Rotation formalisms in three dimensions

In geometry, various **formalisms** exist to express a rotation in three dimensions as a mathematical transformation. In physics, this concept is applied to classical mechanics where rotational (or angular) kinematics is the science of quantitative description of a purely rotational motion. The orientation of an object at a given instant is described with the same tools, as it is defined as an imaginary rotation from a reference placement in space, rather than an actually observed rotation from a previous placement in space.

According to Euler's rotation theorem the rotation of a rigid body (or three-dimensional coordinate system with the fixed origin) is described by a single rotation about some axis. Such a rotation may be uniquely described by a minimum of three real parameters. However, for various reasons, there are several ways to represent it. Many of these representations use more than the necessary minimum of three parameters, although each of them still has only three degrees of freedom.

An example where rotation representation is used is in computer vision, where an automated observer needs to track a target. Consider a rigid body, with three orthogonal unit vectors fixed to its body (representing the three axes of the object's local coordinate system). The basic problem is to specify the orientation of these three unit vectors, and hence the rigid body, with respect to the observer's coordinate system, regarded as a reference placement in space.

Contents

Rotations and motions

Rotation formalisms are focused on proper (orientation-preserving) motions of the Euclidean space with one fixed point, that a *rotation* refers to. Although physical motions with a fixed point are an important case (such as ones described in the center-of-mass frame, or motions of a joint), this approach creates a knowledge about all motions. Any proper motion of the Euclidean space decomposes to a rotation around the origin and a translation. Whichever the order of their composition will be, the "pure" rotation component wouldn't change, uniquely determined by the complete motion.

One can also understand "pure" rotations as linear maps in a vector space equipped with Euclidean structure, not as maps of points of a corresponding affine space. In other words, a rotation formalism captures only the rotational part of a motion, that contains three degrees of freedom, and ignores the translational part, that contains another three.

When representing a rotation as numbers in a computer, some people prefer the quaternion representation or the axis+angle representation, because they avoid the gimbal lock that can occur with Euler rotations.[1]

Formalism alternatives

Rotation matrix

The above-mentioned triad of unit vectors is also called a basis. Specifying the coordinates (*components*) of vectors of this basis in its current (rotated) position, in terms of the reference (non-rotated) coordinate axes, will completely describe the rotation. The three unit vectors, $\hat{\mathbf{u}}$, $\hat{\mathbf{v}}$ and $\hat{\mathbf{w}}$, that form the rotated basis each consist of 3 coordinates, yielding a total of 9 parameters.

These parameters can be written as the elements of a 3×3 matrix $\mathbf{A}$, called a **rotation matrix**. Typically, the coordinates of each of these vectors are arranged along a column of the matrix (however, beware that an alternative definition of rotation matrix exists and is widely used, where the vectors' coordinates defined above are arranged by rows[2])

$$\mathbf{A} = \begin{bmatrix} \hat{\mathbf{u}}_x & \hat{\mathbf{v}}_x & \hat{\mathbf{w}}_x \\ \hat{\mathbf{u}}_y & \hat{\mathbf{v}}_y & \hat{\mathbf{w}}_y \\ \hat{\mathbf{u}}_z & \hat{\mathbf{v}}_z & \hat{\mathbf{w}}_z \end{bmatrix}$$

The elements of the rotation matrix are not all independent—as Euler's rotation theorem dictates, the rotation matrix has only three degrees of freedom.

The rotation matrix has the following properties:

- $\mathbf{A}$ is a real, orthogonal matrix, hence each of its rows or columns represents a unit vector.
- The eigenvalues of $\mathbf{A}$ are

  $$\left\{1, e^{\pm i\theta}\right\} = \{1,\ \cos\theta + i\sin\theta,\ \cos\theta - i\sin\theta\}$$

  where i is the standard imaginary unit with the property $i^2 = -1$
- The determinant of $\mathbf{A}$ is +1, equivalent to the product of its eigenvalues.
- The trace of $\mathbf{A}$ is $1 + 2\cos\theta$, equivalent to the sum of its eigenvalues.

The angle θ which appears in the eigenvalue expression corresponds to the angle of the Euler axis and angle representation. The eigenvector corresponding to the eigenvalue of 1 is the accompanying Euler axis, since the axis is the only (nonzero) vector which remains unchanged by left-multiplying (rotating) it with the rotation matrix.

The above properties are equivalent to

$$\begin{aligned} |\hat{\mathbf{u}}| = |\hat{\mathbf{v}}| = |\hat{\mathbf{w}}| &= 1 \\ \hat{\mathbf{u}} \cdot \hat{\mathbf{v}} &= 0 \\ \hat{\mathbf{u}} \times \hat{\mathbf{v}} &= \hat{\mathbf{w}}\,, \end{aligned}$$

which is another way of stating that $(\hat{\mathbf{u}}, \hat{\mathbf{v}}, \hat{\mathbf{w}})$ form a 3D orthonormal basis. These statements comprise a total of 6 conditions (the cross product contains 3), leaving the rotation matrix with just 3 degrees of freedom, as required.

Two successive rotations represented by matrices $\mathbf{A}_1$ and $\mathbf{A}_2$ are easily combined as elements of a group,

$$\mathbf{A}_{\text{total}} = \mathbf{A}_2 \mathbf{A}_1$$

(Note the order, since the vector being rotated is multiplied from the right).

The ease by which vectors can be rotated using a rotation matrix, as well as the ease of combining successive rotations, make the rotation matrix a useful and popular way to represent rotations, even though it is less concise than other representations.

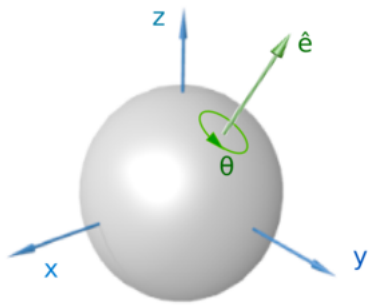
Euler axis and angle (rotation vector)

From Euler's rotation theorem we know that any rotation can be expressed as a single rotation about some axis. The axis is the unit vector (unique except for sign) which remains unchanged by the rotation. The magnitude of the angle is also unique, with its sign being determined by the sign of the rotation axis.

The axis can be represented as a three-dimensional unit vector

$$\hat{\mathbf{e}} = \begin{bmatrix} e_x \\ e_y \\ e_z \end{bmatrix}$$

and the angle by a scalar θ.



A visualization of a rotation represented by an Euler axis and angle.

Since the axis is normalized, it has only two degrees of freedom. The angle adds the third degree of freedom to this rotation representation.

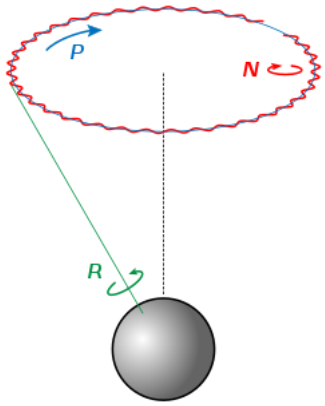
One may wish to express rotation as a **rotation vector**, or **Euler vector**, an un-normalized three-dimensional vector the direction of which specifies the axis, and the length of which is θ,

$$\mathbf{r} = \theta \hat{\mathbf{e}}\,.$$

The rotation vector is useful in some contexts, as it represents a three-dimensional rotation with only three scalar values (its components), representing the three degrees of freedom. This is also true for representations based on sequences of three Euler angles (see below).

If the rotation angle θ is zero, the axis is not uniquely defined. Combining two successive rotations, each represented by an Euler axis and angle, is not straightforward, and in fact does not satisfy the law of vector addition, which shows that finite rotations are not really vectors at all. It is best to employ the rotation matrix or quaternion notation, calculate the product, and then convert back to Euler axis and angle.

Euler rotations



Euler rotations of the Earth. Intrinsic (green), precession (blue) and nutation (red)

The idea behind Euler rotations is to split the complete rotation of the coordinate system into three simpler constitutive rotations, called precession, nutation, and intrinsic rotation, being each one of them an increment on one of the Euler angles. Notice that the outer matrix will represent a rotation around one of the axes of the reference frame, and the inner matrix represents a rotation around one of the moving frame axes. The middle matrix represents a rotation around an intermediate axis called **line of nodes**.

However, the definition of Euler angles is not unique and in the literature many different conventions are used. These conventions depend on the axes about which the rotations are carried out, and their sequence (since rotations are not commutative).

The convention being used is usually indicated by specifying the axes about which the consecutive rotations (before being composed) take place, referring to them by index (1, 2, 3) or letter (X, Y, Z). The engineering and robotics communities typically use 3-1-3 Euler angles. Notice that after composing the independent rotations, they do not rotate about their axis anymore. The most external matrix rotates the other two, leaving the second rotation matrix over the line of nodes, and the third one in a frame comoving with the body. There are $3 \times 3 \times 3 = 27$ possible combinations of three basic rotations but only $3 \times 2 \times 2 = 12$ of them can be used for representing arbitrary 3D rotations as Euler angles. These 12 combinations avoid consecutive rotations around the same axis (such as XXY) which would reduce the degrees of freedom that can be represented.

