A Linear Algebraic Approach to Quaternions

David Eberly
Magic Software, Inc.
`http://www.magic-software.com`

Created: September 16, 2002

Unit quaternions are a powerful way to represent rotations within computer graphics and physics applications. Unfortunately, the mathematical complexity of quaternions seems to discourage some practitioners from any attempts at understanding them. This document provides an alternate approach to the presentation of quaternions, one that is based solely on concepts from trigonometry and linear algebra. The algebra and geometry of quaternions is motivated from a study of certain rotation matrices in four dimensions. This is in contrast to the classical approach that defines unit quaternions as points on a unit hypersphere in four dimensions and lists their important algebraic properties to be taken on faith.

1 Rotation Matrices

Let us review a concepts that you are no doubt already familiar with, rotation in the xy–plane. The rotation of the vector (x, y) the origin by an angle $\theta > 0$ is the vector (x', y') specified by

$$x' = \cos(\theta)x - \sin(\theta)y, \quad y' = \sin(\theta)x + \cos(\theta)y.$$

The formula is derivable using a standard trigonometric construction. The direction of rotation is counterclockwise about the origin. In vector–matrix form the equation is

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

If we now add a third dimension, the rotation of the vector (x, y, z) about the z–axis by an angle $\theta > 0$ is just a rotation of the (x, y) portion about the origin in the xy–plane. The rotated vector (x', y', z') is specified by

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

Setting $\mathbf{v} = [x\, y\, z]^{\mathrm{T}}$, $\mathbf{v}' = [x'\, y'\, z']^{\mathrm{T}}$, and

$$s = \sin(\theta) \text{ and } c = \cos(\theta), \tag{1}$$

the rotation is $\mathbf{v}' = R_0\mathbf{v}$ where R_0 is the rotation matrix

$$R_0 = \begin{bmatrix} c & -s & 0 \\ s & c & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2}$$

The standard coordinate axis directions, represented as 3×1 vectors, are $\boldsymbol{\imath} = [1\,0\,0]^{\mathrm{T}}$, $\boldsymbol{\jmath} = [0\,1\,0]^{\mathrm{T}}$, and $\boldsymbol{k} = [0\,0\,1]^{\mathrm{T}}$. Observe that

$$R_0\boldsymbol{\imath} = [c\,s\,0]^{\mathrm{T}} = c\boldsymbol{\imath} + s\boldsymbol{\jmath},\;\; R_0\boldsymbol{\jmath} = [-s\,c\,0]^{\mathrm{T}} = -s\boldsymbol{\imath} + c\boldsymbol{\jmath},\;\; R_0\boldsymbol{k} = [0\,0\,1]^{\mathrm{T}} = \boldsymbol{k}. \tag{3}$$

The vectors $R_0\boldsymbol{\imath}$, $R_0\boldsymbol{\jmath}$, and $R_0\boldsymbol{k}$ are the columns of the rotation matrix R_0.

The equation for rotation of a vector $\mathbf{v} \in \mathbb{R}^3$ by an angle $\theta > 0$ about an axis with unit–length direction $\mathbf{d}$ is derived next. Let $\mathbf{a}$ and $\mathbf{b}$ be vectors in the plane that contains the origin and has normal $\mathbf{d}$. Moreover, choose these vectors so that $\{\mathbf{a}, \mathbf{b}, \mathbf{d}\}$ is a right–handed orthonormal set: each vector is unit length, the vectors are mutually perpendicular, and $\mathbf{a} \times \mathbf{b} = \mathbf{d}$, $\mathbf{b} \times \mathbf{d} = \mathbf{a}$, and $\mathbf{d} \times \mathbf{a} = \mathbf{b}$. Figure 1 shows a typical choice.

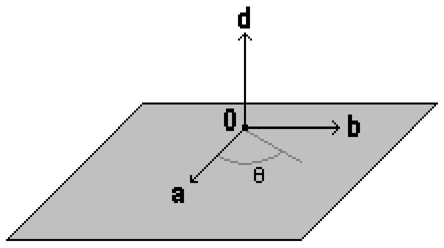

Figure 1. A right–handed orthonormal set of vectors. A rotation is desired about $\mathbf{d}$ by the angle $\theta > 0$.

The orthonormal set of vectors may be used as a *basis* for $\mathbb{R}^3$, both as domain and range of the rotational transformation, denoted $\mathrm{Rot}(\mathbf{u})$. The matrix R_0 in equation (2) represents the rotation in this basis: $\mathrm{Rot}(\mathbf{a}) = c\mathbf{a} + s\mathbf{b}$, $\mathrm{Rot}(\mathbf{b}) = -s\mathbf{a} + c\mathbf{b}$, and $\mathrm{Rot}(\mathbf{d}) = \mathbf{d}$. The similarity between these equations and equations (3) is no coincidence. The representation of $\mathbf{v}$ in the basis is

$$\mathbf{v} = (\mathbf{a} \cdot \mathbf{v})\mathbf{a} + (\mathbf{b} \cdot \mathbf{v})\mathbf{b} + (\mathbf{d} \cdot \mathbf{v})\mathbf{d} =: \alpha\mathbf{a} + \beta\mathbf{b} + \delta\mathbf{d} \tag{4}$$

where the last equality defines α, β, and δ as the dot products of the basis vectors with $\mathbf{v}$. This renaming is done for simplicity of notation in the ensuing constructions. A couple of vector quantities that we will use later are

$$\mathbf{d} \times \mathbf{v} = \mathbf{d} \times (\alpha\mathbf{a} + \beta\mathbf{b} + \delta\mathbf{d}) = \alpha\mathbf{d} \times \mathbf{a} + \beta\mathbf{d} \times \mathbf{b} + \delta\mathbf{d} \times \mathbf{d} = \alpha\mathbf{b} - \beta\mathbf{a} \tag{5}$$

and

$$\mathbf{d} \times (\mathbf{d} \times \mathbf{v}) = \mathbf{d} \times (\alpha\mathbf{b} - \beta\mathbf{a}) = \alpha\mathbf{d} \times \mathbf{b} - \beta\mathbf{d} \times \mathbf{a} = -\alpha\mathbf{a} - \beta\mathbf{b} \tag{6}$$

