re-written as the matrix-like operation

$$\mathbf{q}^a \cdot \mathbf{q}^b = \begin{pmatrix} -q_0^a & \left(-\vec{\mathbf{q}}^a\right)^\top \\ \vec{\mathbf{q}}^a & q_0^a I + \lfloor \vec{\mathbf{q}}^a \rfloor_\times \end{pmatrix} \begin{pmatrix} q_0^b \\ \vec{\mathbf{q}}^b \end{pmatrix}. \tag{4.2}$$

Splitting the quaternion into a vector and scalar portion also allows us to interpret the quaternion in an axis-angle fashion. Given a unit vector describing an axis of rotation $\mathbf{r}$ and angle about that axis θ in radians, the equivalent quaternion is given as

$$\vec{\mathbf{q}} = \mathbf{r} \sin\left(\frac{\theta}{2}\right), \quad q_0 = \cos\left(\frac{\theta}{2}\right). \tag{4.3}$$

This axis-angle interpretation makes it somewhat feasible to interpret a unit quaternion by just looking at numbers. The axis-angle intepretation also makes obvious that inverting a quaternion is the same as computing the complex conjugate

$$\mathbf{q}^{-1} = \bar{\mathbf{q}} = \begin{pmatrix} q_0 \\ -\vec{\mathbf{q}} \end{pmatrix}.$$

There is actually another interesting thing about quaternions we can see in Eq. 4.3. If you think about it, you can actually see that unit quaternions are a double cover[9] of $SO\,(3)$. This is because for each $(\mathbf{r}, \theta)$, $(-\mathbf{r}, -\theta)$ describes the same rotation.

Finally, to convert from a quaternion to a rotation matrix, we employ the following expression

$$R\,(\mathbf{q}) = (2q_0 - 1)\, I - 2q_0 \lfloor \vec{\mathbf{q}} \rfloor_\times + 2\vec{\mathbf{q}}\vec{\mathbf{q}}^\top,$$

and to convert a rotation matrix into its quaternion expression, we can use Algorithm 3.

There are couple of ways to rotate a vector with a quaternion. The first, is to compute the rotation matrix from the quaternion and then use it to rotate the vector, as in

[9] This just means that for every rotation matrix, there are two unit quaternions.

$$\mathbf{r}^b = R\left(\mathbf{q}_a^b\right)\mathbf{r}^a. \tag{4.4}$$

This, however, is very inefficient. Another way to rotate a vector is with quaternion multiplication. We pad the vector with an extra zero in the real component[10] and perform pre- and post-multiplication by the quaternion and its complex conjugate, as in

$$\mathbf{r}^b = \left(\mathbf{q}_a^b\right)^{-1} \cdot \begin{pmatrix} 0 \\ \mathbf{r}^a \end{pmatrix} \cdot \mathbf{q}_a^b. \tag{4.5}$$

While this is more efficient than first computing the rotation matrix and then performing the matrix-vector multiplication there is quite a bit of redundant effort. If you remove this redundant effort, then both Eq. 4.5 and Eq. 4.4 condense into the following expression:

$$\begin{aligned} \mathbf{r}^b &= \mathrm{rot}\left(\mathbf{q}_a^b, \mathbf{r}^a\right) \\ &= \mathbf{r}^a + q_0\mathbf{t} + \lfloor \mathbf{t} \rfloor_\times \vec{\mathbf{q}}_a^b, \qquad \mathbf{t} = 2\lfloor \mathbf{r}^a \rfloor_\times \vec{\mathbf{q}}_a^b. \end{aligned}$$

If implemented efficiently, this method can be just as efficient as the matrix-vector multiplication used when rotating a vector with a rotation matrix.

4.5 Relationship between $SU(2)$ and Unit Quaternions

Unit quaternions are isomorphic[11] to 2×2 complex matrices with $\det = 1$. If we replace the imaginary numbers with the following matrices,

$$\mathbf{1} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \qquad \mathbf{i} = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}$$
$$\mathbf{j} = \begin{pmatrix} 0 & i \\ i & 0 \end{pmatrix}, \qquad \mathbf{k} = \begin{pmatrix} i & 0 \\ 0 & -i \end{pmatrix}$$

then we can re-write the quaternion as the following 2×2 matrix:

$$\begin{aligned} \mathbf{q} &= q_0\mathbf{1} + q_x\mathbf{i} + q_y\mathbf{j} + q_z\mathbf{k} \\ &= \begin{pmatrix} q_0 + q_z i & q_x + q_y i \\ -q_x + q_y i & q_0 - q_z i \end{pmatrix}. \end{aligned}$$

We can see that this representation still follows the original rules for the unit quaternion ($\mathbf{ijk} = -\mathbf{1}$), and instead of unit norm, the matrix has unit determinant and we get the additional constraint, $\mathbf{q}^\dagger\mathbf{q} = 1$. This means that unit quaternions when written as 2×2 matrices fulfill all the conditions of $SU(2)$, the special unitary group. This will show up again when we talk about Lie groups.

[10] A quaternion with no real component is known as a *pure* quaternion.
[11] This means there for every unit quaternion, there is exactly one member of $SU(2)$.

4.6 Body-Centric vs Inertial Representation

In robotics or computer vision, we are typically concerned with rotations so that we can describe the movement of physical objects. When we do this, it turns out that choice of coordinate frame also introduces some interesting consequences that may or may not be intentional. Rotational dynamics of any arbitrary coordinate frame j with respect to another coordinate frame i is given by

$$\dot{R}_i^j = R_i^j \left\lfloor \boldsymbol{\omega}_{i/j}^i \right\rfloor_\times .$$

However, let's study a concrete example: the dynamics of some rigid body undergoing pure rotation. If we define some inertial frame I and the moving, body frame as b, then the dynamics of the *body-centric* rotation matrix (i.e. body→inertial), we get

$$\dot{R}_b^I = R_b^I \left\lfloor \boldsymbol{\omega}_{b/I}^b \right\rfloor_\times , \tag{4.6}$$

and the *inertial* rotation matrix is (inertial→body) evolves according to

$$\dot{R}_I^b = R_I^b \left\lfloor \boldsymbol{\omega}_{I/b}^I \right\rfloor_\times . \tag{4.7}$$

This is all fine, except that we typically measure angular velocity in the body coordinate frame $\boldsymbol{\omega}_{b/I}^b$. Therefore, it is much more convenient to express Eq. 4.7 in terms of body angular rates. If we do this, then we get

$$\begin{aligned} \dot{R}_I^b &= R_I^b \left\lfloor -R_b^I \boldsymbol{\omega}_{b/I}^b \right\rfloor_\times && \text{(Negate and rotate } \boldsymbol{\omega}\text{)} \\ &= -\left\lfloor \boldsymbol{\omega}_{b/I}^b \right\rfloor_\times R_I^b, && \text{(Eq. 2.2)} \end{aligned}$$

which can also be easily derived as the transpose of Eq 4.6. The desire to express our angular rates in terms of the body frame induces a similar effect in unit quaternion dynamics, except that it is reversed due to the reversed order of concatenation:

$$\begin{aligned} \dot{\mathbf{q}}_I^b &= \frac{1}{2}\mathbf{q}_I^b \cdot \begin{pmatrix} 0 \\ \boldsymbol{\omega}_{b/I}^b \end{pmatrix} \\ \dot{\mathbf{q}}_b^I &= -\frac{1}{2}\begin{pmatrix} 0 \\ \boldsymbol{\omega}_{b/I}^b \end{pmatrix} \cdot \mathbf{q}_b^I. \end{aligned} \tag{4.8}$$

Any advantage to expressing attitude in body-centric vs. inertial coordinates will be use-case specific, so both representations are used widely in literature. Because both representations are often used, however, it can sometimes be confusing to compare two works which may be expressing their attitude differently, especially if neither work is specific about their representation.

4.7 Right vs. Left Invariance of Rotation Dynamics

Another non-trivial consequence of differing attitude representations is whether the kinematics are left or right-invariant (LI or RI). Body-centric rotation kinematics are left-invariant. This means

that if we have some constant matrix $A \in SO(3)$, and we multiply the kinematics (Eq. 4.6) by A on the left, we get a new differential equation with the same form as Eq. 4.6.

$$\frac{d}{dt}\left(AR_b^I\right) = AR_b^I \left\lfloor \boldsymbol{\omega}_{b/I}^b \right\rfloor_\times$$

One can see that we do not get a differential equation of the same form if we multiply both sides of Eq. 4.6 on the *right* by A.

Inertial rotation kinematics, on the other hand, are right-invariant. If we multiply both sides of Eq. 4.7 on the right by A, we get

$$\frac{d}{dt}\left(R_I^b A\right) = -\left\lfloor \boldsymbol{\omega}_{b/I}^b \right\rfloor_\times R_I^b A,$$

which has the same form as Eq. 4.7. A similar effect can be observed in quaternion dynamics, but with the order reversed.

The left-right invariance shows up because of our choice in representing $\boldsymbol{\omega}$ in the body frame. I know this is jumping ahead a bit, but the Adjoint representation will allow us to flip back and forth between left- and right-vectors as necessary, making this distinction also a matter of convenience.

4.8 The Solution to the Rotational Dynamics Equation

Equations 4.6 and 4.7 are first-order matrix differential equations. These are solved with the matrix exponential we discussed in Section 3. The solution to the body-centric equation is

$$R_b^I(t) = R_b^I(t_0) \exp\left(\left\lfloor \boldsymbol{\omega}_{b/I}^b \right\rfloor_\times t\right)$$

and the inertial dynamics are solved with

$$R_I^b(t) = \exp\left(-\left\lfloor \boldsymbol{\omega}_{b/I}^b \right\rfloor_\times t\right) R_I^b(t_0).$$

Quaternion dynamics are also easily solved in a similar fashion, as in

$$\mathbf{q}_b^I(t) = \exp\begin{pmatrix} 0 \\ -\boldsymbol{\omega}_{b/I}^b t \end{pmatrix} \cdot \mathbf{q}_b^I(t_0)$$

$$\mathbf{q}_I^b(t) = \mathbf{q}_I^b(t_0) \cdot \exp\begin{pmatrix} 0 \\ \boldsymbol{\omega}_{b/I}^b t \end{pmatrix}.$$

The infinite series Eq. 3.3 can be used to compute these directly. However, efficient closed-form implementations are derived in Section 7.

5 Transformations

Now we venture into the world of rigid body transformations. These allow us to represent both translation and rotation simultaneously. As with rotations, there are two popular representations that I will consider in this document, the first is matrix-based, the set of 4×4 matrices, commonly

known as $SE(3)$. The second (like rotations) will be based on complex numbers, dual unit quaternions. There are actually other representations, but we will see later all of these groups share the same Lie Algebra! Therefore, we can choose the representation that we feel most comfortable with, or is the most efficient, etc... It really doesn't matter, which is a great thing.

5.1 Homogeneous Transform Matrices

Homogeneous transforms take the form of 4×4 matrices. These are members of the Special Euclidean group of 3 dimensions, or $SE(3)$. This is by far the most common representation in robotics; it's efficient, and the Lie group/Lie algebra is probably the most straight-forward. A homogeneous transform matrix has the following block structure:

$$T_a^b = \begin{bmatrix} R_a^b & \mathbf{t}_{b/a}^a \\ 0 & 1 \end{bmatrix},$$

where you have the rotation matrix in the upper left corner and the translation vector in the third column.

Let's walk through in a block-wise format what happens when two transforms are multiplied together. All we need to do is follow the matrix multiplication rules and see how the coordinate frames work out. I found this to be a very useful exercise when I first got started. It really hit home to me how the matrices concatenate in the same way as rotation matrices, and the potentially confusing parameterization of the translation vector. This procedure looks like the this:

$$\begin{aligned} T_a^c &= T_b^c \cdot T_a^b \\ &= \begin{bmatrix} R_b^c & \mathbf{t}_{c/b}^c \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} R_a^b & \mathbf{t}_{b/a}^b \\ 0 & 1 \end{bmatrix} \\ &= \begin{bmatrix} R_b^c R_a^b & R_b^c \mathbf{t}_{b/a}^b + \mathbf{t}_{c/b}^c \\ 0 & 1 \end{bmatrix} \\ &= \begin{bmatrix} R_a^c & \mathbf{t}_{c/a}^c \\ 0 & 1 \end{bmatrix}. \end{aligned}$$

So we see, we get what we expect: all the coordinate frames land the way we want.

We can also perform the same exercise for taking the inverse. Remember that the inverse of a matrix is $1/\det(A)\mathrm{Adj}(A)$, where $\mathrm{Adj}(A)$ refers to the adjugate of A. Because it gets a little nasty, I'm going to skip actually computing the adjugate, but if you work it out, you can see how this apparently super convenient result just pops out:

$$
\begin{aligned}
\left(T_a^b\right)^{-1} &= \begin{bmatrix} R_a^b & \mathbf{t}_{b/a}^b \\ 0 & 1 \end{bmatrix}^{-1} \\
&= \frac{1}{\det\left(T_a^b\right)} \operatorname{Adj}\left(T_a^b\right) \\
&= \frac{1}{1} \begin{bmatrix} \left(R_a^b\right)^\top & -\left(R_a^b\right)^\top \mathbf{t}_{b/a}^b \\ 0 & 1 \end{bmatrix} \\
&= \begin{bmatrix} R_b^a & -R_b^a \mathbf{t}_{b/a}^b \\ 0 & 1 \end{bmatrix} \\
&= \begin{bmatrix} R_b^a & \mathbf{t}_{a/b}^a \\ 0 & 1 \end{bmatrix} \\
&= T_b^a.
\end{aligned}
$$

Boom.

If you didn't notice, it's really important to note is that the translation vector is defined in the *destination* frame, rather than the origin frame of the rotation matrix. This isn't a problem, but it was very confusing for me at first.

5.2 Passive and Active Transformations

The same notion of passive and active transformations discussed with rotations applies here. To perform a passive transformation with $SE(3)$, we just pad the vector to be rotated with an extra 1 at the bottom, and multiply by our transform matrix. Just for completeness, let us walk through what happens when we do this in a block-wise format

$$
\begin{aligned}
\mathbf{r}_{b/p}^b &= T_a^b \cdot \mathbf{r}_{a/p}^a \\
&= \begin{bmatrix} R_a^b & \mathbf{t}_{b/a}^b \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \mathbf{r}_{a/p}^a \\ 1 \end{bmatrix} \\
&= \begin{bmatrix} R_a^b \mathbf{r}_{a/p}^a + \mathbf{t}_{b/a}^b \\ 1 \end{bmatrix} \\
&= \mathbf{r}_{b/p}^b.
\end{aligned}
$$

Doing this at least once is another exercise that I found to be very helpful when first trying to use $SE(3)$.

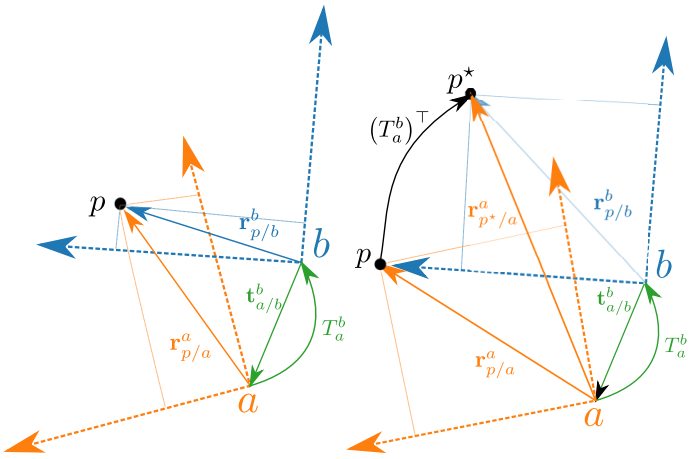

Fig. 5.1: Illustration of a passive transformation (left) and an active transformation (right)

An active transformation occurs by multiplying by the inverse of the passive transform as shown in Figure 5.1. Just as with rotations, If you walk through the coordinate frames of an active transformation, you'll see that the coordinate frames don't line up right in our notation. This is because the active transformation does not preserve the location. Instead we have defined a new location, based on the coordinates of p in frame a projected into frame b. I have tried to represent this with the $p^\star$ notation in Figure 5.1.