Therefore, Euler angles are never expressed in terms of the external frame, or in terms of the co-moving rotated body frame, but in a mixture. Other conventions (e.g., rotation matrix or quaternions) are used to avoid this problem.

In aviation orientation of the aircraft is usually expressed as intrinsic Tait-Bryan angles following the z-y'-x'' convention, which are called ***heading***, ***elevation***, and ***bank*** (or synonymously, ***yaw***, ***pitch***, and ***roll***).

Quaternions

Quaternions, which form a four-dimensional vector space, have proven very useful in representing rotations due to several advantages over the other representations mentioned in this article.

A quaternion representation of rotation is written as a versor (normalized quaternion):

$$\hat{\mathbf{q}} = q_i \mathbf{i} + q_j \mathbf{j} + q_k \mathbf{k} + q_r = \begin{bmatrix} q_i \\ q_j \\ q_k \\ q_r \end{bmatrix}$$

The above definition stores the quaternion as an array following the convention used in (Wertz 1980) and (Markley 2003). An alternative definition, used for example in (Coutsias 1999) and (Schmidt 2001), defines the "scalar" term as the first quaternion element, with the other elements shifted down one position.

In terms of the Euler axis

$$\hat{\mathbf{e}} = \begin{bmatrix} e_x \\ e_y \\ e_z \end{bmatrix}$$

and angle θ this versor's components are expressed as follows:

$$\begin{aligned} q_i &= e_x \sin \frac{\theta}{2} \\ q_j &= e_y \sin \frac{\theta}{2} \\ q_k &= e_z \sin \frac{\theta}{2} \\ q_r &= \cos \frac{\theta}{2} \end{aligned}$$

Inspection shows that the quaternion parametrization obeys the following constraint:

$$q_i^2 + q_j^2 + q_k^2 + q_r^2 = 1$$

The last term (in our definition) is often called the scalar term, which has its origin in quaternions when understood as the mathematical extension of the complex numbers, written as

$$a + bi + cj + dk \qquad \text{with } a, b, c, d \in \mathbb{R}$$

and where $\{i, j, k\}$ are the hypercomplex numbers satisfying

$$
\begin{aligned}
i^2 &= j^2 = k^2 = -1 \\
ij &= -ji = k \\
jk &= -kj = i \\
ki &= -ik = j
\end{aligned}
$$

Quaternion multiplication, which is used to specify a composite rotation, is performed in the same manner as multiplication of complex numbers, except that the order of the elements must be taken into account, since multiplication is not commutative. In matrix notation we can write quaternion multiplication as

$$
\tilde{\mathbf{q}} \otimes \mathbf{q} = \begin{bmatrix} q_r & q_k & -q_j & q_i \\ -q_k & q_r & q_i & q_j \\ q_j & -q_i & q_r & q_k \\ -q_i & -q_j & -q_k & q_r \end{bmatrix} \begin{bmatrix} \tilde{q}_i \\ \tilde{q}_j \\ \tilde{q}_k \\ \tilde{q}_r \end{bmatrix} = \begin{bmatrix} \tilde{q}_r & -\tilde{q}_k & \tilde{q}_j & \tilde{q}_i \\ \tilde{q}_k & \tilde{q}_r & -\tilde{q}_i & \tilde{q}_j \\ -\tilde{q}_j & \tilde{q}_i & \tilde{q}_r & \tilde{q}_k \\ -\tilde{q}_i & -\tilde{q}_j & -\tilde{q}_k & \tilde{q}_r \end{bmatrix} \begin{bmatrix} q_i \\ q_j \\ q_k \\ q_r \end{bmatrix}
$$

Combining two consecutive quaternion rotations is therefore just as simple as using the rotation matrix. Just as two successive rotation matrices, $\mathbf{A}_1$ followed by $\mathbf{A}_2$, are combined as

$$\mathbf{A}_3 = \mathbf{A}_2 \mathbf{A}_1,$$

we can represent this with quaternion parameters in a similarly concise way:

$$\mathbf{q}_3 = \mathbf{q}_2 \otimes \mathbf{q}_1$$

Quaternions are a very popular parametrization due to the following properties:

- More compact than the matrix representation and less susceptible to round-off errors
- The quaternion elements vary continuously over the unit sphere in $\mathbb{R}^4$, (denoted by S^3) as the orientation changes, avoiding discontinuous jumps (inherent to three-dimensional parameterizations)
- Expression of the rotation matrix in terms of quaternion parameters involves no trigonometric functions
- It is simple to combine two individual rotations represented as quaternions using a quaternion product

Like rotation matrices, quaternions must sometimes be renormalized due to rounding errors, to make sure that they correspond to valid rotations. The computational cost of renormalizing a quaternion, however, is much less than for normalizing a 3 × 3 matrix.

Quaternions also capture the spinorial character of rotations in three dimensions. For a three-dimensional object connected to its (fixed) surroundings by slack strings or bands, the strings or bands can be untangled after *two* complete turns about some fixed axis from an initial untangled state. Algebraically, the quaternion describing such a rotation changes from a scalar +1 (initially), through (scalar + pseudovector) values to scalar −1 (at one full turn), through (scalar + pseudovector)

values back to scalar +1 (at two full turns). This cycle repeats every 2 turns. After $2n$ turns (integer $n > 0$), without any intermediate untangling attempts, the strings/bands can be partially untangled back to the $2(n-1)$ turns state with each application of the same procedure used in untangling from 2 turns to 0 turns. Applying the same procedure n times will take a $2n$-tangled object back to the untangled or 0 turn state. The untangling process also removes any rotation-generated twisting about the strings/bands themselves. Simple 3D mechanical models can be used to demonstrate these facts.

Rodrigues vector

The **Rodrigues vector** (sometimes called the **Gibbs vector**, with coordinates called **Rodrigues parameters**)[3][4] can be expressed in terms of the axis and angle of the rotation as follows:

$$\mathbf{g} = \hat{\mathbf{e}} \tan \frac{\theta}{2}$$

This representation is a higher-dimensional analog of the gnomonic projection, mapping unit quaternions from a 3-sphere onto the 3-dimensional pure-vector hyperplane.

It has a discontinuity at 180° (π radians): as any rotation vector $\mathbf{r}$ tends to an angle of π radians, its tangent tends to infinity.

A rotation $\mathbf{g}$ followed by a rotation $\mathbf{f}$ in the Rodrigues representation has the simple rotation composition form

$$(\mathbf{g}, \mathbf{f}) = \frac{\mathbf{g} + \mathbf{f} - \mathbf{f} \times \mathbf{g}}{1 - \mathbf{g} \cdot \mathbf{f}}.$$

Today, the most straightforward way to prove this formula is in the (faithful) doublet representation, where $\mathbf{g} = \hat{\mathbf{n}} \tan a$, etc.

The combinatoric features of the Pauli matrix derivation just mentioned are also identical to the equivalent quaternion derivation below. Construct a quaternion associated with a spatial rotation R as,

$$S = \cos \frac{\phi}{2} + \sin \frac{\phi}{2} \mathbf{S}.$$

Then the composition of the rotation R_B with R_A is the rotation $R_C = R_B R_A$, with rotation axis and angle defined by the product of the quaternions,

$$A = \cos \frac{\alpha}{2} + \sin \frac{\alpha}{2} \mathbf{A} \quad \text{and} \quad B = \cos \frac{\beta}{2} + \sin \frac{\beta}{2} \mathbf{B},$$

that is

$$C = \cos \frac{\gamma}{2} + \sin \frac{\gamma}{2} \mathbf{C} = \left(\cos \frac{\beta}{2} + \sin \frac{\beta}{2} \mathbf{B} \right) \left(\cos \frac{\alpha}{2} + \sin \frac{\alpha}{2} \mathbf{A} \right).$$

Expand this quaternion product to

$$\cos\frac{\gamma}{2}+\sin\frac{\gamma}{2}\mathbf{C}=\left(\cos\frac{\beta}{2}\cos\frac{\alpha}{2}-\sin\frac{\beta}{2}\sin\frac{\alpha}{2}\mathbf{B}\cdot\mathbf{A}\right)+\left(\sin\frac{\beta}{2}\cos\frac{\alpha}{2}\mathbf{B}+\sin\frac{\alpha}{2}\cos\frac{\beta}{2}\mathbf{A}+\right.$$

Divide both sides of this equation by the identity resulting from the previous one,

$$\cos\frac{\gamma}{2}=\cos\frac{\beta}{2}\cos\frac{\alpha}{2}-\sin\frac{\beta}{2}\sin\frac{\alpha}{2}\mathbf{B}\cdot\mathbf{A},$$

and evaluate

$$\tan\frac{\gamma}{2}\mathbf{C}=\frac{\tan\frac{\beta}{2}\mathbf{B}+\tan\frac{\alpha}{2}\mathbf{A}+\tan\frac{\beta}{2}\tan\frac{\alpha}{2}\mathbf{B}\times\mathbf{A}}{1-\tan\frac{\beta}{2}\tan\frac{\alpha}{2}\mathbf{B}\cdot\mathbf{A}}.$$

This is Rodrigues' formula for the axis of a composite rotation defined in terms of the axes of the two component rotations. He derived this formula in 1840 (see page 408).[3] The three rotation axes **A**, **B**, and **C** form a spherical triangle and the dihedral angles between the planes formed by the sides of this triangle are defined by the rotation angles.