The rotation applied to $\mathbf{v}$ is represented as a matrix R_1 with respect to the standard basis via:

$$\begin{aligned}
R_1\mathbf{v} &= R_1(\alpha\mathbf{a} + \beta\mathbf{b} + \delta\mathbf{d}) \\
&= \alpha R_1\mathbf{a} + \beta R_1\mathbf{b} + \delta R_1\mathbf{d} \\
&= \alpha(c\mathbf{a} + s\mathbf{b}) + \beta(-s\mathbf{a} + c\mathbf{b}) + \delta\mathbf{d} \\
&= (\alpha\mathbf{a} + \beta\mathbf{b} + \delta\mathbf{d}) + \alpha(c\mathbf{a} + s\mathbf{b} - \mathbf{a}) + \beta(-s\mathbf{a} + c\mathbf{b} - \mathbf{b}) \\
&= (\alpha\mathbf{a} + \beta\mathbf{b} + \delta\mathbf{d}) + s(\alpha\mathbf{b} - \beta\mathbf{a}) + (1-c)(-\alpha\mathbf{a} - \beta\mathbf{b}) \\
&= \mathbf{v} + s\mathbf{d}\times\mathbf{v} + (1-c)\mathbf{d}\times(\mathbf{d}\times\mathbf{v})
\end{aligned} \tag{7}$$

where the last equality is just an application of the equations (4), (5), and (6). The matrix R_1 itself is constructed by observing that the cross product $\mathbf{d}\times\mathbf{u}$ can be written as a matrix times vector:

$$\mathbf{d}\times\mathbf{u} = \begin{bmatrix} d_1 \\ d_2 \\ d_3 \end{bmatrix} \times \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} = \begin{bmatrix} d_2u_3 - d_3u_2 \\ d_3u_1 - d_1u_3 \\ d_1u_2 - d_2u_1 \end{bmatrix} = \begin{bmatrix} 0 & -d_3 & d_2 \\ d_3 & 0 & -d_1 \\ -d_2 & d_1 & 0 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} = D\mathbf{u} \tag{8}$$

where the last equality defines the 3×3 matrix D. This matrix is *skew–symmetric* since $D^{\mathrm{T}} = -D$. The rotation formula (7) becomes

$$R_1\mathbf{v} = \mathbf{v} + sD\mathbf{v} + (1-c)D^2\mathbf{v} = \left(I + sD + (1-c)D^2\right)\mathbf{v}$$

where I is the 3×3 identity matrix. Since this equation is true for all vectors $\mathbf{v}$, the rotation matrix must be

$$R_1 = I + sD + (1-c)D^2. \tag{9}$$

We have arrived at our first goal of understanding: The matrix R_0 in (2) is the representation of the rotation with respect to the basis $\{\mathbf{a}, \mathbf{b}, \mathbf{d}\}$ and the matrix R_1 in (9) is the representation of the rotation with respect to the standard basis $\{\boldsymbol{\imath}, \boldsymbol{\jmath}, \boldsymbol{k}\}$. In the linear algebra terminology, the two matrices are related by a *similarity transformation*: $R_1 = PR_0P^{-1}$ where P is an invertible 3×3 matrix. P will be constructed here shortly.

First, observe that equation (4) may be manipulated as

$$I\mathbf{v} = \mathbf{v} = (\mathbf{a}\cdot\mathbf{v})\mathbf{a} + (\mathbf{b}\cdot\mathbf{v})\mathbf{b} + (\mathbf{d}\cdot\mathbf{v})\mathbf{d} = \mathbf{a}(\mathbf{a}^{\mathrm{T}}\mathbf{v}) + \mathbf{b}(\mathbf{b}^{\mathrm{T}}\mathbf{v}) + \mathbf{d}(\mathbf{d}^{\mathrm{T}}\mathbf{v}) = (\mathbf{a}\mathbf{a}^{\mathrm{T}} + \mathbf{b}\mathbf{b}^{\mathrm{T}} + \mathbf{d}\mathbf{d}^{\mathrm{T}})\mathbf{v}.$$

The equation is true for all vectors $\mathbf{v}$, so

$$I = \mathbf{a}\mathbf{a}^{\mathrm{T}} + \mathbf{b}\mathbf{b}^{\mathrm{T}} + \mathbf{d}\mathbf{d}^{\mathrm{T}} \tag{10}$$

Keep in mind that $\mathbf{u}\mathbf{u}^{\mathrm{T}}$ is the product of a 3×1 matrix and a 1×3 matrix, the result being a 3×3 matrix. This is not the same as $\mathbf{u}^{\mathrm{T}}\mathbf{u}$, a product of a 1×3 matrix and a 3×1 matrix, the result being a 1×1 matrix (a scalar).

Second, equations (5) and (8) imply the relationship

$$D\mathbf{v} = \mathbf{d}\times\mathbf{v} = \alpha\mathbf{b} - \beta\mathbf{a} = (\mathbf{a}\cdot\mathbf{v})\mathbf{b} - (\mathbf{b}\cdot\mathbf{v})\mathbf{a} = \mathbf{b}(\mathbf{a}^{\mathrm{T}}\mathbf{v}) - \mathbf{a}(\mathbf{b}^{\mathrm{T}}\mathbf{v}) = (\mathbf{b}\mathbf{a}^{\mathrm{T}} - \mathbf{a}\mathbf{b}^{\mathrm{T}})\mathbf{v}.$$

This equation is true for all vectors $\mathbf{v}$, so

$$D = \mathbf{b}\mathbf{a}^{\mathrm{T}} - \mathbf{a}\mathbf{b}^{\mathrm{T}}. \tag{11}$$