Transform matrices are an obvious extension to rotation matrices. However, just like rotation matrices, the complex number representation is more efficient. This brings us to our next transform parameterization: dual unit quaternions.

5.3 Dual Unit Quaternions

The use of dual unit quaternions is another approach to representing a rigid body transform that relies on complex numbers. As one might expect, it can also be considered as two copies of $SU(2)$ as well. Dual quaternions are more popular in the computer graphics and physics community, and for the same reasons that quaternions often out-perform rotation matrices, dual quaternions are often yield the most efficient representation for rigid body transformations.

Before I jump into dual quaternions, I'm going to first introduce dual numbers. Dual numbers are similar in many ways to imaginary numbers. The idea is that we define a scalar ϵ as a mathematical construct that splits a numerical representation into two components. For example, we might have the dual number

$$z = r + d\epsilon,$$

where r is the real part, and d is the dual part. We also define ϵ as having special rules where $\epsilon^2 = 0$, but $\epsilon \neq 0$. Again, this is similar to complex numbers where we use i to distinguish imaginary components from real components. The dual operator ϵ is used in much the same way. It should be noted that ϵ in this context is just a special symbol. It is *not* related to small perturbations, as it is used in other areas of mathematics. Trying to interpret it in this way can sometimes be confusing.

Dual numbers follow these basic rules. If two dual numbers are added, then the real part and dual part are simply summed separately (just like imaginary numbers). For example,

$$x = a + b\epsilon, \quad y = c + d\epsilon$$

$$x + y = (a + c) + (b + d)\,\epsilon.$$

Multiplication also is performed similarly to complex multiplication, but with the rule that $\epsilon^2 = 0$. For example,

$$\begin{aligned} x \cdot y &= ac + ad\epsilon + bc\epsilon + bd\epsilon^2 \\ &= ac + (ad + bc)\,\epsilon. \end{aligned}$$

Finally, dual variables have their own concept of conjugate, given in the expected way and denoted as $(\cdot)^*$. For example,

$$\begin{aligned} z &= r + d\epsilon \\ z^* &= r - d\epsilon. \end{aligned}$$

If r and d are also complex variables, then we actually have three forms of conjugation. Imaginary conjugation, dual conjugation, and imaginary-dual conjugation, given as

$$\begin{aligned} \bar{z} &= \bar{r} + \bar{d}\epsilon && \text{imag} \\ z^* &= r - d\epsilon. && \text{dual} \\ \bar{z}^* &= \bar{r} - \bar{d}\epsilon. && \text{imag+dual} \end{aligned}$$

Now we are ready to talk about dual quaternions, $\mathbb{D}\mathcal{S}^3$ and $\mathbb{D}SU\,(2)$[12]. We can represent 3D transforms with the dual unit quaternion pair

$$\Gamma = \mathbf{q}_r + \epsilon\mathbf{q}_d,$$

where $\mathbf{q}_r$ is called the real part and $\mathbf{q}_d$ is the dual part. The real part contains the rotation, while the dual part is one-half the pure quaternion containing the translation half rotated, as in[13]

$$\begin{aligned} \mathbf{q}_r &= \mathbf{q}_a^b \\ \mathbf{q}_d &= \frac{1}{2} \begin{pmatrix} 0 \\ \mathbf{t}_{b/a}^a \end{pmatrix} \cdot \mathbf{q}_a^b. \end{aligned}$$

If we combine the quaternion algebra operations with the dual number, we get dual quaternion arithmetic. Dual quaternion multiplication is given as one might expect, with

[12] The $\mathbb{D}$ here refers to the dual formulation.

[13] This may seem like a weird parameterization, and some (like myself) may question if there is a simpler way, rather than handling this half-rotated vector. It comes from using the simplest form of the underlying Lie Algebra, and finding the natural expression of that group in dual quaternions. Trying to form a convenient group and mangling the underlying algebra to match results in losing a lot of the beauty of Lie Theory. It's better to just use the natural expression, starting at the algebra and working up.

$$\Gamma_1 \circ \Gamma_2 = \mathbf{q}_{1r} \cdot \mathbf{q}_{2r} + (\mathbf{q}_{1r} \cdot \mathbf{q}_{2d} + \mathbf{q}_{1d} \cdot \mathbf{q}_{2r})\,\epsilon.$$

Let's walk through this together and make sure the coordinate frames all line up. Before we do this, however, we need the following trick. If we take the quaternion rotation formula, and split it up, we get an expression for dealing with half-rotated vectors:

$$\begin{aligned} \mathbf{t}^b &= \left(\mathbf{q}_a^b\right)^{-1} \cdot \mathbf{t}^a \cdot \mathbf{q}_a^b \\ \mathbf{q}_a^b \cdot \mathbf{t}^b &= \mathbf{q}_a^b \cdot \left(\mathbf{q}_a^b\right)^{-1} \cdot \mathbf{t}^a \cdot \mathbf{q}_a^b \\ \mathbf{q}_a^b \cdot \mathbf{t}^b &= \mathbf{t}^a \cdot \mathbf{q}_a^b. \end{aligned} \tag{5.1}$$

Now can work through the dual quaternion arithmetic, and see how dual quaternion multiplication results in concatenating transforms.

$$\begin{aligned} \Gamma_a^b \circ \Gamma_b^c &= \mathbf{q}_a^b \cdot \mathbf{q}_b^c + \frac{\epsilon}{2}\left(\mathbf{q}_a^b \cdot \begin{pmatrix} 0 \\ \mathbf{t}_{c/b}^b \end{pmatrix} \cdot \mathbf{q}_b^c + \begin{pmatrix} 0 \\ \mathbf{t}_{b/a}^a \end{pmatrix} \cdot \mathbf{q}_a^b \cdot \mathbf{q}_b^c\right) \\ &= \mathbf{q}_a^c + \frac{\epsilon}{2}\left(\begin{pmatrix} 0 \\ \mathbf{t}_{c/b}^a \end{pmatrix} \cdot \mathbf{q}_a^b \cdot \mathbf{q}_b^c + \begin{pmatrix} 0 \\ \mathbf{t}_{b/a}^a \end{pmatrix} \cdot \mathbf{q}_a^b \cdot \mathbf{q}_b^c\right) && \text{(Eq. 5.1)} \\ &= \mathbf{q}_a^c + \frac{\epsilon}{2}\left(\begin{pmatrix} 0 \\ \mathbf{t}_{c/a}^a \end{pmatrix} \cdot \mathbf{q}_a^c\right). \end{aligned}$$

Inverting (or computing the complex conjugate of) a dual quaternion is as simple as inverting the two components. To work through this operation, line by line, we will also need to use the trick from Eq. 5.1.

$$\begin{aligned} \left(\Gamma_a^b\right)^{-1} &= \left(\mathbf{q}_a^b\right)^{-1} + \frac{\epsilon}{2}\left(\begin{pmatrix} 0 \\ \mathbf{t}_{b/a}^a \end{pmatrix} \cdot \mathbf{q}_a^b\right)^{-1} \\ &= \mathbf{q}_b^a + \frac{\epsilon}{2}\left(\left(\mathbf{q}_a^b\right)^{-1} \cdot \begin{pmatrix} 0 \\ \mathbf{t}_{b/a}^a \end{pmatrix}^{-1}\right) \\ &= \mathbf{q}_b^a + \frac{\epsilon}{2}\left(\mathbf{q}_b^a \cdot \begin{pmatrix} 0 \\ \mathbf{t}_{a/b}^a \end{pmatrix}\right) \\ &= \mathbf{q}_b^a + \frac{\epsilon}{2}\left(\begin{pmatrix} 0 \\ \mathbf{t}_{a/b}^b \end{pmatrix} \cdot \mathbf{q}_b^a\right) && \text{(Eq. 5.1)} \\ &= \Gamma_b^a \end{aligned}$$

Transforming a vector is done by creating a pure quaternion (with no rotational component) from the translation vector, then pre- and post-multiplying by the dual quaternion and its dual-complex conjugate. If we work through the steps we can see how this works:

$$
\begin{aligned}
\mathbf{r}_{p/b}^{b} &= \left(\bar{\Gamma}_{a}^{b}\right)^{*} \circ \mathbf{r}_{p/a}^{a} \circ \Gamma_{a}^{b} \\
&= \left[\mathbf{q}_{b}^{a} - \frac{\epsilon}{2}\left(\mathbf{t}_{a/b}^{b} \cdot \mathbf{q}_{b}^{a}\right)\right] \circ \left[\mathbf{1} + \epsilon \mathbf{r}_{p/a}^{a}\right] \circ \left[\mathbf{q}_{a}^{b} + \frac{\epsilon}{2}\left(\mathbf{t}_{b/a}^{a} \cdot \mathbf{q}_{a}^{b}\right)\right] \\
&= \left[\mathbf{q}_{b}^{a} - \frac{\epsilon}{2}\left(\mathbf{t}_{a/b}^{b} \cdot \mathbf{q}_{b}^{a}\right)\right] \circ \left[\left(\mathbf{1} \cdot \mathbf{q}_{a}^{b}\right) + \frac{\epsilon}{2}\left(\mathbf{1} \cdot \left(\mathbf{t}_{b/a}^{a} \cdot \mathbf{q}_{a}^{b}\right) + 2\mathbf{r}_{p/a}^{a} \cdot \mathbf{q}_{a}^{b}\right)\right] \\
&= \left(\mathbf{q}_{b}^{a} \cdot \mathbf{q}_{a}^{b}\right) + \frac{\epsilon}{2}\left(\mathbf{q}_{b}^{a} \cdot \left(\left(-\mathbf{t}_{b/a}^{a} \cdot \mathbf{q}_{a}^{b}\right) + 2\mathbf{r}_{p/a}^{a} \cdot \mathbf{q}_{a}^{b}\right) + \left(\mathbf{t}_{a/b}^{b} \cdot \mathbf{q}_{b}^{a}\right) \cdot \mathbf{q}_{a}^{b}\right) \\
&= \mathbf{1} + \frac{\epsilon}{2}\left(\mathbf{q}_{b}^{a} \cdot \left(-\mathbf{t}_{b/a}^{a} \cdot \mathbf{q}_{a}^{b}\right) + \mathbf{q}_{b}^{a} \cdot 2\mathbf{r}_{p/a}^{a} \cdot \mathbf{q}_{a}^{b} + \left(\mathbf{t}_{a/b}^{b} \cdot \mathbf{q}_{b}^{a}\right) \cdot \mathbf{q}_{a}^{b}\right) \\
&= \mathbf{1} + \frac{\epsilon}{2}\left(-\mathbf{t}_{b/a}^{b} + 2\mathbf{r}_{p/a}^{b} + \mathbf{t}_{a/b}^{b}\right) \\
&= \mathbf{1} + \frac{\epsilon}{2}\left(2\mathbf{r}_{p/a}^{b} - 2\mathbf{t}_{b/a}^{b}\right) \\
&= \mathbf{1} + \epsilon\left(\mathbf{r}_{p/b}^{b}\right).
\end{aligned}
$$

5.4 Dynamics and Invariance

As with rotations, the same concept of left- and right-invariance also applies to transforms. If we parameterize rigid body dynamics in body-centric coordinates, we get the left-invariant dynamics

$$
\dot{T}_{b}^{I} = T_{b}^{I} \begin{pmatrix} \left\lfloor \boldsymbol{\omega}_{b/I}^{b} \right\rfloor_{\times} & \mathbf{v}_{b/I}^{b} \\ 0 & 1 \end{pmatrix}.
$$

and the inertial coordinates have right-invariant dynamics, given as

$$
\dot{T}_{I}^{b} = \begin{pmatrix} -\left\lfloor \boldsymbol{\omega}_{b/I}^{b} \right\rfloor_{\times} & -\mathbf{v}_{b/I}^{b} \\ 0 & 1 \end{pmatrix} T_{I}^{b}
$$

Like ordinary quaternions and rotation matrices, dual quaternions also concatenate opposite $SE\left(3\right)$, so the order of the dynamic equations is flipped, as in

$$
\begin{aligned}
\dot{\Gamma}_{b}^{I} &= -\frac{1}{2}\left(\begin{pmatrix} 0 \\ \boldsymbol{\omega}_{b/I}^{b} \end{pmatrix} + \epsilon \begin{pmatrix} 0 \\ \mathbf{v}_{b/I}^{b} \end{pmatrix}\right) \circ \Gamma_{b}^{I}\left(t_{0}\right) \\
\dot{\Gamma}_{I}^{b} &= \frac{1}{2}\Gamma_{I}^{b} \circ \left(\begin{pmatrix} 0 \\ \boldsymbol{\omega}_{b/I}^{b} \end{pmatrix} + \epsilon \begin{pmatrix} 0 \\ \mathbf{v}_{b/I}^{b} \end{pmatrix}\right).
\end{aligned}
$$

6 Lie Groups

Lie Group theory is a very old discipline deeply rooted in mathematics and theoretical physics. It was developed to better understand and model nature at the most fundamental levels. In fact, much of particle physics can be modeled using Lie group theory, including special relativity and quantum dynamics. While we don't often need to properly model Minkowski space, we can learn a lot about modeling physical systems by piggybacking on physics literature.

If you can't tell from the last 4 sections, we end up doing a lot of reasoning about coordinate frames in robotics. We often need to reason about how they move, and how they relate to one another. We also find ourselves trying to infer things about these coordinate frames given noisy sensor measurements. There are a variety of ways that we could do this, but the real problems show up when we need to this in real time under realistic computational limitations.

To meet actual computational requirements, we generally want to be able to leverage linear algebra and all of its tools. Unfortunately, however, coordinate frame transformations are not vectors, so it's not completely obvious how we can use these tools in a principled manner. Luckily, we have Lie group theory that can bridge this gap. We will see in this section how we can use Lie Groups to take non-vector group objects, and map them into a vector space where we can perform linear algebra and reason about things in an efficient manner, then map our results back into the group so we can do useful things.

As a further motivating example, consider the case where we might want to represent our uncertainty of some estimate of a rotation matrix. In general, assuming that our uncertainty of some vector quantity $\mathbf{x}$ can be represented with a multivariate Gaussian distribution, the covariance of this distribution is computed as

$$\Sigma_{\mathbf{x}} = E\left[(\mathbf{x} - \hat{\mathbf{x}})(\mathbf{x} - \hat{\mathbf{x}})^\top\right],$$

where $\hat{\mathbf{x}}$ was our best estimate and $\mathbf{x}$ was the true value. However, if we consider putting our rotation matrix R into this equation, what does $R - \hat{R}$ even mean? Subtraction is not a sensible operation when talking about rotation matrices. First off, it would no longer be a rotation matrix, as it would not longer have unit determinant, and second, it would have nine parameters, when we expect there to only be three[14]. Lie group theory solves this problem for us, as we will soon find out.

6.1 Group Theory

Before we launch into Lie groups, let us consider groups generically. A group is simply a set of members which are *closed* under some group operator. This means that if I had two group members g_1 and g_2 and I act on these members with the group operator $\circ$, then the result should also be a part of the group. For example,

$$g_1 \circ g_2 = g_3, \qquad g_1, g_2, g_3 \in G.$$

We use groups all the time, maybe without knowing it. Some commonly used groups in robotics are vectors (closed under vector addition), rotation matrices (closed under matrix multiplication), and quaternions (closed under quaternion multiplication). Some more exotic groups could include binary vectors closed under bit-wise-XOR, positive definite matrices closed under matrix addition and integers closed under addition.[15]

[14] Using Euler angles doesn't get you out of this, either. It's just more subtle. Think about what it really means to subtract two "vectors" of euler angles. The intermediate axes defining the Euler angle parameterization will not line up, so it's not a sensible operation.

