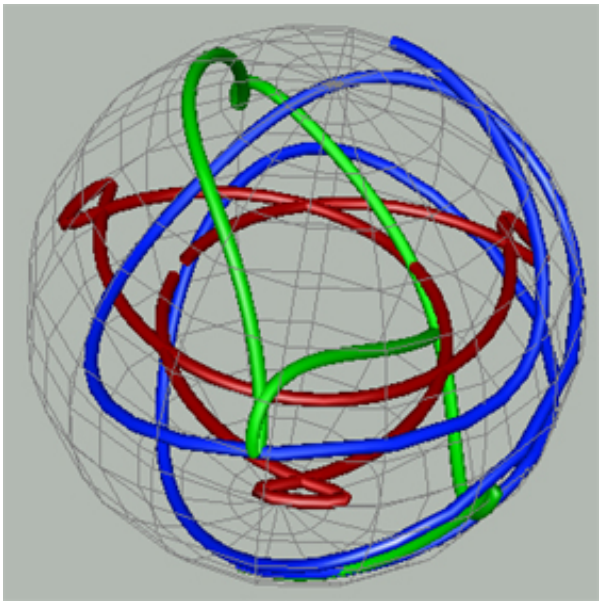
Rotations in 3 Dimensions with Quaternions

Sean Turner

April 8, 2017



Contents

List of Figures

Abstract

William Rowan Hamilton first described quaternions in 1843. Quaternions are used to describe transformation in 3D space and have many applications in aeronautics, robotics, and computer graphics. This manuscript will provide a brief overview of quaternions and spatial geometries, specifically relating to algebra, geometry, and differential calculus. This will be followed by a comparision between quaternions, euler angles, and rotational matrices, and then a discussion of their applications.

1 Introduction

Nobody knows how long vectors have been used in mathematics. Some speculate that the parallelogram method for addition of vectors was lost in a work of Aristotle. However quaternions, often defined as the quotient of two vectors, were not described until 1843 by William Rowan Hamilton. Quaternions are now extensively used in aeronautics and computer graphics for their advantages over tradional transformtion methods.

1.1 History of Quaternions

On October 17, 1843, William Rowan Hamilton wrote a letter to his friend John Graves, Esq. on the subject of "a very curious train of mathematical speculation." The letter details his "theory of quaternions", and follows his mathematical reasoning behind the development of his "quaternions." The day before, while walking across the Royal Canal in Dublin, Hamilton had the idea for the formula of quaternions as shown below in Equation 1. [5]

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

This equations and its implications will be investigated in Section 2. The letter was only a few pages long, but it quite thorough in its' scope, and was eventually published in the *London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science* the following year in 1844. Shortly after Hamilton's death in 1865, his son edited and published the longest of Hamilton's books, at over 800 pages. Titled *Elements of Quaternions*, this book was the go-to book on quaternions for several decades. The wake of Hamilton's exploration into quaternions led to research associations like the Quaternion Society who described themselves as an "International Association for Promoting the Study of Quaternions."

1.2 Basic Geometric Transformations

The primary topic of this manuscript is the application of quaternions to rotations in 3D space, so some terms will be defined here. There are two methods of transformation: translation and rotation [1].

Definition 1.2.1 (Translation)**.** A *translation* is a point in space moved from one position to another. Let a point $P \in \mathbb{R}^3$ be denoted as $(x, y, z), x, y, x \in \mathbb{R}$ and the translation by a vector $(\Delta x, \Delta y, \Delta z)$. Then the new position P' is $(x + \Delta x, y + \Delta y, z + \Delta z)$. There is only one translation vector that takes P to P'.

A *rotation* can be defined in multiple ways. The following definition is given by Euler, and will be used here.

Definition 1.2.2 (Euler's Definition of Rotation)**.** Let $O, O' \in \mathbb{R}^3$ be two orientations. Then there exists an axis $l \in \mathbb{R}^3$ and an angle of rotation $\Theta \in [-\pi, \pi]$ such that O yields O' when rotated Θ about l.

It is important to distinguish between *orientation* and *rotation*. Orientation here is the normal vector to an object in 3D space. A rotation comprised of

an axis and angle of rotation. Unlike the translation between two points, the rotation between two orientations in 3D space is not unique.

Section 2 will define and discuss quaternions in detail. Section 3 will compare quaternions and alternate methods of expressing rotations, while Section 4 will discuss real world applications. Section 5 will conclude the report.

2 Discussion

The quaternion equation was briefly introduced in Equation 1. The following definition is a rigorous definion that follows from that equation.

Definition 2.0.1 (Quaternion)**.** A *quaternion* is a number of the form

$$a + b\mathbf{i} + c\mathbf{j} + c\mathbf{k}$$

where a, b, c, and d are real numbers, **i**, **j**, **k**, are square roots of -1, and **ijk**= -1.

The following section will describe quaternions in more depth, specifically related to algebra, geometry, and differential calculus.