Modified Rodrigues parameters (MRPs) can be expressed in terms of Euler axis and angle by

$$\mathbf{p}=\hat{\mathbf{e}}\tan\frac{\theta}{4}\,.$$

Its components can be expressed in terms of the components of a unit quaternion representing the same rotation as

$$p_{x,y,z}=\frac{q_{i,j,k}}{1+q_r}\,.$$

The modified Rodrigues vector is a stereographic projection mapping unit quaternions from a 3-sphere onto the 3-dimensional pure-vector hyperplane. The projection of the opposite quaternion **-q** results in a different modified Rodrigues vector $\mathbf{p}^s$ than the projection of the original quaternion **q**. Comparing components one obtains that

$$p^s_{x,y,z}=\frac{-q_{i,j,k}}{1-q_r}=\frac{-p_{x,y,z}}{\mathbf{p}^2}\,.$$

Notably, if one of these vectors lies inside the unit 3-sphere, the other will lie outside.

Cayley–Klein parameters

See definition at Wolfram Mathworld (http://mathworld.wolfram.com/Cayley-KleinParameters.html).

Higher-dimensional analogues

Vector transformation law

Active rotations of a 3D vector **p** in Euclidean space around an axis **n** over an angle η can be easily written in terms of dot and cross products as follows:

$$\mathbf{p}' = p_{\parallel}\mathbf{n} + \cos\eta\,\mathbf{p}_{\perp} + \sin\eta\,\mathbf{p}\wedge\mathbf{n}$$

wherein

$$p_{\parallel} = \mathbf{p}\cdot\mathbf{n}$$

is the longitudinal component of **p** along **n**, given by the dot product,

$$\mathbf{p}_{\perp} = \mathbf{p} - (\mathbf{p}\cdot\mathbf{n})\mathbf{n}$$

is the transverse component of **p** with respect to **n**, and

$$\mathbf{p}\wedge\mathbf{n}$$

is the cross product, of **p** with **n**.

The above formula shows that the longitudinal component of **p** remains unchanged, whereas the transverse portion of **p** is rotated in the plane perpendicular to **n**. This plane is spanned by the transverse portion of **p** itself and a direction perpendicular to both **p** and **n**. The rotation is directly identifiable in the equation as a 2D rotation over an angle η.

Passive rotations can be described by the same formula, but with an inverse sign of either η or **n**.

Conversion formulae between formalisms

Rotation matrix ↔ Euler angles

The Euler angles (φ, θ, ψ) can be extracted from the rotation matrix $\mathbf{A}$ by inspecting the rotation matrix in analytical form.

Rotation matrix → Euler angles (*z*-*x*-*z* extrinsic)

Using the *x*-convention, the 3-1-3 extrinsic Euler angles φ, θ and ψ (around the *z*-axis, *x*-axis and again the *Z*-axis) can be obtained as follows:

$$\phi = \operatorname{atan2}(A_{31}, A_{32})$$
$$\theta = \arccos(A_{33})$$
$$\psi = -\operatorname{atan2}(A_{13}, A_{23})$$

Note that atan2(a, b) is equivalent to arctan $\frac{a}{b}$ where it also takes into account the quadrant that the point (b, a) is in; see atan2.

When implementing the conversion, one has to take into account several situations:[5]

- There are generally two solutions in the interval $[-\pi, \pi]^3$. The above formula works only when θ is within the interval $[0, \pi]$.
- For the special case $A_{33} = 0$, φ and ψ will be derived from A_{11} and A_{12}.
- There are infinitely many but countably many solutions outside of the interval $[-\pi, \pi]^3$.
- Whether all mathematical solutions apply for a given application depends on the situation.

Euler angles (z-y'-x'' intrinsic) → rotation matrix

The rotation matrix $\mathbf{A}$ is generated from the 3-2-1 intrinsic Euler angles by multiplying the three matrices generated by rotations about the axes.

$$\mathbf{A} = \mathbf{A}_3\mathbf{A}_2\mathbf{A}_1 = \mathbf{A}_Z\mathbf{A}_Y\mathbf{A}_X$$

The axes of the rotation depend on the specific convention being used. For the x-convention the rotations are about the x-, y- and z-axes with angles ϕ, θ and ψ, the individual matrices are as follows:

$$\mathbf{A}_X = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi & \cos\phi \end{bmatrix}$$
$$\mathbf{A}_Y = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix}$$
$$\mathbf{A}_Z = \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

This yields

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

Note: This is valid for a right-hand system, which is the convention used in almost all engineering and physics disciplines.

The interpretation of these right-handed rotation matrices is that they express coordinate transformations (passive) as opposed to point transformations (active). Because $\mathbf{A}$ expresses a rotation from the local frame $\mathbf{1}$ to the global frame $\mathbf{0}$ (i.e., $\mathbf{A}$ encodes the axes of frame $\mathbf{1}$ w.r.t frame $\mathbf{0}$), the elementary rotation matrices are composed as above. Because the inverse rotation is just the rotation transposed, if we wanted the global-to-local rotation from frame $\mathbf{0}$ to frame $\mathbf{1}$, we would write $\mathbf{A}^\top = (\mathbf{A}_Z\mathbf{A}_Y\mathbf{A}_X)^\top = \mathbf{A}_X^\top\mathbf{A}_Y^\top\mathbf{A}_Z^\top$.

Rotation matrix ↔ Euler axis/angle

If the Euler angle θ is not a multiple of π, the Euler axis $\hat{\mathbf{e}}$ and angle θ can be computed from the elements of the rotation matrix A as follows:

$$\theta = \arccos\frac{A_{11} + A_{22} + A_{33} - 1}{2}$$
$$e_1 = \frac{A_{32} - A_{23}}{2\sin\theta}$$
$$e_2 = \frac{A_{13} - A_{31}}{2\sin\theta}$$
$$e_3 = \frac{A_{21} - A_{12}}{2\sin\theta}$$

Alternatively, the following method can be used:

Eigendecomposition of the rotation matrix yields the eigenvalues 1 and $\cos\theta \pm i\sin\theta$. The Euler axis is the eigenvector corresponding to the eigenvalue of 1, and θ can be computed from the remaining eigenvalues.

The Euler axis can be also found using singular value decomposition since it is the normalized vector spanning the null-space of the matrix $\mathbf{I} - \mathbf{A}$.

To convert the other way the rotation matrix corresponding to an Euler axis $\hat{\mathbf{e}}$ and angle θ can be computed according to Rodrigues' rotation formula (with appropriate modification) as follows:

$$\mathbf{A} = \mathbf{I}_3\cos\theta + (1 - \cos\theta)\hat{\mathbf{e}}\hat{\mathbf{e}}^\mathsf{T} + [\hat{\mathbf{e}}]_\times\sin\theta$$

with $\mathbf{I}_3$ the 3 × 3 identity matrix, and

$$[\hat{\mathbf{e}}]_\times = \begin{bmatrix} 0 & -e_3 & e_2 \\ e_3 & 0 & -e_1 \\ -e_2 & e_1 & 0 \end{bmatrix}$$

is the cross-product matrix.

This expands to:

$$\begin{aligned}
A_{11} &= (1-\cos\theta)e_1^2 + \cos\theta \\
A_{12} &= (1-\cos\theta)e_1 e_2 - e_3 \sin\theta \\
A_{13} &= (1-\cos\theta)e_1 e_3 + e_2 \sin\theta \\
A_{21} &= (1-\cos\theta)e_2 e_1 + e_3 \sin\theta \\
A_{22} &= (1-\cos\theta)e_2^2 + \cos\theta \\
A_{23} &= (1-\cos\theta)e_2 e_3 - e_1 \sin\theta \\
A_{31} &= (1-\cos\theta)e_3 e_1 - e_2 \sin\theta \\
A_{32} &= (1-\cos\theta)e_3 e_2 + e_1 \sin\theta \\
A_{33} &= (1-\cos\theta)e_3^2 + \cos\theta
\end{aligned}$$

Rotation matrix ↔ quaternion

When computing a quaternion from the rotation matrix there is a sign ambiguity, since $\mathbf{q}$ and $-\mathbf{q}$ represent the same rotation.