[15] In case you need another reason to stop using Euler angles, Euler angles aren't even a group, unless, of course, the group operator is defined as "convert to rotation matrix, multiply, and then convert back to Euler angles"

6.2 Lie Group Definition

A Lie group is a group that is also a differentiable manifold. What this means is that every group element A induces a map from one group element B to another $C = A \circ B$ and that this mapping is differentiable. Of all the groups mentioned above, vectors, rotation matrices, quaternions and positive definite matrices are also Lie Groups, while binary vectors closed under XOR and integers are not, because the mappings are not differentiable.

Each Lie groups has an associated Lie algebra, typically indicated by using the fraktur font, (i.e. $\mathfrak{so}(3)$). A Lie algebra is a vector space $\mathfrak{g}$ equipped with a binary operation called the *Lie bracket,* $[,]$. For matrix Lie algebras, the Lie bracket is given as

$$[A, B] = AB - BA.$$

The Lie bracket abides by the following rules:

- Bilinearity: $[aX + bY, Z] = a[X, Z] + b[Y, Z]$
- Anticommutativity: $[X, Y] = -[Y, X] \quad \forall X, Y \in \mathfrak{g}$
- The Jacobi Identity: $[X, [Y, Z]] + [Z, [X, Y]] + [Y, [Z, X]] = 0 \quad \forall X, Y, Z \in \mathfrak{g}$

The Lie Algebra defines something equivalent to the basis of the Lie Group, except instead of basis vectors, we have *generators*. The generators are the building blocks of the group, and typically encode the degrees of freedom in a orthonormal way. Each member of any Lie Algebra can be expressed as the exponential of a linear combination of the generators.

There is an important theorem that we will use a little later called the Baker-Campbell-Hausdorff Theorem (BCH). This theorem states that the solution Z to

$$e^X e^Y = e^Z,$$

can be expressed as a power series involving commutators X and Y in the Lie bracket. The first couple terms of this series is given as

$$Z = X + Y + \frac{1}{2}[X, Y] + \frac{1}{12}[X, [X, Y]] - \frac{1}{12}[Y, [X, Y]] + \ldots.$$

This formula is central to many proofs in the Lie group-Lie algebra correspondence.

6.3 The Generators of $SO(3)$

The easiest way to come up with the generators of a Lie Group is to consider the infinitesimal transformations that the group can accommodate. For example, the generators of $\mathfrak{so}(3)$ are given by

$$J_1 = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{pmatrix} \qquad J_2 = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ -1 & 0 & 0 \end{pmatrix} \qquad J_3 = \begin{pmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}.$$

Obviously, there is a lot more rigor here than I can really provide, not being a mathematician myself[16]. However, we can prove that these generators are correct with this simple exercise. We know that all elements of $SO(3)$ must follow these conditions:

$$A^\top A = I \qquad\qquad \det(A) = 1.$$

We also know that every element of $SO(3)$ can be written in terms of some linear combination of generators J as

$$A = \exp(\theta J).$$

We can back out the structure of J through putting it into the definition conditions of $SO(3)$. The first condition yields

$$\begin{aligned} A^\top A &= I \\ \exp(\theta J)^\top \exp(\theta J) &= I \\ J^\top + J &= 0. \end{aligned}$$

If we use the second condition (and the identity $\det(\exp(A)) = \exp(\operatorname{tr}(A))$) we see

$$\begin{aligned} \det(A) &= 1 \\ \det\left(e^{\theta J}\right) &= 1 \\ \operatorname{tr}(J) &= 0. \end{aligned}$$

So, the generators for the algebra $\mathfrak{so}(3)$ must be anti-symmetric, traceless 3×3 matrices. Three linearly independent matrices fulfilling these conditions *and* the rules for a lie algebra are[17]

$$J_1 = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{pmatrix} \qquad J_2 = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ -1 & 0 & 0 \end{pmatrix} \qquad J_3 = \begin{pmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}.$$

Therefore, these must be valid generators of $\mathfrak{so}(3)$.

As mentioned before, all members of a Lie group can be created by computing the exponential of a linear combination of the generators. Furthermore, because the generators are orthogonal, $\mathfrak{so}(3)$ is isomorphic to $\mathbb{R}^3$. Therefore, to keep things concise, we can actually form a vector whose coefficients are then multiplied by the generators. In the case of $SO(3)$, this is the same as computing the skew-symmetric matrix of a vector. But in general, the $(\cdot)^\wedge$ and $(\cdot)^\vee$ notation is used to indicate this mapping. For example, for $\mathfrak{so}(3)$:

[16] For example, I'm not sure if the choice to make these line up exactly with the skew-symmetric matrix simply a matter of convenience or something more fundamental.

[17] Specifically, we need generators that obey the right-hand rule through the Lie bracket so they can satisfy the Jacobi Identity.

$$
\begin{aligned}
(\mathbf{v}^\wedge) &= \sum_i J_i \mathbf{e}_i^\top \mathbf{v} \\
&= J_1 \mathbf{v}_x + J_2 \mathbf{v}_y + J_3 \mathbf{v}_z \\
&= \begin{pmatrix} 0 & -\mathbf{v}_z & \mathbf{v}_y \\ \mathbf{v}_z & 0 & -\mathbf{v}_x \\ -\mathbf{v}_y & -\mathbf{v}_x & 0 \end{pmatrix},
\end{aligned}
$$

$$
(\mathbf{v}^\wedge)^\vee = \mathbf{v}.
$$

This notation is used in some literature [1, 5], but not in others [3, 4]. Since the two representations (the matrix algebra and the vector) are isomorphic, the mapping to the matrix Lie Algebra is often skipped notationally, and $\exp(\mathbf{v})$ is written while $\exp(\mathbf{v}^\wedge)$ would technically be more correct. As computing the matrix exponential of $\mathbf{v} \in \mathbb{R}^3$ is actually not possible given the definition above, the mapping of the vector to the matrix Lie Algebra is implied in literature where the mapping is ignored.

6.4 The Generators of $SU(2)/\mathcal{S}^3$

We can follow the same procedure that we performed to derive the generators of $SO(3)$ to get the generators of $SU(2)/\mathcal{S}^3$. First, let's start with the conditions of $SU(2)$

$$
U^\dagger U = U U^\dagger = 1 \qquad\qquad \det(U) = 1.
$$

Now, as before, let us write these conditions in terms of the generators, and use BCH in tandem with the fact that $\left[U^\dagger, U\right] = 0$ to get

$$
\begin{aligned}
U^\dagger U &= 1 \\
(\exp(iJ))^\dagger \exp(iJ) &= 1 \\
\exp\left(-iJ^\dagger\right) \exp(iJ) &= 1 \\
\exp\left(-iJ^\dagger + iJ + \frac{1}{2}\left[J^\dagger, J\right]\right) + \ldots &= 1 \qquad \text{(BCH)} \\
\exp\left(-iJ^\dagger + iJ\right) &= 1 \\
-iJ^\dagger + iJ &= 0 \\
J^\dagger &= J. \qquad (6.1)
\end{aligned}
$$

Now, if we look at the second condition, (again, leveraging $\det(\exp(A)) = \exp(\mathrm{tr}(A))$), we can see

$$
\begin{aligned}
\det(\exp(iJ)) &= 1 \\
\exp(i\mathrm{tr}(J)) &= 1 \\
\mathrm{tr}(J) &= 0. \qquad (6.2)
\end{aligned}
$$

So the generators must be Hermetian (Eq. 6.1), and traceless (Eq. 6.2) complex 2×2 matrices. The standard basis for this set is

$$J_1 = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \qquad J_2 = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} \qquad J_3 = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}.$$

This choice of generators corresponds with infinitesimal rotations in $SU(2)$, and because of the isomorphism between $SU(2)$ and $\mathcal{S}^3$ (unit quaternions), we have also just derived the generators for unit quaternions as well:

$$J_1 = i \qquad J_2 = j \qquad J_3 = k.$$

This means that the $(\cdot)^\wedge$ operations for $SU(2)$ and $\mathcal{S}^3$ are given as

$$\boldsymbol{\omega}^\wedge = \begin{pmatrix} \boldsymbol{\omega}_z & \boldsymbol{\omega}_x - \boldsymbol{\omega}_y i \\ \boldsymbol{\omega}_x + \boldsymbol{\omega}_y i & \boldsymbol{\omega}_z \end{pmatrix},$$

and

$$\boldsymbol{\omega}^\wedge = \begin{pmatrix} 0 \\ \boldsymbol{\omega} \end{pmatrix},$$

respectively, while the $(\cdot)^\vee$operator is defined as the inverse of either operation.

There is a very interesting relationship between $\mathfrak{su}(2)$ and $\mathfrak{so}(3)$. Both algebras have the following property:

$$\begin{aligned} [J_1, J_2] &= J_3 & [J_2, J_1] &= -J_3 \\ [J_2, J_3] &= J_1 & [J_3, J_2] &= -J_1 \\ [J_3, J_1] &= J_2 & [J_1, J_3] &= -J_2. \end{aligned}$$

It turns out, this is indicative of a more fundamental result. One can say that $SU(2)$ and $SO(3)$ have the same Lie algebra, because we define Lie algebras by their Lie bracket. Forgive the extended quotation, but this summary from *Physics From Symmetry[7]* explains this concept in detail, as well as giving a very nice motivation for why we use Lie Algebras in the first place.

> There is precisely one simply-connected Lie group corresponding to each Lie algebra. This simply-connected group can be thought of as the "mother" of all those groups having the same Lie algebra, because there are maps to all other groups with the same Lie algebra from the simply connected group, but not vice versa. We could call it the mother group of this particular Lie algebra, but mathematicians tend to be less dramatic and call it the covering group. All other groups having the same Lie algebra are said to be covered by the simply connected one. $SU(2)$ is the double cover of $SO(3)$. This means there is a two-to-one map from $SU(2)$ to $SO(3)$.
>
> Furthermore, $SU(2)$ is the three sphere, which is a simply connected manifold. Therefore, we have found the "most important" group belonging to the Lie algebra. We can get all other groups belonging to this Lie algebra through maps from $SU(2)$.

> We can now understand what manifold $SO(3)$ is. The map from $SU(2)$ to $SO(3)$ identifies with two points of $SU(2)$, one point of $SO(3)$. Therefore, we can think of $SO(3)$ as the top half of $\mathcal{S}^3$
>
> We can see, from the point of view that Lie groups are manifolds that $SU(2)$ is a more complete object than $SO(3)$. $SO(3)$ is just "part" of the complete object.
>
> ...
>
> To describe nature at the most fundamental level, we must use the covering group, instead of any of the other groups that one can map to from the covering group. We are able to derive the representations of the most fundamental group, belonging to a given Lie algebra, by deriving representations of the Lie algebra. We can then put the matrices representing the Lie algebra elements (the generators) into the exponential function to get matrices representing group elements.
>
> Herein lies the strength of Lie theory. By using pure mathematics we are able to reveal something fundamental about nature. The standard symmetry group of special relativity hides something from us [the spin of elementary particles], because it is not the most fundamental group belonging to this symmetry. The covering group of the Poincaré group is the fundamental group and therefore we will use it to describe nature.

What Schwichtenberg is talking about is the fact that the double cover of the Lorentz group is able to represent the spin of a particle, while the standard Lorentz group cannot. This is analogous to the difference between the use of $\mathcal{S}^3/SU(2)$ versus $SO(3)$. In robotics, we are typically unconcerned with the bottom half of the $\mathcal{S}^3$ unit sphere, so there is no fundamental drawback to using $SO(3)$ like there is in particle physics. However, there is something about nature and pure mathematics that encodes rotations in a double cover, and $SO(3)$ only tells us half of the story.

Finally, it's crucial to realize that the Lie groups arise from and are defined by the Lie algebra, and therefore by the generators. Not the other way around. By starting at the generators we can capture at a more fundamental level what is going on in 3D rotations, and what the manifold really is.

6.5 The generators of $SE(3)$

As mentioned before, $SE(3)$ is the set of rigid body transforms. Lie Theory as developed by physicists has very little to say about $SE(3)$, as it is actually a subset of the set of 4×4 matrices, known classically as the Lorentz group. The Lorentz group (or the complexified version, octonians) are used to encode not only rigid body transforms, but also the dilation of space and time due to relativistic effects. In robotics, we can usually restrict ourselves to time and space-preserving transformations, so we can use a simplified version of the Lorentz space generators.[18]

The generators of $SE(3)$ are just the basis of infinitesimal transformations along each degree of freedom.

[18] The bottom row of the 4×4 matrix is used in the Lorentz group, whereas is it not used in $SE(3)$. The Lorentz group generators continue the skew-symmetric pattern we are used to in this bottom row and right column. For more information about using Lie groups to model spacetime, [7] is a phenomenal introductory text to Lie groups in the context of theoretical physics. It includes a very good explanation of the Lorentz group and its double-cover.

$$J_1 = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \qquad J_2 = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \qquad J_3 = \begin{pmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$
$$J_4 = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \qquad J_5 = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \qquad J_6 = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix}.$$

The use of these generators defines the $(\cdot)^\wedge$ operator for $SE(3)$ as

$$\begin{pmatrix} \boldsymbol{\omega} \\ \mathbf{v} \end{pmatrix}^\wedge = \begin{pmatrix} \lfloor \boldsymbol{\omega} \rfloor_\times & \mathbf{v} \\ 0 & 0 \end{pmatrix},$$

and $(\cdot)^\vee$ as the inverse operation.

6.6 The generators of $\mathbb{D}SU(2)$ and $\mathbb{D}\mathcal{S}^3$

As with $SE(3)$, we use only the subset of the two copies of $\mathfrak{su}(2)$ necessary to encode time and space-preserving 3D transformations. The generators for this group are

$$J_1 = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} + \epsilon \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix} \qquad J_2 = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} + \epsilon \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix} \qquad J_3 = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} + \epsilon \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}$$
$$J_4 = \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix} + \epsilon \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \qquad J_5 = \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix} + \epsilon \begin{pmatrix} 0 & i \\ i & 0 \end{pmatrix} \qquad J_6 = \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix} + \epsilon \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}.$$

As $\mathbb{D}SU(2)$ is isomorphic to $\mathbb{D}S^3$ (dual unit quaternions), the generators for dual quaternions are given by

$$J_1 = i \qquad J_2 = j \qquad J_3 = k$$
$$J_4 = \epsilon i \qquad J_5 = \epsilon j \qquad J_5 = \epsilon k.$$

By this point, we should already be able to tell that the $(\cdot)^\wedge$ operator for these groups is given by

$$\begin{pmatrix} \boldsymbol{\omega} \\ \mathbf{v} \end{pmatrix}^\wedge = \begin{pmatrix} \boldsymbol{\omega} & \boldsymbol{\omega}_x - \boldsymbol{\omega}_y i \\ \boldsymbol{\omega}_x + \boldsymbol{\omega}_y i & \boldsymbol{\omega}_z \end{pmatrix} + \epsilon \begin{pmatrix} \mathbf{v}_z & \mathbf{v}_x - \mathbf{v}_y i \\ \mathbf{v}_x + \mathbf{v}_y i & \mathbf{v}_z \end{pmatrix}$$

for $\mathbb{D}SU(2)$ and

$$\begin{pmatrix} \boldsymbol{\omega} \\ \mathbf{v} \end{pmatrix}^\wedge = \begin{pmatrix} 0 \\ \boldsymbol{\omega} \end{pmatrix} + \epsilon \begin{pmatrix} 0 \\ \mathbf{v} \end{pmatrix}$$

for $\mathbb{D}\mathcal{S}^3$.