Third, equations (4) and (6) imply the relationship

$$D^2\mathbf{v} = \mathbf{d} \times (\mathbf{d} \times \mathbf{v}) = -\alpha\mathbf{a} - \beta\mathbf{b} = (\mathbf{d} \cdot \mathbf{v})\mathbf{d} - \mathbf{v} = \mathbf{d}(\mathbf{d}^{\mathrm{T}}\mathbf{v}) - \mathbf{v} = (\mathbf{d}\mathbf{d}^{\mathrm{T}} - I)\mathbf{v}.$$

This equation is true for all vectors $\mathbf{v}$, so

$$D^2 = \mathbf{d}\mathbf{d}^{\mathrm{T}} - I. \tag{12}$$

Now to construct P. Equation (4) can be factored into a product of a 1×3 block matrix (each block a 3×1 vector) and a 3×1 block matrix (each block a single scalar):

$$\mathbf{v} = \alpha\mathbf{a} + \beta\mathbf{b} + \delta\mathbf{d} = \mathbf{a}\alpha + \mathbf{b}\beta + \mathbf{d}\delta = \left[\begin{array}{c|c|c} \mathbf{a} & \mathbf{b} & \mathbf{d} \end{array}\right] \left[\begin{array}{c} \alpha \\ \hline \beta \\ \hline \delta \end{array}\right] = P\mathbf{u} \tag{13}$$

where the last equality defines P as the matrix whose columns are $\mathbf{a}$, $\mathbf{b}$, and $\mathbf{d}$ and $\mathbf{u} = [\alpha\ \beta\ \delta]^{\mathrm{T}}$. Since the columns of P form a right–handed orthonormal set, P is itself a rotation matrix itself, so $P^{-1} = P^{\mathrm{T}}$.

Let $\mathbf{v}'$ denote the rotated vector corresponding to $\mathbf{v}$. The equation $\mathbf{v}' = R_1\mathbf{v}$ represents the rotation when the standard basis $\{\boldsymbol{\imath}, \boldsymbol{\jmath}, \boldsymbol{k}\}$ is used for both the domain and the range. The components of $\mathbf{v}$ and $\mathbf{v}'$ are the coordinates with respect to this basis. Similarly, the $\mathbf{u}' = R_0\mathbf{u}$ represents the rotation when the basis $\{\mathbf{a}, \mathbf{b}, \mathbf{d}\}$ is used for both the domain and range. The components of $\mathbf{u}$ and $\mathbf{u}'$ are the coordinates of $\mathbf{v}$ and $\mathbf{v}'$, respectively, with respect to this basis. Substituting $\mathbf{u} = P^{\mathrm{T}}\mathbf{v}$ and $\mathbf{u}' = P^{\mathrm{T}}\mathbf{v}$ to convert back to the standard basis, we have

$$\mathbf{u}' = R_0\mathbf{u} \ \rightarrow\ P^{\mathrm{T}}\mathbf{v}' = R_0P^{\mathrm{T}}\mathbf{v} \ \rightarrow\ P^{\mathrm{T}}R_1\mathbf{v} = R_0P^{\mathrm{T}}\mathbf{v}.$$

Since this is true for all vectors $\mathbf{v}$, it must be that $P^{\mathrm{T}}R_1 = R_0P^{\mathrm{T}}$ leading to the claimed similarity relationship $R_1 = PR_0P^{\mathrm{T}}$. Equation (9) may be alternately derived from the similarity transformation using block matrix calculations:

$$
\begin{aligned}
R_1 &= P R_0 P^{\mathrm{T}} \\
&= \left[\begin{array}{c|c|c} \mathbf{a} & \mathbf{b} & \mathbf{d} \end{array}\right] \left[\begin{array}{c|c|c} c & -s & 0 \\ \hline s & c & 0 \\ \hline 0 & 0 & 1 \end{array}\right] \left[\begin{array}{c} \mathbf{a}^{\mathrm{T}} \\ \hline \mathbf{b}^{\mathrm{T}} \\ \hline \mathbf{d}^{\mathrm{T}} \end{array}\right] \\
&= \left[\begin{array}{c|c|c} \mathbf{a} & \mathbf{b} & \mathbf{d} \end{array}\right] \left[\begin{array}{c} c\mathbf{a}^{\mathrm{T}} - s\mathbf{b}^{\mathrm{T}} \\ \hline s\mathbf{a}^{\mathrm{T}} + c\mathbf{b}^{\mathrm{T}} \\ \hline \mathbf{d} \end{array}\right] \\
&= \mathbf{a}(c\mathbf{a}^{\mathrm{T}} - s\mathbf{b}^{\mathrm{T}}) + \mathbf{b}(s\mathbf{a}^{\mathrm{T}} + c\mathbf{b}^{\mathrm{T}}) + \mathbf{d}\mathbf{d}^{\mathrm{T}} \\
&= c(\mathbf{a}\mathbf{a}^{\mathrm{T}} + \mathbf{b}\mathbf{b}^{\mathrm{T}}) + s(\mathbf{b}\mathbf{a}^{\mathrm{T}} - \mathbf{a}\mathbf{b}^{\mathrm{T}}) + \mathbf{d}\mathbf{d}^{\mathrm{T}} \\
&= c(I - \mathbf{d}\mathbf{d}^{\mathrm{T}}) + s(\mathbf{b}\mathbf{a}^{\mathrm{T}} - \mathbf{a}\mathbf{b}^{\mathrm{T}}) + \mathbf{d}\mathbf{d}^{\mathrm{T}} && \text{by equation (10)} \\
&= c(I - \mathbf{d}\mathbf{d}^{\mathrm{T}}) + sD + \mathbf{d}\mathbf{d}^{\mathrm{T}} && \text{by equation (11)} \\
&= I + sD + (1-c)(\mathbf{d}\mathbf{d}^{\mathrm{T}} - I) \\
&= I + sD + (1-c)D^2 && \text{by equation (12)}
\end{aligned}
\tag{14}
$$