One way of computing the quaternion

$$\mathbf{q} = \begin{bmatrix} q_i \\ q_j \\ q_k \\ q_r \end{bmatrix} = q_i \mathbf{i} + q_j \mathbf{j} + q_k \mathbf{k} + q_r$$

from the rotation matrix $\mathbf{A}$ is as follows:

$$\begin{aligned}
q_r &= \frac{1}{2}\sqrt{1 + A_{11} + A_{22} + A_{33}} \\
q_i &= \frac{1}{4q_r}(A_{32} - A_{23}) \\
q_j &= \frac{1}{4q_r}(A_{13} - A_{31}) \\
q_k &= \frac{1}{4q_r}(A_{21} - A_{12})
\end{aligned}$$

There are three other mathematically equivalent ways to compute $\mathbf{q}$. Numerical inaccuracy can be reduced by avoiding situations in which the denominator is close to zero. One of the other three methods looks as follows:[6]

$$
\begin{aligned}
q_i &= \frac{1}{2}\sqrt{1 + A_{11} - A_{22} - A_{33}} \\
q_j &= \frac{1}{4q_i}\left(A_{12} + A_{21}\right) \\
q_k &= \frac{1}{4q_i}\left(A_{13} + A_{31}\right) \\
q_r &= \frac{1}{4q_i}\left(A_{32} - A_{23}\right)
\end{aligned}
$$

The rotation matrix corresponding to the quaternion $\mathbf{q}$ can be computed as follows:

$$\mathbf{A} = \left(q_r^2 - \check{\mathbf{q}}^\mathsf{T}\check{\mathbf{q}}\right)\mathbf{I}_3 + 2\check{\mathbf{q}}\check{\mathbf{q}}^\mathrm{T} + 2q_r\mathcal{Q}$$

where

$$\check{\mathbf{q}} = \begin{bmatrix} q_i \\ q_j \\ q_k \end{bmatrix}, \quad \mathcal{Q} = \begin{bmatrix} 0 & -q_k & q_j \\ q_k & 0 & -q_i \\ -q_j & q_i & 0 \end{bmatrix}$$

which gives

$$\mathbf{A} = \begin{bmatrix} 1 - 2q_j^2 - 2q_k^2 & 2\left(q_i q_j - q_k q_r\right) & 2\left(q_i q_k + q_j q_r\right) \\ 2\left(q_i q_j + q_k q_r\right) & 1 - 2q_i^2 - 2q_k^2 & 2\left(q_j q_k - q_i q_r\right) \\ 2\left(q_i q_k - q_j q_r\right) & 2\left(q_j q_k + q_i q_r\right) & 1 - 2q_i^2 - 2q_j^2 \end{bmatrix}$$

or equivalently

$$\mathbf{A} = \begin{bmatrix} -1 + 2q_i^2 + 2q_r^2 & 2\left(q_i q_j - q_k q_r\right) & 2\left(q_i q_k + q_j q_r\right) \\ 2\left(q_i q_j + q_k q_r\right) & -1 + 2q_j^2 + 2q_r^2 & 2\left(q_j q_k - q_i q_r\right) \\ 2\left(q_i q_k - q_j q_r\right) & 2\left(q_j q_k + q_i q_r\right) & -1 + 2q_k^2 + 2q_r^2 \end{bmatrix}$$

Euler angles ↔ quaternion

Euler angles (z-x-z extrinsic) → quaternion

We will consider the x-convention 3-1-3 extrinsic Euler angles for the following algorithm. The terms of the algorithm depend on the convention used.

We can compute the quaternion

$$\mathbf{q} = \begin{bmatrix} q_i \\ q_j \\ q_k \\ q_r \end{bmatrix} = q_i\mathbf{i} + q_j\mathbf{j} + q_k\mathbf{k} + q_r$$

from the Euler angles (ϕ, θ, ψ) as follows:

$$\begin{aligned}
q_i &= \cos\frac{\phi-\psi}{2}\sin\frac{\theta}{2} \\
q_j &= \sin\frac{\phi-\psi}{2}\sin\frac{\theta}{2} \\
q_k &= \sin\frac{\phi+\psi}{2}\cos\frac{\theta}{2} \\
q_r &= \cos\frac{\phi+\psi}{2}\cos\frac{\theta}{2}
\end{aligned}$$

Euler angles (z-y'-x'' intrinsic) → quaternion

A quaternion equivalent to yaw (ψ), pitch (θ) and roll (ϕ) angles. or intrinsic Tait–Bryan angles following the z-y'-x'' convention, can be computed by

$$\begin{aligned}
q_i &= \sin\frac{\phi}{2}\cos\frac{\theta}{2}\cos\frac{\psi}{2} - \cos\frac{\phi}{2}\sin\frac{\theta}{2}\sin\frac{\psi}{2} \\
q_j &= \cos\frac{\phi}{2}\sin\frac{\theta}{2}\cos\frac{\psi}{2} + \sin\frac{\phi}{2}\cos\frac{\theta}{2}\sin\frac{\psi}{2} \\
q_k &= \cos\frac{\phi}{2}\cos\frac{\theta}{2}\sin\frac{\psi}{2} - \sin\frac{\phi}{2}\sin\frac{\theta}{2}\cos\frac{\psi}{2} \\
q_r &= \cos\frac{\phi}{2}\cos\frac{\theta}{2}\cos\frac{\psi}{2} + \sin\frac{\phi}{2}\sin\frac{\theta}{2}\sin\frac{\psi}{2}
\end{aligned}$$

Quaternion → Euler angles (z-x-z extrinsic)

Given the rotation quaternion

$$\mathbf{q} = \begin{bmatrix} q_i \\ q_j \\ q_k \\ q_r \end{bmatrix} = q_i \mathbf{i} + q_j \mathbf{j} + q_k \mathbf{k} + q_r \,,$$

the x-convention 3-1-3 extrinsic Euler Angles (φ, θ, ψ) can be computed by

$$\begin{aligned}
\phi &= \operatorname{atan2}\left(\left(q_i q_k + q_j q_r\right), -\left(q_j q_k - q_i q_r\right)\right) \\
\theta &= \arccos\left(-q_i^2 - q_j^2 + q_k^2 + q_r^2\right) \\
\psi &= \operatorname{atan2}\left(\left(q_i q_k - q_j q_r\right), \left(q_j q_k + q_i q_r\right)\right)
\end{aligned}$$

Quaternion → Euler angles (z-y'-x'' intrinsic)

Given the rotation quaternion

$$\mathbf{q} = \begin{bmatrix} q_i \\ q_j \\ q_k \\ q_r \end{bmatrix} = q_i \mathbf{i} + q_j \mathbf{j} + q_k \mathbf{k} + q_r \,,$$

yaw, pitch and roll angles, or intrinsic Tait–Bryan angles following the z-y'-x'' convention, can be computed by

$$\begin{aligned}
\mathrm{roll} &= \operatorname{atan2}\left(2\left(q_r q_i + q_j q_k\right), 1 - 2\left(q_i^2 + q_j^2\right)\right) \\
\mathrm{pitch} &= \arcsin\left(2\left(q_r q_j - q_k q_i\right)\right) \\
\mathrm{yaw} &= \operatorname{atan2}\left(2\left(q_r q_k + q_i q_j\right), 1 - 2\left(q_j^2 + q_k^2\right)\right)
\end{aligned}$$

Euler axis–angle ↔ quaternion

Given the Euler axis $\hat{\mathbf{e}}$ and angle θ, the quaternion

$$\mathbf{q} = \begin{bmatrix} q_i \\ q_j \\ q_k \\ q_r \end{bmatrix} = q_i \mathbf{i} + q_j \mathbf{j} + q_k \mathbf{k} + q_r \,,$$

can be computed by

$$
\begin{aligned}
q_i &= \hat{e}_1 \sin\frac{\theta}{2} \\
q_j &= \hat{e}_2 \sin\frac{\theta}{2} \\
q_k &= \hat{e}_3 \sin\frac{\theta}{2} \\
q_r &= \cos\frac{\theta}{2}
\end{aligned}
$$

Given the rotation quaternion $\mathbf{q}$, define

$$
\check{\mathbf{q}} = \begin{bmatrix} q_i \\ q_j \\ q_k \end{bmatrix}.
$$

Then the Euler axis $\hat{\mathbf{e}}$ and angle θ can be computed by

$$
\begin{aligned}
\hat{\mathbf{e}} &= \frac{\check{\mathbf{q}}}{\|\check{\mathbf{q}}\|} \\
\theta &= 2\arccos q_r
\end{aligned}
$$

Rotation matrix ↔ Rodrigues vector

Rodrigues vector → Rotation matrix

Since the definition of the Rodrigues vector can be related to rotation quaternions:

$$
\begin{cases}
g_i = \dfrac{q_i}{q_r} = e_x \tan\left(\dfrac{\theta}{2}\right) \\
g_j = \dfrac{q_j}{q_r} = e_y \tan\left(\dfrac{\theta}{2}\right) \\
g_k = \dfrac{q_k}{q_r} = e_z \tan\left(\dfrac{\theta}{2}\right)
\end{cases}
$$

By making use of the following property

$$
1 = q_r^2 + q_i^2 + q_j^2 + q_k^2 = q_r^2 \left(1 + \frac{q_i^2}{q_r^2} + \frac{q_j^2}{q_r^2} + \frac{q_k^2}{q_r^2}\right) = q_r^2 \left(1 + g_i^2 + g_j^2 + g_k^2\right)
$$