7 Computing the Matrix Exponential

Now that we have introduced the primary four 3D Lie groups used in our field, the rest of this document focuses on making efficient implementations in software. We'll start in this section by deriving closed-form expressions for the exponential and logarithm of each group. The next section focuses on the Adjoint, and computing Jacobians of expressions involving Lie group operations.

As noted in Section 3, the matrix exponential is defined with an infinite series. Given a square matrix A, the matrix exponential of A is defined as

$$\begin{aligned}\exp(A) &= I + A + \frac{1}{2}A^2 + \frac{1}{6}A^3 + \cdots \\ &= \sum_{k=0}^{\infty} \frac{1}{k!}A^k.\end{aligned}$$

and the matrix logarithm is defined as the inverse operation.

If you wanted to, you could just go ahead and compute this series up to a high enough order, and you would get the right answer[19]. However, for all of the Lie groups discussed above, the matrix exponential has a closed form that make computing it much, much faster. In this section, we will derive these forms for each of the Groups discussed.

Disclaimer: this section is going to be a bit dry, as it is pretty much entirely nitty-gritty algebra. You may want to just skip to the section you're most interested in.

7.1 Closed-Form Matrix Exponential for $SO(3)$

We know that the combination of the generators will give us a skew-symmetric matrix, therefore the matrix exponential of $\mathfrak{so}(3)$ will look like

$$\exp\left(\lfloor \boldsymbol{\omega} \rfloor_\times\right) = I + \lfloor \boldsymbol{\omega} \rfloor_\times + \frac{1}{2!}\lfloor \boldsymbol{\omega} \rfloor_\times^2 + \frac{1}{3!}\lfloor \boldsymbol{\omega} \rfloor_\times^3 + \cdots.$$

If we collect the even and odd pairs together, we will get

$$\exp\left(\lfloor \boldsymbol{\omega} \rfloor_\times\right) = I + \sum_{k=0}^{\infty}\left[\frac{\lfloor \boldsymbol{\omega} \rfloor_\times^{2k+1}}{(2k+1)!} + \frac{\lfloor \boldsymbol{\omega} \rfloor_\times^{2k+2}}{(2k+2)!}\right],$$

and we use the property of skew-symmetric matrices that

$$\lfloor \boldsymbol{\omega} \rfloor_\times^3 = -\|\boldsymbol{\omega}\|^2 \lfloor \boldsymbol{\omega} \rfloor_\times,$$

then we can rewrite the coefficients of this series in terms of the norm of $\boldsymbol{\omega}$ squared and $\lfloor w \rfloor_\times$ as

$$\begin{aligned}\theta^2 &= \boldsymbol{\omega}^\top \boldsymbol{\omega} \\ \lfloor \boldsymbol{\omega} \rfloor_\times^{2k+1} &= (-1)^k \theta^{2k} \lfloor \boldsymbol{\omega} \rfloor_\times \\ \lfloor \boldsymbol{\omega} \rfloor_\times^{2k+2} &= (-1)^k \theta^{2k} \lfloor \boldsymbol{\omega} \rfloor_\times^2.\end{aligned}$$

[19] In fact, this is a really great way to check your implementation for accuracy.

Let's plug these coefficients back into the equation and munge around a little bit, factoring out the $\lfloor \boldsymbol{\omega} \rfloor_\times$ and $\lfloor \boldsymbol{\omega} \rfloor_\times^2$ terms until we get

$$\begin{aligned}
\exp\left(\lfloor \boldsymbol{\omega} \rfloor_\times\right) &= I + \sum_{k=0}^{\infty} \left(\frac{(-1)^k \, \theta^{2k} \, \lfloor \boldsymbol{\omega} \rfloor_\times}{(2k+1)!} \right) + \sum_{k=0}^{\infty} \left(\frac{(-1)^k \, \theta^{2k} \, \lfloor \boldsymbol{\omega} \rfloor_\times^2}{(2k+2)!} \right) \\
&= I + \sum_{k=0}^{\infty} \left(\frac{(-1)^k \, \theta^{2k}}{(2k+1)!} \right) \lfloor \boldsymbol{\omega} \rfloor_\times + \sum_{k=0}^{\infty} \left(\frac{(-1)^k \, \theta^{2k}}{(2k+2)!} \right) \lfloor \boldsymbol{\omega} \rfloor_\times^2 \\
&= I + \left(1 - \frac{\theta^2}{3!} + \frac{\theta^4}{5!} + \cdots \right) \lfloor \boldsymbol{\omega} \rfloor_\times + \left(\frac{1}{2!} - \frac{\theta^2}{4!} + \frac{\theta^4}{6!} + \cdots \right) \lfloor \boldsymbol{\omega} \rfloor_\times^2 . \qquad (7.1)
\end{aligned}$$

At this point, we can refer to Table 3 and recognize the sinc function and a cosine-related function. We can plug these expressions into Eq. 7.1 and get the well-known Rodrigues formula,

$$\boxed{\exp\left(\lfloor \boldsymbol{\omega} \rfloor_\times\right) = I + \left(\frac{\sin(\theta)}{\theta} \right) \lfloor \boldsymbol{\omega} \rfloor_\times + \left(\frac{1 - \cos(\theta)}{\theta^2} \right) \lfloor \boldsymbol{\omega} \rfloor_\times^2 .} \qquad (7.2)$$

This formula will have numerical problems if $\theta \approx 0$, so best practice is to use the Taylor series version with two or three terms if you've got a small θ.

Now, if you wanted to compute the matrix logarithm for $SO(3)$, you can just compute the inverse of Eq. 7.2, as

$$\boxed{\log(R) \quad = \tfrac{\theta}{2\sin\theta} \left(R - R^\top \right),}$$

where

$$\theta = \cos^{-1}\left(\frac{\operatorname{tr}(R) - 1}{2} \right).$$

This equation has two points where numerical issues come into play, the first is when $\theta \approx 0$, the other is when $\theta \approx \pi$. As with Eq. 7.2, just use the Taylor series of $\log(R)$ if θ is close to these values (See Table 3).

7.2 Closed-Form Exponential for $\mathcal{S}^3$

Let's start with the definition of the quaternion exponential

$$\exp\left(\boldsymbol{\omega}^\wedge\right) = \sum_{k=0}^{\infty} \frac{\left(\boldsymbol{\omega}^\wedge\right)^k}{k!}, \qquad \boldsymbol{\omega}^\wedge = \begin{pmatrix} 0 \\ \boldsymbol{\omega} \end{pmatrix} \qquad (7.3)$$

where the term $(\mathbf{q})^k$ refers to multiplying $\mathbf{q}$ with itself $(\mathbf{q} \cdot \mathbf{q} \cdot \cdots)$ k times. We also note that

$$\begin{aligned}
\left(\boldsymbol{\omega}^\wedge\right)^2 &= \left(\boldsymbol{\omega}_x i + \boldsymbol{\omega}_y j + \boldsymbol{\omega}_z k\right) \cdot \left(\boldsymbol{\omega}_x i + \boldsymbol{\omega}_y j + \boldsymbol{\omega}_z k\right) \\
&= -\boldsymbol{\omega}_x^2 - \boldsymbol{\omega}_d^2 - \boldsymbol{\omega}_z^2 \\
&= -\left\| \boldsymbol{\omega} \right\|^2
\end{aligned}$$

Therefore, if $\theta = \|\boldsymbol{\omega}\|$, then

$$(\boldsymbol{\omega}^\wedge)^2 = -\theta^2, \qquad (\boldsymbol{\omega}^\wedge)^3 = -\theta^2\boldsymbol{\omega}^\wedge \qquad (\boldsymbol{\omega}^\wedge)^4 = \theta^4 \cdots .$$

Now, we can rewrite the series in Eq. 7.3 as

$$\begin{aligned}
\exp(\boldsymbol{\omega}^\wedge) &= \sum_{k=0}^{\infty} \frac{(\boldsymbol{\omega}^\wedge)^k}{k!} \\
&= 1 + \boldsymbol{\omega}^\wedge - \frac{\theta^2}{2!} - \frac{\theta^2}{3!}\boldsymbol{\omega}^\wedge + \frac{\theta^4}{4!} + \frac{\theta^4}{5!}\boldsymbol{\omega}^\wedge - \cdots \\
&= 1 + \frac{\theta}{\theta}\boldsymbol{\omega}^\wedge - \frac{\theta^2}{2!} - \frac{\theta^3}{3!\theta}\boldsymbol{\omega}^\wedge + \frac{\theta^4}{4!} + \frac{\theta^5}{5!\theta}\boldsymbol{\omega}^\wedge - \cdots \\
&= \left(1 - \frac{\theta^2}{2!} + \frac{\theta^4}{4!} \cdots\right) + \frac{1}{\theta}\left(\theta - \frac{\theta^3}{3!} + \frac{\theta^5}{5!} \cdots\right)\boldsymbol{\omega}^\wedge \\
&= \cos(\theta) + \frac{\sin(\theta)}{\theta}\boldsymbol{\omega}^\wedge. \qquad (7.4)
\end{aligned}$$

As with the rotation matrix exponential, we will want to use the Taylor series approximation if $\theta \approx 0$ to avoid numerical errors. (See Table 3).

Computing the closed-form logarithm is done by inverting Eq. 7.4 and is given by

$$\log(\mathbf{q}) = \operatorname{atan2}(\|\vec{\mathbf{q}}\|, q_0)\frac{\mathbf{q}}{\|\vec{\mathbf{q}}\|}.$$

Because of the fact that quaternion dynamics have a $\frac{1}{2}$ in front of them (See Eq. 4.8), it is common practice in both physics and robotics applications to embed the $\frac{1}{2}$ into the exp function itself such that[20]

$$\boxed{\exp(\boldsymbol{\omega}) = \cos\left(\frac{\|\boldsymbol{\omega}\|}{2}\right) + \sin\left(\frac{\|\boldsymbol{\omega}\|}{2}\right)\frac{\boldsymbol{\omega}}{\|\boldsymbol{\omega}\|}}$$

$$\boxed{\log(\mathbf{q}) = 2\tan^{-1}\left(\frac{\|\vec{\mathbf{q}}\|}{q_0}\right)\cdot\frac{\vec{\mathbf{q}}}{\|\vec{\mathbf{q}}\|}.}$$

This is actually the same as if we redefined the generators of the unit quaternion to all be $\frac{1}{2}J_i$. This is fine, and totally proper, however we just need to be explicit about this choice.

7.3 Closed-Form Matrix Exponential for $SE(3)$

To derive the closed-form expression for the matrix exponential of $SE(3)$ we start with the series expression for the matrix exponential and play with it until we can find patterns we can convert into known expressions. In the case of $SE(3)$, this procedure starts out as follows:

[20] You might recognize this as the axis-angle conversion to unit quaternions (Eq. 4.3)

$$\begin{aligned}
\exp\begin{pmatrix}\lfloor\boldsymbol{\omega}\rfloor_\times & \mathbf{v}\\ 0 & 1\end{pmatrix} &= I + \begin{pmatrix}\lfloor\boldsymbol{\omega}\rfloor_\times & \mathbf{v}\\ 0 & 1\end{pmatrix} + \frac{1}{2!}\begin{pmatrix}\lfloor\boldsymbol{\omega}\rfloor_\times & \mathbf{v}\\ 0 & 1\end{pmatrix}^2 + \frac{1}{3!}\begin{pmatrix}\lfloor\boldsymbol{\omega}\rfloor_\times & \mathbf{v}\\ 0 & 1\end{pmatrix}^3 + \cdots \\
&= I + \begin{pmatrix}\lfloor\boldsymbol{\omega}\rfloor_\times & \mathbf{v}\\ 0 & 1\end{pmatrix} + \frac{1}{2!}\begin{pmatrix}\lfloor\boldsymbol{\omega}\rfloor_\times^2 & \lfloor\boldsymbol{\omega}\rfloor_\times\mathbf{v}\\ 0 & 1\end{pmatrix} + \frac{1}{3!}\begin{pmatrix}\lfloor\boldsymbol{\omega}\rfloor_\times^3 & \lfloor\boldsymbol{\omega}\rfloor_\times^2\mathbf{v}\\ 0 & 1\end{pmatrix} + \cdots \\
&= \begin{pmatrix}\exp\left(\lfloor\boldsymbol{\omega}\rfloor_\times\right) & V\mathbf{v}\\ 0 & 1\end{pmatrix} \\
V &= I + \frac{1}{2}\lfloor\boldsymbol{\omega}\rfloor_\times + \frac{1}{3!}\lfloor\boldsymbol{\omega}\rfloor_\times^2 .
\end{aligned}$$

At this point, we can see that the upper right block is just the rotation matrix, but we have this matrix V that couples some amount of angular rate in with linear velocity. This is the term that describes the screw motions we expect from this exponential map. We can find a closed-form expression for V by splitting the even and odd pairs which leads to:

$$\begin{aligned}
V &= I + \sum_{k=0}^{\infty}\left(\frac{\lfloor\boldsymbol{\omega}\rfloor_\times^{(2k+1)}}{(2k+2)!} + \frac{\lfloor\boldsymbol{\omega}\rfloor_\times^{(2k+2)}}{(2k+3)!}\right) \\
&= I + \sum_{k=0}^{\infty}\left(\frac{(-1)^k\,\theta^{2k}}{(2k+2)!}\right)\lfloor\boldsymbol{\omega}\rfloor_\times + \sum_{k=0}^{\infty}\left(\frac{(-1)^k\,\theta^{2k}}{(2k+3)!}\right)\lfloor\boldsymbol{\omega}\rfloor_\times^2 \\
&= I + \left(\frac{1}{2} - \frac{\theta^2}{4!} + \frac{\theta^4}{6!} + \cdots\right)\lfloor\boldsymbol{\omega}\rfloor_\times + \left(\frac{1}{3!} - \frac{\theta^2}{5!} + \frac{\theta^4}{7!} + \cdots\right)\lfloor\boldsymbol{\omega}\rfloor_\times^2 \\
&= I + \left(\frac{1-\cos(\theta)}{\theta^2}\right)\lfloor\boldsymbol{\omega}\rfloor_\times + \left(\frac{\theta-\sin(\theta)}{\theta^3}\right)\lfloor\boldsymbol{\omega}\rfloor_\times^2 .
\end{aligned}$$

In summary, the final form for the matrix exponential is given as

$$\boxed{\exp\begin{pmatrix}\lfloor\boldsymbol{\omega}\rfloor_\times & \mathbf{v}\\ 0 & 1\end{pmatrix} = \begin{pmatrix}\exp\left(\lfloor\boldsymbol{\omega}\rfloor_\times\right) & V\mathbf{v}\\ 0 & 1\end{pmatrix},}$$

where

$$V = I + \left(\frac{1-\cos(\theta)}{\theta^2}\right)\lfloor\boldsymbol{\omega}\rfloor_\times + \left(\frac{\theta-\sin(\theta)}{\theta^3}\right)\lfloor\boldsymbol{\omega}\rfloor_\times^2 ,$$

and the logarithm has this closed-form:

$$\boxed{\log\begin{pmatrix}R & \mathbf{t}\\ 0 & 1\end{pmatrix} = \begin{pmatrix}\log(R) & V^{-1}\mathbf{t}\\ 0 & 0\end{pmatrix},}$$

where

$$V^{-1} = I - \frac{1}{2}\lfloor\boldsymbol{\omega}\rfloor_\times + \frac{1}{\theta^2}\left(1 - \frac{\theta\sin\theta}{2\,(1-\cos\theta)}\right)\lfloor\boldsymbol{\omega}\rfloor_\times^2 .$$

As with the other results in this document, Taylor series expansions for the numerically unstable trigonometric terms can be found in Table 3.