2 An Alternate View of 3D Rotation

Let us return to the standard basis $\{\boldsymbol{\imath}, \boldsymbol{\jmath}, \boldsymbol{k}\}$ and the rotation matrix R_0 from equation (2) that represents rotation about the z–axis by angle θ. Quite clearly the z–axis is an invariant set under the rotation since $R(z\boldsymbol{k}) = zR(\boldsymbol{k}) = z\boldsymbol{k}$ (vectors on the z–axis are transformed to vectors on the z–axis). The xy–plane is also an invariant set under the rotation since $R(x\boldsymbol{\imath} + y\boldsymbol{\jmath}) = xR\boldsymbol{\imath} + yR\boldsymbol{\jmath} = (cx - sy)\boldsymbol{\imath} + (sc + cy)\boldsymbol{\jmath}$ (vectors in the xy–plane are transformed to vectors in the xy–plane). More precisely, the z–axis and xy–plane are *invariant subspaces* of $\mathbb{R}^3$ with respect to the rotation. The precise definition is: Let V be a vector space with subspace $S \subseteq V$ and let $T : V \to V$ be a linear transformation. Define the set $T(S) = \{T(\mathbf{v}) : \mathbf{v} \in S\}$. The subspace S is *invariant* with respect to T whenever $T(S) \subseteq S$. Note that invariance does not necessarily imply $\mathbf{v} = T(\mathbf{v})$.

Now for an unlikely twist that hints at the essence of quaternions. Instead of viewing the rotation as an operation on vectors $(x, y, z) \in \mathbb{R}^3$, let us look at it as an operation on vectors $(x, y, z, w) \in \mathbb{R}^4$. The inclusion of the w component gives us an additional degree of freedom that allows the creation of a more efficient representation of rotations than what is possible in three dimensions. By efficient, I mean in the sense of a computer implementation. The 4D representation requires less memory than its 3D counterpart. Composition of rotations in 3D involves multiplication of rotation matrices. Composition using the 4D representation can be computed faster than its 3D counterpart.

A natural choice for representing the rotation in 4D is to choose

$$\mathcal{R}_0 = \begin{bmatrix} c & -s & 0 & 0 \\ s & c & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \left[\begin{array}{c|c} R_0 & \mathbf{0} \\ \hline \mathbf{0}^{\mathrm{T}} & 1 \end{array}\right] = \left[\begin{array}{c|c} R_{xy} & 0 \\ \hline 0^{\mathrm{T}} & I \end{array}\right] \tag{15}$$

where the first equality defines a 2×2 block matrix whose upper–left block is the 3×3 rotation matrix, whose upper–right block is the 3×1 zero vector, whose lower–left block is the 1×3 zero vectors, and whose lower–right block is the scalar 1. The second equality defines a 2×2 block matrix where each block is itself a 2×2 matrix. The matrix R_{xy} is just the rotation within the xy–plane, the matrices 0 and 0^{T} have all zeros, and I is the 2×2 identity matrix. The vector (x, y, z, w) is transformed to $(cx - sy, sx + cy, z, w)$. The xy–plane is still an invariant subspace, but now of its parent space $\mathbb{R}^4$. The zw–plane is also an invariant subspace of $\mathbb{R}^4$ since both z and w are unchanged by the transformation. The rotation by angle θ within the xy–plane can be thought of as a composition of two rotations, each by angle $\theta/2$:

$$R_{xy} = \begin{bmatrix} c & -s \\ s & c \end{bmatrix} = \begin{bmatrix} \gamma & -\sigma \\ \sigma & \gamma \end{bmatrix} \begin{bmatrix} \gamma & -\sigma \\ \sigma & \gamma \end{bmatrix} = H^2$$

where

$$\sigma = \sin(\theta/2) \text{ and } \gamma = \cos(\theta/2) \tag{16}$$

and where H is the rotation matrix for the half–angle $\theta/2$ that controls the rotation in the xy–plane. The matrix (15) may be factored into

$$\mathcal{R}_0 = \left[\begin{array}{c|c} H & 0 \\ \hline 0^{\mathrm{T}} & I \end{array}\right] \left[\begin{array}{c|c} H & 0 \\ \hline 0^{\mathrm{T}} & I \end{array}\right]$$

Now for the surprise. The identity matrix I that keeps z and w fixed during the rotations can be replaced by nonidentity matrices. That is, we actually can allow z and w to change during each half–angle rotation in the xy–plane *as long as we make sure z returns to its original value after both operations.* Generally any invertible linear tranformation will do, say

$$\mathcal{R}_0 = \left[\begin{array}{c|c} H & 0 \\ \hline 0^{\mathrm{T}} & M^{-1} \end{array}\right] \left[\begin{array}{c|c} H & 0 \\ \hline 0^{\mathrm{T}} & M \end{array}\right]$$

where M is an invertible 2×2 matrix. Why complicate matters? Let us just choose $M = H^{\mathrm{T}}$. This choice (rather than $M = H$) will keep us consistent with the standard use of quaternions to represent rotations. The factorization is

$$\mathcal{R}_0 = \left[\begin{array}{c|c} H & 0 \\ \hline 0^{\mathrm{T}} & H \end{array}\right] \left[\begin{array}{c|c} H & 0 \\ \hline 0^{\mathrm{T}} & H^{\mathrm{T}} \end{array}\right] =: \overline{\mathcal{Q}}_0 \mathcal{Q}_0 \tag{17}$$

where the last equality defines the matrices $\mathcal{Q}_0$ and $\overline{\mathcal{Q}}_0$, themselves rotations in 4D. In summary, the half–angle rotation H is applied twice to (x, y) to obtain the full angle rotation in the xy–plane. The inverse half–angle rotation H^{T} is applied to (z, w), a rotation within the zw–plane, but that rotation is undone by H in the second operation, the end result being that (z, w) is unchanged by the composition.

