Topic: Using quaternions to describe 3D rotations

Research Question: How and why are quaternions used to compute 3D rotations?

Subject: Mathematics

Word Count: 3978

Table of Contents

1. Introduction

Quaternions are an extension of the complex numbers first introduced by William Rowan Hamilton in the 19th century(Byrne). His goal was to find a number system which incorporates rotations in 3 dimensions, analogous to how the product of complex numbers results in rotations in 2 dimensions. Initially, he tried to work with two imaginary numbers i and j, which, unfortunately, led to contradictions. Famously, while walking along the Royal Canal in Dublin with his wife, he realised that the solution was introducing a third imaginary number, k. Hamilton instantly carved this onto Brougham Bridge, so as not to forget it(Graves 434-435).

As is the case with such things in mathematics, Hamilton's ideas were initially ridiculed, with some like Oliver Heaviside going so far as to call them evil: "So far as the vector analysis I required was concerned, the quaternion was not only not required, but was a positive evil of no considerable magnitude"(Crowe 171). Quaternions have, nonetheless, become incredibly popular as robotics, virtual reality, augmented reality, etc. rise in popularity as they all require 3D rotations. Furthermore, there is little doubt that the need for computing 3D rotations will only continue to increase, as these grow more popular. In the past few years, quaternions have become the standard way of performing these rotations. As was shown by Andy Matuschak, an ex-employee of Apple, even our phones use them on a regular basis in order to determine orientation and such.

There are other ways of computing 3D rotations, each with their own advantages and disadvantages. Understanding the differences is important in order to implement the different methods effectively. Therefore, in this essay I would like to demonstrate how quaternions operate and provide some intuition as to how they can be used to compute

these rotations, while also discussing the nuances of using them in order to gain a better understanding of when to use them.

2. Quaternions

2.1 Definition

As mentioned previously, quaternions are an extension of the complex numbers. A complex number is a number z can be defined as:

$$z = a + bi,\ a, b \in \mathbb{R} \text{ and } i^2 = -1$$

Similarly, a quaternion is a number q that can be defined as:

$$q = a + bi + cj + dk,\ q \in \mathbf{H},\ a, b, c, d \in \mathbb{R}$$

Where $\mathbf{H}$ represents the set of quaternions and i, j, k represent imaginary numbers which Hamilton(46) defined as

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

a is often referred to as the real or scalar component of the quaternion, while $bi + cj + dk$ is called the imaginary or vector component. This allows us to write quaternions in a different form:

$$q = (a, \vec{q}), a \in \mathbb{R}, \vec{q} \in \mathbb{R}^3$$

A quaternion whose vector component is zero is called a scalar quaternion(or simply scalar) and a quaternion whose scalar component is zero is called a pure or vector quaternion. Despite vector notation being used for $\vec{q}$, it should be noted that it is still a quaternion. The notation is simply a reminder of its imaginary nature. The reason for vector notation will be made apparent in the following section.

2.2 Vector space of quaternions

As we will see, it is useful to conceptualise quaternions as vectors. Hence, we will start by showing this is a valid interpretation of quaternions by proving that they form a vector space over the real numbers.

Adding two quaternions is a simple component wise addition.

Let $p = p_0 + p_1 i + p_2 j + p_3 k$ and $q = q_0 + q_1 i + q_2 j + q_3 k$

$$p + q = (p_0 + q_0) + (p_1 + q_1) i + (p_2 + q_2) j + (p_3 + q_3) k$$

$$= (q_0 + p_0) + (q_1 + p_1) i + (q_2 + p_2) j + (q_3 + p_3) k$$

$$= q + p$$

The second step above is justified by the commutativity of addition of real numbers. The associativity of addition is also simple to see.

For $p, q, r \in \mathbf{H}$

$$(p + q) + r = ((p_0 + q_0) + r_0) + ((p_1 + q_1) + r_1) i + ((p_2 + q_2) + r_2) j + ((p_3 + q_3) + r_3) k$$

$$= (p_0 + (q_0 + r_0)) + (p_1 + (q_1 + r_1)) i + (p_2 + (q_2 + r_2)) j + (p_3 + (q_3 + r_3)) k$$

$$= p + (q + r)$$

Once again, we can use associativity of addition of real numbers to justify the second step.

The additive identity, 0, in $\mathbf{H}$ is $0 + 0i + 0j + 0k$,

$$q + 0 = (q_0 + 0) + (q_1 + 0) i + (q_2 + 0) j + (q_3 + 0) k$$

$$= q_0 + q_1 i + q_2 j + q_3 k = q$$

For every $q \in \mathbf{H}$, there exists a corresponding $-q \in \mathbf{H}$, such that $q + (-q) = 0$.

Let $-q = (-q_0) + (-q_1) i + (-q_2) j + (-q_3) k$, then

$$q + (-q) = (q_0 + (-q_0)) + (q_1 + (-q_1)) i + (q_2 + (-q_2)) j + (q_3 + (-q_3)) k$$

$$= 0 + 0i + 0j + 0k$$

We can hence conclude that quaternions form an abelian group under addition.

To establish that quaternions form a vector space, we need to confirm the properties of scalar multiplication. The product of a quaternion and a real number is defined as:

$$aq = qa = aq_0 + aq_1 i + aq_2 j + aq_3 k, a \in \mathbb{R}, q \in \mathbf{H}$$

For some $a, b \in \mathbb{R}$ and $q \in \mathbf{H}$:

$$a(bq) = a(bq_0 + bq_1 i + bq_2 j + bq_3 k)$$

$$= abq_0 + abq_1 i + abq_2 j + abq_3 k$$

$$= b(aq_0 + aq_1 i + aq_2 j + aq_3 k)$$

$$= b(aq)$$

1 from the field of reals also acts as the identity element for quaternions.

$$1q = 1q_0 + 1q_1 i + 1q_2 j + 1q_3 k$$

$$= q_0 + q_1 i + q_2 j + q_3 k$$

$$= q$$

Finally, we need to confirm the distributivity of scalar multiplication with respect to vector and scalar addition. For $a,b \in \mathbb{R}$ and $p,q \in \mathbf{H}$:

$$a(p+q) = (ap_0 + aq_0) + (ap_1 + aq_1)i + (ap_2 + aq_2)j + (ap_3 + aq_3)k$$

$$= (ap_0 + ap_1 i + ap_2 j + ap_3 k) + (aq_0 + aq_1 i + aq_2 j + aq_3 k)$$

$$= ap + aq$$

$$(a+b)q = (aq_0 + bq_0) + (aq_1 + bq_1)i + (aq_2 + bq_2)j + (aq_3 + bq_3)k$$

$$= (aq_0 + aq_1 i + aq_2 j + aq_3 k) + (bq_0 + bq_1 i + bq_2 j + bq_3 k)$$

$$= aq + bq$$

We have thus proven that quaternions form a vector space over the real numbers, which allows us to think of quaternions as vectors in $\mathbb{R}^4$ with the basis vectors $\{1, i, j, k\}$. In fact, the imaginary components are often thought of as spanning the familiar x, y and z axes while the real numbers live in the 'fourth' dimension. We can now use the tools of linear algebra if and when necessary to understand how quaternionic operations affect the space.