The formula can be obtained by factoring q_r^2 from the final expression obtained for quaternions:

$$\mathbf{A} = q_r^2 \begin{bmatrix} \frac{1}{q_r^2} - 2\frac{q_j^2}{q_r^2} - 2\frac{q_k^2}{q_r^2} & 2\left(\frac{q_i}{q_r}\frac{q_j}{q_r} - \frac{q_k}{q_r}\right) & 2\left(\frac{q_i}{q_r}\frac{q_k}{q_r} + \frac{q_j}{q_r}\right) \\ 2\left(\frac{q_i}{q_r}\frac{q_j}{q_r} + \frac{q_k}{q_r}\right) & \frac{1}{q_r^2} - 2\frac{q_i^2}{q_r^2} - 2\frac{q_k^2}{q_r^2} & 2\left(\frac{q_j}{q_r}\frac{q_k}{q_r} - \frac{q_i}{q_r}\right) \\ 2\left(\frac{q_i}{q_r}\frac{q_k}{q_r} - \frac{q_j}{q_r}\right) & 2\left(\frac{q_j}{q_r}\frac{q_k}{q_r} + \frac{q_i}{q_r}\right) & \frac{1}{q_r^2} - 2\frac{q_i^2}{q_r^2} - 2\frac{q_j^2}{q_r^2} \end{bmatrix}$$

Leading to the final formula:

$$\mathbf{A} = \frac{1}{1 + g_i^2 + g_j^2 + g_k^2} \begin{bmatrix} 1 + g_i^2 - g_j^2 - g_k^2 & 2\left(g_i g_j - g_k\right) & 2\left(g_i g_k + g_j\right) \\ 2\left(g_i g_j + g_k\right) & 1 - g_i^2 + g_j^2 - g_k^2 & 2\left(g_j g_k - g_i\right) \\ 2\left(g_i g_k - g_j\right) & 2\left(g_j g_k + g_i\right) & 1 - g_i^2 - g_j^2 + g_k^2 \end{bmatrix}$$

Conversion formulae for derivatives

Rotation matrix ↔ angular velocities

The angular velocity vector

$$\boldsymbol{\omega} = \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix}$$

can be extracted from the time derivative of the rotation matrix $\frac{\mathrm{d}\mathbf{A}}{\mathrm{d}t}$ by the following relation:

$$[\boldsymbol{\omega}]_\times = \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix} = \frac{\mathrm{d}\mathbf{A}}{\mathrm{d}t}\mathbf{A}^\mathsf{T}$$

The derivation is adapted from Ioffe[7] as follows:

For any vector $\mathbf{r}_0$, consider $\mathbf{r}(t) = \mathbf{A}(t)\mathbf{r}_0$ and differentiate it:

$$\frac{\mathrm{d}\mathbf{r}}{\mathrm{d}t} = \frac{\mathrm{d}\mathbf{A}}{\mathrm{d}t}\mathbf{r}_0 = \frac{\mathrm{d}\mathbf{A}}{\mathrm{d}t}\mathbf{A}^\mathsf{T}(t)\mathbf{r}(t)$$

The derivative of a vector is the linear velocity of its tip. Since $\mathbf{A}$ is a rotation matrix, by definition the length of $\mathbf{r}(t)$ is always equal to the length of $\mathbf{r}_0$, and hence it does not change with time. Thus, when $\mathbf{r}(t)$ rotates, its tip moves along a circle, and the linear velocity of its tip is tangential to the circle; i.e., always perpendicular to $\mathbf{r}(t)$. In this specific case, the relationship between the linear velocity vector and the angular velocity vector is

$$\frac{\mathrm{d}\mathbf{r}}{\mathrm{d}t} = \boldsymbol{\omega}(t) \times \mathbf{r}(t) = [\boldsymbol{\omega}]_\times \mathbf{r}(t)$$

(see circular motion and cross product).

By the transitivity of the abovementioned equations,

$$\frac{\mathrm{d}\mathbf{A}}{\mathrm{d}t}\mathbf{A}^\mathsf{T}(t)\mathbf{r}(t) = [\boldsymbol{\omega}]_\times \mathbf{r}(t)$$

which implies

$$\frac{\mathrm{d}\mathbf{A}}{\mathrm{d}t}\mathbf{A}^\mathsf{T}(t) = [\boldsymbol{\omega}]_\times$$

Quaternion ↔ angular velocities

The angular velocity vector

$$\boldsymbol{\omega} = \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix}$$

can be obtained from the derivative of the quaternion $\frac{\mathrm{d}\mathbf{q}}{\mathrm{d}t}$ as follows:[8]

$$\begin{bmatrix} 0 \\ \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} = 2\frac{\mathrm{d}\mathbf{q}}{\mathrm{d}t}\tilde{\mathbf{q}}$$

where $\tilde{\mathbf{q}}$ is the conjugate (inverse) of $\mathbf{q}$.

Conversely, the derivative of the quaternion is

$$\frac{\mathrm{d}\mathbf{q}}{\mathrm{d}t} = \frac{1}{2}\begin{bmatrix} 0 \\ \omega_x \\ \omega_y \\ \omega_z \end{bmatrix}\mathbf{q}\,.$$

Rotors in a geometric algebra

The formalism of geometric algebra (GA) provides an extension and interpretation of the quaternion method. Central to GA is the geometric product of vectors, an extension of the traditional inner and cross products, given by

$$\mathbf{ab} = \mathbf{a} \cdot \mathbf{b} + \mathbf{a} \wedge \mathbf{b}$$

where the symbol $\wedge$ denotes the exterior product or wedge product. This product of vectors $\mathbf{a}$, and $\mathbf{b}$ produces two terms: a scalar part from the inner product and a bivector part from the wedge product. This bivector describes the plane perpendicular to what the cross product of the vectors would return.

Bivectors in GA have some unusual properties compared to vectors. Under the geometric product, bivectors have a negative square: the bivector $\hat{\mathbf{x}}\hat{\mathbf{y}}$ describes the *xy*-plane. Its square is $(\hat{\mathbf{x}}\hat{\mathbf{y}})^2 = \hat{\mathbf{x}}\hat{\mathbf{y}}\hat{\mathbf{x}}\hat{\mathbf{y}}$. Because the unit basis vectors are orthogonal to each other, the geometric product reduces to the antisymmetric outer product – $\hat{\mathbf{x}}$ and $\hat{\mathbf{y}}$ can be swapped freely at the cost of a factor of −1. The square reduces to $-\hat{\mathbf{x}}\hat{\mathbf{x}}\hat{\mathbf{y}}\hat{\mathbf{y}} = -1$ since the basis vectors themselves square to +1.

This result holds generally for all bivectors, and as a result the bivector plays a role similar to the imaginary unit. Geometric algebra uses bivectors in its analogue to the quaternion, the *rotor*, given by

$$\mathbf{R} = \exp\left(\frac{-\hat{\mathbf{B}}\theta}{2}\right) = \cos\frac{\theta}{2} - \hat{\mathbf{B}}\sin\frac{\theta}{2}\,,$$

where $\hat{\mathbf{B}}$ is a unit bivector that describes the plane of rotation. Because $\hat{\mathbf{B}}$ squares to −1, the power series expansion of $\mathbf{R}$ generates the trigonometric functions. The rotation formula that maps a vector $\mathbf{a}$ to a rotated vector $\mathbf{b}$ is then

$$\mathbf{b} = \mathbf{R}\mathbf{a}\mathbf{R}^\dagger$$

where

$$\mathbf{R}^\dagger = \exp\left(\frac{1}{2}\hat{\mathbf{B}}\theta\right) = \cos\frac{\theta}{2} + \hat{\mathbf{B}}\sin\frac{\theta}{2}$$

is the *reverse* of R (reversing the order of the vectors in B is equivalent to changing its sign).

Example. A rotation about the axis

$$\hat{\mathbf{v}} = \frac{1}{\sqrt{3}}\left(\hat{\mathbf{x}} + \hat{\mathbf{y}} + \hat{\mathbf{z}}\right)$$

can be accomplished by converting $\hat{\mathbf{v}}$ to its dual bivector,

$$\hat{\mathbf{B}} = \hat{\mathbf{x}}\hat{\mathbf{y}}\hat{\mathbf{z}}\hat{\mathbf{v}} = \mathbf{i}\hat{\mathbf{v}}\,,$$

where $\mathbf{i} = \hat{\mathbf{x}}\hat{\mathbf{y}}\hat{\mathbf{z}}$ is the unit volume element, the only trivector (pseudoscalar) in three-dimensional space. The result is

$$\hat{\mathbf{B}} = \frac{1}{\sqrt{3}}\left(\hat{\mathbf{y}}\hat{\mathbf{z}} + \hat{\mathbf{z}}\hat{\mathbf{x}} + \hat{\mathbf{x}}\hat{\mathbf{y}}\right) .$$

In three-dimensional space, however, it is often simpler to leave the expression for $\hat{\mathbf{B}} = \mathbf{i}\hat{\mathbf{v}}$, using the fact that $\mathbf{i}$ commutes with all objects in 3D and also squares to −1. A rotation of the $\hat{\mathbf{x}}$ vector in this plane by an angle θ is then

$$\hat{\mathbf{x}}' = \mathbf{R}\hat{\mathbf{x}}\mathbf{R}^\dagger = e^{-i\hat{\mathbf{v}}\frac{\theta}{2}}\hat{\mathbf{x}}e^{i\hat{\mathbf{v}}\frac{\theta}{2}} = \hat{\mathbf{x}}\cos^2\frac{\theta}{2} + \mathbf{i}\left(\hat{\mathbf{x}}\hat{\mathbf{v}} - \hat{\mathbf{v}}\hat{\mathbf{x}}\right)\cos\frac{\theta}{2}\sin\frac{\theta}{2} + \hat{\mathbf{v}}\hat{\mathbf{x}}\hat{\mathbf{v}}\sin^2\frac{\theta}{2}$$