What does this really gain us? For the 3D rotation matrix R_0, we have no gain (rather, a loss). The 3D matrix requires storing two precomputed numbers, s and c. The zeros and one are in known positions and do not need to be stored in general memory. The application of R to (x, y, z) is computed as $R_{xy}(x, y)$ since z is unchanged. This requires a product of a 2×2 matrix and a 2×1 vector that uses 6 operations (4 multiplications and 2 additions). The 4D matrix requires storing σ and γ–no change in memory requirements. However, the blind application of the right–hand–side matrices in (17) leads to computing terms $H(x, y)$, $H(H(x, y))$, $H^{\mathrm{T}}(z, w)$, and $H(H^{\mathrm{T}}(z, w))$ for a total of 24 operations. We could be clever and realize that (z, w) will not change, but that still leaves us with computing $H(x, y)$ and $H(H(x, y))$ for a total of 12 operations. Being even more clever, we realize that $H^2 = R_{xy}$ and just compute $R_{xy}(x, y)$. This just brings us full circle with no gain.

The real gain occurs by constructing a 4D rotation matrix $\mathcal{R}_1$ that corresponds to the general 3D rotation matrix R_1 of equation (9). The construction in equation (14) is what we use for motivation. We need to "lift" all our basis vectors into $\mathbb{R}^4$ by appending a zero w–component. These vectors will be written as block matrices to preserve the notion of the first three components living in $\mathbb{R}^3$. Additional vectors are defined to allow us to have a standard basis and an alternate basis for $\mathbb{R}^4$. The standard basis is $\{\hat{\boldsymbol{\imath}}, \hat{\boldsymbol{\jmath}}, \hat{\boldsymbol{k}}, \hat{\boldsymbol{\ell}}\}$ where

$$\hat{\boldsymbol{\imath}} = \left[\begin{array}{c} \boldsymbol{\imath} \\ \hline 0 \end{array}\right], \quad \hat{\boldsymbol{\jmath}} = \left[\begin{array}{c} \boldsymbol{\jmath} \\ \hline 0 \end{array}\right], \quad \hat{\boldsymbol{k}} = \left[\begin{array}{c} \boldsymbol{k} \\ \hline 0 \end{array}\right], \quad \hat{\boldsymbol{\ell}} = \left[\begin{array}{c} \mathbf{0} \\ \hline 1 \end{array}\right]. \tag{18}$$

The alternate basis is $\{\hat{\mathbf{a}}, \hat{\mathbf{b}}, \hat{\mathbf{d}}, \hat{\boldsymbol{\ell}}\}$ where

$$\hat{\mathbf{a}} = \left[\begin{array}{c} \mathbf{a} \\ \hline 0 \end{array}\right], \quad \hat{\mathbf{b}} = \left[\begin{array}{c} \mathbf{b} \\ \hline 0 \end{array}\right], \quad \hat{\mathbf{d}} = \left[\begin{array}{c} \mathbf{d} \\ \hline 0 \end{array}\right]. \tag{19}$$

A construction analogous to the one that produces equation (10) may be used to obtain

$$\mathcal{I} = \hat{\mathbf{a}}\hat{\mathbf{a}}^{\mathrm{T}} + \hat{\mathbf{b}}\hat{\mathbf{b}}^{\mathrm{T}} + \hat{\mathbf{d}}\hat{\mathbf{d}}^{\mathrm{T}} + \hat{\boldsymbol{\ell}}\hat{\boldsymbol{\ell}}^{\mathrm{T}} \tag{20}$$

where $\mathcal{I}$ is the 4×4 identity matrix. A construction similar to the one that leads to equation (11) may be used to obtain

$$\mathcal{D} = \left[\begin{array}{r|c} D & \mathbf{d} \\ \hline -\mathbf{d}^{\mathrm{T}} & 0 \end{array}\right] = \hat{\mathbf{b}}\hat{\mathbf{a}}^{\mathrm{T}} - \hat{\mathbf{a}}\hat{\mathbf{b}}^{\mathrm{T}} + \hat{\mathbf{d}}\hat{\boldsymbol{\ell}}^{\mathrm{T}} - \hat{\boldsymbol{\ell}}\hat{\mathbf{d}}^{\mathrm{T}}. \tag{21}$$

The matrix $\mathcal{Q}_0$ represents a general 4D rotation with respect to the alternate basis. We need a matrix $\mathcal{Q}_1$ that represents the same rotation, but with respect to the standard basis. As in equation (9), a similarity relationship exists: $\mathcal{Q}_1 = \mathcal{P}\mathcal{Q}_0\mathcal{P}^{\mathrm{T}}$ where $\mathcal{P} = [\hat{\mathbf{a}}\ \hat{\mathbf{b}}\ \hat{\mathbf{d}}\ \hat{\boldsymbol{\ell}}]$, a 4D rotation matrix. Block matrix calculations allow us to construct $\mathcal{Q}_1$:

$$\begin{aligned}
\mathcal{Q}_1 &= \mathcal{P}\mathcal{Q}_0\mathcal{P}^{\mathrm{T}} \\
&= \left[\begin{array}{c|c|c|c} \hat{\mathbf{a}} & \hat{\mathbf{b}} & \hat{\mathbf{d}} & \hat{\boldsymbol{\ell}} \end{array}\right] \left[\begin{array}{c|c|c|c} \gamma & -\sigma & 0 & 0 \\ \hline \sigma & \gamma & 0 & 0 \\ \hline 0 & 0 & \gamma & \sigma \\ \hline 0 & 0 & -\sigma & \gamma \end{array}\right] \left[\begin{array}{c} \hat{\mathbf{a}}^{\mathrm{T}} \\ \hline \hat{\mathbf{b}}^{\mathrm{T}} \\ \hline \hat{\mathbf{d}}^{\mathrm{T}} \\ \hline \hat{\boldsymbol{\ell}}^{\mathrm{T}} \end{array}\right] \qquad (22) \\
&= \gamma(\hat{\mathbf{a}}\hat{\mathbf{a}}^{\mathrm{T}} + \hat{\mathbf{b}}\hat{\mathbf{b}}^{\mathrm{T}} + \hat{\mathbf{d}}\hat{\mathbf{d}}^{\mathrm{T}} + \hat{\boldsymbol{\ell}}\hat{\boldsymbol{\ell}}^{\mathrm{T}} + \sigma(\hat{\mathbf{b}}\hat{\mathbf{a}}^{\mathrm{T}} - \hat{\mathbf{a}}\hat{\mathbf{b}}^{\mathrm{T}} + \hat{\mathbf{d}}\hat{\boldsymbol{\ell}}^{\mathrm{T}} - \hat{\boldsymbol{\ell}}\hat{\mathbf{d}}^{\mathrm{T}}) \\
&= \gamma\mathcal{I} + \sigma\mathcal{D} \quad \text{by equations (20) and (21)}
\end{aligned}$$