7.4 Closed-Form Exponential for $\mathbb{D}\mathcal{S}^3$

To derive the exponential map of the dual quaternion, we start with the exponential map for the ordinary quaternion, except with dual numbers. For some syntactical relief, I'm going to abuse notation a little bit and define $\boldsymbol{\omega}$ and $\mathbf{v}$ as pure quaternions and use the $\tilde{(\cdot)}$ syntax to remind us that a value has both real and dual components. The dual exponential map is given as

$$\exp\left(\boldsymbol{\omega}+\epsilon\mathbf{v}\right)=\cos\left(\frac{\tilde{\phi}}{2}\right)+\sin\left(\frac{\tilde{\phi}}{2}\right)\frac{\left(\boldsymbol{\omega}+\epsilon\mathbf{v}\right)}{\tilde{\phi}}\qquad\qquad\tilde{\phi}=\left\|\boldsymbol{\omega}+\epsilon\mathbf{v}\right\|. \tag{7.5}$$

This is absolutely correct, but unwieldy. Let's use the identities from Table 2 to break this into the real and dual parts. First the dual angle can be re-written as

$$\begin{aligned}\tilde{\phi}&=\sqrt{\left(\boldsymbol{\omega}+\epsilon\mathbf{v}\right)^{\top}\left(\boldsymbol{\omega}+\epsilon\mathbf{v}\right)}\\&=\sqrt{\left(\boldsymbol{\omega}\right)^{\top}\left(\boldsymbol{\omega}\right)+\epsilon 2\boldsymbol{\omega}^{\top}\mathbf{v}}\\&=\sqrt{\boldsymbol{\omega}^{\top}\boldsymbol{\omega}}+\frac{\boldsymbol{\omega}^{\top}\mathbf{v}}{\sqrt{\boldsymbol{\omega}^{\top}\boldsymbol{\omega}}}\epsilon\\&=\theta+\frac{\gamma}{\theta}\epsilon.\end{aligned}$$

where θ is the angle of rotation that we saw for the ordinary quaternion and γ is $\boldsymbol{\omega}^{\top}\mathbf{v}$. With this result, we can now split the dual sin and cos functions as

$$\begin{aligned}\cos\left(\frac{\tilde{\phi}}{2}\right)&=\cos\left(\frac{\theta}{2}+\epsilon\frac{\gamma}{2\theta}\right)\\&=\cos\frac{\theta}{2}-\epsilon\frac{\gamma}{2\theta}\sin\frac{\theta}{2}\\\sin\left(\frac{\tilde{\phi}}{2}\right)&=\sin\frac{\theta}{2}+\epsilon\frac{\gamma}{2\theta}\cos\frac{\theta}{2},\end{aligned}$$

and the sinc-like function expands as

$$\begin{aligned}\frac{\sin\frac{\tilde{\phi}}{2}}{\tilde{\phi}}&=\frac{\sin\frac{\theta}{2}+\epsilon\frac{\gamma}{2\theta}\cos\frac{\theta}{2}}{\theta+\frac{\gamma}{\theta}\epsilon}\\&=\frac{\sin\frac{\theta}{2}+\epsilon\frac{\gamma}{2\theta}\cos\frac{\theta}{2}}{\theta+\frac{\gamma}{\theta}\epsilon}\left(\frac{\theta-\frac{\gamma}{\theta}\epsilon}{\theta-\frac{\gamma}{\theta}\epsilon}\right)\\&=\frac{\theta\sin\frac{\theta}{2}+\epsilon\left(\frac{\gamma}{2\theta}\theta\cos\frac{\theta}{2}-\frac{\gamma}{\theta}\sin\frac{\theta}{2}\right)}{\theta^2}\\&=\frac{\sin\frac{\theta}{2}}{\theta}+\epsilon\gamma\left(\frac{\frac{1}{2}\cos\frac{\theta}{2}-\frac{\sin\frac{\theta}{2}}{\theta}}{\theta^2}\right).\end{aligned}$$

We can finally expand and factor Eq. 7.5 into the real and dual components as follows

$$\begin{aligned}
\exp\left(\boldsymbol{\omega}+\epsilon\mathbf{v}\right) &= \cos\frac{\tilde{\phi}}{2}+\frac{\sin\frac{\tilde{\phi}}{2}}{\tilde{\phi}}\left(\boldsymbol{\omega}+\epsilon\mathbf{v}\right)\\
&= \left(\cos\frac{\theta}{2}-\epsilon\frac{\gamma}{2\theta}\sin\frac{\theta}{2}\right)+\left(\frac{\sin\frac{\theta}{2}}{\theta}+\epsilon\gamma\left(\frac{\frac{1}{2}\cos\frac{\theta}{2}-\frac{\sin\frac{\theta}{2}}{\theta}}{\theta^2}\right)\right)\left(\boldsymbol{\omega}+\epsilon\mathbf{v}\right)\\
&= \left(\cos\frac{\theta}{2}-\epsilon\frac{\gamma}{2\theta}\sin\frac{\theta}{2}\right)+\left(\frac{\sin\frac{\theta}{2}}{\theta}\boldsymbol{\omega}+\epsilon\left(\gamma\left(\frac{\frac{1}{2}\cos\frac{\theta}{2}-\frac{\sin\frac{\theta}{2}}{\theta}}{\theta^2}\right)\boldsymbol{\omega}+\frac{\sin\frac{\theta}{2}}{\theta}\mathbf{v}\right)\right)\\
&= \left(\cos\frac{\theta}{2}+\frac{\sin\frac{\theta}{2}}{\theta}\boldsymbol{\omega}\right)+\epsilon\left(-\frac{\gamma}{2\theta}\sin\frac{\theta}{2}+\gamma\left(\frac{\frac{1}{2}\cos\frac{\theta}{2}-\frac{\sin\frac{\theta}{2}}{\theta}}{\theta^2}\right)\boldsymbol{\omega}+\frac{\sin\frac{\theta}{2}}{\theta}\mathbf{v}\right)
\end{aligned}$$

$$\boxed{\exp\left(\boldsymbol{\omega}+\epsilon\mathbf{v}\right)=\begin{pmatrix}\cos\frac{\theta}{2}\\ \frac{\sin\frac{\theta}{2}}{\theta}\boldsymbol{\omega}\end{pmatrix}+\epsilon\begin{pmatrix}-\frac{\gamma}{2\theta}\sin\frac{\theta}{2}\\ \gamma\left(\frac{\frac{1}{2}\cos\frac{\theta}{2}-\frac{\sin\frac{\theta}{2}}{\theta}}{\theta^2}\right)\boldsymbol{\omega}+\frac{\sin\frac{\theta}{2}}{\theta}\mathbf{v}\end{pmatrix},}$$

which is the final form of the matrix exponential for the dual quaternion representation.

In similar fashion, we can derive the dual quaternion logarithm by expanding the dual form of the ordinary quaternion expression. We start with the ordinary quaternion logarithm

$$\log\left(\Gamma\right)=2\tan^{-1}\left(\frac{\left\|\tilde{\vec{\mathbf{q}}}\right\|}{\tilde{q_0}}\right)\frac{\tilde{\vec{\mathbf{q}}}}{\left\|\tilde{\vec{\mathbf{q}}}\right\|},$$

which can now separate into real and dual parts.

First, let us start by separating the expression for the norm of the dual quaternion imaginary component. For this derivation, I have defined

$$\mathbf{r}=\vec{\mathbf{q}}_r \qquad r_0=q_{r0} \qquad \mathbf{d}=\vec{\mathbf{q}}_d \qquad d_0=q_{d0}$$

to clean up the expressions somewhat. Let's start first with the norm of the dual complex portion:

$$\begin{aligned}
\left\|\tilde{\vec{\mathbf{q}}}\right\| &= \sqrt{\left(\mathbf{r}+\epsilon\mathbf{d}\right)^{\top}\left(\mathbf{r}+\epsilon\mathbf{d}\right)}\\
&= \sqrt{\mathbf{r}^{\top}\mathbf{r}+\epsilon 2\mathbf{r}^{\top}\mathbf{d}}\\
&= \sqrt{\mathbf{r}^{\top}\mathbf{r}}+\frac{\gamma}{\sqrt{\mathbf{r}^{\top}\mathbf{r}}}\epsilon && \gamma = \mathbf{r}^{\top}\mathbf{d}\\
&= \theta+\frac{\gamma}{\theta}\epsilon. && \theta = \sqrt{\mathbf{r}^{\top}\mathbf{r}},
\end{aligned}$$

We can use this to separate the argument of the arc-tangent function

$$\begin{aligned}\frac{\|\tilde{\vec{\mathbf{q}}}\|}{\tilde{q}_0} &= \frac{\theta + \frac{\gamma}{\theta}\epsilon}{r_0 + \epsilon d_0} \\ &= \frac{\theta + \frac{\gamma}{\theta}\epsilon}{r_0 + \epsilon d_0}\left(\frac{r_0 - \epsilon d_0}{r_0 - \epsilon d_0}\right) \\ &= \frac{r_0\theta}{r_0^2} + \epsilon\frac{\left(r_0\frac{\gamma}{\theta} - d_0\theta\right)}{r_0^2} \\ &= \frac{\theta}{r_0} + \epsilon\frac{\left(r_0\frac{\gamma}{\theta} - d_0\theta\right)}{r_0^2},\end{aligned}$$

the screw angle:

$$\begin{aligned}\tan^{-1}\left(\frac{\|\tilde{\vec{\mathbf{q}}}\|}{\tilde{q}_0}\right) &= \tan^{-1}\left(\frac{\theta}{r_0} + \epsilon\frac{\left(r_0\frac{\gamma}{\theta} - d_0\theta\right)}{r_0^2}\right) \\ &= \tan^{-1}\left(\frac{\theta}{r_0}\right) + \epsilon\left(\frac{\frac{\left(r_0\frac{\gamma}{\theta} - d_0\theta\right)}{r_0^2}}{\left(\frac{\theta}{r_0}\right)^2 + 1}\right) \\ &= \tan^{-1}\left(\frac{\theta}{r_0}\right) + \epsilon\left(\frac{\frac{\left(r_0\frac{\gamma}{\theta} - d_0\theta\right)}{r_0^2}}{\frac{\theta^2 + r_0^2}{r_0^2}}\right) \\ &= \tan^{-1}\left(\frac{\theta}{r_0}\right) + \epsilon\left(r_0\frac{\gamma}{\theta} - d_0\theta\right), \qquad (\theta^2 + r_0^2 = 1)\end{aligned}$$

and the screw axis:

$$\begin{aligned}\frac{\tilde{\vec{\mathbf{q}}}}{\|\tilde{\vec{\mathbf{q}}}\|} &= \frac{\mathbf{r} + \epsilon\mathbf{d}}{\theta + \frac{\gamma}{\theta}\epsilon} \\ &= \frac{\mathbf{r} + \epsilon\mathbf{d}}{\theta + \frac{\gamma}{\theta}\epsilon}\left(\frac{\theta - \frac{\gamma}{\theta}\epsilon}{\theta - \frac{\gamma}{\theta}\epsilon}\right) \\ &= \frac{\mathbf{r}}{\theta} + \epsilon\frac{\mathbf{d}\theta - \frac{\gamma}{\theta}\mathbf{r}}{\theta^2}.\end{aligned}$$

With these expressions, we can write the full form of the dual quaternion logarithm as

$$\begin{aligned}
\log(\Gamma) &= 2\tan^{-1}\left(\left\|\tilde{\tilde{\mathbf{q}}}\right\|, \tilde{q_0}\right)\frac{\tilde{\tilde{\mathbf{q}}}}{\left\|\tilde{\tilde{\mathbf{q}}}\right\|} \\
&= 2\left(\tan^{-1}\left(\frac{\theta}{r_0}\right) + \epsilon\left(r_0\frac{\gamma}{\theta} - d_0\theta\right)\right)\left(\frac{\mathbf{r}}{\theta} + \epsilon\frac{\mathbf{d}\theta - \frac{\gamma}{\theta}\mathbf{r}}{\theta^2}\right) \\
&= 2\left(\frac{\mathbf{r}}{\theta}\tan^{-1}\left(\frac{\theta}{r_0}\right)\right) + 2\epsilon\left(\tan^{-1}\left(\frac{\theta}{r_0}\right)\left(\frac{\mathbf{d}\theta - \frac{\gamma}{\theta}\mathbf{r}}{\theta^2}\right) + \left(r_0\frac{\gamma}{\theta} - d_0\theta\right)\frac{\mathbf{r}}{\theta}\right) \\
&= 2\left(\frac{\tan^{-1}\left(\frac{\theta}{r_0}\right)}{\theta}\mathbf{r}\right) + 2\epsilon\left(\frac{\tan^{-1}\left(\frac{\theta}{r_0}\right)}{\theta}\left(\mathbf{d} - \frac{\gamma}{\theta^2}\mathbf{r}\right) + \left(r_0\frac{\gamma}{\theta^2} - d_0\right)\mathbf{r}\right).
\end{aligned}$$

Again, to make things a little more compact, we can define

$$a = \frac{\tan^{-1}\left(\frac{\theta}{r_0}\right)}{\theta}, \qquad\qquad b = \frac{\gamma}{\theta^2},$$

and simplify the expression above to get

$$\begin{aligned}
\log(\Gamma) &= 2\left(a\mathbf{r} + \epsilon\left(a\left(\mathbf{d} - b\mathbf{r}\right) + \left(r_0 b - d_0\right)\mathbf{r}\right)\right) \\
&= 2a\mathbf{r} + 2\epsilon\left(a\mathbf{d} + \left(\left(r_0 - a\right)b - d_0\right)\mathbf{r}\right).
\end{aligned}$$

This gives us the final form of the logarithm for the dual quaternion:

$$\boxed{\log(\Gamma) = 2\begin{pmatrix} a\mathbf{r} \\ a\mathbf{d} + \left(\left(r_0 - a\right)b - d_0\right)\mathbf{r}\end{pmatrix}.}$$

To deal with the case that θ is small, we have to re-arrange the expression so we can find a numerically stable Taylor series expression[2]:

$$\begin{aligned}
\log(\Gamma) &= 2a\mathbf{r} + 2\epsilon\left(a\mathbf{d} + \left(\left(r_0 - a\right)b - d_0\right)\mathbf{r}\right) \\
&= 2a\mathbf{r} + 2\epsilon\left(\frac{\tan^{-1}\left(\frac{\theta}{r_0}\right)}{\theta}\mathbf{d} + \left(\left(r_0 - \frac{\tan^{-1}\left(\frac{\theta}{r_0}\right)}{\theta}\right)\frac{\gamma}{\theta^2} - d_0\right)\mathbf{r}\right) \\
&= 2a\mathbf{r} + 2\epsilon\left(\frac{\tan^{-1}\left(\frac{\theta}{r_0}\right)}{\theta}\mathbf{d} + \left(\gamma\frac{\theta\cos\left(\frac{\theta}{2}\right) - \tan^{-1}\left(\frac{\theta}{r_0}\right)}{\theta^3} - d_0\right)\mathbf{r}\right) \qquad \left(r_0 = \cos\left(\frac{\theta}{2}\right)\right) \\
&= 2a\mathbf{r} + 2\epsilon\left(a\mathbf{d} + \left(\gamma\frac{\theta\cos\left(\frac{\theta}{2}\right) - \tan^{-1}\left(\frac{\theta}{r_0}\right)}{\theta^3} - d_0\right)\mathbf{r}\right).
\end{aligned}$$

At this point, we can pump the nasty trig expression into Wolfram Alpha to get a useful Taylor series representation to use when $\theta \approx 0$.

$$\frac{\theta\cos\left(\frac{\theta}{2}\right)-\tan^{-1}\left(\frac{\theta}{r_0}\right)}{\theta^3}=\frac{5}{24}-\frac{379}{1920}\theta^2+\frac{46073}{322560}\theta^4-\frac{127431}{1146880}\theta^6$$

The Taylor series expression for $\frac{1}{\theta}\tan^{-1}\left(\frac{\theta}{r_0}\right)$ is given in Table 3. It should be noted that this equation is highly non-linear, which is why for this series I have added the 6th-order term.