Recognizing that

$$\mathbf{i}(\hat{\mathbf{x}}\hat{\mathbf{v}} - \hat{\mathbf{v}}\hat{\mathbf{x}}) = 2\mathbf{i}(\hat{\mathbf{x}} \wedge \hat{\mathbf{v}})$$

and that $-\hat{\mathbf{v}}\hat{\mathbf{x}}\hat{\mathbf{v}}$ is the reflection of $\hat{\mathbf{x}}$ about the plane perpendicular to $\hat{\mathbf{v}}$ gives a geometric interpretation to the rotation operation: the rotation preserves the components that are parallel to $\hat{\mathbf{v}}$ and changes only those that are perpendicular. The terms are then computed:

$$\hat{\mathbf{v}}\hat{\mathbf{x}}\hat{\mathbf{v}} = \frac{1}{3}\left(-\hat{\mathbf{x}} + 2\hat{\mathbf{y}} + 2\hat{\mathbf{z}}\right)$$
$$2\mathbf{i}\hat{\mathbf{x}} \wedge \hat{\mathbf{v}} = 2\mathbf{i}\frac{1}{\sqrt{3}}\left(\hat{\mathbf{x}}\hat{\mathbf{y}} + \hat{\mathbf{x}}\hat{\mathbf{z}}\right) = \frac{2}{\sqrt{3}}\left(\hat{\mathbf{y}} - \hat{\mathbf{z}}\right)$$

The result of the rotation is then

$$\hat{\mathbf{x}}' = \hat{\mathbf{x}}\left(\cos^2\frac{\theta}{2} - \frac{1}{3}\sin^2\frac{\theta}{2}\right) + \frac{2}{3}\hat{\mathbf{y}}\sin\frac{\theta}{2}\left(\sin\frac{\theta}{2} + \sqrt{3}\cos\frac{\theta}{2}\right) + \frac{2}{3}\hat{\mathbf{z}}\sin\frac{\theta}{2}\left(\sin\frac{\theta}{2} - \sqrt{3}\cos\right.$$

A simple check on this result is the angle $\theta = \frac{2}{3}\pi$. Such a rotation should map $\hat{\mathbf{x}}$ to $\hat{\mathbf{y}}$. Indeed, the rotation reduces to

$$\begin{aligned}\hat{\mathbf{x}}' &= \hat{\mathbf{x}}\left(\frac{1}{4} - \frac{1}{3}\frac{3}{4}\right) + \frac{2}{3}\hat{\mathbf{y}}\frac{\sqrt{3}}{2}\left(\frac{\sqrt{3}}{2} + \sqrt{3}\frac{1}{2}\right) + \frac{2}{3}\hat{\mathbf{z}}\frac{\sqrt{3}}{2}\left(\frac{\sqrt{3}}{2} - \sqrt{3}\frac{1}{2}\right) \\ &= 0\hat{\mathbf{x}} + \hat{\mathbf{y}} + 0\hat{\mathbf{z}} = \hat{\mathbf{y}}\end{aligned}$$

exactly as expected. This rotation formula is valid not only for vectors but for any multivector. In addition, when Euler angles are used, the complexity of the operation is much reduced. Compounded rotations come from multiplying the rotors, so the total rotor from Euler angles is

$$\mathbf{R} = \mathbf{R}_{\gamma'}\mathbf{R}_{\beta'}\mathbf{R}_{\alpha} = \exp\left(\frac{-\mathbf{i}\hat{\mathbf{z}}'\gamma}{2}\right)\exp\left(\frac{-\mathbf{i}\hat{\mathbf{x}}'\beta}{2}\right)\exp\left(\frac{-\mathbf{i}\hat{\mathbf{z}}\alpha}{2}\right)$$

but

$$\hat{\mathbf{x}}' = \mathbf{R}_{\alpha}\hat{\mathbf{x}}\mathbf{R}_{\alpha}^{\dagger} \quad \text{and}$$
$$\hat{\mathbf{z}}' = \mathbf{R}_{\beta'}\hat{\mathbf{z}}\mathbf{R}_{\beta'}^{\dagger}\ .$$

These rotors come back out of the exponentials like so:

$$\mathbf{R}_{\beta'} = \cos\frac{\beta}{2} - \mathbf{i}\mathbf{R}_{\alpha}\hat{\mathbf{x}}\mathbf{R}_{\alpha}^{\dagger}\sin\frac{\beta}{2} = \mathbf{R}_{\alpha}\mathbf{R}_{\beta}\mathbf{R}_{\alpha}^{\dagger}$$

where $\mathbf{R}_{\beta}$ refers to rotation in the original coordinates. Similarly for the γ rotation,

$$\mathbf{R}_{\gamma'} = \mathbf{R}_{\beta'}\mathbf{R}_{\gamma}\mathbf{R}_{\beta'}^{\dagger} = \mathbf{R}_{\alpha}\mathbf{R}_{\beta}\mathbf{R}_{\alpha}^{\dagger}\mathbf{R}_{\gamma}\mathbf{R}_{\alpha}\mathbf{R}_{\beta}^{\dagger}\mathbf{R}_{\alpha}^{\dagger}\ .$$

Noting that $\mathbf{R}_{\gamma}$ and $\mathbf{R}_{\alpha}$ commute (rotations in the same plane must commute), and the total rotor becomes

$$\mathbf{R} = \mathbf{R}_{\alpha}\mathbf{R}_{\beta}\mathbf{R}_{\gamma}$$

Thus, the compounded rotations of Euler angles become a series of equivalent rotations in the original fixed frame.

While rotors in geometric algebra work almost identically to quaternions in three dimensions, the power of this formalism is its generality: this method is appropriate and valid in spaces with any number of dimensions. In 3D, rotations have three degrees of freedom, a degree for each linearly independent plane (bivector) the rotation can take place in. It has been known that pairs of quaternions can be used to generate rotations in 4D, yielding six degrees of freedom, and the geometric algebra approach verifies this result: in 4D, there are six linearly independent bivectors that can be used as the generators of rotations.

Angle-Angle-Angle

Rotations can be modeled as an axis and an angle; as illustrated with a gyroscope which has an axis through the rotor, and the amount of spin around that axis demonstrated by the rotation of the rotor; this rotation can be expressed as **angle ∗ (axis)** where axis is a unit vector specifying the direction of the rotor axis. From the origin, in any direction, is the same rotation axis, with the scale of the angle equivalent to the distance from the origin. From any other point in space, similarly the same direction vector applied relative to the orientation represented by the starting point rather than the origin applies the same change around the same axes that the unit vector specifies. The **angle ∗ axis** scaling each point gives a unique coordinate in Angle-Angle-Angle notation. The difference between two coordinates immediately yields the single axis of rotation and angle between the two orientations.

The natural log of a quaternion represents curving space by 3 angles around 3 axles of rotation, and is expressed in arc-length; similar to Euler angles, but order independent.[9] There is a Lie product formula definition of the addition of rotations, which is that they are sum of infinitesimal steps of

each rotation applied in series; this would imply that rotations are the result of all rotations in the same instant are applied, rather than a series of rotations applied subsequently.

The axes of rotation are aligned to the standard Cartesian X, Y, Z axes. These rotations may be simply added and subtracted, especially when the frames being rotated are fixed to each other as in IK chains. Differences between two objects that are in the same reference frame are found by simply subtracting their orientations. Rotations that are applied from external sources, or are from sources relative to the current rotation still require multiplications, application of the Rodriguez Formula is provided.

The rotation from each axle coordinate represent rotating the plane perpendicular to the specified axis simultaneously with all other axles. Although the measures can be considered in angles, the representation is actually the arc-length of the curve; an angle implies a rotation around a point, where a curvature is a delta applied to the current point in an inertial direction.

Just an observational note: log quaternions have rings, or octaves of rotations; that is for rotations greater than 4π have related curves. Curvatures of things that approach this boundary appear to chaotically jump orbits.

For 'human readable' angles the 1-norm can be used to rescale the angles to look more 'appropriate'; much like Celsius might be considered more correct than Fahrenheit.

$$Q = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$

Other related values are immediately derivable:

$$\|V\| \text{ or } \|V\|_2 = \sqrt{XX + YY + ZZ}$$
$$\|V\|_1 = |X| + |Y| + |Z|$$

The total angle of rotation....

$$\theta = \|V\|$$

The axis of rotation...

$$\text{Axis}(\ln Q) = \begin{bmatrix} \frac{X}{\theta} \\ \frac{Y}{\theta} \\ \frac{Z}{\theta} \end{bmatrix}$$

Quaternion Representation

$$q = \begin{bmatrix} \cos\frac{\theta}{2} \\ \sin\frac{\theta}{2}\frac{X}{\|Q\|} \\ \sin\frac{\theta}{2}\frac{Y}{\|Q\|} \\ \sin\frac{\theta}{2}\frac{Z}{\|Q\|} \end{bmatrix}$$

Basis Matrix Computation

This was built from rotating the vectors (1,0,0), (0,1,0), (0,0,1), and reducing constants.