2.1 Algebra & Quaternions

The addition and subtraction of quaternions is the same as 4D vector addition. That is, adding quaternions is simply separately adding the coefficients of **i**, **j**, **k**. [4] For example,

$$(1 + 2\mathbf{i} + 3\mathbf{j} + 4\mathbf{k}) + (-2 + 3\mathbf{i} - 1\mathbf{j} + 4\mathbf{k}) = -1 + 5\mathbf{i} + 2\mathbf{j} + 8\mathbf{k}$$

Multiplication is a little more involved. Clearly, since **i**, **j**, and **k** are square roots of -1, it is true that $\mathbf{i}^2 = \mathbf{j}^2 = \mathbf{k}^2 = -1$. However, it is not so clear what, for example, **ij** is. We know that $\mathbf{ij} \neq -1$ because $\mathbf{ijk} = -1$, and $\mathbf{k} \neq 1$. To find **ij** we must use Equation 1 as shown:

$$\mathbf{ij} = -\mathbf{ij}(-1) = -\mathbf{ijk}^2 = -(\mathbf{ijk})\mathbf{k} = -(-1)\mathbf{k} = \mathbf{k}$$

It is important to note that quaternion multiplication is not commutative, since $\mathbf{ij} = \mathbf{k} \neq \mathbf{ji} = -\mathbf{k}$. A full table of the quaternion relationships between **i**, **j**, and **k** are shown below in Table 1:

Table 1: Quaternion Characteristics

	i	**j**	**k**
i	**-1**	**k**	**-j**
j	**-k**	**-1**	**i**
k	**-j**	**i**	**-1**

Quaternion multiplication is not commutative as shown above, but other algebraic properties are satisfied as shown in Theorem 2.1.1

Theorem 2.1.1. *Properties of Quaternion Multiplication*

1. Associativity: $\mathbf{q}(\mathbf{rs}) = (\mathbf{qr})\mathbf{s}$
2. Distributivity: $\mathbf{q}(\mathbf{r} + \mathbf{s}) = \mathbf{qr} + \mathbf{qs}$
3. Inverses: $\forall$ quaternions $\mathbf{q} \neq 0$, $\exists$ a quaternion $\mathbf{r}$ s.t. $\mathbf{qr} = 1$
4. Cancellation: If $\mathbf{qr} = \mathbf{qs}$, then $\mathbf{r} = \mathbf{s}$

When quaternions are written in the form $\mathbf{q} = a + b\mathbf{i} + c\mathbf{j} + d\mathbf{k}$, they are said to be in *Cartesian form*, similar to the method of displaying a complex number in the form $a + bi$. Just as we can separate a complex number into real and imaginary parts, so we can split a quaternion **q** into a *scalar* part $S\mathbf{q} = a$ and a

vector part $V\mathbf{q} = b\mathbf{i} + c\mathbf{j} + d\mathbf{k}$. We can also define the *conjugate* of a quaternion as

$$\mathbf{q}^* = S\mathbf{q} - V\mathbf{q} = a - b\mathbf{i} - c\mathbf{j} - d\mathbf{k},$$

and the *absolute value* of $\mathbf{q}$ as

$$|\mathbf{q}| = \sqrt{a^2 + b^2 + c^2 + d^2}.$$

Quaternions also appear in abstract algebra as a non-abelian group, which means that $a * b \neq b * a, \forall a, b \in Q$. [2] Figure 3 shows the cycle diagram for the quaternions.

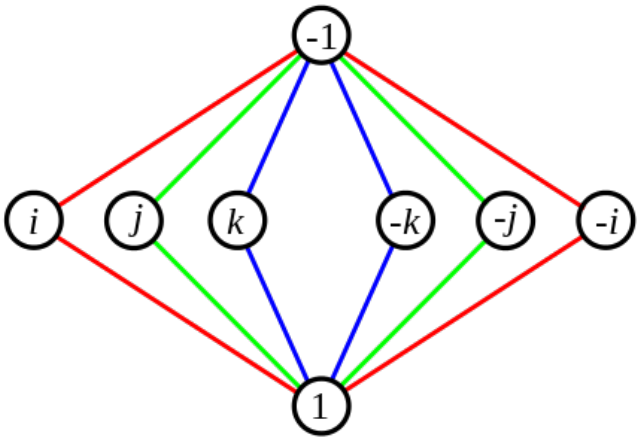


Figure 1: Quaternions Cycle Diagram

Despite being a non-abelian group, every subgroup of the group is normal subgroup. This property is known as being *Hamiltonian*. However, the quaternions as a group are not closed under multiplication.

Theorem 2.1.2. The quaternions are not closed under multiplication.

Proof: Let us assume, by way of contradiction, that the quaternions are closed under multiplication.