8 The Adjoint as the Jacobian of Group Action

Most people's first introduction to calculus typically focuses on computing derivatives. For completeness, we're going to start here and work our way to computing Jacobians of the Lie Group action. The formula for computing the derivative of a scalar function f, is given as[21]

$$\frac{\partial f}{\partial x}=\lim_{\varepsilon\to 0}\frac{1}{\varepsilon}\left(f\left(x+\varepsilon\right)-f\left(x\right)\right).$$

In conversational English, this means, "How does the output of the function f change if I bump the input a tiny bit?" This is very naturally adapted for a vector function $g\in\mathbb{R}^m\to\mathbb{R}^n$ as

$$\begin{aligned}\frac{\partial g}{\partial\mathbf{x}}&=\lim_{\varepsilon\to 0}\frac{1}{\varepsilon}\left[\boldsymbol{\delta}_1\left(\mathbf{x},\varepsilon\right)\quad\boldsymbol{\delta}_2\left(\mathbf{x},\varepsilon\right)\cdots\boldsymbol{\delta}_m\left(\mathbf{x},\varepsilon\right)\right]\in\mathbb{R}^{n\times m}\\ \boldsymbol{\delta}_i\left(\mathbf{x},\varepsilon\right)&=g\left(\mathbf{x}+\mathbf{e}_i\varepsilon\right)-g\left(\mathbf{x}\right)\end{aligned}$$

Again, in conversational English, we're saying that g is a function that takes in m-vectors and produces n-vectors. We want to know what g looks like along each of the input axes, so we're going to make a matrix where each column is the derivative along each of these directions. This is known as the Jacobian matrix, and can also be interpreted as, "If I change the input in some way, what does it do to the output?" For example, if I go further along the $\mathbf{e}_1$-axis, would I expect the output to go up or down? This would be captured in the first column of the Jacobian. This relationship lets us look at Jacobians as a sort of mapping from input space to output space, and lets us quickly compute how changes in input should affect the output of the function g.

The concept of differentiation works quite naturally with Lie groups as well. For example, consider the case of some function $h\in SE\left(3\right)\to\mathbb{R}^n$. What we want is a matrix that tells us how the output of h behaves as we tweak the argument of h along the *generators* of $\mathfrak{se}\left(3\right)$. To compute this, we use the fact that every member of the group can be described by the exponential of some linear combination of generators, as in

$$T=\exp\left(\mathbf{x}^\wedge\right),\qquad\qquad\mathbf{x}^\wedge=\log\left(T\right).$$

When we write it this way, we can see how we might perturb $\mathbf{x}$ (and therefore, T) in the J_i direction:

$$\begin{aligned}T_i^+&=\exp\left(\mathbf{x}^\wedge+J_i\varepsilon\right)\\&=T\cdot\exp\left(J_i\varepsilon\right).\end{aligned}$$

[21] Note, to be consistent with the vast majority of literature on this topic, I am using ε to mean a small perturbation. This is a separate and distinct concept from ϵ, the dual number.

Next, we need to think about how we might compute the difference between T_i^+ and the original T. Again, looking to the Lie Algebra, we should expect the difference $\boldsymbol{\delta}_i \in \mathfrak{se}(3)$ to be

$$\begin{aligned} \exp(\boldsymbol{\delta}_i) &= T \cdot \exp(J_i \varepsilon) \cdot \exp(-\mathbf{x}^\wedge) \\ \boldsymbol{\delta}_i &= \log\left(T \cdot \exp(J_i \varepsilon) \cdot T^{-1}\right)^\vee . \end{aligned} \tag{8.1}$$

If we make a matrix out of the six $\boldsymbol{\delta}_i$ for $\mathfrak{se}(3)$, take the limit as $\varepsilon \to 0$ and attach coordinate frames to our transform T_a^b, then we get a 6×6 matrix that tells us how things change along the generators in the b frame if we perturb along the generators the a frame. This has a special name–the Adjoint–and it just so happens to be the Jacobian of the group action by a member of the group.

We can come at the Adjoint in another way: because the Adjoint is the Jacobian of the group action of a Lie group, we can use it to map vectors from the "output" of a group action to the "input", and vice-versa. In mathematical terms[22]:

$$T \cdot \exp(\boldsymbol{\delta}) = \exp(\mathrm{Ad}_T \cdot \boldsymbol{\delta}) \cdot T.$$

See how we have "moved" $\boldsymbol{\delta}$ from the "input" of T (the right side) to the "output" (the left side)? If we attach coordinate frames to T this becomes even more clear:

$$\begin{aligned} T_a^b \cdot \exp(\boldsymbol{\delta}^a) &= \exp\left(\mathrm{Ad}\left(T_a^b\right) \cdot \boldsymbol{\delta}^a\right) \cdot T_a^b. \\ &= \exp\left(\boldsymbol{\delta}^b\right) \cdot T_a^b. \end{aligned} \tag{8.2}$$

The Adjoint transforms $\boldsymbol{\delta}$ from a to b.

Remember all that talk about left-invariance and right-invariance? This property of the Adjoint of the Lie Group lets us swap back and forth however we need to. This is really helpful when computing Jacobians of complex expressions involving Lie Groups.

9 Closed-Form Expressions for the Adjoint

In this section, I'm going to derive the Adjoint for each of our four Lie Groups, but before we do that, let's first talk about a concrete example of where the Adjoint is useful.

Let's consider some kind of nonlinear controller that is computing error on the Lie Algebra of $SE(3)$, where we have some desired state $\check{T}$ and our current state T. We're going to form some control law around the error between these two states. Naturally, we want to use linear algebra as the backbone of our implementation, so we'd like to use $\mathfrak{se}(3)^\vee$ to represent our error. We could compute the error in one of 4 ways:

[22] for the rest of this section, I'm dropping the $(\cdot)^\wedge$ so we can focus on the coordinate frame notation. The mapping from the vector to algebra should be implied by context.

$$\begin{aligned}
\boldsymbol{\delta}_{\check{a}}^{a} &= \log\left(\left(\check{T}_{a}^{b}\right)^{-1} \cdot T_{a}^{b}\right) \\
\boldsymbol{\delta}_{\check{b}}^{b} &= \log\left(\check{T}_{a}^{b} \cdot \left(T_{a}^{b}\right)^{-1}\right) \\
\boldsymbol{\delta}_{\check{a}}^{a} &= \log\left(\left(T_{a}^{b}\right)^{-1} \cdot \check{T}_{a}^{b}\right) \\
\boldsymbol{\delta}_{\check{b}}^{b} &= \log\left(\left(T_{a}^{b}\right) \cdot \left(\check{T}_{a}^{b}\right)^{-1}\right).
\end{aligned}$$

The choice of error depends largely on which frames are moving, and which are not. Let's assume that in this case the a frame is shared between the transform (i.e. perhaps a refers to the inertial frame), and that the b and $\check{b}$ frames are the current and desired body frame of a robot. The goal of the controller in this case is to make $b = \check{b}$. In this case, it makes the most sense to define our error as one of the following two options (or else the coordinate frames will not match up):

$$\begin{aligned}
\boldsymbol{\delta}_{b}^{\check{b}} &= \log\left(\check{T}_{a}^{b} \cdot \left(T_{a}^{b}\right)^{-1}\right), \\
\boldsymbol{\delta}_{\check{b}}^{b} &= \log\left(T_{a}^{b} \cdot \left(\check{T}_{a}^{b}\right)^{-1}\right).
\end{aligned}$$

Driving either of these errors to zero will achieve our control objectives. However, watch what happens when we want to compute the Jacobian of our error with respect to our current state. We can find these Jacobians using the chain rule:

$$\begin{aligned}
\frac{\partial}{\partial T_{a}^{b}} \boldsymbol{\delta}_{b}^{\check{b}} &= \left.\frac{\partial \log(\mathbf{x})}{\partial \mathbf{x}}\right|_{\mathbf{x}=\left(T_{a}^{b} \cdot \left(T_{a}^{\check{b}}\right)\right)} \cdot \frac{\partial}{\partial T_{a}^{b}}\left(\check{T}_{a}^{b} \cdot \left(T_{a}^{b}\right)^{-1}\right) && = A \cdot B^{\check{b}} \\
\frac{\partial}{\partial T_{a}^{b}} \boldsymbol{\delta}_{\check{b}}^{b} &= \left.\frac{\partial \log(\mathbf{x})}{\partial \mathbf{x}}\right|_{\mathbf{x}=\left(T_{a}^{b} \cdot \left(T_{a}^{\check{b}}\right)\right)} \cdot \frac{\partial}{\partial T_{a}^{b}}\left(T_{a}^{b} \cdot \left(\check{T}_{a}^{b}\right)^{-1}\right) && = A \cdot B^{b}
\end{aligned}$$

The expression for A is in Table 6, and it's the same for both parameterizations. However, we are left with the two B expressions. In the first case, we have to go through our desired state to get to the current state. This isn't too bad, we just look at Table 4 to get (using the left Jacobian):

$$\begin{aligned}
B^{\check{b}} &= \frac{\partial}{\partial T_{a}^{b}}\left(\check{T}_{a}^{b} \cdot \left(T_{a}^{b}\right)^{-1}\right) \\
&= -\mathrm{Ad}\left(\check{T}_{a}^{b} \cdot \left(T_{a}^{b}\right)^{-1}\right)
\end{aligned}$$

for the first cast, and

$$\begin{aligned}
B^{b} &= \frac{\partial}{\partial T_{a}^{b}}\left(T_{a}^{b} \cdot \left(\check{T}_{a}^{b}\right)^{-1}\right) \\
&= I \in \mathbb{R}^{6\times 6}.
\end{aligned}$$

for the second. Having a closed form expression for the Adjoint makes computing these kinds of expressions very straight-forward.

Most modern forms of control and estimation rely heavily on quickly-computed Jacobians. With a table of Jacobians for exp, log, Ad and a few other Lie group expressions, plus the Matrix Cookbook [6], it's pretty easy to chain-rule out even the hairiest Jacobians pretty quickly to get analytical expressions. Appendix A contains most of these Jacobians for easy reference.

9.1 Adjoint for $SO(3)$

For all four Lie groups, we can derive the Adjoint by finding the closed-form expression of Eq. 8.1. $SO(3)$ is quite easy:

$$\begin{aligned}
\operatorname{Ad}(R) &= \lim_{\varepsilon \to 0} \frac{1}{\varepsilon} \begin{bmatrix} R \cdot \exp(J_1 \varepsilon) \cdot R^{-1} & \cdots \end{bmatrix} \\
&= \lim_{\varepsilon \to 0} \frac{1}{\varepsilon} \begin{bmatrix} R \cdot \left(I + \lfloor \mathbf{e}_1 \epsilon \rfloor_\times + \cdots \right) \cdot R^{-1} & \cdots \end{bmatrix} \\
&= \lim_{\varepsilon \to 0} \frac{1}{\varepsilon} \begin{bmatrix} R \cdot R^{-1} + R \cdot \lfloor \mathbf{e}_1 \epsilon \rfloor_\times \cdot R^{-1} & \cdots \end{bmatrix} \\
&= \begin{bmatrix} R \cdot \lfloor \mathbf{e}_1 \rfloor_\times \cdot R^{-1} & \cdots \end{bmatrix} && R \cdot R^{-1} = I \\
&= \begin{bmatrix} \lfloor R\mathbf{e}_1 \rfloor_\times & \lfloor R\mathbf{e}_2 \rfloor_\times & \lfloor R\mathbf{e}_3 \rfloor_\times \end{bmatrix} && \text{(Eq. 2.2)} \\
&= R
\end{aligned}$$

We see here that $SO(3)$ is *self-adjoint*, and it makes sense why. Consider an $SO(3)$ version of Eq. 8.2. Rotating the vector is the natural way to transform $\boldsymbol{\delta}$ from the a frame to the b frame.

$$\boxed{\operatorname{Ad}(R) = R}$$

9.2 Adjoint for Unit Quaternions

Remember, the Adjoint action is given by

$$\operatorname{Ad} = \lim_{\varepsilon \to 0} \frac{1}{\varepsilon} \begin{bmatrix} \log\left(\mathbf{q} \cdot \exp(J_1 \varepsilon) \cdot \mathbf{q}^{-1} \right) & \cdots \end{bmatrix}.$$

In the case of unit quaternions, we can see that they too, are self-adjoint. To compute the adjoint of a quaternion, all we need to do is pre- and post-multiply a pure quaternion of the lie algebra member by the quaternion.

$$\operatorname{Ad}(\mathbf{q}) = \mathbf{q} \cdot \boldsymbol{\delta} \cdot \mathbf{q}^{-1}.$$

However, sometimes we might want the matrix form of the adjoint. This can be computed as follows:

$$
\begin{aligned}
&= \lim_{\varepsilon \to 0} \frac{1}{\varepsilon} \left[\log \left(\mathbf{q} \cdot \begin{pmatrix} 0 \\ \mathbf{e}_1 \sin\left(\frac{\varepsilon}{2}\right) \end{pmatrix} \cdot \mathbf{q}^{-1} \right) \quad \cdots \right] \\
&= \lim_{\varepsilon \to 0} \frac{1}{\varepsilon} \left[\log \begin{pmatrix} 0 \\ \frac{R^\top \mathbf{e}_1 \varepsilon}{2} \end{pmatrix} \quad \cdots \right] \\
&= \lim_{\varepsilon \to 0} \frac{1}{\varepsilon} \begin{bmatrix} R^\top \mathbf{e}_1 \varepsilon & R^\top \mathbf{e}_2 \varepsilon & R^\top \mathbf{e}_3 \varepsilon \end{bmatrix} \\
&= R^\top.
\end{aligned}
$$

You could probably come up with this only just from intuition as well. If you remember that quaternions and rotation matrices share a Lie Algebra, and that the order of concatenation is backwards, then $R^\top$ is the logical choice.

$$\boxed{\mathrm{Ad}\left(\mathbf{q}\right) = R^\top\left(\mathbf{q}\right)}$$

9.3 Adjoint for $SE\left(3\right)$

Computing this adjoint is more complicated than pure rotation. Unfortunately, $SE\left(3\right)$ isn't self-adjoint,[23] so the limit gets a little more hairy. We have two main blocks: the first three generators are the rotation generators, while the last three are the translation generators. Let's consider these separately. First, the rotation blocks

$$
\begin{aligned}
\mathrm{Ad}_{1-3} &= \lim_{\varepsilon \to 0} \frac{1}{\varepsilon} \left[\log\left(T \cdot \exp\left(J_1 \varepsilon\right) \cdot T^{-1} \right) \quad \cdots \right] \\
&= \lim_{\varepsilon \to 0} \frac{1}{\varepsilon} \left[\log \left(T \cdot \begin{bmatrix} \lfloor \mathbf{e}_1 \varepsilon \rfloor_\times & 0 \\ 0 & 0 \end{bmatrix} \cdot T^{-1} \right) \quad \cdots \right] \\
&= \lim_{\varepsilon \to 0} \frac{1}{\varepsilon} \left[\log \left(\begin{bmatrix} R & \mathbf{t} \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \lfloor \mathbf{e}_1 \varepsilon \rfloor_\times & 0 \\ 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} R^\top & -R^\top \mathbf{t} \\ 0 & 1 \end{bmatrix} \right) \quad \cdots \right] \\
&= \lim_{\varepsilon \to 0} \frac{1}{\varepsilon} \left[\log \left(\begin{bmatrix} R \lfloor \mathbf{e}_1 \varepsilon \rfloor_\times & 0 \\ 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} R^\top & -R^\top \mathbf{t} \\ 0 & 1 \end{bmatrix} \right) \quad \cdots \right] \\
&= \lim_{\varepsilon \to 0} \frac{1}{\varepsilon} \left[\log \left(\begin{bmatrix} R \lfloor \mathbf{e}_1 \varepsilon \rfloor_\times R^\top & -R \lfloor \mathbf{e}_1 \varepsilon \rfloor_\times R^\top \mathbf{t} \\ 0 & 0 \end{bmatrix} \right) \quad \cdots \right] \\
&= \lim_{\varepsilon \to 0} \frac{1}{\varepsilon} \left[\log \left(\begin{bmatrix} \lfloor R \mathbf{e}_1 \varepsilon \rfloor_\times & -\lfloor R \mathbf{e}_1 \varepsilon \rfloor_\times \mathbf{t} \\ 0 & 0 \end{bmatrix} \right) \quad \cdots \right] && \text{(Eq. 2.2)} \\
&= \lim_{\varepsilon \to 0} \frac{1}{\varepsilon} \left[\log \left(\begin{bmatrix} \lfloor R \mathbf{e}_1 \varepsilon \rfloor_\times & \lfloor \mathbf{t} \rfloor_\times R \mathbf{e}_1 \varepsilon \\ 0 & 0 \end{bmatrix} \right) \quad \cdots \right] && \text{(Eq. 2.1)} \\
&= \lim_{\varepsilon \to 0} \frac{1}{\varepsilon} \begin{bmatrix} R\varepsilon \\ \lfloor \mathbf{t} \rfloor_\times R \varepsilon \end{bmatrix} \\
&= \begin{bmatrix} R \\ \lfloor \mathbf{t} \rfloor_\times R \end{bmatrix}.
\end{aligned}
$$

[23] This should be pretty obvious, seeing as $\mathfrak{se}\left(3\right)$ is 6-dimensional while $SE\left(3\right)$ is a group of 4×4 matrices.