2.3 Quaternionic multiplication

Quaternions have an additional operation, multiplication. The product of any two quaternions can be found using the definition of the imaginary numbers. For example, since $i \times i = -1$ and $i \times jk = -1$, then $i = jk$. Similarly, we can conclude that $ij = k$. Right multiplying both sides by j results in $-i = kj$. This means that multiplication with quaternions is non-commutative. Similar methods can be used to form a set of equations, dictating the product of any two imaginary numbers.

$$\begin{aligned} i &= jk \quad -i = kj \\ j &= ki \quad -j = ik \\ k &= ij \quad -k = ji \end{aligned}$$

The above identities can be used to find the product of any two quaternions. First let us find the product of two vector quaternions.

If $r = (0, \vec{r}) = r_1 i + r_2 j + r_3 k$ and $s = (0, \vec{s}) = s_1 i + s_2 j + s_3 k$, then

$$rs = (-r_1 s_1 - r_2 s_2 - r_3 s_3) + (r_2 s_3 - r_3 s_2) i + (r_3 s_1 - r_1 s_3) j + (r_1 s_2 - r_2 s_1) k,$$

The real component is computationally equivalent to the negative dot product of the two vectors, while the imaginary component is equivalent to the cross product. We can therefore write the product more succinctly as:

$$rs = (-\vec{r} \cdot \vec{s}, \vec{r} \times \vec{s})$$

This allows us to find the product of two general quaternions.

$$pq = (p_0 q_0 - \vec{p} \cdot \vec{q}, p_0 \vec{q} + q_0 \vec{p} + \vec{p} \times \vec{q})$$

$$= (p_0 q_0 - p_1 q_1 - p_2 q_2 - p_3 q_3) + (p_0 q_1 + p_1 q_0 + p_2 q_3 - p_3 q_2) i$$

$$+ (p_0 q_2 - p_1 q_3 + p_2 q_0 + p_3 q_1) j + (p_0 q_3 + p_1 q_2 - p_2 q_1 + p_3 q_0) k$$

2.3 Conjugates

Similar to complex numbers, all quaternions have conjugates. For a quaternion, $q = q_0 + q_1 i + q_2 j + q_3 k$, its conjugate is $q^* = q_0 - (q_1 i + q_2 j + q_3 k)$. Multiplying a quaternion with its conjugate produces a positive real number:

$$qq^* = {q_0}^2 + {q_1}^2 + {q_2}^2 + {q_3}^2 = \|q\|^2$$

$\|q\|$ is the modulus of q, commonly referred to as the norm of q. It can be thought of as the 'length' of q. A quaternion of norm 1 is called a unit quaternion.

A unit quaternion, q, can also be represented as such:

$$q = (\cos\theta, \sin\theta\ \vec{v}) = \cos\theta + v_1 \sin\theta i + v_2 \sin\theta j + v_3 \sin\theta k,\ \vec{v} \in \mathbb{R}^3,\ \|\vec{v}\| = 1, \theta \in [0, 2\pi]$$

This refers to a quaternion that has been rotated θ radians from the real number line on the $1-\vec{v}$ plane(the plane spanned out by the real numbers and $\vec{v}$). These two lines are necessarily perpendicular because the real numbers are all perpendicular to the three imaginary numbers. $\vec{v}$ is simply a linear combination of the imaginary numbers therefore it must also be perpendicular to the real number line. This can easily be confirmed by the fact that if represented as vectors in $\mathbb{R}^4$ their dot product is 0. A diagram has been drawn below.

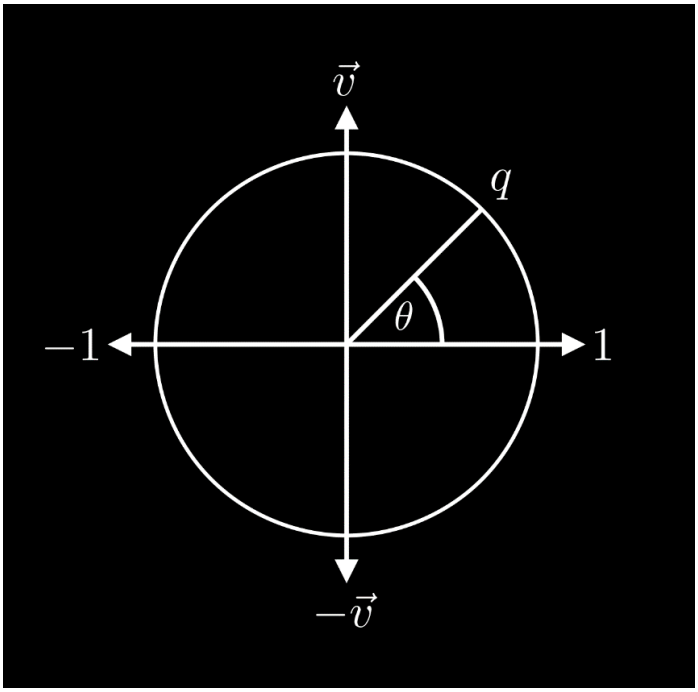

Figure 2:1 Quaternion q on the 1-v plane

An important property of conjugates is the following:

$$(pq)^* = (p_0 q_0 - \vec{p}\cdot\vec{q}, -p_0\vec{q} - q_0\vec{p} - \vec{p}\times\vec{q})$$

$$= \left(q_0 p_0 - \vec{q} \cdot \vec{p}, q_0 \left(-\vec{p} \right) + p_0 \left(-\vec{q} \right) + \left(-\vec{q} \times -\vec{p} \right) \right)$$

$$= q^* p^*$$

Conjugates are also very useful because they allow us to find the multiplicative inverse of a quaternion, q^{-1} which is defined like so:

$$qq^{-1} = 1$$

We can rearrange the equation of norm above to get:

$$q \frac{q^*}{\|q\|^2} = 1$$

Hence, it becomes simple to see that

$$q^{-1} = \frac{q^*}{\|q\|^2}$$

It also becomes clear that if q is a unit quaternion, then its inverse is equivalent to its conjugate.

Conjugation also helps us realise an important aspect of quaternionic multiplication.

$$\|pq\|^2 = (pq)(pq)^*$$

$$= pqq^* p^*$$

$$= p\left(qq^* \right) p^*$$

$$= \|q\|^2 \, pp^*$$

$$= \|q\|^2 \|p\|^2$$

$$\therefore \|pq\| = \|p\|\|q\|$$

This means that the norm of the product of two quaternions is equal to the product of their norms.

3. Rotations

3.1 Intuition for Quaternionic Rotation

In this section, we will explore the nature of quaternionic multiplication and see how rotations are a natural consequence of how quaternions transform $\mathbb{R}^4$.