This means that $\exists a, b, c \in \mathbb{R}$ such that $\mathbf{ij} = a + \mathbf{i}b + \mathbf{j}c$. If we multiply this with $\mathbf{i}$ we get $\mathbf{j} = b + \mathbf{i}a + \mathbf{ij}c$. Substituting the first equation into the second yields $-\mathbf{j} = -b + \mathbf{i}a + (a + \mathbf{i}b + \mathbf{j}c)$. Therefore $0 = (ac - b) + \mathbf{i}(a + bc) + \mathbf{j}(c^2 + a)$, and we have $ac - b = 0$, $a + bc = 0$, and $c^2 + 1 = 0$. However, we assume c is real, so there is a contradiction. □

2.2 Geometry & Quaternions

As discussed in Section 1.2, the primary methods of transformations that will be dicussed are translation and rotation. When using quaternions for geometry, usually the *Pure Quaternion* is used. [3] A pure quaternion is simply a quaternion in the form of $\mathbf{q} = x\mathbf{i} + y\mathbf{j} + z\mathbf{k}$ because it can be used to represent a point in 3D. We define a *dot product* of two quaternions $\mathbf{q}$ and $\mathbf{r}$ by the following equation.

$$-S(\mathbf{qr}) = x_1x_2 + y_1y_2 + z_1z_2 \tag{2}$$

This is analagous to the traditional dot product $\mathbf{q} \cdot \mathbf{r} = |\mathbf{q}||\mathbf{r}|cos(\phi)$, where ϕ is the angle between the vectors $\mathbf{q}$and $\mathbf{r}$in 3D. Similarly, we can define a cross product bewteen $\mathbf{q}$ and $\mathbf{r}$ as the following:

$$V(\mathbf{qr}) = (y_1z_2 - y_2z_1)\mathbf{i} - (x_1z_2 - x_2z_1)\mathbf{j} - (x_1y_2 - x_2y_1)\mathbf{k} \tag{3}$$

Again, this is the same as the geometric $\mathbf{q} \times \mathbf{r} = |\mathbf{q}||\mathbf{r}|sin(\phi)\mathbf{u}$, where ϕ is the same as before, and $\mathbf{u}$ is a pure unit quaternion and represents a unit vector perpedicular to the plane formed by $\mathbf{q}$ and $\mathbf{r}$. Figure 2 shows the geometric representation of the cross product.

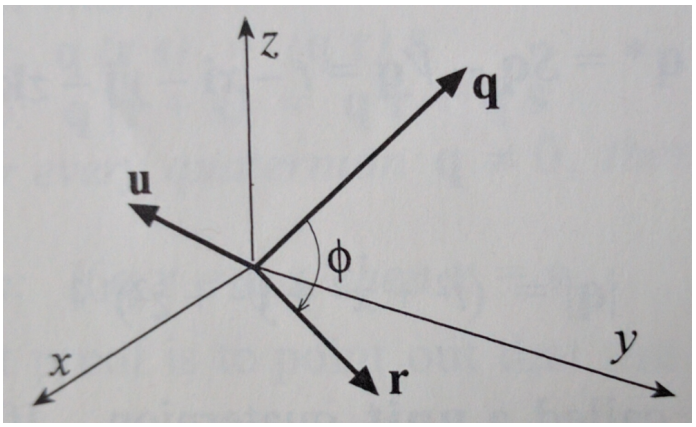


Figure 2: Visualizing $\mathbf{q} \times \mathbf{r}$

By combining the scalar result of the dot product and the vector result of the cross product, we can construct quaternion multiplication:

$$\mathbf{qr} = -(\mathbf{q} \cdot \mathbf{r}) + (\mathbf{q} \times \mathbf{r}) \tag{4}$$

This give us a geometric way of visualizing quaternion multiplication.

There is one more prerequisite before we can do rotations with quaternions. Just like complex numbers can be represented in a polar form $z = |z|(cos(\theta) + isin(\theta))$, we can express quaternions in a similar form of $\mathbf{q} = |\mathbf{q}|(cos(\theta) + \mathbf{u}sin(\theta))$. Here, instead of the imaginary number i, we use the pure unit quaternion $\mathbf{u}$. Therefore, we have the following theorem:

Theorem 2.2.1. Every quaternion can be represented in the form:

$$\mathbf{q} = |\mathbf{q}|(cos(\theta) + \mathbf{u}sin(\theta))$$

With the above theorems and definitions, we are now able to represent rotation in 3D space with quaternions.

Theorem 2.2.2. Let $\mathbf{r}$ be a unit quaternion. Let R be the transformation on the pure quaternion $\mathbf{q}$ defined by

$$\mathrm{R}\mathbf{q} = \mathbf{r}\mathbf{q}\mathbf{r}^*.$$

Then R is a rotation of the three dimensional space of pure quaternions about an axis passing through the origin. If we take $\mathbf{r}$ in the polar form

$$\mathbf{r} = cos(\theta) + \mathbf{u}sin(\theta),$$

where $\mathbf{u}$ is a unit quaternion, then R$\mathbf{q}$ is the pure quaternion obtained by rotating $\mathbf{q}$ about $\mathbf{u}$ by the angle 2θ. Every rotation about an axis passing through the origin can be expressed this way.

Proof: Let us look at R$\mathbf{q}$ for three cases, $\mathbf{q} = \mathbf{u}$, $\mathbf{q}$ is perpendicular to $\mathbf{u}$, and $\mathbf{q}$ is any pure quaternion.