Given an input $Q = [X, Y, Z]$,

$$\begin{aligned} q_r &= \cos\theta \\ q_i &= \sin\theta \cdot \frac{X}{\|Q\|} \\ q_j &= \sin\theta \cdot \frac{Y}{\|Q\|} \\ q_k &= \sin\theta \cdot \frac{Z}{\|Q\|} \end{aligned}$$

Which are used to compute the resulting matrix...

$$\begin{bmatrix} 1 - 2q_j^2 - 2q_k^2 & 2(q_i q_j - q_k q_r) & 2(q_i q_k + q_j q_r) \\ 2(q_i q_j + q_k q_r) & 1 - 2q_i^2 - 2q_k^2 & 2(q_j q_k - q_i q_r) \\ 2(q_i q_k - q_j q_r) & 2(q_j q_k + q_i q_r) & 1 - 2q_i^2 - 2q_j^2 \end{bmatrix}$$

Alternate Basis Calculation

Alternatively this can be used

given: $A = [X, Y, Z]$

convert to angle-axis $\theta = \|A\|$, and $[x, y, z] = \dfrac{A}{\|A\|}$

Compute some partial expressions:

$$\begin{aligned} x_y &= xy(1 - \cos\theta) & w_x &= x\sin\theta & x_x &= xx(1 - \cos\theta) \\ y_z &= yz(1 - \cos\theta) & w_y &= y\sin\theta & y_y &= yy(1 - \cos\theta) \\ x_z &= xz(1 - \cos\theta) & w_z &= z\sin\theta & z_z &= zz(1 - \cos\theta) \end{aligned}$$

Compute the resulting matrix:

$$\begin{bmatrix} \cos\theta + x_x & x_y + w_z & w_y + x_z \\ w_z + x_y & \cos\theta + y_y & y_z - w_x \\ x_z - w_y & w_x + y_z & \cos\theta + z_z \end{bmatrix}$$

Expanded:

$$\begin{bmatrix} \cos\theta + x^2(1-\cos\theta) & xy(1-\cos\theta) - z\sin\theta & y\sin\theta + xz(1-\cos\theta) \\ z\sin\theta + xy(1-\cos\theta) & \cos\theta + y^2(1-\cos\theta) & yz(1-\cos\theta) - x\sin\theta \\ xz(1-\cos\theta) - y\sin\theta & x\sin\theta + yz(1-\cos\theta) & \cos\theta + z^2(1-\cos\theta) \end{bmatrix}$$

Vector Rotation

Rotate the vector $v = (X, Y, Z)$ around the rotation vector $Q = (X, Y, Z)$.

The angle of rotation will be $\theta = \|Q\|$.

Calculate the cosine of the angle times the vector to rotate, plus sine of the angle times the axis of rotation, plus one minus cosine of the angle times the dot product of the vector and rotation axis times the axis of rotation.

$$V' = \cos(\theta) * v + \sin(\theta) * \left(\frac{Q}{\|Q\|} \times v\right) + (1 - \cos(\theta)) \left(\frac{Q}{\|Q\|} \cdot v\right) * \frac{Q}{\|Q\|}$$

Some notes: the dot product includes the cosine of the angle between the vector being rotated and the axis of rotation times the length of V; the cross product includes the sine of the angle between the vector being rotated and the axis of rotation.

Rotate a Rotation Vector

Using Rodrigues Composite Rotation Formula

For a given rotation vector $Q = (X, Y, Z)$, and another rotation vector $A = (X, Y, Z)$ to rotate the frame around.

From the initial rotation vectors, extract the angles and axes:

$$\theta = \frac{\|Q\|}{2}$$

$$\gamma = \frac{\|A\|}{2}$$

Normalized axis of rotation for the current frame:

$$Q_n = \frac{Q}{\|Q\|}$$

Normalized axis of rotation to rotate the frame around:

$$A_n = \frac{A}{\|A\|}$$

The result angle angle of the rotation is

$$\alpha = 2\cos^{-1}(\cos(\theta)\cos(\gamma) + \sin(\theta)\sin(\gamma)Q_n \cdot A_n)$$

or

$$\alpha = 2cos^{-1}(\cos(\theta - \gamma)(1 - (Q_n \cdot A_n)) + \cos(\theta + \gamma)(1 + (Q_n \cdot A_n)))$$

The result, un-normalized axis of rotation:

$$r = \sin(\gamma)\cos(\theta)A_n + \sin(\theta)\cos(\gamma)Q_n + \sin(\theta)\sin(\gamma)A_n \times Q_n$$

or

$$r = (A_n \times Q_n)(\cos(\theta - \gamma) - \cos(\theta + \gamma)) + A_n(\sin(\theta + \gamma) + \sin(\theta - \gamma)) + Q_n(\sin(\theta + \gamma) - s$$

The Rodrigues Rotation Formula would lead that the sin of above resulting angle can be used to normalize the vector, however this fails for large ranges; so normalize the result axis as any other vector.

$$R_n = \frac{r}{\|r\|}$$

And the final frame rotation coordinate:

$$R = \alpha R_n$$

Spin rotation around a fixed axis

A rotation vector Q represents three axes; these may be used as a shorthand to rotate the rotation around using the above 'Rotate a Rotation Vector'. These expressions are best represented as code fragments.

Setup some constants used in other expressions.

$$
\begin{aligned}
n_x &= \frac{Q_x}{\|Q\|} \\
n_y &= \frac{Q_y}{\|Q\|} \\
n_z &= \frac{Q_z}{\|Q\|} \\
\text{angle} &= \|Q\| \\
s &= \sin(\text{angle}) \\
c_1 &= \cos(\text{angle}) \\
c &= 1 - c_1
\end{aligned}
$$

using the above values...

$$\text{x-axis} = \left[x = cn_x^2 + c_1,\ y = cn_xn_y + sn_z,\ z = cn_xn_z - sn_y\right]$$

or

$$\text{y-axis} = \left[x = cn_yn_x - sn_z,\ y = cn_y^2 + c_1,\ z = cn_yn_z + sn_x\right]$$

or

$$\text{z-axis} = \left[x = cn_zn_x + sn_y,\ y = cn_zn_y - sn_x,\ z = cn_z^2 + c_1\right]$$

Conversion from Basis Matrix

Compute the determinant of the matrix...

$$d = \frac{(\text{basis}_{\text{right}_X} + \text{basis}_{\text{up}_Y} + \text{basis}_{\text{forward}_Z}) - 1}{2}$$

Convert to the angle of rotation...

$$\theta = 2\arccos(d)$$

$$yz = \text{basis}_{\text{up}_Z} - \text{basis}_{\text{forward}_Y}$$

$$xz = \text{basis}_{\text{forward}_X} - \text{basis}_{\text{right}_Z}$$

$$xy = \text{basis}_{\text{right}_Y} - \text{basis}_{\text{up}_X}$$

compute the normal factor...

$$\mathbf{normal} = \frac{1}{\sqrt{yz^2 + xz^2 + xy^2}}$$

$$n = \begin{bmatrix} yz \cdot \mathbf{normal} \\ xz \cdot \mathbf{normal} \\ xy \cdot \mathbf{normal} \end{bmatrix}$$

the resulting angle-angle-angle:

$$n \cdot \theta$$

Conversion from Normal Vector(Y)

Representation of a normal as a rotation, this assumes that the vector $(0, 1, 0)$ is 'up'. If some other axle is considered primary, the coordinates can be simply swapped.

This assume a normalized input vector in the direction of the normal

$$N = \begin{bmatrix} \mathbf{normal}_X \\ \mathbf{normal}_Y \\ \mathbf{normal}_Z \end{bmatrix}$$

The angle is simply the sum of the x/z coordinate(or y,x if Z is 'up', or y,z if X is 'up')...

$$\mathbf{angle} = |N_x| + |N_z|$$

if angle is 0, the job is done, result with $(0, 0, 0)$

$$r = 1/(\mathbf{angle})$$

Some temporary values; these values are just partials referenced later...

$$t = \begin{bmatrix} N_x \cdot r \\ N_y \\ N_z \cdot r \end{bmatrix}$$

Use the projected normal on the Y axis as the angle to rotate...

$$\mathbf{target}_{\mathbf{angle}} = \cos^{-1}(t_Y)$$

$$\mathbf{result} = \begin{bmatrix} t_Z \cdot \mathbf{target}_{\mathbf{angle}} \\ 0 \\ -t_X \cdot \mathbf{target}_{\mathbf{angle}} \end{bmatrix}$$

Align Normal using Basis

The default tangent and bi-tangent of rotations which only have their normal set, results in tangents and bi-tangents that are irregular. Alternatively build a basis matrix, and convert from basis using the above mentioned method. Compute the normal of the above, and the matrix to convert...

$$\text{normal}_{\text{twist}} = \sqrt{t_Z^2 + t_X^2}$$

$$\begin{bmatrix} \left(N_y \cdot \frac{-t_X}{\text{normal}_{\text{twist}}}\right) & N_x & \frac{t_Z}{\text{normal}_{\text{twist}}} \\ \left(N_z \cdot \frac{t_Z}{\text{normal}_{\text{twist}}}\right) - \left(N_x \cdot \frac{-t_X}{\text{normal}_{\text{twist}}}\right) & N_y & 0 \\ \left(-N_y \cdot \frac{t_Z}{\text{normal}_{\text{twist}}}\right) & N_z & \frac{-t_X}{\text{normal}_{\text{twist}}} \end{bmatrix}$$

And then use the basis to log quaternion conversion...