Since quaternions can be treated as vectors in $\mathbb{R}^4$ we will start by studying how quaternionic multiplication transforms this space. The properties of quaternion multiplication ensure that the transformation(or mapping) is linear. This means we can decompose this transformation to its effects on the basis vectors and construct a matrix based on this. In this section, all quaternions will be unit quaternions.

Let $q = (\cos\theta, \sin\theta\ \vec{v}) = \cos\theta + v_1 \sin\theta i + v_2 \sin\theta j + v_3 \sin\theta k$

$$q1 = (\cos\theta + v_1 \sin\theta i + v_2 \sin\theta j + v_3 \sin\theta k)1 = \cos\theta + v_1 \sin\theta i + v_2 \sin\theta j + v_3 \sin\theta k$$

$$qi = (\cos\theta + v_1 \sin\theta i + v_2 \sin\theta j + v_3 \sin\theta k)i = -v_1 \sin\theta + \cos\theta i + v_3 \sin\theta j - v_2 \sin\theta k$$

$$qj = (\cos\theta + v_1 \sin\theta i + v_2 \sin\theta j + v_3 \sin\theta k)\, j = -v_2 \sin\theta - v_3 \sin\theta i + \cos\theta j + v_1 \sin\theta k$$

$$qk = (\cos\theta + v_1 \sin\theta i + v_2 \sin\theta j + v_3 \sin\theta k)\, k = -v_3 \sin\theta + v_2 \sin\theta i - v_1 \sin\theta j + \cos\theta k$$

We can represent these transformations concisely using a matrix:

$$Q = \begin{bmatrix} \cos\theta & -v_1\sin\theta & -v_2\sin\theta & -v_3\sin\theta \\ v_1\sin\theta & \cos\theta & -v_3\sin\theta & v_2\sin\theta \\ v_2\sin\theta & v_3\sin\theta & \cos\theta & -v_1\sin\theta \\ v_3\sin\theta & -v_2\sin\theta & v_1\sin\theta & \cos\theta \end{bmatrix}$$

As can be seen the matrix is constructed of four-unit vectors(even without checking, we know this to be true since the product of norms is equal to the norm of the product). Importantly, all the vectors remain orthogonal to one another. This can be confirmed by taking the dot product of any two columns. If we take a look at the transformed 1(real numbers) and i, for example, we get:

$$\begin{bmatrix} \cos\theta \\ v_1\sin\theta \\ v_2\sin\theta \\ v_3\sin\theta \end{bmatrix} \cdot \begin{bmatrix} -v_1\sin\theta \\ \cos\theta \\ v_3\sin\theta \\ -v_2\sin\theta \end{bmatrix}$$

$$= -v_1\cos\theta\sin\theta + v_1\sin\theta\cos\theta + v_2v_3\sin^2\theta - v_2v_3\sin^2\theta = 0$$

A similar situation occurs when the dot product of any of the five remaining pairs is taken.

Another important property of Q is:

$$QQ^T = \begin{bmatrix} \cos\theta & -v_1\sin\theta & -v_2\sin\theta & -v_3\sin\theta \\ v_1\sin\theta & \cos\theta & -v_3\sin\theta & v_2\sin\theta \\ v_2\sin\theta & v_3\sin\theta & \cos\theta & -v_1\sin\theta \\ v_3\sin\theta & -v_2\sin\theta & v_1\sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} \cos\theta & v_1\sin\theta & v_2\sin\theta & v_3\sin\theta \\ -v_1\sin\theta & \cos\theta & v_3\sin\theta & -v_2\sin\theta \\ -v_2\sin\theta & -v_3\sin\theta & \cos\theta & v_1\sin\theta \\ -v_3\sin\theta & v_2\sin\theta & -v_1\sin\theta & \cos\theta \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = I$$

This allows us to conclude that Q is an orthogonal matrix. A fundamental property of an orthogonal matrix is conservation of dot product, and hence the angle between two vectors. The preservation of angles and determinant of +1(which ensures no scaling or changes in orientation) allow us to conclude that Q acts as a rotation in $\mathbb{R}^4$.

One might imagine since unit quaternions can perform rotations in $\mathbb{R}^4$, then correctly chosen quaternions must also perform rotations in $\mathbb{R}^3$ which is simply a subspace of $\mathbb{R}^4$. This, unfortunately, is not the case as unit quaternions do not encode every possible rotation in $\mathbb{R}^4$. As Aurélie Richard et al. discussed, a general rotation in $\mathbb{R}^4$ produces two distinct rotations on(at least) two orthogonal planes that only intersect at one point, usually the origin. Two planes, A and B are said to be orthogonal if for every $v \in A$ and $w \in B$, $v \cdot w = 0$ (Weisstein "Orthogonal Subspaces"). These orthogonal planes are called the invariant planes since vectors that start on this plane remain on this plane after the rotation. A simple example of such a rotation would be:

$$\begin{bmatrix} \cos a & -\sin\alpha & 0 & 0 \\ \sin\alpha & \cos\alpha & 0 & 0 \\ 0 & 0 & \cos\beta & -\sin\beta \\ 0 & 0 & \sin\beta & \cos\beta \end{bmatrix}$$

This matrix corresponds to a rotation by α in the $1-i$ plane and by β in the $j-k$ plane(similar methods as before can be used to confirm that the matrix is orthogonal and has a determinant of +1). Quaternions allow us to encode the specific case where the magnitude of the two angles is the same, which are called isoclinic rotations. If the angles are of the same sign, the rotation is called left-isoclinic, while rotations with oppositely signed angles are called right-isoclinic. We will see that left and right multiplication correspond to left and right-isoclinic rotations respectively