Case 1: $\mathbf{q} = \mathbf{u}$
Then $\mathrm{R}\mathbf{u} = \mathbf{r}\mathbf{u}\mathbf{r}^* = (cos(\theta) + \mathbf{u}sin(\theta))\mathbf{u}(cos(\theta) - \mathbf{u}sin(\theta))$. Expanding this yields

$$\mathrm{R}\mathbf{u} = cos(\theta)^2\mathbf{u} - sin(\theta)^2\mathbf{u}^3 = cos(\theta)^2\mathbf{u} - sin(\theta)^2(-\mathbf{u}) = \mathbf{u}.$$

Thus $\mathbf{u}$ is a fixed point of R. This is expected because R$\mathbf{u}$ is the rotation of $\mathbf{u}$ about the axis $\mathbf{u}$.

Case 2: $\mathbf{q}$ is perpendicular to $\mathbf{u}$.
Then $\mathrm{R}\mathbf{u} = \mathbf{r}\mathbf{q}\mathbf{r}^* = (cos(\theta) + \mathbf{u}sin(\theta))\mathbf{q}(cos(\theta) - \mathbf{u}sin(\theta))$

$= cos(\theta)^2\mathbf{q} + \mathbf{uq}cos(\theta)sin(\theta) - \mathbf{qu}cos(\theta)sin(\theta) - \mathbf{uqu}sin(\theta)^2$. Using Equation 4, we get

$$\mathbf{uq} = -(\mathbf{u} \cdot \mathbf{q}) + (\mathbf{u} \times \mathbf{q}) = (\mathbf{u} \times \mathbf{q}),$$

because $\mathbf{u}$ and $\mathbf{q}$ are perpendicular. Similarly,

$$\mathbf{qu} = (\mathbf{q} \times \mathbf{u}) = -(\mathbf{u} \times \mathbf{q})$$

so

$$\mathbf{uqu} = (\mathbf{u} \cdot (\mathbf{u} \times \mathbf{q})) - (\mathbf{u} \times (\mathbf{u} \times \mathbf{q})) = -(\mathbf{u} \times (\mathbf{u} \times \mathbf{q})).$$

Therefore $\mathbf{uqu} = \mathbf{q}$. When we put this all together we get

$$\mathrm{R}\mathbf{q} = cos(\theta)^2\mathbf{q} + 2cos(\theta)sin(\theta)(\mathbf{u} \times \mathbf{q}) - sin(\theta)^2\mathbf{q} = cos(2\theta)\mathbf{q} + sin(2\theta)(\mathbf{u} \times \mathbf{q}).$$

Therefore, R$\mathbf{q}$ is the rotation of $\mathbf{q}$ through an angle of 2θ about the axis of $\mathbf{u}$.

Case 3: $\mathbf{q}$ is any pure quaternion. First note that R is a linear transformation, that is, $R(\mathbf{q} + \mathbf{r}) = R(\mathbf{q}) + R(\mathbf{r})$ and $R(\alpha\mathbf{q}) = \alpha R(\mathbf{q})$. Secondly notice that the rotation of 3D space is also a linear transformation. To prove that two linear transformations are equal, it is enough to prove that they have the same effect on the vectors from some basis of 3D space. We have shown that the rotation R and the rotation about the axis $\mathbf{u}$ by the angle 2θ are equal when $\mathbf{u} = \mathbf{q}$ and $\mathbf{u}$ is perpendicular to $\mathbf{u}$. However, a basis for 3D space can be found consisting of the vector $\mathbf{u}$ plus two orthogonal vectors. Therefore, R acts like a rotation for all 3D vectors. □

3 Comparison

This section will investigate alternative methods for rotating objects in 3D, detail the conversion between them, and compare their advantages and disadvantages to quaternions.

3.1 Other Non-Euclidean Transformation Methods

There are two alternate primary methods for expressing rotations of vectors in 3D: Euler angles and rotation matrices. Euler angles, originally developed by Euler as a tool for solving differential equations, have become a widely used method for expressing rotation. Euler angles $(\theta_x, \theta_y, \theta_z)$ are the rotation angles about the x,y, and z axes respectively. There are a few problems with Euler angles that mostly result from ambiguity. The main problem is that there are multiple methods of using Euler angles to achieve the same rotation. Take for example, a rotation about the z-axis by 180°. Clearly, the simplest way to express this is with the Euler angles $(\theta_x = 0°, \theta_y = 0°, \theta_z = 180°)$. However, another equally valid solution is $(\theta_x = 180°, \theta_y = 180°, \theta_z = 0°)$. It is important to note that the *order* that the Euler angles are applied makes a difference. In the previous case it happened to not make a differnce, but in general, the angles are not commutative. There are various conventions in place to help with the ambiguity. In this paper, we will use one of two methods for representing Euler angles. The first method, is "proper" Euler angles (ϕ, θ, ψ), where ϕ is the rotation about the z-axis, $\theta \in [0, \pi]$ is the rotation about the x-axis, and ψ is another rotation about the z-axis. The next method is subset of Euler angles called "Tait-Bryan" angles, where (α, β, γ) correspond to the $(\theta_x, \theta_y, \theta_z)$ method used above. These are both common methods for representing Euler angles.