To emphasize that vectors in $\mathbb{R}^4$ are thought of as a $\mathbb{R}^3$ part, the first three components, and a last component we can write the matrix as

$$\mathcal{Q}_1 = \left[\begin{array}{c|c} \gamma I + \sigma D & \sigma\mathbf{d} \\ \hline -\sigma\mathbf{d}^{\mathrm{T}} & \gamma \end{array}\right]. \qquad (23)$$

where I is the 3×3 identity and D is the skew–symmetric matrix defined in the last section. A similar construction leads to

$$\overline{\mathcal{Q}}_1 = \left[\begin{array}{c|c} \gamma I + \sigma D & -\sigma\mathbf{d} \\ \hline \sigma\mathbf{d}^{\mathrm{T}} & \gamma \end{array}\right]. \qquad (24)$$

A quick calculation that uses equation (9) will verify that

$$\mathcal{R}_1 = \overline{\mathcal{Q}}_1\mathcal{Q}_1 = \left[\begin{array}{c|c} R_1 & \mathbf{0} \\ \hline \mathbf{0}^{\mathrm{T}} & 1 \end{array}\right].$$

3 A Closer Look

We now have an interesting formulation for a 3D rotation by relating it to a pair of 4D rotations. In summary, if R is a 3D rotation matrix with unit–length axis $\mathbf{d} = (d_1, d_2, d_3)$ and angle θ, let $\sigma = \sin(\theta/2)$, $\gamma = \cos(\theta/2)$, and

$$\mathcal{Q} = \left[\begin{array}{c|c} \gamma I + \sigma D & \sigma\mathbf{d} \\ \hline -\sigma\mathbf{d}^{\mathrm{T}} & \gamma \end{array}\right] \quad \text{and} \quad \overline{\mathcal{Q}} = \left[\begin{array}{c|c} \gamma I + \sigma D & -\sigma\mathbf{d} \\ \hline \sigma\mathbf{d}^{\mathrm{T}} & \gamma \end{array}\right].$$

The 4D rotation matrix that represents the 3D rotation is

$$\mathcal{R} = \overline{\mathcal{Q}}\mathcal{Q} = \left[\begin{array}{c|c} \gamma I + \sigma D & -\sigma\mathbf{d} \\ \hline \sigma\mathbf{d}^{\mathrm{T}} & \gamma \end{array}\right] \left[\begin{array}{c|c} \gamma I + \sigma D & \sigma\mathbf{d} \\ \hline -\sigma\mathbf{d}^{\mathrm{T}} & \gamma \end{array}\right] = \left[\begin{array}{c|c} R & \mathbf{0} \\ \hline \mathbf{0}^{\mathrm{T}} & 1 \end{array}\right].$$

The application to a vector $\mathbf{v} \in \mathbb{R}^3$ is

$$\left[\begin{array}{c} \mathbf{v}' \\ \hline 0 \end{array}\right] = \mathcal{R} \left[\begin{array}{c} \mathbf{v} \\ \hline 0 \end{array}\right]$$

and the 3D result is $\mathbf{v}' = R\mathbf{v}$. The matrix $\mathcal{Q}$ has a very special form,

$$\mathcal{Q} = \begin{bmatrix} \gamma & -\sigma d_3 & \sigma d_2 & \sigma d_1 \\ \sigma d_3 & \gamma & -\sigma d_1 & \sigma d_2 \\ -\sigma d_2 & \sigma d_1 & \gamma & \sigma d_3 \\ -\sigma d_1 & -\sigma d_2 & -\sigma d_3 & \gamma \end{bmatrix} =: \begin{bmatrix} w & -z & y & x \\ z & w & -x & y \\ -y & x & w & z \\ -x & -y & -z & w \end{bmatrix} \tag{25}$$

where the last equality defines $x = \sigma d_1$, $y = \sigma d_2$, $z = \sigma d_3$, and $w = \gamma$. The names x, y, z, and w of these quantites are not to be confused with the names of the variable components for vectors in $\mathbb{R}^4$. I use the names because that is the standard choice for the components of a quaternion which $\mathcal{Q}$ happens to be related to. Although $\mathcal{Q}$ has 16 entries, only 4 of them are unique–the last column values. Moreover, $\overline{\mathcal{Q}}$ uses those same 4 values. The matrix R has 9 entries to be stored in memory, but in our fancy formulation we only need 4 entries, a significant reduction when an application's data contains a lot of rotation matrices. If memory were really at a premium, we could use the fact that

$$x^2 + y^2 + z^2 + w^2 = \sigma^2 d_1^2 + \sigma^2 d_2^2 + \sigma^2 d_3^3 + \gamma^2 = \sigma^2 |\mathbf{d}|^2 + \gamma^2 = \sigma^2 + \gamma^2 = 1,$$

restrict angles $\theta \in [-\pi, \pi)$ so that $w \geq 0$, and store only x, y, and z. The w value is reconstructed as $w = \sqrt{1 - (x^2 + y^2 + z^2)}$.