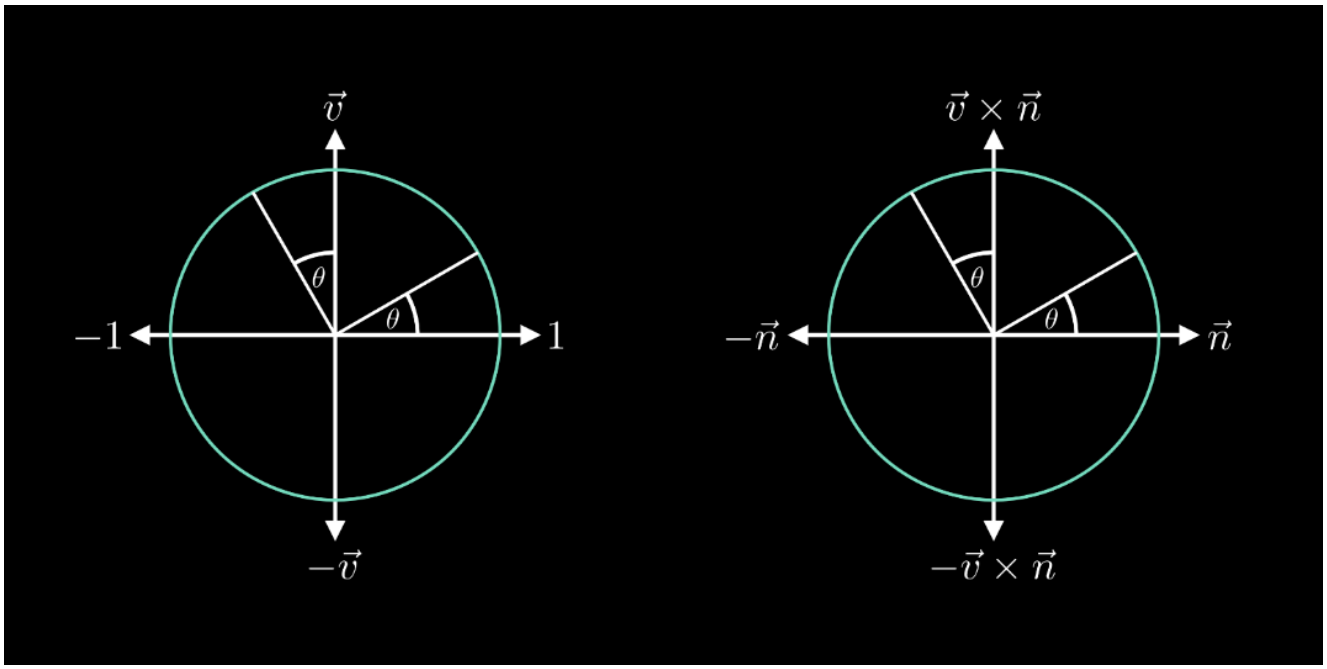

Figure 3:1 Left-isoclinic rotations in 2 orthogonal planes

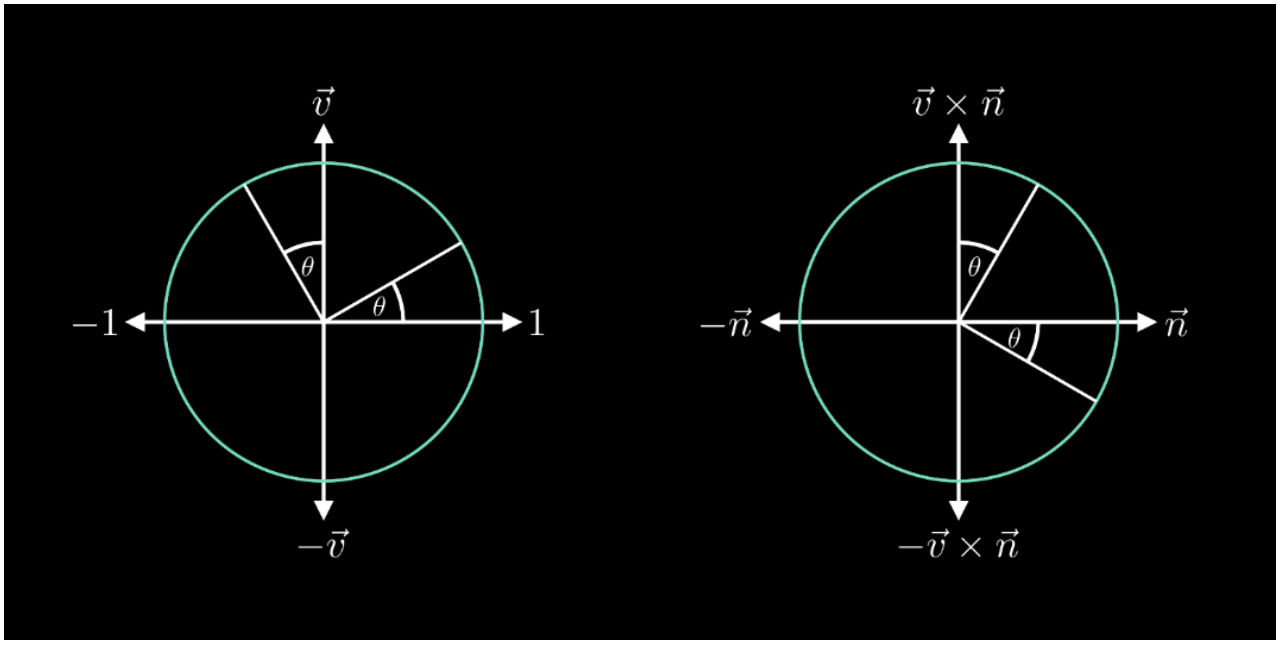

Figure 3:2 Right-Isoclinic rotations in 2 orthogonal planes

In the case of quaternions, the invariant planes are relatively easy to find and have already been shown above.

Let $q = (\cos\theta, \sin\theta\ \vec{v}), v = (0, \vec{v})$

$$qv = (\cos\theta, \sin\theta\ \vec{v})(0, \vec{v})$$

$$= (-\sin\theta(\vec{v}\cdot\vec{v}), \cos\theta\vec{v} + \sin\theta(\vec{v}\times\vec{v}))$$

$$= (-\sin\theta, \cos\theta\vec{v})$$

$$= -\sin\theta + \cos\theta\vec{v}$$

Hence, we can see with left multiplication, the $1-\vec{v}$ plane is invariant. To find the other invariant plane we can use the definition of orthogonal planes. Let us view how this transformation acts on a vector orthogonal to $\vec{v}$.

Let $\vec{n}$ be a vector such that $\vec{v}\cdot\vec{n} = 0$

$$qn = (\cos\theta, \sin\theta\vec{v})(0, \vec{n})$$

$$= (-\sin\theta(\vec{v}\cdot\vec{n}), \cos\theta\vec{n} + \sin\theta(\vec{v}\times\vec{n}))$$

$$= \cos\theta\vec{n} + \sin\theta(\vec{v}\times\vec{n})$$

$\vec{n}$ and $\vec{v}\times\vec{n}$ are both orthogonal to $\vec{v}$ and 1, providing us with the second invariant plane. As additional assurance, we can confirm that $\vec{v}\times\vec{n}$ remains on this plane as well.

$$(\cos\theta, \sin\theta\vec{v})(0, \vec{v}\times\vec{n})$$

$$= (-\sin\theta\vec{v}\cdot(\vec{v}\times\vec{n}), \cos\theta(\vec{v}\times\vec{n}) + \sin\theta\vec{v}\times(\vec{v}\times\vec{n}))$$

$$= (0, \cos\theta(\vec{v}\times\vec{n}) + \sin\theta(\vec{v}(\vec{v}\cdot\vec{n}) - \vec{n}(\vec{v}\cdot\vec{v})))$$

$$= -\sin\theta\vec{n} + \cos\theta(\vec{v}\times\vec{n})$$

A visualisation of these rotations can be seen in Figure 3.1 above. Note how the rotation in the $\vec{n} - \vec{v} \times \vec{n}$ plane is precisely what we want in order to perform an axis-angle rotation, a transformation that performs a rotation by θ in the plane orthogonal to some given axis $\vec{v}$. Unfortunately, this only occurs in the case where the vector to be rotated is orthogonal to $\vec{v}$, which will of course, not be true for some general vector in $\mathbb{R}^3$. The component parallel to $\vec{v}$ will be rotated 'into the fourth dimension' towards the real numbers. In this case, the idea of left and right isoclinic rotations becomes useful. Above we performed left-isoclinic rotations using quaternions. Let us now see how we can perform right-isoclinic rotations.