The other primary method for expressing rotation in 3D is a rotation matrix.

A rotation matrix is usually method for applying Euler angles. Each rotation about the x,y, and z axes are used to generate a 3x3 rotation matrix. These are then multiplied together to get a single rotation matrix. Sometimes a 4x4 homogenous matrix is used because it can hold translation and rotation in the same matrix. We know that matrix multiplication does not commute, which is also true of rotations. The following rotation matrices are used to describe rotation about the given axis:

$$R_x(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos(\theta) & -sin(\theta) \\ 0 & sin(\theta) & cos(\theta) \end{bmatrix}, R_y(\theta) = \begin{bmatrix} cos(\theta) & 0 & sin(\theta) \\ 0 & 1 & 0 \\ -sin(\theta) & 0 & cos(\theta) \end{bmatrix}, R_z(\theta) = \begin{bmatrix} cos(\theta) & sin(\theta) & 0 \\ -sin(\theta) & cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

So a rotation about the z-axis by 180° would be represented by multiplying the three individual rotation matrices together.

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Notice that this is the same final rotation matrix as if we had instead rotated the x-axis by 180° and then the y-axis by 180°. This can be verified as shown:

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{bmatrix} * \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{bmatrix} * \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Compare this to a quaternion of the same rotation. For any rotation about axis $< \beta_x, \beta_y, \beta_z >$, with an angle of α, then

$$\mathbf{q} = \cos\,(\frac{\alpha}{2}) + \ \sin\,(\frac{\alpha}{2})\beta_x\mathbf{i} + \ \sin\,(\frac{\alpha}{2})\beta_y\mathbf{j} + \ \sin\,(\frac{\alpha}{2})\beta_x\mathbf{k}$$

So a rotation of 180° about the z-axis would simply be represented by $\mathbf{q} = \mathbf{k}$.

3.2 Euler Angles to Rotation Matrix

To convert Euler angles to rotation matrices depends on which Euler angle convention is used. Recall that we are using the (ϕ, θ, ψ) method. We can convert each of these to matrics as shown below:

$$x_1 = \begin{bmatrix} \cos\phi & \sin\phi & 0 \\ \text{-sin } \phi & \cos\phi & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$x_2 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & \sin\theta \\ 0 & -\sin\theta & \cos\theta \end{bmatrix}$$

$$x_3 = \begin{bmatrix} \cos\psi & \sin\psi & 0 \\ -\sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

The complete rotation matrix is the multiplication of $x_1 x_2 x_3$, which is shown below.

$$R = \begin{bmatrix} \cos\psi\cos\phi - \cos\theta\sin\phi\sin\psi & \cos\psi\sin\theta + \cos\theta\cos\phi\sin\psi & \sin\psi\sin\theta \\ -\sin\psi\cos\phi - \cos\theta\sin\phi\cos\psi & -\sin\psi\sin\phi + \cos\theta\cos\phi\sin\psi & \cos\psi\sin\theta \\ \sin\theta\sin\phi & -\sin\theta\cos\theta & \cos\theta \end{bmatrix}$$

This allows us to easily convert from Euler angles to rotation matrices. Remember though, once we have a rotation matrix, it is not possible to unambiguously find which angles produced it.

3.3 Comparison Between Methods

3.3.1 Euler Angles and Rotation Matrices Disadvantages

Euler angles and rotation matrices have historically been used to represent rotation in 3D, and the mathematics behind them are well known. Euler angles and rotation matrices make rotations about the x,y, or z axes simple. However an arbitrary rotation about an arbitrary axis makes it difficult to derive the Euler angles. Another disadvantage, as briefly discussed above, is that the order that the rotation matrices are applied in matter. Different conventions lead to different results so care must be taken when coding and using libraries. Euler angles and rotation matrices also inherantly contain the problem of gimbal lock, which will be investiagated in more depth in the Applications section. In addition, given a rotation matrix, it can be difficult to find the inverse because there is no unambiguous inverse. Euler angles and rotation matrices make interpolation difficult because each axis must be interpolated separately. Interpolation is often used in computer animation, but interpolation with Euler angles requires triple the computation time, and power. Lastly, the homogenous matrix holds extra information. When coding applications, extra information is a waste of program space, and in an increase to processing time and power.

3.3.2 Euler Angles and Rotation Matrices Advantages

The biggest advantage to Euler angles and the corresponding rotation matrices, other than for historical reasons, is the ability to encode translation, rotation, scaling, and projection into one matrix. That is to say, if the extra information is needed, rotation matrices can be a good choice. This advantage does not extend to Euler angles however. In practive, Euler angles may be used to derive the rotation matrix, but is unlikely to be used by themselves.

3.3.3 Quaternions Disadvantages

One of the main disadvantages to quaternions is that they can only represent rotation. They do not have the extra information that rotation matrices can hold. Another disadvantage is that quaternions are not usually taught in mathematics courses, so the math may appear complicated. Quaternions may also be more difficult to visualize than Euler angles, or even rotation matrices.