One issue that has not yet been addressed is composition of rotations. Let us do so now. Let $\mathcal{R} = \overline{\mathcal{Q}}\mathcal{Q}$ and $\mathcal{S} = \overline{\mathcal{P}}\mathcal{P}$ be two of our 4D rotations that correspond to 3D rotation matrices R and S, respectively. The composition in the 4D setting is

$$\mathcal{S}\mathcal{R} = \left(\overline{\mathcal{P}}\mathcal{P}\right)\left(\overline{\mathcal{Q}}\mathcal{Q}\right) = \overline{\mathcal{P}}\left(\mathcal{P}\overline{\mathcal{Q}}\right)\mathcal{Q} = \overline{\mathcal{P}}\left(\overline{\mathcal{Q}}\mathcal{P}\right)\mathcal{Q} = \left(\overline{\mathcal{P}\mathcal{Q}}\right)\left(\mathcal{P}\mathcal{Q}\right). \tag{26}$$

The astute reader will say "Wait a moment." and remind me that matrix multiplication is generally not commutative, so how can I switch the order in $\mathcal{P}\overline{\mathcal{Q}} = \overline{\mathcal{Q}}\mathcal{P}$? As it turns out, the matrices $\mathcal{P}$ and $\overline{\mathcal{Q}}$ do commute. This can be verified with a moderate amount of symbolic manipulation. (Okay, I was lazy and used Mathematica to verify this first, then manually checked the symbolic calculations.) What equation (26) says is that I can store $\mathcal{Q}$ to represent $\mathcal{R}$, $\mathcal{P}$ to represent $\mathcal{S}$, and compute the product $\mathcal{P}\mathcal{Q}$ and store to represent the composition $\mathcal{S}\mathcal{R}$. If $\mathcal{Q}$ is stored as the 4–tuple (x_1, y_1, z_1, w_1) and $\mathcal{P}$ is stored as the 4–tuple (x_2, y_2, z_2, w_2), we only need to compute the unique values in $\mathcal{P}\mathcal{Q}$, call them (x_3, y_3, z_3, w_3). We can do this by computing $\mathcal{P}$ times the last column of $\mathcal{Q}$:

$$\begin{bmatrix} x_3 \\ y_3 \\ z_3 \\ w_3 \end{bmatrix} = \begin{bmatrix} w_2 & -z_2 & y_2 & x_2 \\ z_2 & w_2 & -x_2 & y_2 \\ -y_2 & x_2 & w_2 & z_2 \\ -x_2 & -y_2 & -z_2 & w_2 \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \\ z_1 \\ w_1 \end{bmatrix} = \begin{bmatrix} w_2 x_1 - z_2 y_1 + y_2 z_1 + x_2 w_1 \\ z_2 x_1 + w_2 y_1 - x_2 z_1 + y_2 w_1 \\ -y_2 x_1 + x_2 y_1 + w_2 z_1 + z_2 w_1 \\ -x_2 x_1 - y_2 y_1 - z_2 z_1 + w_2 w_1 \end{bmatrix}. \tag{27}$$

The product $\mathcal{P}\mathcal{Q}$ effectively represents the composition of the 3D rotations. Computing the product requires 28 operations (16 multiplications and 12 additions). The product SR of the 3D rotations requires 3 multiplications and 2 additions per entry for a total of 45 operations. Clearly our special representation is cheaper to compute compositions of rotations.

The one deficiency of the special representation is in actually transforming the vectors. In 3D, the product $\mathbf{v}' = R\mathbf{v}$ requires 15 operations (9 multiplications and 6 additions). The iterated product

$$\left[\begin{array}{c} u_1 \\ u_2 \\ u_3 \\ \hline 0 \end{array}\right] = \left(\left[\begin{array}{ccc|c} w & -z & y & -x \\ z & w & -x & -y \\ -y & x & w & -z \\ \hline x & y & z & w \end{array}\right]\left(\left[\begin{array}{ccc|c} w & -z & y & x \\ z & w & -x & y \\ -y & x & w & z \\ \hline -x & -y & -z & w \end{array}\right]\left[\begin{array}{c} v_1 \\ v_2 \\ v_3 \\ \hline 0 \end{array}\right]\right)\right)$$

requires, in worst case, 28 operations for each of two generic multiplies of a 4×4 matrix and a 4×1 vector for a total of 56 operations, a dismal result compared to the 15 operations for the rotation in 3D. However, a smarter implementation notices that the last component of the input vector is zero, so the corresponding multiplications and additions do not need to be calculated. The first product of the 4×4 matrix and the input vector therefore needs only 5 operations per each of 4 intermediate terms for a total of 20 operations. Moreover, the last component of the output vector is zero, so it does not have to be formally computed. The second product of the 4×4 matrix and the intermediate 4×1 vector from the last product will require 7 operations per each of 3 output terms for a total of 21 operations. The grand total in the optimized scheme is 41 operations–still not in the ballpark to compete with the natural 3D product. Yet we can optimize one last time. The product $\overline{\mathcal{Q}}\mathcal{Q}$ contains R itself as the upper–left 3×3 block. In terms of the 4–tuple (x, y, z, w) that represents $\mathcal{Q}$, we have

$$R = \begin{bmatrix} 1-2(y^2+z^2) & 2(xy-zw) & 2(xz+yw) \\ 2(xy+zw) & 1-2(x^2+z^2) & 2(yz-xw) \\ 2(xz-yw) & 2(yz+xw) & 1-2(x^2+y^2) \end{bmatrix} \tag{28}$$

where we have used the fact that $x^2+y^2+z^2+w^2 = 1$ to reduce some of the matrix entries. The number of operations required to compute R in this manner is 24 (12 multiplications and 12 additions). The calculations are: $t_x = 2x$, $t_y = 2y$, $t_z = 2z$, $t_{xx} = t_x x$, $t_{xy} = t_x y$, $t_{xz} = t_x z$, $t_{xw} = t_x w$, $t_{yy} = t_y y$, $t_{yz} = t_y z$, $t_{yw} = t_y w$, $t_{zz} = t_z z$, $t_{zw} = t_z w$, $R_{11} = 1-t_{yy}-t_{zz}$, $R_{12} = t_{xy}-t_{wz}$, $R_{13} = t_{xz}+t_{yw}$, $R_{21} = t_{xy}+t_{zw}$, $R_{22} = 1-t_{xx}-t_{zz}$, $R_{23} = t_{yz}-t_{xw}$, $R_{31} = t_{xz}-t_{yw}$, $R_{32} = t_{yz}+t_{xw}$, and $R_{33} = 1-t_{xx}-t_{yy}$. After computing R, the rotation is calculated as $R\mathbf{v}$. The total operation count is 39 (24 to compute R and 15 to transform). This is slightly better than 41, but still more costly than if we stored R directly instead of (x, y, z, w). The saving factor here, though, is that in computer graphics and physics applications, large sets of vectors are rotated (model space to world space conversion of triangle meshes, for example). The conversion to R occurs once followed by the transformation of a large set of vectors. The reduced memory for storing (x, y, z, w) instead of R and the increased speed for computing compositions certainly justify the 24 operation fixed cost of conversion to R for the purposes of vector transformations.