Since quaternion multiplication is non-commutative, it is natural to wonder how reversing the order of multiplication affects the transformation.

$$1\left(\cos\theta, \sin\theta\vec{v}\right)$$

$$= \left(\cos\theta, \sin\theta\vec{v}\right)$$

$$\left(0, \vec{v}\right)\left(\cos\theta, \sin\theta\vec{v}\right)$$

$$= \left(-\sin\theta\left(\vec{v}\cdot\vec{v}\right), \cos\theta\vec{v} + \sin\theta\left(\vec{v}\times\vec{v}\right)\right)$$

$$= -\sin\theta + \cos\theta\vec{v}$$

The rotation in the $1 - \vec{v}$ plane stay the same. Let us now consider the other plane.

$$\left(0, \vec{n}\right)\left(\cos\theta, \sin\theta\vec{v}\right) = \cos\theta\vec{n} - \sin\theta\left(\vec{v}\times\vec{n}\right)$$

$$= \cos\left(-\theta\right)\vec{n} + \sin\left(-\theta\right)\left(\vec{v}\times\vec{n}\right)$$

$$\left(0, \vec{v}\times\vec{n}\right)\left(\cos\theta, \sin\theta\vec{v}\right) = \cos\theta\left(\vec{v}\times\vec{n}\right) + \sin\theta\vec{n}$$

$$= -\sin(-\theta)\vec{n} + \cos(-\theta)(\vec{v} \times \vec{n})$$

The direction of rotation has been reversed in the $\vec{n} - \vec{v} \times \vec{n}$ plane, hence we can conclude that right multiplication by a unit quaternion corresponds to a right-isoclinic rotation. It now becomes clear that to perform the desired rotation, we require a right-isoclinic rotation by $-\theta$ to restore the component parallel to $\vec{v}$. This quaternion must be:

$$q' = (\cos(-\theta), \sin(-\theta)\vec{v})$$

$$= (\cos\theta, -\sin\theta\vec{v})$$

$$= q^{-1}$$

Hence to rotate about some arbitrary axis $\vec{v}$ in $\mathbb{R}^3$, we first left multiply by q and then right multiply by q^{-1} in order to restore the component parallel to $\vec{v}$. A natural consequence of the right multiplication is that the plane where we wish the rotation, $\vec{n} - \vec{v} \times \vec{n}$ plane, is further rotated by θ, resulting in an overall rotation of 2θ. Therefore, to rotate a vector, $\vec{r}$, θ about $\vec{v}$, the operation is:

$$r' = \left(\cos\frac{\theta}{2}, \sin\frac{\theta}{2}\vec{v}\right)(0, \vec{r})\left(\cos\frac{\theta}{2}, -\sin\frac{\theta}{2}\vec{v}\right) = qrq^{-1}$$

3.2 Quaternions versus Euler angles/Matrices

One might question the use quaternions to perform these rotations, when other methods such as Euler angles and matrices are available. In fact, Euler angles, which tell us the amount 3 perpendicular axes have rotated, only require storing three numbers, while quaternions require four, making them more efficient in terms of

storage. Furthermore, the matrix multiplication to perform this rotation can be done quite quickly by computers as many have optimised matrix multiplication to a large extent(Dam et al. 31).

One issue with using Euler angles is their dependence on the hierarchy of axes. Essentially, it is important to keep track of the order in which the individual axes are being rotated, since rotations in 3D are non-commutative. There is little agreement between industries(and even within industries) about this order(Rotenberg 4).

The biggest issue with Euler angles, however, is gimbal lock, which occurs when two of axes of rotation line up with one another causing a loss of a degree of freedom(Strickland).

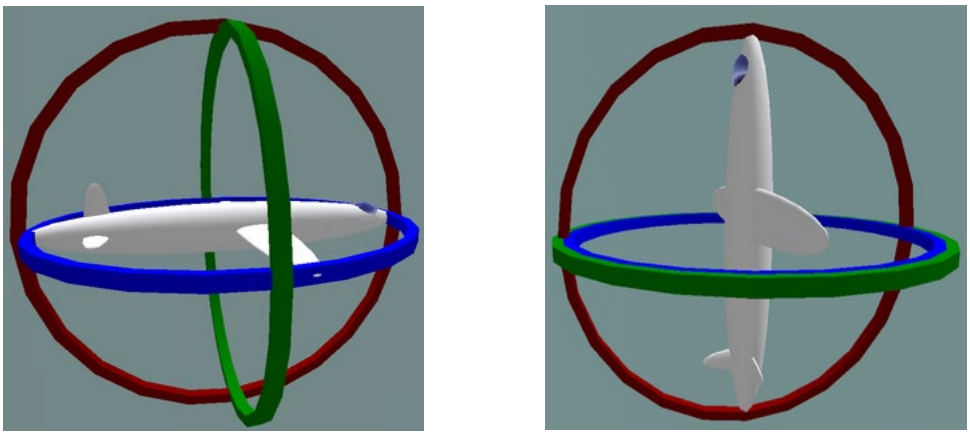

Figure 3:1 Illustration of Gimbal lock(Mathspoetry)

The different colours represent rotations about different axes. As we can see in the second picture, the green and blue circles have lined up therefore the plane has lost the ability to rotate about what is commonly referred to as the ‘yaw’ axis, signified by the blue circle. This means that we would not be able orient the plane towards us from the second position. This will always be a problem with Euler angles.

As Brooks et al. pointed out in their book *Chariots for Apollo : The NASA History of Manned Lunar Spacecraft to 1969*, a terrifying example of gimbal lock occurred during the Apollo 10 mission. As the crew were returning to Earth in their lunar module, Snoopy, experienced gimbal lock and “seemed to be throwing a fit, lurching wildly about”(310). Luckily the pilots onboard were able to Snoopy back under control without any serious damage. Modern rockets tend to use quaternions; especially as greater research is done into Reusable Launch Vehicles(RLVs)(Ambar et al.).

This is not to say that Euler angles are useless. They can be incredibly useful for simple rotations, because of their temporal and spatial efficiency. The aircraft industry still uses Euler angles to talk about and perform rotations(“Axis of rotation”). Since planes only tend to move along one axis at a time, this means that interpolation is not as big of an issue. It also provides more intuitive control for the pilots which is likely one of their greater concerns. Additionally, as mentioned by Rotenberg, having a solid ground allows other vehicles to be represented intuitively using Euler angles(5).