3.3.4 Quaternions Advantages

Quaternions offer multiple advantages over traditional methods. First, there is the geometric implication of representing an arbitrary rotation about an abritrary axis, rather than the combination of x,y, and z rotations. Quaternions are unabmiguous, in that there are only two quaternions to represent a rotation: **q** and -**q**. The negative quaternion is the opposite rotation about the opposite axis. Quaternions are simpler to interpolate than the other methods. Again, this is ideal for animators and other computer graphic visualizations. Quaternions are compact and other take 4 number to represent a single roation. Rotations can be composed by multiplying quaternions. Again, these make for easier applications than Euler angles and rotation matrices. Quaternions do not have any gimabl lock because that is a problem inherant with rotation matrices. This will be investigated further in the following section. Overall, quaternions offer significant adavantages over traditional rotation choices. The only reason to use rotation matrices is when the application also requires translation.

4 Applications

4.1 Aeronautics & Quaternions

Aeronautics often deal with the problem of rotation. An airplane has an "internial frame", which needs to be corresponded to the " body frame." A common way to represent rotation on an airplane is with the following convention:

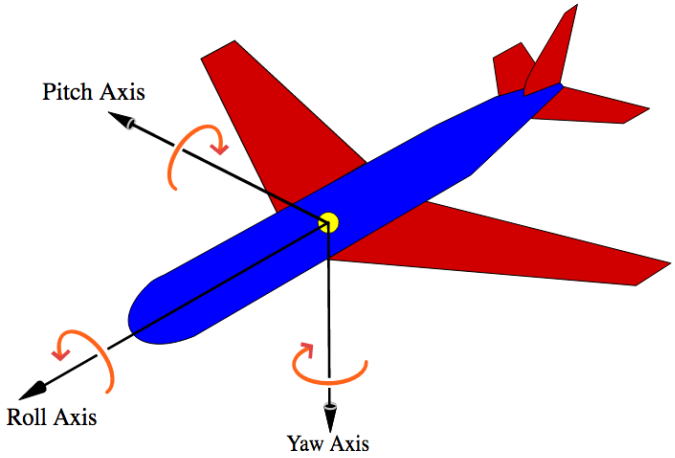


Figure 3: Roll, Pitch, and Yaw

Roll, pitch, and yaw are simply Euler angles. When flying a small plane, and rotating about a single axis, it may be easier to use Euler angles. However, many large commerical airlines use computer assisted flight with many maneuvers done on autopilot. Complicated rotations may be represented as an arbitrary rotation about an arbitrary axis, the perfect use case for quaternions. There is another reason to use quaternions rather than rotation matrices when flying. Rotations in 3D space can be thought of as a three gimbal system, where a gimbal is " a pivoted support that allows the rotation of an object about a single axis." Figure 4 shows an example of a 3-axis gimbal system.

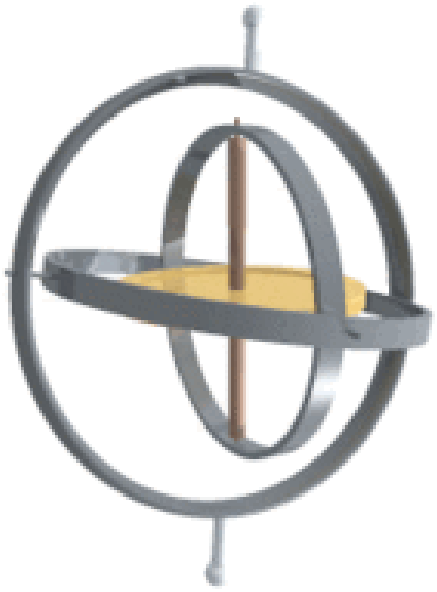


Figure 4: A 3-Axis Gimbal System

This is essentially a visualization of Euler angles, where the deviation from the traditional 3-axis orientation can be used to keep track of the orientation of the system. Gimbals are commonly used in gyroscopes, aeronautics, and rocket systems. Gimbals in a 3-axis configuration have 3 degrees of freedom. A phenomenon can occur where gimbals end up in the same plane, and you lose a degree of freedom. Figure 5 shows a locked gimbal system.