4 Spherical Linear Interpolation

The 4–tuple (x, y, z, w) that represents the matrix $\mathcal{Q}$ was already shown to be unit length when viewed as a vector in $\mathbb{R}^4$. That means it is a point on the hypersphere of radius 1 that is centered at the origin of $\mathbb{R}^4$. This is just a fancy way of stating the geometry associated with the algebraic equation $x^2+y^2+z^2+w^2 = 1$.

A standard problem in computer graphics and animation is to interpolate two 3D rotation matrices R_0 and R_1 for various choices of $t \in [0, 1]$. The interpolant is denoted $R(t)$, a rotation matrix itself, and it is required that $R(0) = R_0$ and $R(1) = R_1$. The 4–tuple representations of the rotation matrices and the corresponding hypersphere geometry allow for a simple yet elegant interpolation called *spherical linear interpolation* or *slerp* for short. If $\mathbf{q}_i = (x_i, y_i, z_i, w_i)$ are the 4–tuple representations for R_i $(i = 0, 1)$, and if $\mathbf{q}(t)$ is the 4–tuple representing $R(t)$, then a reasonable geometric condition to impose is that $\mathbf{q}(t)$ lie on the hyperspherical arc connecting $\mathbf{q}_0$ and $\mathbf{q}_1$. Moreover, the angle between $\mathbf{q}(t)$ and $\mathbf{q}_0$ should be proportional to the angle ϕ between $\mathbf{q}_0$ and $\mathbf{q}_1$ with constant of proportionality t. Figure 2 illustrates this by showing the plane spanned by $\mathbf{q}_0$ and $\mathbf{q}_1$ and the circular arc connecting them within that plane.

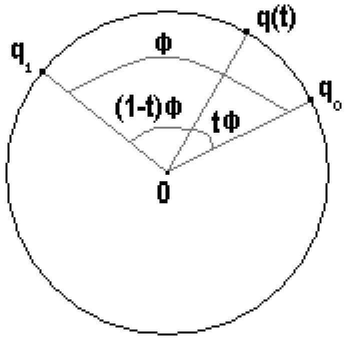

Figure 2. Illustration of the slerp of two vectors.

The angle ϕ between $\mathbf{q}_0$ and $\mathbf{q}_1$ is indirectly obtained by a dot product, $\cos(\phi) = \mathbf{q}_0 \cdot \mathbf{q}_1$. The interpolant is required to be of the form $\mathbf{q}(t) = c_0(t)\mathbf{q}_0 + c_1(t)\mathbf{q}_1$ for some to–be–determined coefficient functions $c_0(t)$ and $c_1(t)$. Construction of $\mathbf{q}(t)$ uses only trigonometry and solving two equations in two unknowns. As t uniformly varies between 0 and 1, the values $\mathbf{q}(t)$ are required to uniformly vary along the circular arc from $\mathbf{q}_0$ to $\mathbf{q}_1$. That is, the angle between $\mathbf{q}(t)$ and $\mathbf{q}_0$ is $t\phi$ and the angle between $\mathbf{q}(t)$ and $\mathbf{q}_1$ is $(1-t)\phi$. Dotting the equation for $\mathbf{q}(t)$ with $\mathbf{q}_0$ yields

$$\cos(t\phi) = c_0(t) + \cos(\phi)c_1(t)$$

and dotting the equation with $\mathbf{q}_1$ yields

$$\cos((1-t)\phi) = \cos(\phi)c_0(t) + c_1(t).$$

These are two equations in the two unknowns c_0 and c_1. The solution for c_0 is

$$c_0(t) = \frac{\cos(t\phi) - \cos(\phi)\cos((1-t)\phi)}{1 - \cos^2(\phi)} = \frac{\sin((1-t)\phi)}{\sin(\phi)}.$$

The last equality is obtained by applying double–angle formulas for sine and cosine. By symmetry, $c_1(t) = c_0(1-t)$. Or solve the equations for

$$c_1(t) = \frac{\cos((1-t)\phi) - \cos(\phi)\cos(t\phi)}{1 - \cos^2(\phi)} = \frac{\sin(t\phi)}{\sin(\phi)}.$$

The spherical linear interpolation is

$$\text{slerp}(t; \mathbf{q}_0, \mathbf{q}_1) = \frac{\sin((1-t)\phi)\mathbf{q}_0 + \sin(t\phi)\mathbf{q}_1}{\sin\phi} \tag{29}$$

for $0 \le t \le 1$.

5 The Relationship to Quaternions

There you have it. The previous sections, based only on linear algebraic methods, shows an alternate system for rotation vectors that has the advantages of reduced memory and increased speed of compositions compared to the standard 3D rotation system. The 4–tuple (x, y, z, w) that represents the matrix $\mathcal{Q}$ is a disguised version of a quaternion that is formally written as $q = xi + yj + zk + w$. Equation (27) is a disguised version of the product of two quaternions. Equation (25) tells you how to calculate a quaternion from a rotation matrix. Equation (28) tells you how to calculate a rotation matrix from a quaternion.

If you want to find out about the classical approach to defining quaternions and showing how they relate to rotations, you will have to read other documents. After all, my point was to give you an alternate formulation without all the mathematical complexity!