For more intricate tasks, Euler angles unfortunately do not suffice. Instead, one could use a matrix . A rotation matrix, R , to represent axis-angle rotations would be:

$$R = \begin{bmatrix} \cos\theta + u_x^2(1-\cos\theta) & u_x u_y(1-\cos\theta) - u_z\sin\theta & u_x u_z(1-\cos\theta) + u_y\sin\theta \\ u_y u_x(1-\cos\theta) + u_z\sin\theta & \cos\theta + u_y^2(1-\cos\theta) & u_y u_z(1-\cos\theta) - u_x\sin\theta \\ u_z u_x(1-\cos\theta) - u_y\sin\theta & u_z u_y(1-\cos\theta) + u_x\sin\theta & \cos\theta + u_z^2(1-\cos\theta) \end{bmatrix}$$

Where θ is the angle of rotation and $\vec{u}$ is the axis of rotation, $\vec{u} = \begin{bmatrix} u_x \\ u_y \\ u_z \end{bmatrix}$ (Cole 257)

Matrices are also incredibly versatile since nearly every transformation, scaling, reflection, rotation, can be encoded within them which makes concatenating transformations simpler. This also means that programmers and developers don’t

have to work with different mathematical objects, allowing for greater compatibility. Furthermore, most programmers are likely going to be quite familiar with matrices, which means that they will be more comfortable using them and will find it easier to find mistakes and bugs in their code. Although this may seem trivial, there is an undeniable cost to learning something new. Being familiar with the framework could help save a lot of time and effort.

On the other hand, matrices are less efficient in terms of storage, since they require storing nine numbers, more than double required for quaternions. This trade-off between temporal and spatial efficiency is a common one in the world of computers. The trait to sacrifice depends entirely on the use case, as some applications may value time over space.

One of the bigger issues with working with matrices is orthogonalization. Rounding and floating-point errors are inevitable in computation. Successive rotations with matrices will cause errors to accumulate and lead to deviations from intended results. This can be solved by orthogonalizing the resultant matrix. A popular algorithm for this is Singular Value Decomposition(SVD) which can be computationally expensive(Vasudevan).

By contrast, quaternions are far easier to normalise as it simply requires dividing the quaternion by its norm, which itself requires taking the square root of the dot product of the vector with itself. Taking a brief side-step into complexity theory, allows us to make quantitative comparisons. Since finding the dot product of a vector involves n multiplications and n additions, it can be said to take linear time, or $O(n)$. On the other hand, SVD decomposition for an $n \times n$ matrix takes a time complexity of $O(n^3)$ (Vasudevan). This means matrix orthogonalization will take approximately 6.75 times

more operations and hence take about 6.75 times longer to compute. In fact, normalisation can be made even more efficient if done regularly as it allows use of approximations (e.g. Taylor series or Padé approximants) that remove the need to compute a square root altogether. One of the biggest advantages of quaternions, however, is interpolation.

4. Interpolation

Interpolation is the process by which values between two known values of a function can be found(Weisstein, “Interpolation”). This is very useful in this case as it allows us to find the positions of an object between any two given orientations, making animation possible, for example. Quaternions allow easy implementation of Spherical Linear Interpolation(SLERP), which allows us to find the parametric formula for the unit-radius arc between any two unit vectors(Shoemake, “Animating Rotation with Quaternion Curves” 248). This is a fairly popular rotation interpolation method.

The formula to perform SLERP using quaternions is (Shoemake, “Animating Rotation with Quaternion Curves” 248):

$$q = \left(q_1 q_0^{-1}\right)^t q_0$$

q_0 and q_1 represent the initial and final positions as unit vector quaternions while t is some parameter between 0 and 1. Before proving this result, we need to define exponentiation with respect to quaternions. For this, we can use the following definition of exponents:

$$e^a = 1 + a + \frac{a^2}{2!} + \frac{a^3}{3!} + \cdots + \frac{a^n}{n!} + \cdots$$

Euler's formula can then be extended to find that any unit quaternion can be written like so:

$$e^{\vec{v}\theta} = \cos\theta + \sin\theta\ \vec{v} = \left(\cos\theta, \sin\theta\ \vec{v}\right), \theta \in \left[0, 2\pi\right], \vec{v} \in \mathbb{R}^3, \left\|\vec{v}\right\| = 1$$

Here is a short proof of the above result

$$\left(\vec{v}\right)^2 = \left(0, \vec{v}\right)\left(0, \vec{v}\right)$$

$$= \left(-\vec{v}\cdot\vec{v}, \vec{v}\times\vec{v}\right)$$

$$= \left(-\left\|\vec{v}\right\|^2, 0\right)$$

$$= -1$$

Therefore,

$$e^{\vec{v}\theta} = 1 + \frac{\vec{v}\theta}{1!} + \frac{\left(\vec{v}\theta\right)^2}{2!} + \frac{\left(\vec{v}\theta\right)^3}{3!} + \frac{\left(\vec{v}\theta\right)^4}{4!} + \frac{\left(\vec{v}\theta\right)^5}{5!} + \cdots$$

$$= 1 + \vec{v}\theta - \frac{\theta^2}{2!} - \frac{\vec{v}\theta^3}{3!} + \frac{\theta^4}{4!} + \frac{\vec{v}\theta^5}{5!}\cdots$$

Since the series is absolutely convergent, the terms can be rearranged to:

$$e^{\vec{v}\theta} = \left(1 - \frac{\theta^2}{2!} + \frac{\theta^4}{4!} - \cdots\right) + \left(\vec{v} - \frac{\vec{v}\theta^3}{3!} + \frac{\vec{v}\theta^5}{5!} - \cdots\right)$$

$$= \left(1 - \frac{\theta^2}{2!} + \frac{\theta^4}{4!} - \cdots\right) + \vec{v}\left(1 - \frac{\theta^3}{3!} + \frac{\theta^5}{5!} - \cdots\right)$$

$$= \cos\theta + \vec{v}\sin\theta$$

The final step is achieved by recognising the sums as the Maclaurin series for $\cos\theta$ and $\sin\theta$. What this means is that for any unit quaternion q,

$$q^t = \left(e^{v\theta}\right)^t$$

$$= e^{v(\theta t)}$$

$$= \cos\left(\theta t\right) + \sin\left(\theta t\right)\vec{v}$$

This is analogous to exponentiation with complex numbers.