Now for the translation blocks

$$\begin{aligned}
\mathrm{Ad}_{4-6} &= \lim_{\varepsilon\to 0}\frac{1}{\varepsilon}\left[\log\left(T\cdot\exp\left(J_4\varepsilon\right)\cdot T^{-1}\right) \quad \cdots\right] \\
&= \lim_{\varepsilon\to 0}\frac{1}{\varepsilon}\left[\log\left(\begin{bmatrix} R & \mathbf{t} \\ 0 & 1\end{bmatrix}\cdot\begin{bmatrix} 0 & \mathbf{e}_1\varepsilon \\ 0 & 0\end{bmatrix}\cdot\begin{bmatrix} R^\top & -R^\top\mathbf{t} \\ 0 & 1\end{bmatrix}\right) \quad \cdots\right] \\
&= \lim_{\varepsilon\to 0}\frac{1}{\varepsilon}\left[\log\left(\begin{bmatrix} 0 & R\mathbf{e}_1\varepsilon \\ 0 & 0\end{bmatrix}\cdot\begin{bmatrix} R^\top & -R^\top\mathbf{t} \\ 0 & 1\end{bmatrix}\right) \quad \cdots\right] \\
&= \lim_{\varepsilon\to 0}\frac{1}{\varepsilon}\left[\log\left(\begin{bmatrix} 0 & R\mathbf{e}_1\varepsilon \\ 0 & 0\end{bmatrix}\right) \quad \cdots\right] \\
&= \lim_{\varepsilon\to 0}\frac{1}{\varepsilon}\begin{bmatrix} 0 & 0 & 0 \\ R\mathbf{e}_1\varepsilon & R\mathbf{e}_2\varepsilon & R\mathbf{e}_3\varepsilon\end{bmatrix} \\
&= \begin{bmatrix} 0 \\ R\end{bmatrix}.
\end{aligned}$$

We now concatenate these blocks to get the final Adjoint matrix:

$$\boxed{\mathrm{Ad}\left(T\right) = \begin{pmatrix} R & 0 \\ \lfloor\mathbf{t}\rfloor_\times R & R\end{pmatrix}.}$$

Note that some literature[3, 4] derives this block-transposed. All that means is that they define generators 1-3 as the translation generators and 4-6 as the rotation generators.

Before moving on, let's take a second to develop a little bit of intuition regarding the adjoint when it comes to rigid transforms. Consider a rod rotating about a fixed origin, as shown in Figure 9.1. Let us say we have the linear and angular velocity of coordinate frame a, but we want the velocity of coordinate frame b. Because the coordinate frames are fixed by some rigid transform, any angular rate value of a is the same for b, only rotated. However, to compute the linear velocity of coordinate frame b, we must account for the "lever arm" and the angular rate. This is what is going on in the bottom-left corner of the Adjoint in $SE\left(3\right)$.

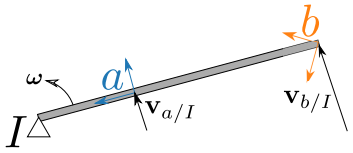

Fig. 9.1: Illustration of a rotating rigid body with two coordinate frames. The Adjoint of $SE\left(3\right)$ will convert $\boldsymbol{\omega}$ and $\mathbf{v}$ from frame a to frame b, and account for the change in translation.

9.4 Adjoint of Dual Quaternions

As with unit quaternions, dual unit quaternions are also self-adjoint:

$$\mathrm{Ad}\left(\Gamma\right) = \Gamma\cdot\exp\left(\mathbf{d}\right)\cdot\Gamma^{-1}.$$

We have no problem dealing with six-vectors here (the problem with $SE(3)$,which led us to compute that hairy limit). However, just like unit quaternions, we can compute the matrix form of the Adjoint using the limit expression. Again, let's split up the generators into two groups to help keep things a little more concise

$$
\begin{aligned}
\mathrm{Ad}_{1-3} &= \lim_{\varepsilon\to 0}\frac{1}{\varepsilon}\begin{bmatrix}\log\left(\Gamma\cdot\exp\left(J_1\varepsilon\right)\cdot\Gamma^{-1}\right) & \cdots\end{bmatrix} \\
&= \lim_{\varepsilon\to 0}\frac{1}{\varepsilon}\left[\log\left(\begin{bmatrix}\mathbf{q}_r\\ \epsilon\mathbf{q}_d\end{bmatrix}\cdot\left[\begin{pmatrix}0\\ \mathbf{e}_1\sin\left(\frac{\varepsilon}{2}\right)\\ 0\epsilon\end{pmatrix}\right]\cdot\begin{bmatrix}\mathbf{q}_r\\ \epsilon\mathbf{q}_d\end{bmatrix}^{-1}\right) \quad \cdots\right] \\
&= \lim_{\varepsilon\to 0}\frac{1}{\varepsilon}\left[\log\left(\begin{bmatrix}\mathbf{q}_r\\ \epsilon\mathbf{q}_d\end{bmatrix}\cdot\begin{bmatrix}\mathbf{e}_1\varepsilon\\ 0\epsilon\end{bmatrix}\cdot\begin{bmatrix}\mathbf{q}_r\\ \epsilon\mathbf{q}_d\end{bmatrix}^{-1}\right) \quad \cdots\right] && \sin\varepsilon\to\varepsilon \\
&= \lim_{\varepsilon\to 0}\frac{1}{\varepsilon}\left[\log\left(\begin{bmatrix}\mathbf{q}_r\cdot\mathbf{e}_1^{\wedge}\varepsilon\\ \epsilon\left(\mathbf{e}_1^{\wedge}\varepsilon\cdot\mathbf{q}_d\right)\end{bmatrix}\cdot\begin{bmatrix}\mathbf{q}_r\\ \epsilon\mathbf{q}_d\end{bmatrix}^{-1}\right) \quad \cdots\right] \\
&= \lim_{\varepsilon\to 0}\frac{1}{\varepsilon}\left[\log\left(\begin{bmatrix}\mathbf{q}_r\cdot\mathbf{e}_1^{\wedge}\varepsilon\cdot\mathbf{q}_r^{-1}\\ \epsilon\left(\mathbf{q}_r\cdot\mathbf{e}_1^{\wedge}\varepsilon\cdot\mathbf{q}_d^{-1}+\mathbf{q}_r^{-1}\cdot\mathbf{e}_1^{\wedge}\varepsilon\cdot\mathbf{q}_d\right)\end{bmatrix}\right) \quad \cdots\right] && \mathbf{e}_1^{\wedge}=\begin{pmatrix}0\\ \mathbf{e}_1\end{pmatrix} \\
&= \lim_{\varepsilon\to 0}\frac{1}{\varepsilon}\left[\log\left(\begin{bmatrix}\mathbf{q}_r\cdot\mathbf{e}_1^{\wedge}\varepsilon\cdot\mathbf{q}_r^{-1}\\ \epsilon\left(\mathbf{q}_r\cdot\mathbf{e}_1^{\wedge}\varepsilon\cdot\left(\frac{1}{2}\mathbf{t}^{\wedge}\cdot\mathbf{q}_r\right)^{-1}+\mathbf{q}_r^{-1}\cdot\mathbf{e}_1^{\wedge}\varepsilon\cdot\frac{1}{2}\mathbf{t}^{\wedge}\cdot\mathbf{q}_r\right)\end{bmatrix}\right) \quad \cdots\right] \\
&= \lim_{\varepsilon\to 0}\frac{1}{\varepsilon}\left[\log\left(\begin{bmatrix}\mathbf{q}_r\cdot\mathbf{e}_1^{\wedge}\varepsilon\cdot\mathbf{q}_r^{-1}\\ \frac{1}{2}\epsilon\left(\mathbf{q}_r\cdot\mathbf{e}_1^{\wedge}\varepsilon\cdot\mathbf{q}_r^{-1}\cdot(-\mathbf{t}^{\wedge})+\mathbf{q}_r^{-1}\cdot\mathbf{e}_1^{\wedge}\varepsilon\cdot\mathbf{t}^{\wedge}\cdot\mathbf{q}_r\right)\end{bmatrix}\right) \quad \cdots\right] \\
&= \lim_{\varepsilon\to 0}\frac{1}{\varepsilon}\left[\log\left(\begin{bmatrix}\mathbf{q}_r\cdot\mathbf{e}_1^{\wedge}\varepsilon\cdot\mathbf{q}_r^{-1}\\ \frac{1}{2}\epsilon\left(-\left\lfloor R^{\top}\mathbf{e}_1\varepsilon\right\rfloor_{\times}\mathbf{t}+R\left\lfloor\mathbf{e}_1\varepsilon\right\rfloor_{\times}\mathbf{t}\right)\end{bmatrix}\right) \quad \cdots\right] && \text{(Eq. 4.2)} \\
&= \lim_{\varepsilon\to 0}\frac{1}{\varepsilon}\left[\log\left(\begin{bmatrix}\mathbf{q}_r\cdot\mathbf{e}_1^{\wedge}\varepsilon\cdot\mathbf{q}_r^{-1}\\ \frac{1}{2}\epsilon\left(-R^{\top}\left\lfloor\mathbf{e}_1\varepsilon\right\rfloor_{\times}R\mathbf{t}+R\left\lfloor\mathbf{e}_1\varepsilon\right\rfloor_{\times}\mathbf{t}\right)\end{bmatrix}\right) \quad \cdots\right] && \text{(Eq. 2.2)} \\
&= \lim_{\varepsilon\to 0}\frac{1}{\varepsilon}\left[\log\left(\begin{bmatrix}\mathbf{q}_r\cdot\mathbf{e}_1^{\wedge}\varepsilon\cdot\mathbf{q}_r^{-1}\\ \frac{1}{2}\epsilon\left(R^{\top}\left\lfloor R\mathbf{t}\right\rfloor_{\times}\mathbf{e}_1\varepsilon-R\left\lfloor\mathbf{t}\right\rfloor_{\times}\mathbf{e}_1\varepsilon\right)\end{bmatrix}\right) \quad \cdots\right] && \text{(Eq. 2.1)} \\
&= \lim_{\varepsilon\to 0}\frac{1}{\varepsilon}\left[\log\left(\begin{bmatrix}\mathbf{q}_r\cdot\mathbf{e}_1^{\wedge}\varepsilon\cdot\mathbf{q}_r^{-1}\\ \frac{1}{2}\epsilon\left(\left\lfloor\mathbf{t}\right\rfloor_{\times}R^{\top}\mathbf{e}_1\varepsilon-R\left\lfloor\mathbf{t}\right\rfloor_{\times}\mathbf{e}_1\varepsilon\right)\end{bmatrix}\right) \quad \cdots\right] && \text{(Eq. 2.1)} \\
&= \begin{bmatrix}R^{\top}\\ \left\lfloor\mathbf{t}\right\rfloor_{\times}R^{\top}\end{bmatrix}.
\end{aligned}
$$

Whew! Luckily, the translation generators are much easier:

$$
\begin{aligned}
\mathrm{Ad}_{4-6} &= \lim_{\varepsilon \to 0} \frac{1}{\varepsilon} \begin{bmatrix} \log\left(\Gamma \cdot \exp\left(J_1 \varepsilon\right) \cdot \Gamma^{-1}\right) & \cdots \end{bmatrix} \\
&= \lim_{\varepsilon \to 0} \frac{1}{\varepsilon} \begin{bmatrix} \log\left(\begin{bmatrix} \mathbf{q}_r \\ \epsilon \mathbf{q}_d \end{bmatrix} \cdot \begin{bmatrix} 0 \\ \mathbf{e}_1 \epsilon \end{bmatrix} \cdot \begin{bmatrix} \mathbf{q}_r \\ \epsilon \mathbf{q}_d \end{bmatrix}^{-1} \right) & \cdots \end{bmatrix} \\
&= \lim_{\varepsilon \to 0} \frac{1}{\varepsilon} \begin{bmatrix} \log\left(\begin{bmatrix} 0 \\ \epsilon \left(\mathbf{q}_r \cdot \mathbf{e}_q \varepsilon\right) \end{bmatrix} \cdot \begin{bmatrix} \mathbf{q}_r \\ \epsilon \mathbf{q}_d \end{bmatrix}^{-1} \right) & \cdots \end{bmatrix} \\
&= \lim_{\varepsilon \to 0} \frac{1}{\varepsilon} \begin{bmatrix} \log\left[\begin{matrix} 0 \\ \epsilon \left(\mathbf{q}_r \cdot \mathbf{e}_q \varepsilon \cdot \mathbf{q}_r^{-1}\right) \end{matrix} \right] & \cdots \end{bmatrix} \\
&= \lim_{\varepsilon \to 0} \frac{1}{\varepsilon} \begin{bmatrix} \log\left[\begin{matrix} 0 \\ \epsilon \left(R^\top \mathbf{e}_1 \varepsilon\right) \end{matrix} \right] & \cdots \end{bmatrix} \\
&= R^\top.
\end{aligned}
$$

Finally, we can write the full Adjoint for dual quaternions as

$$
\boxed{\mathrm{Ad}\left(\Gamma\right) = \begin{pmatrix} R^\top & 0 \\ \lfloor \mathbf{t} \rfloor_\times R^\top & R^\top \end{pmatrix}},
$$

which, unsurprisingly, is the same as the Adjoint of $SE\left(3\right)$, except with all the rotations transposed (due to the reverse order of quaternions rotation with respect to $SO\left(3\right)$.) Note that this is also the matrix inverse of the Adjoint of $SE\left(3\right)$.

Now that we have each of the Adjoints, we can use the tables in Appendix A to find Jacobians for almost any arbitrary expression involving Lie groups!

10 Left vs Right Jacobians

For now, just read [5]. They have a great discussion on this topic.

TL;DR: Most of the time, I end up using the left Jacobian. Depending on how you define error, however, you may want to use the right Jacobian. If things are still hazy after reading Sola, write unit tests of your implementations and make sure everything is self-consistent.