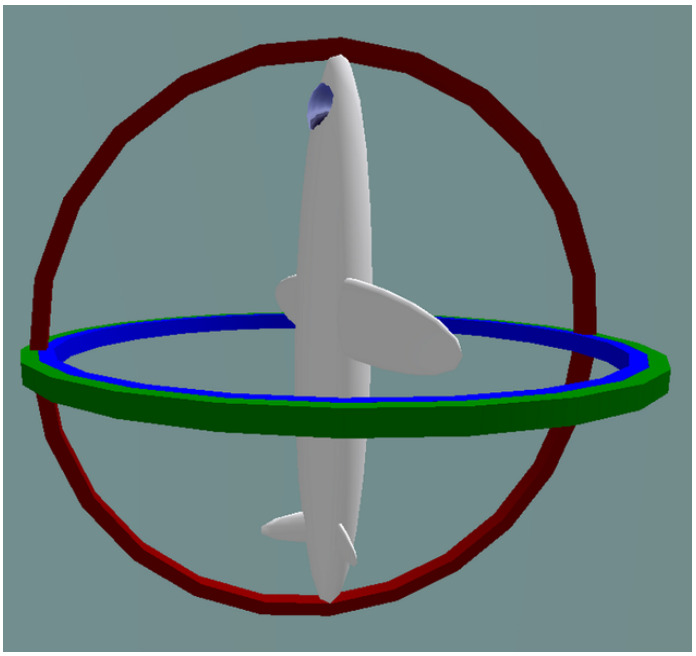


Figure 5: Roll and Yaw Gimbal Locked System

In the above figure, the roll and yaw axes are now represented with a single gimbal. If the plane is told to rotate roll or yaw, the rotation is applied to the same axis. The only way to get out of gimbal lock is to rotate about the single gimbal. Gimbal lock is a property of Euler angles and rotation matrices. Here is the rotation matrix for the (α, β, γ) angles that we did not calculate earlier.

$$R = \begin{bmatrix} \cos\beta\cos\gamma & -\sin\gamma\cos\beta & \sin\beta \\ \sin\beta\cos\alpha\cos\gamma + \cos\alpha\sin\gamma & -\sin\gamma\sin\alpha\sin\beta + \cos\alpha\cos\gamma & -\cos\beta\sin\alpha \\ -\sin\beta\cos\gamma\cos\alpha + \sin\alpha\sin\gamma & \sin\gamma\sin\beta\cos\alpha + \sin\alpha\cos\gamma & \cos\alpha\cos\beta \end{bmatrix}$$

Let us take $\beta = \frac{\pi}{2}$. This results in the following rotation matrix:

$$R = \begin{bmatrix} 0 & 0 & 1 \\ \cos\,\alpha\,\cos\,\gamma + \cos\,\alpha\,\sin\,\gamma & -\sin\,\gamma\,\sin\,\alpha + \cos\,\alpha\,\cos\,\gamma & 0 \\ -\cos\,\gamma\,\cos\,\alpha + \sin\,\alpha\,\sin\,\gamma & \sin\,\gamma\,\cos\,\alpha + \sin\,\alpha\,\cos\,\gamma & 0 \end{bmatrix}$$

Using trigonometric identities, we can simplify this rotation matrix as shown below.

$$R = \begin{bmatrix} 0 & 0 & 1 \\ \sin\,(\alpha+\gamma) & \cos\,(\alpha+\gamma) & 0 \\ -\cos\,(\alpha+\gamma) & \sin\,(\alpha+\gamma) & 0 \end{bmatrix}$$

Clearly, changing α or γ has the same affect. The gimbal is in effect, "locked." To fix this and get out of the gimbal lock, β must be rotated first. This is a problem inherant to Euler angles and their use in rotation matrices, but does not affect quaternions.

4.2 Computer Graphics & Quaternions

Computer graphics, and specifically computer animation, rely on quaternions to express rotation between orientations. Several game engines come with the ability to use quaternions on the back end. The 2D and 3D cross-platform game engine called "Unity", has instructions for coders and animators for implementing quaternions in their projects. There are several reasons that quaternions offer advantages that have already been discussed, namely computation time, memory needed to represent a single rotation, and avoiding gimbal lock. There is one other reason that quaternions offer and advantage to rotation matrices. To investigate this, let us look at the basics of animation.

4.2.1 Animation

Animation relies on simulating three dimensional objects on a rigid frame and projecting them onto a 2D surface. Tranformations (rotations and translations) are used to move objects between orientations and positions. Methods such as spherical interpolation can be used to control the position of infinitely distant light sources, or to model features on a globe. Let's say we are trying to animate a health bar in a video game. We might pick several points that we want the bar to go through, and then perform a linear interpolation to fill in the points between. It would not be too hard to pick some points that are evenly spaced apart, so that the movement does not appear to be jerky when rendered in an animation. This method, shortened to *lerp*, is commonly used for interpolating linear distance or velocity. However, *lerp* does not work well when trying to do spherical interpolation, shortened to *slerp*. [1]

Imagine we are trying to animate an object about arbitrary axis, β, and by an arbitrary angle, θ. We have discussed several methods for obtaining the rotation about β by θ. We could, for example, compute several different Euler angles and then individually rotate about each axis to achieve the desired rotation. However, despite being valid mathematically, this would look pretty odd when animated. Another solution might be to rotate using the quaternion, or rotation matrix, but linearly interpolated. This is a natural rotation, but might appear jerky because the interpolation is not smooth, and if using a rotation matrix, we could still run into the problem of gimbal lock. The solution is to use *slerp* to animate a smooth interpolation with quaternions. This achieves the desired smoothness, without fear of gimbal lock. The mechanics behind *slerp* are outside of the scope of this paper, but it works by keeping a constant speed. The geometric formula for *slerp*, as detailed by Ken Shoemaker, is given below

in Equation 5.