Let's say we wish to interpolate between unit vectors q_0 and q_1 in $\mathbb{R}^3$. This will correspond to finding the shortest path between them on the unit sphere, which translates to finding the geodesic between q_0 and q_1. This geodesic must necessarily lie on the plane spanned out by these two vectors allowing us to draw a diagram like so:

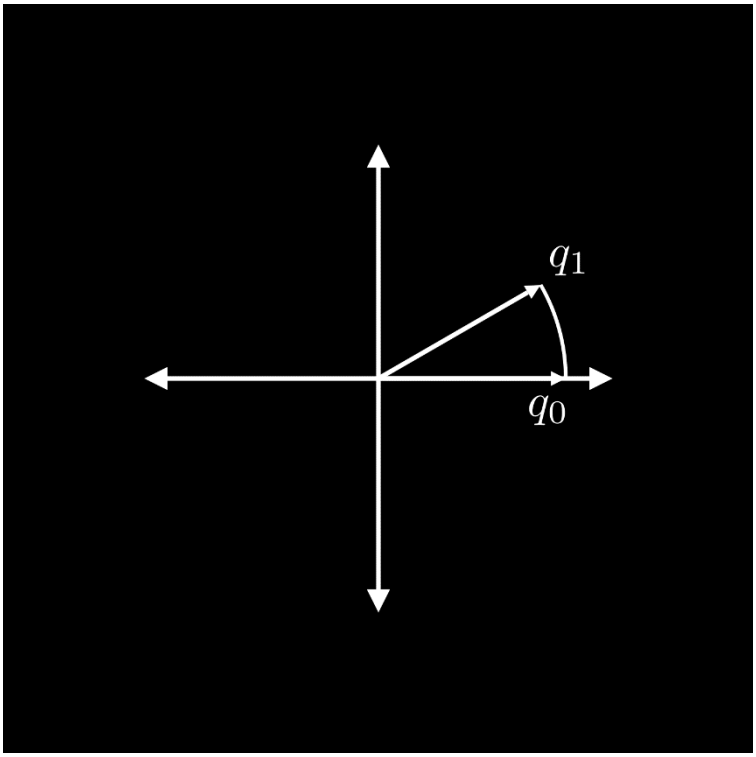

Figure 4:1 Plane spanned out by q0 and q1

Our goal is to find a parametric equation to describe the arc between q_0 and q_1 in terms of q_0 and q_1. If we had some properly oriented $\vec{n}$ that was orthogonal to both q_0 and q_1 then we could perform a simple multiplication and achieve a rotation in the $q_0 - q_1$ plane, as desired.

Let $r = \left(\cos\theta, \sin\theta\vec{n}\right)$

$$rq_0 = \left(\cos\theta, \sin\theta\vec{n}\right)\left(0, \vec{q}_0\right)$$

$$= \cos\theta\vec{q}_0 + \sin\theta\left(\vec{n} \times \vec{q}_0\right)$$

As we allow θ to vary from 0 to 2π, the resulting quaternion traces out a circle like so:

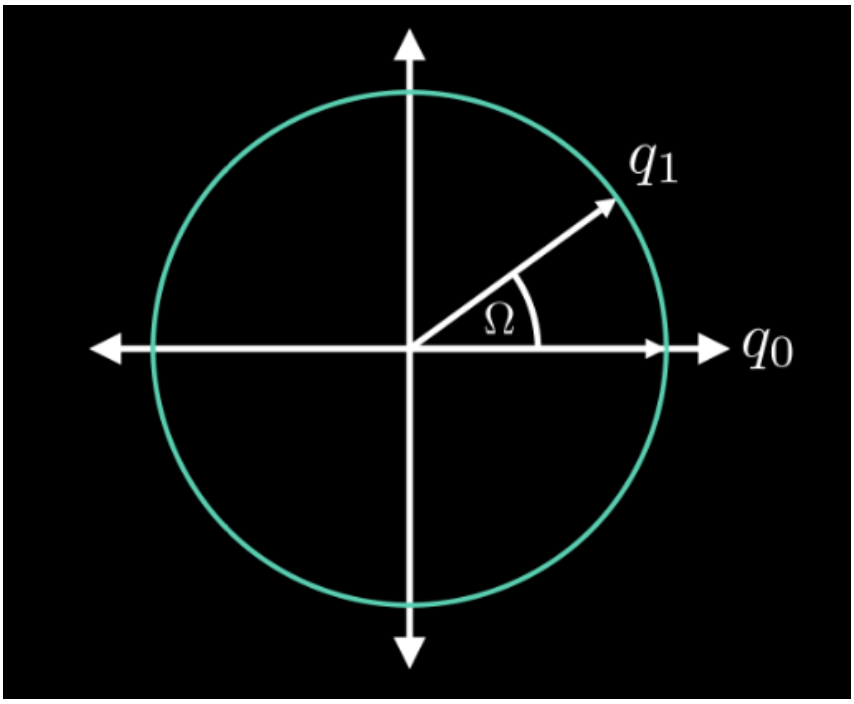

Figure 4:2 Circle spanned out by q0 and n x q0

This means that if we set the value of r as $\left(\cos\Omega, \sin\Omega\vec{n}\right)$ where $\Omega = \cos^{-1}\left(\vec{q}_0 \cdot \vec{q}_1\right)$, then we can write an equation for the arc as:

$$\cos t\Omega\vec{q}_0 + \sin t\Omega\left(\vec{n} \times \vec{q}_0\right) = r^t q_0, \; 0 \le t \le 1$$

Looking at the formula for r, we can see that it is incredibly similar to the product of the quaternions q_0 and q_1.

$$q_0 q_1 = \left(-q_0 \cdot q_1, q_0 \times q_1\right)$$

$$= \left(-\cos\Omega, \sin\Omega\vec{n}\right)$$

$\vec{n}$ even has the correct orientation relative to q_0 and q_1. The only difference is the sign of the real component. This is easily fixed by changing the sign of q_0, which is the same as taking its inverse, as it is a unit vector quaternion. This changes the formula for r to $q_0^{-1} q_1$. However, because of the negative sign the direction of the cross-product vector is reversed. To correct the direction of the cross-product vector we can reverse the order of multiplication allowing us to construct the following function to perform spherical interpolation entirely out of quaternions.

$$Slerp\left(q_0, q_1, t\right) = \left(q_1 q_0^{-1}\right)^t q_0$$

As Dam et al.(43) showed in their paper, this can equivalently be written as:

$$Slerp\left(q_0, q_1, t\right) = q_1 \left(q_1^{-1} q_0\right)^{1-t}$$

$$= \left(q_0 q_1^{-1}\right)^{1-t} q_1$$

$$= q_0 \left(q_0^{-1} q_1\right)^t$$

Because the parameter t directly controls the angle between the resultant vector and q_0, SLERP allows the rotation to maintain a constant angular velocity. This makes the rotation both intuitive and aesthetically pleasing. This kind of interpolation is nearly impossible to achieve with Euler angles alone. The obvious method of interpolating

the angles independently in the three axes can produce highly erratic results(Shoemake, “Animating Rotation with Quaternion Curves” 247). Direct linear interpolation using matrices may cause the image to collapse into a line or even a point(Shoemake, “Matrix Animation and Polar Decomposition” 260). To properly interpolate using matrices would involve extracting the axis vector and such which is very expensive computationally. Furthermore, there is very little guarantee that either of these methods would produce a rotation with constant angular velocity or span a geodesic.