A Useful Tables

Tab. 2: Common dual number formulae

$$
\begin{aligned}
f\left(r+d\epsilon\right) &= f\left(r\right)+\epsilon d\left(f'\left(r\right)\right)\\
\cos\left(r+d\epsilon\right) &= \cos r-\epsilon d\sin r\\
\sin\left(r+d\epsilon\right) &= \sin r+\epsilon d\cos r\\
\arctan\left(r+d\epsilon\right) &= \arctan r+\epsilon\frac{d}{r^2+1}\\
\left(r+d\epsilon\right)^2 &= r^2+\epsilon 2rd\\
\sqrt{r+d\epsilon} &= \sqrt{r}+\epsilon\frac{d}{2\sqrt{r}}
\end{aligned}
$$

Tab. 3: Taylor series expansions

$$
\begin{aligned}
\sin\left(\theta\right) &= \theta-\frac{1}{3!}\theta^3+\frac{1}{5!}\theta^5\cdots\\
\cos\left(\theta\right) &= 1-\frac{1}{2}\theta^2+\frac{1}{4!}\theta^4\cdots\\
\cos\left(\frac{\theta}{2}\right) &= 1-\frac{1}{8}\theta^2+\frac{1}{46080}\theta^4\cdots\\
\frac{\sin\left(\theta\right)}{\theta} &= 1-\frac{1}{3!}\theta^2+\frac{1}{5!}\theta^4\cdots\\
\frac{\sin\left(\frac{\theta}{2}\right)}{\theta} &= \frac{1}{2}-\frac{1}{48}\theta^2+\frac{1}{3840}\theta^4+\cdots\\
\frac{\theta}{\sin\left(\theta\right)} &= 1+\frac{1}{3!}\theta^2+\frac{7}{360}\theta^4\cdots\\
\frac{1-\cos\left(\theta\right)}{\theta^2} &= \frac{1}{2}-\frac{1}{24}\theta^2+\frac{1}{720}\theta^4\cdots\\
\frac{\cos\theta-\frac{\sin\theta}{\theta}}{\theta^2} &= -\frac{1}{3}+\frac{1}{30}\theta^2-\frac{1}{840}\theta^4\cdots\\
\frac{\frac{1}{2}\cos\frac{\theta}{2}-\frac{\sin\frac{\theta}{2}}{\theta}}{\theta^2} &= -\frac{1}{24}+\frac{1}{960}\theta^2-\frac{1}{107520}\theta^4\cdots\\
\frac{\tan^{-1}\left(\frac{\theta}{y}\right)}{\theta} &= \frac{1}{y}-\frac{1}{3y^3}\theta^2+\frac{1}{5y^5}\theta^4
\end{aligned}
$$

Tab. 4: Group-Group Jacobians

Expression	Left Jacobian	Right Jacobian
$\frac{\partial}{\partial \mathbf{x}} \mathbf{y} \cdot \mathbf{x}$	$\mathrm{Ad}\left(\mathbf{y}\right)^{-1}$	I
$\frac{\partial}{\partial \mathbf{x}} \mathbf{x}^{-1} \cdot \mathbf{y}$	$-\mathrm{Ad}\left(\mathbf{x}\right)^{-1}$	$-\mathrm{Ad}\left(\mathbf{x}^{-1} \cdot \mathbf{y}\right)^{-1}$
$\frac{\partial}{\partial \mathbf{x}} \mathbf{y} \cdot \mathbf{x}^{-1}$	$-\mathrm{Ad}\left(\mathbf{y} \cdot \mathbf{x}^{-1}\right)$	$-\mathrm{Ad}\left(\mathbf{x}\right)$
$\frac{\partial}{\partial \mathbf{x}} \mathbf{x} \cdot \mathbf{y}$	I	$\mathrm{Ad}\left(\mathbf{y}\right)^{-1}$

$\mathbf{x}, \mathbf{y} \in G$

Tab. 5: Useful Jacobians for $SO\left(3\right)$

Expression	Left Jacobian	Right Jacobian
$\frac{\partial}{\partial \mathbf{x}} R \cdot \mathbf{v}$	$-\left\lfloor R \cdot \mathbf{v} \right\rfloor_{\times}$	$-R \cdot \left\lfloor \mathbf{v} \right\rfloor_{\times}$
$\frac{\partial}{\partial \mathbf{x}} R^{\top} \cdot \mathbf{v}$	$R^{\top} \left\lfloor \mathbf{v} \right\rfloor_{\times}$	$\left\lfloor R^{\top} \cdot \mathbf{v} \right\rfloor_{\times}$
$\frac{\partial}{\partial \boldsymbol{\omega}} \exp\left(\left\lfloor \boldsymbol{\omega} \right\rfloor_{\times}\right)$	$aI + b\left\lfloor \boldsymbol{\omega} \right\rfloor_{\times} + c\boldsymbol{\omega}\boldsymbol{\omega}^{\top}$	$aI - b\left\lfloor \boldsymbol{\omega} \right\rfloor_{\times} + c\boldsymbol{\omega}\boldsymbol{\omega}^{\top}$
$\frac{\partial}{\partial R} \log\left(R\right)$	$I - \frac{1}{2}\left\lfloor \boldsymbol{\delta} \right\rfloor_{\times} + e\left\lfloor \boldsymbol{\delta} \right\rfloor_{\times}^{2}$	$I + \frac{1}{2}\left\lfloor \boldsymbol{\delta} \right\rfloor_{\times} + e\left\lfloor \boldsymbol{\delta} \right\rfloor_{\times}^{2}$

$R \in SO\left(3\right), \quad \mathbf{v}, \boldsymbol{\omega} \in \mathbb{R}^{3} \quad \boldsymbol{\delta} = \log\left(R\right) \quad \theta = \left\|\boldsymbol{\delta}\right\|$

$a = \frac{\sin\theta}{\theta} \quad b = \frac{1-\cos(\theta)}{\theta^{2}} \quad c = \frac{1-a}{\theta^{2}} \quad e = \frac{b-2c}{2a}$

Tab. 6: Useful Jacobians Jacobians for $SE\left(3\right)$

Expression	Left Jacobian	Right Jacobian
$\frac{\partial}{\partial \mathbf{x}} T \cdot \mathbf{v}$	$\begin{pmatrix} -\lfloor R \cdot \mathbf{v} + \mathbf{t} \rfloor_\times & I \end{pmatrix}$	$\begin{pmatrix} -R \cdot \lfloor \mathbf{v} \rfloor_\times & R \end{pmatrix}$
$\frac{\partial}{\partial \mathbf{x}} T^{-1} \cdot \mathbf{v}$	$\begin{pmatrix} R^\top \cdot \lfloor \mathbf{v} \rfloor_\times & -R^\top \end{pmatrix}$	$\begin{pmatrix} \left\lfloor R^\top \cdot (\mathbf{v} - \mathbf{t}) \right\rfloor_\times & -I \end{pmatrix}$
$\frac{\partial}{\partial \boldsymbol{\xi}} \exp\left(\boldsymbol{\xi}^\wedge\right)$	$\begin{pmatrix} A & 0 \\ D & A \end{pmatrix}$	$\begin{pmatrix} A^\top & 0 \\ D^\top & A^\top \end{pmatrix}$
$\frac{\partial}{\partial T} \log\left(T\right)$	$\begin{pmatrix} E & 0 \\ -E \cdot D \cdot E & E \end{pmatrix}$	$\begin{pmatrix} E^\top & 0 \\ -\left(E \cdot D \cdot E\right)^\top & E^\top \end{pmatrix}$

$$T \in SE\left(3\right), \quad \mathbf{v} \in \mathbb{R}^3, \quad \boldsymbol{\xi} = \begin{pmatrix} \boldsymbol{\omega} & \mathbf{v} \end{pmatrix}^\top \in \mathfrak{se}\left(3\right)$$

$$a = \tfrac{\sin\theta}{\theta} \quad b = \tfrac{1-\cos(\theta)}{\theta^2} \quad c = \tfrac{1-a}{\theta^2} \quad d = \boldsymbol{\omega}^\top \mathbf{v} \quad A = \tfrac{\partial}{\partial \boldsymbol{\omega}} \exp\left(\lfloor \boldsymbol{\omega} \rfloor_\times\right)$$

$$B = \boldsymbol{\omega}\mathbf{v}^\top + \mathbf{v}\boldsymbol{\omega}^\top \quad C = \left(c - b\right) I + \left(\tfrac{a-2b}{\theta^2}\right) \lfloor \boldsymbol{\omega} \rfloor_\times + \left(\tfrac{b-3c}{\theta^2}\right) \boldsymbol{\omega}\boldsymbol{\omega}^\top$$

$$D = b \lfloor \mathbf{v} \rfloor_\times + cB + dC \quad E = \tfrac{\partial}{\partial R} \log\left(R\right)$$

Tab. 7: Useful Jacobians for $\mathcal{S}^3$

Expression	Left Jacobian	Right Jacobian
$\frac{\partial}{\partial \mathbf{x}} \mathbf{q}^{-1} \cdot \mathbf{v}^\wedge \cdot \mathbf{q}$	$R\left(\mathbf{q}\right)^\top \cdot \lfloor \mathbf{v} \rfloor_\times$	$\left\lfloor R\left(\mathbf{q}\right)^\top \cdot \mathbf{v} \right\rfloor_\times$
$\frac{\partial}{\partial \mathbf{x}} \mathbf{q} \cdot \mathbf{v}^\wedge \cdot \mathbf{q}^{-1}$	$-\lfloor R\left(\mathbf{q}\right) \rfloor_\times \cdot \mathbf{v}$	$-R\left(\mathbf{q}\right) \cdot \lfloor \mathbf{v} \rfloor_\times$
$\frac{\partial}{\partial \boldsymbol{\omega}} \exp\left(\boldsymbol{\omega}^\wedge\right)$	$aI + b \lfloor \boldsymbol{\omega} \rfloor_\times + c\boldsymbol{\omega}\boldsymbol{\omega}^\top$	$aI - b \lfloor \boldsymbol{\omega} \rfloor_\times + c\boldsymbol{\omega}\boldsymbol{\omega}^\top$
$\frac{\partial}{\partial \mathbf{q}} \log\left(\mathbf{q}\right)$	$I - \frac{1}{2} \lfloor \boldsymbol{\delta} \rfloor_\times + e \lfloor \boldsymbol{\delta} \rfloor_\times^2$	$I + \frac{1}{2} \lfloor \boldsymbol{\delta} \rfloor_\times + e \lfloor \boldsymbol{\delta} \rfloor_\times^2$

$$\mathbf{q} \in \mathcal{S}^3 \quad \mathbf{v}, \boldsymbol{\omega} \in \mathfrak{s}^3 \quad \boldsymbol{\delta} = \log\left(\mathbf{q}\right) \quad \theta = \|\boldsymbol{\delta}\|$$
$$a = \tfrac{\sin\theta}{\theta} \quad b = \tfrac{1-\cos(\theta)}{\theta^2} \quad c = \tfrac{1-a}{\theta^2} \quad e = \tfrac{b-2c}{2a}$$

Tab. 8: Useful Jacobians for $\mathbb{D}\mathcal{S}^3$

Expression	Left Jacobian	Right Jacobian
$\frac{\partial}{\partial \mathbf{x}} \Gamma^{-1} \cdot \mathbf{v}^{\wedge} \cdot \Gamma$	$\begin{pmatrix} R(\mathbf{q}_r) \lfloor \mathbf{v} \rfloor_{\times} & R(\mathbf{q}) \end{pmatrix}$	$\begin{pmatrix} \lfloor R(\mathbf{q}_r)(\mathbf{v}-\mathbf{t}) \rfloor_{\times} & -I \end{pmatrix}$
$\frac{\partial}{\partial \mathbf{x}} \Gamma \cdot \mathbf{v}^{\wedge} \cdot \Gamma^{-1}$	$\begin{pmatrix} -\left\lfloor R(\mathbf{q}_r)^{\top} \right\rfloor_{\times} \mathbf{v} + \mathbf{t} & I \end{pmatrix}$	$\begin{pmatrix} R(\mathbf{q}_r)^{\top} \lfloor -\mathbf{v} \rfloor_{\times} & R(\mathbf{q}_r)^{\top} \end{pmatrix}$
$\frac{\partial}{\partial \boldsymbol{\xi}} \exp(\boldsymbol{\xi}^{\wedge})$	$\begin{pmatrix} A & 0 \\ D & A \end{pmatrix}$	$\begin{pmatrix} A^{\top} & 0 \\ D^{\top} & A^{\top} \end{pmatrix}$
$\frac{\partial}{\partial T} \log(\Gamma)$	$\begin{pmatrix} E & 0 \\ -E \cdot D \cdot E & E \end{pmatrix}$	$\begin{pmatrix} E^{\top} & 0 \\ (-E \cdot D \cdot E)^{\top} & E^{\top} \end{pmatrix}$

$$\Gamma \in \mathbb{D}\mathcal{S}^3, \quad \boldsymbol{\xi} = \begin{pmatrix} \boldsymbol{\omega} & \mathbf{v} \end{pmatrix}^{\top} \in \mathfrak{d}\mathfrak{s}^3$$

$$a = \tfrac{\sin\theta}{\theta} \quad b = \tfrac{1-\cos(\theta)}{\theta^2} \quad c = \tfrac{1-a}{\theta^2} \quad d = \boldsymbol{\omega}^{\top}\mathbf{v} \quad A = \tfrac{\partial}{\partial \boldsymbol{\omega}} \exp\left(\lfloor \boldsymbol{\omega} \rfloor_{\times}\right)$$

$$B = \boldsymbol{\omega}\mathbf{v}^{\top} + \mathbf{v}\boldsymbol{\omega}^{\top} \quad C = (c-b)\, I + \left(\tfrac{a-2b}{\theta^2}\right) \lfloor \boldsymbol{\omega} \rfloor_{\times} + \left(\tfrac{b-3c}{\theta^2}\right) \boldsymbol{\omega}\boldsymbol{\omega}^{\top}$$

$$D = b \lfloor \mathbf{v} \rfloor_{\times} + cB + dC \quad E = \tfrac{\partial}{\partial R} \log(R)$$

References

[1] Timothy D. Barfoot. *State Estimation For Robotics*. 2019. Available at `http://asrl.utias.utoronto.ca/~tdb/bib/barfoot_ser17.pdf`,.

[2] Neil T. Dantam. Practical exponential coordinates using implicit dual quaternions. 2018. Available at `http://www.neil.dantam.name/papers/dantam2018practical.pdf`.

[3] Tom Drummond. Lie groups, lie algebras, projective geometry and optimization for 3d geometry, engineering and computer vision, 2014. Available at `https://www.dropbox.com/s/5y3tvypzps59s29/3DGeometry.pdf?dl=0`.

[4] Ethan Eade. Lie groups for 2d and 3d transformations, 2017. Available at `http://ethaneade.com/lie.pdf`,.

[5] Dinesh Atchuthan Joan Sola, Jeremie Deray. A micro lie theory for state estimation in robotics. 2019. Available at `https://arxiv.org/pdf/1812.01537.pdf`.

[6] Michael Syskind Pedersen Kaare Brandt Petersen. The matrix cookbook, 2012. Available at `https://www.math.uwaterloo.ca/~hwolkowi/matrixcookbook.pdf`.

[7] Jakob Schwichtenberg. *Physics from symmetry*. Springer International Publishing, 2015. Available at `http://www.infis.ufu.br/~gerson/grupos/3%20-%20Great%20books/Physics%20from%20Symmetry.pdf`.

[8] Gregory G. Slabaugh. Computing euler angles from a rotation matrix. Available at `http://www.gregslabaugh.net/publications/euler.pdf`.