Align Normal Directly

Or This is the direct computation to result with a log quaternion; compute the above result vector and then...

$$\begin{aligned} t_{X_n} &= t_X \cdot \text{normal}_{\text{twist}} \\ t_{Z_n} &= t_Z \cdot \text{normal}_{\text{twist}} \\ s &= \sin(\text{target}_{\text{angle}}) \\ c &= 1 - \cos(\text{target}_{\text{angle}}) \end{aligned}$$

This is the angle

$$\text{angle} = \arccos((t_Y + 1) * (1 - t_{X_n})/2 - 1);$$

These partial products are used below...

$$\begin{aligned} yz &= s \cdot n_X \\ xz &= (2 - c \cdot (n_X^2 + n_Z^2)) \cdot t_{Z_n} \\ xy &= s \cdot n_X * t_{Z_n} + s \cdot n_Z \cdot (1 - t_{X_n}) \end{aligned}$$

Compute the normalized rotation vector (axle of rotation)...

$$n = \begin{bmatrix} \frac{yz}{\sqrt{yz^2+xz^2+xy^2}} \\ \frac{xz}{\sqrt{yz^2+xz^2+xy^2}} \\ \frac{xy}{\sqrt{yz^2+xz^2+xy^2}} \end{bmatrix}$$

and finally compute the resulting log quaternion.

$$\text{final}_{\text{result}} = \text{angle} \cdot n$$

Conversion from axis-angle

This assumes the input axis $a = [X, Y, Z]$ is normalized. If there is 0 rotation, result with $(0, 0, 0)$

$$\theta = \text{angle}$$

$$\text{result} = \theta * a$$

See also

- Euler filter
- Orientation (geometry)
- Rotation around a fixed axis
- Three-dimensional rotation operator

References

1. "Fiducial Marker Tracking for Augmented Reality" (http://www.questionablyartificial.com/2016---fiducial-marker-tracking-for-augmented-reality.html).
2. Weisstein, Eric W. "Rotation Matrix" (https://mathworld.wolfram.com/RotationMatrix.html). *MathWorld*.
3. Rodrigues, Olinde (1840). "Des lois géometriques qui regissent les déplacements d' un systéme solide dans l' espace, et de la variation des coordonnées provenant de ces déplacement considérées indépendant des causes qui peuvent les produire". *J. Math. Pures Appl*. **5**: 380–440. online (http://sites.mathdoc.fr/JMPA/PDF/JMPA_1840_1_5_A39_0.pdf)
4. cf. J Willard Gibbs (1884). *Elements of Vector Analysis*, New Haven, p. 67
5. Direct and inverse kinematics (http://cmp.felk.cvut.cz/cmp/courses/ROB/roblec/serial-noteeng.pdf) lecture notes, page 5
6. Mebius, Johan (2007). "Derivation of the Euler–Rodrigues formula for three-dimensional rotations from the general formula for four-dimensional rotations". arXiv:math/0701759 (https://arxiv.org/abs/math/0701759).
7. [1] (http://www.euclideanspace.com/physics/kinematics/angularvelocity/mark.htm) Physics - Mark Ioffe - W(t) in terms of matrices
8. Quaternions and Rotation (http://web.cs.iastate.edu/~cs577/handouts/quaternion.pdf) lecture notes, p. 14-15

9. d3x0r. "STFRPhysics Repository" (https://github.com/d3x0r/STFRPhysics).

Further reading

- Shuster, M.D. (1993). "A Survey of Attitude Representations" (https://web.archive.org/web/20190925150540/http://www.malcolmdshuster.com/Pub_1993h_J_Repsurv_scan.pdf) (PDF). *Journal of the Astronautical Sciences*. **41** (4): 439–517. Bibcode:1993JAnSc..41..439S (https://ui.adsabs.harvard.edu/abs/1993JAnSc..41..439S). Archived from the original (http://www.malcolmdshuster.com/Pub_1993h_J_Repsurv_scan.pdf) (PDF) on 2019-09-25.
- Taubin, G. (2011). "3D Rotations" (http://mesh.brown.edu/rotations). *IEEE Computer Graphics and Applications*. **31** (6): 84–89. doi:10.1109/MCG.2011.92 (https://doi.org/10.1109%2FMCG.2011.92). PMID 24808261 (https://pubmed.ncbi.nlm.nih.gov/24808261).
- Coutsias, E.; Romero, L. (2004). "The Quaternions with an application to Rigid Body Dynamics" (https://digitalrepository.unm.edu/math_fsp/4/). *Sandia Technical Report*. Sandia National Laboraties. SAND2004-0153.
- Markley, F. Landis (2003). "Attitude Error Representations for Kalman Filtering". *Journal of Guidance, Control and Dynamics*. **26** (2): 311–7. Bibcode:2003JGCD...26..311M (https://ui.adsabs.harvard.edu/abs/2003JGCD...26..311M). doi:10.2514/2.5048 (https://doi.org/10.2514%2F2.5048). hdl:2060/20020060647 (https://hdl.handle.net/2060%2F20020060647).
- Goldstein, H. (1980). *Classical Mechanics* (2nd ed.). Addison–Wesley. ISBN 0-201-02918-9.
- Wertz, James R. (1980). *Spacecraft Attitude Determination and Control*. D. Reidel. ISBN 90-277-1204-2.
- Schmidt, J.; Niemann, H. (2001). "Using Quaternions for Parametrizing 3-D Rotations in Unconstrained Nonlinear Optimization". *Proceedings of the Vision Modeling and Visualization Conference 2001*. pp. 399–406. ISBN 3898380289.
- Landau, L.; Lifshitz, E.M. (1976). *Mechanics* (https://archive.org/details/mechanics00land) (3rd ed.). Pergamon Press. ISBN 0-08-021022-8.
- Klumpp, A.R. (December 1976). "Singularity-Free Extraction of a Quaternion from a Direction-Cosine Matrix". *Journal of Spacecraft and Rockets*. **13** (12): 754–5. Bibcode:1976JSpRo..13..754K (https://ui.adsabs.harvard.edu/abs/1976JSpRo..13..754K). doi:10.2514/3.27947 (https://doi.org/10.2514%2F3.27947).
- Doran, C.; Lasenby, A. (2003). *Geometric Algebra for Physicists*. Cambridge University Press. ISBN 978-0-521-71595-9.
- Terzakis, G.; Lourakis, M.; Ait-Boudaoud, D. (2018). "Modified Rodrigues Parameters: An Efficient Representation of Orientation in 3D Vision and Graphics" (http://rdcu.be/wIBC). *Journal of Mathematical Imaging and Vision*. **60** (3): 422–442. doi:10.1007/s10851-017-0765-x (https://doi.org/10.1007%2Fs10851-017-0765-x).
- Rowenhorst, D.; Rollett, A.D.; Rohrer, G.S.; Groeber, M.; Jackson, M.; Konijnenberg, P.J.; De Graef, M. (2015). "Consistent representations of and conversions between 3D rotations" (https://semanticscholar.org/paper/932b4eaa22afb4384815db12dfd197c24c7ba355). *Modelling and Simulation in Materials Science and Engineering*. **23** (8): 083501. Bibcode:2015MSMSE..23h3501R (https://ui.adsabs.harvard.edu/abs/2015MSMSE..23h3501R).

doi:10.1088/0965-0393/23/8/083501 (https://doi.org/10.1088%2F0965-0393%2F23%2F8%2F083501).

External links

- EuclideanSpace (http://www.euclideanspace.com/maths/geometry/rotations/index.htm) has a wealth of information on rotation representation
- Q36. How do I generate a rotation matrix from Euler angles? (http://www.j3d.org/matrix_faq/matrfaq_latest.html#Q36) and Q37. How do I convert a rotation matrix to Euler angles? (http://www.j3d.org/matrix_faq/matrfaq_latest.html#Q37) — The Matrix and Quaternions FAQ
- Imaginary numbers are not Real – the Geometric Algebra of Spacetime (http://www.mrao.cam.ac.uk/~clifford/introduction/intro/node9.html#SECTION00030000000000000000) – Section "Rotations and Geometric Algebra" derives and applies the rotor description of rotations
- Starlino's DCM Tutorial (http://www.starlino.com/dcm_tutorial.html) – Direction cosine matrix theory tutorial and applications. Space orientation estimation algorithm using accelerometer, gyroscope and magnetometer IMU devices. Using complimentary filter (popular alternative to Kalman filter) with DCM matrix.