Admittedly, there are some issues with SLERP. Interpolating between successive orientations can cause sharp turns in the rotations. A smoother version of SLERP was suggested by Dam et al. called Squad, which is comparable to how Bezier curves make linear interpolation smoother.

One major disadvantage of SLERP is its computational slowness. As Jonathan Blow wrote in his article, “if you’re calling SLERP in the first place, you are already in a world of slowness”. Nevertheless, even the alternatives he provided, such as quaternion normalised LERP(linear interpolation) and log-quaternion LERP, involve using quaternions. Although SLERP itself may not be the optimal function, it seems that quaternions as a whole certainly provide a very nice framework for interpolation.

5. Conclusion

In conclusion, quaternions provide a natural way of encoding 4D rotations through their multiplication. Careful manipulation of this multiplication allows us to perform rotations in 3D. Expanding known definitions of exponentiation also allowed us to find a method to animate any given rotation known as SLERP. This simple method can also be refined to produce more pleasing animations.

Quaternions help avoid serious issues such as gimbal lock, while also taking comparatively little storage space. However, the choice of whether to use quaternions or other methods depends entirely on the use case. For aircraft, especially manned, the intuition provided Euler angles proves to be of greater importance. For other tasks, like space travel, quaternions are more effective. Additionally, although quaternions may be spatially efficient, they can take longer to compute than matrices. This sacrifice of computational time for space may be undesirable in certain circumstances such as computer games, where you want rotations and transformations to occur as fast as possible, without incurring any time lag for the player. And while matrices may be quicker to compute, they take longer to orthogonalise, making them undesirable when many consecutive rotations are to take place. A summary of the comparisons has been provided below.

	Advantages	**Disadvantages**
Euler angles	- Only require three numbers - Simple to use and understand - Computationally fast	- Gimbal lock - Interpolation is difficult
Matrices(axis-angle)	- Axis-angle representation more intuitive and useful - Matrices are versatile - Computationally fast - Familiar to developers/coders	- Require nine numbers - Computationally expensive to orthogonalise
Quaternions	- Only require four numbers - Easy normalisation - Easy interpolation - SLERP maintains constant angular velocity	- Foreign to developers/coders - SLERP computationally slow

We would be doing ourselves a disservice if we thought that quaternions were only limited to computing rotations. They can be found everywhere in physics. Maxwell's equations of electromagnetism were originally written using quaternions(Jack). Research into physical space it itself has suggested that it has a quaternion structure(Jack). This illustrates the surprising usefulness of mathematical structures in describing the universe. With the emergence of octonions and sedenions, which are further extensions of complex numbers into 8 and 16 dimensions, one can only imagine their potential uses and the wonderful mathematics that awaits within them.

6. Works Cited

@andy_matuschak. “I’ve tried five different times to understand quaternions; I’ve shipped production code to hundreds of millions of devices involving them 😈, and I *still* don’t get them”. *Twitter*, 6 September 2018, twitter.com/andy_matuschak/status/1037722339809452032

Ambar, Radzi et al. (2014). *Quaternion-based adaptive attitude control for a winged rocket*. MOVIC 2014 - 12th International Conference on Motion and Vibration Control.

“Axis of Rotation.” SKYbrary Aviation Safety, *SKYbrary*, 27 July 2007, www.skybrary.aero/index.php/Axis_of_Rotation.

Blow, Jonathan. “Understanding Slerp, Then Not Using It.” *Game Developer Magazine*, 2004.

Brooks, Courtney G., et al. *Chariots for Apollo: the NASA History of Manned Lunar Spacecraft to 1969*. National Aeronautics and Space Administration, 1979. history.nasa.gov/SP-4205.pdf.

Byrne, Eimear. *The Ring of Real Quaternions*. UCD School of Mathematics and Statistics. 2005, maths.ucd.ie/courses/mst2013/ch5.pdf.

Cole, I. R. “Rotation About an Arbitrary Axis”. *Modelling CPV*. 1, figshare, 12 Aug. 2019, hdl.handle.net/2134/18050.

Crowe, Michael J. *A History of Vector Analysis: The Evolution of the Idea of a Vectorial System*. Courier Corporation, 1994.

Dam, Erik B, et al. *Quaternions, Interpolation and Animation*. University of Copenhagen, 1998, pp. 1–95.

Graves, Robert P. *Life of Sir William Rowan Hamilton. Vol. 1*, New York Arno Pr, 1882, pp. 434–435.

Hamilton, William Rowan. *Lectures on Quaternions*. Hodges and Smith, 1853.

Jack, Peter Michael. “Physical Space as a Quaternion Structure, I: Maxwell Equations. A Brief Note.” *ArXiv*, 2003, arxiv.org/abs/math-ph/0307038.

Mathspoetry. “Gimbal Lock.” *Wikimedia*, 14 Feb. 2009, commons.wikimedia.org/wiki/File:No_gimbal_lock.png.

Maxwell, James Clerk. *A Treatise on Electricity and Magnetism*. Vol. 1, Clarendon Press, 1873.

Richard, Aurélie, et al. “Decomposition of ND-Rotations: Classification, Properties and Algorithm.” *Graphical Models*, vol. 73, no. 6, 2011, pp. 346–353., doi:10.1016/j.gmod.2011.06.004.

Rotenburg, Steve. “Orientation and Quaternions” University of Washington Computer Science and Engineering, 20 Dec. 2017, courses.cs.washington.edu/courses/cse474/18wi/pdfs/lectures/Orientation-Quaternions.pdf.

Serge, Belongie. “Rodrigues' Rotation Formula.” *Wolfram MathWorld*, *Wolfram*, mathworld.wolfram.com/RodriguesRotationFormula.html. Accessed 30 May 2019.

Shoemake, Ken. “Animating Rotation with Quaternion Curves.” *ACM SIGGRAPH Computer Graphics*, vol. 19, no. 3, 1985, pp. 245–254., doi:10.1145/325165.325242.

Shoemake, Ken, and Tom Duff. “Matrix Animation and Polar Decomposition.” *Graphics Interface Proceedings 1992*, Canadian Information Processing Society, 1992, pp. 258–264.

Strickland, Jonathan. “What Is a Gimbal -- and What Does It Have to Do with NASA?” HowStuffWorks Science, *HowStuffWorks*, 28 June 2018, science.howstuffworks.com/gimbal1.htm.

Vasudevan, Vinita and M. Ramakrishna. “A Hierarchical Singular Value Decomposition Algorithm for Low Rank Matrices.” *ArXiv* abs/1710.02812 (2017)

Weisstein, Eric W. "Interpolation." From *MathWorld*--A Wolfram Web Resource. mathworld.wolfram.com/Interpolation.html

Weisstein, Eric W. "Orthogonal Subspaces." From *MathWorld*--A Wolfram Web Resource. mathworld.wolfram.com/OrthogonalSubspaces.html