$$Slerp(p_0, p_1, t) = \frac{sin[(1-t)\Omega]}{sin\Omega} p_0 + \frac{sin[t\Omega]}{sin\Omega} p_1 \tag{5}$$

When using Slerp to interpolate quaternions, it produces a uniform angular velocity around a rotation axis. The same equation using quaternions is shown below in Equation 6.

$$Slerp(q_0, q_1, t) = (q_1 q_0^{-1})^t q_0 \tag{6}$$

To investigate the application of slerp, some MATLAB code was written to interpolate a vector rotation. This code for this is shown below.

```
v = [1 .5 1]; % vector
iterations = 100;
p = [1 0 0 0]; % initial quaternion, does not rotate
q = [1 0 1 0]; % Random Quaternion
quat2eul(q) %Makes visualization easier, return euler angles in ZYX

pn = quatnormalize(p); % normalization to unit quaternions
qn = quatnormalize(q);
qi = quatinterp(pn,qn,1/iterations,'slerp'); % performs interpolation between extremes

plot3([0 v(1)], [0 v(2)], [0 v(3)]); % plot initial vector
grid on;
hold on;
r = quatrotate(p,v); % doesn't rotate, but initializes r

for i = 1:iterations % loops over iterations and applies rotation about quaternion
```

```
    r = quatrotate(qi, r);
    plot3([0 r(1)], [0 r(2)], [0 r(3)]);
end

rotate3d; % allows for click and drag in 3d plot
```

Using an iteration value of 2 gives the following result.

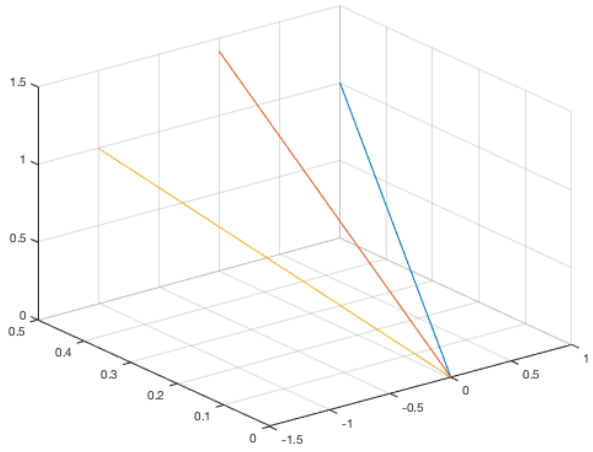


Figure 6: Slerp Example with 3 Iterations

The quaternion used in the code rotates the vector about the y-axis by an angle of 90°. Notice the spherical interpolation of the second vector. Using an iteration value of 100 yields the following plot.

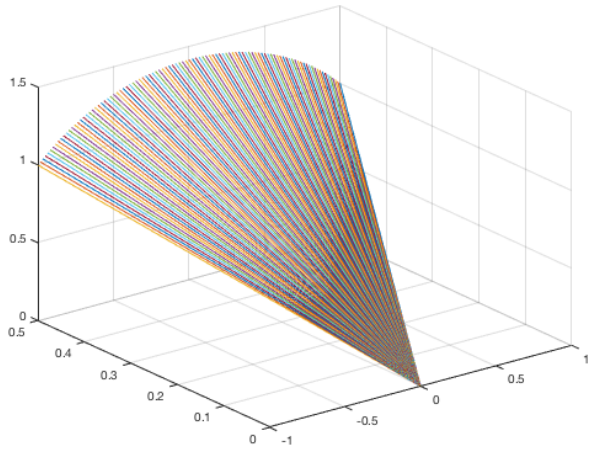


Figure 7: Slerp Example with 100 Iterations

Here the spherical interpolation is more clearly noticable. The MATLAB code is easy to use, and performs the necessary interpolation quickly.

5 Conclusion

This paper details several different transformation methods for rotation in three dimensions. Euler angles, rotation matrices, and quaternions were all described, and compared, and the conversions between them were presented. Overall, quaternions offer better usability for almost every application, except in applications were translations are also needed. Quaternions have uses in aeronautics, robotics, and computer graphics. They prevent gimbal lock, a phenomenon whereby three degrees of freedom becomes two degrees of freedom. Quaternions are also used for spherical interpolation, also known as *slerp*. MATLAB code was used to investigate quaternions and slerp.

References

[1] Erik B. Dam et al, *Quaternions, interpolation, and animation*, 1998, http://web.mit.edu/2.998/www/QuaternionReport1.pdf.

[2] William Hamilton, *Elements of quaternions*, London, Longmans, Green & co., 1866.

[3] Michael Henle, *Modern geometries*, Prentice-Hall, Upper Saddle River, New Jersey, 2001.

[4] Lieut.-Colonel H. W. L. Hime, *Outlines of quaternions*, London, Longmans, Green & co., 1894.

[5] Charles Jasper Joly, *A manual of quaternions*, MacMillan and Co., Limited, New York, NY, 1905.