WIKIPEDIA
The Free Encyclopedia

Conversion between quaternions and Euler angles

Spatial rotations in three dimensions can be parametrized using both Euler angles and unit quaternions. This article explains how to convert between the two representations. Actually this simple use of "quaternions" was first presented by Euler some seventy years earlier than Hamilton to solve the problem of magic squares. For this reason the dynamics community commonly refers to quaternions in this application as "Euler parameters".

Definition

There are two representations of quaternions. This article uses the more popular Hamilton.

A quaternion has 4 scaler values: q_w (the real part) and q_x q_y q_z (the imaginary part).

A *unit quaternion* satisfies this equation:

$$q_w^2 + q_x^2 + q_y^2 + q_z^2 = 1$$

We can associate a quaternion with a rotation around an axis by the following expression

$$\begin{aligned}
\mathbf{q}_w &= \cos(\alpha/2) \\
\mathbf{q}_x &= \sin(\alpha/2)\cos(\beta_x) \\
\mathbf{q}_y &= \sin(\alpha/2)\cos(\beta_y) \\
\mathbf{q}_z &= \sin(\alpha/2)\cos(\beta_z)
\end{aligned}$$

where α is a simple rotation angle (the value in radians of the angle of rotation) and $\cos(\beta_x)$, $\cos(\beta_y)$ and $\cos(\beta_z)$ are the "direction cosines" of the angles between the three coordinate axes and the axis of rotation. (Euler's Rotation Theorem).

Intuition

To better understand how "direction cosines" work with quaternions:

$$\begin{aligned}
\mathbf{q}_w &= \cos(\text{rotation angle}/2) \\
\mathbf{q}_x &= \sin(\text{rotation angle}/2)\cos(\text{angle between axis of rotation and x axis}) \\
\mathbf{q}_y &= \sin(\text{rotation angle}/2)\cos(\text{angle between axis of rotation and y axis}) \\
\mathbf{q}_z &= \sin(\text{rotation angle}/2)\cos(\text{angle between axis of rotation and z axis})
\end{aligned}$$

If the axis of rotation is the *x*-axis:

$$\begin{aligned}
\mathbf{q}_w &= \cos(\alpha/2) \\
\mathbf{q}_x &= \sin(\alpha/2)\cdot 1 \\
\mathbf{q}_y &= \sin(\alpha/2)\cdot 0 \\
\mathbf{q}_z &= \sin(\alpha/2)\cdot 0
\end{aligned}$$

If the axis of rotation is the *y*-axis:

$$\mathbf{q}_w = \cos(\alpha/2)$$
$$\mathbf{q}_x = \sin(\alpha/2) \cdot 0$$
$$\mathbf{q}_y = \sin(\alpha/2) \cdot 1$$
$$\mathbf{q}_z = \sin(\alpha/2) \cdot 0$$

If the axis of rotation is the *z*-axis:

$$\mathbf{q}_w = \cos(\alpha/2)$$
$$\mathbf{q}_x = \sin(\alpha/2) \cdot 0$$
$$\mathbf{q}_y = \sin(\alpha/2) \cdot 0$$
$$\mathbf{q}_z = \sin(\alpha/2) \cdot 1$$

If the axis of rotation is a vector located 45° ($\frac{\pi}{4}$ radians) between the *x* and *y* axes:

$$\mathbf{q}_w = \cos(\alpha/2)$$
$$\mathbf{q}_x = \sin(\alpha/2) \cdot 0.707\ldots$$
$$\mathbf{q}_y = \sin(\alpha/2) \cdot 0.707\ldots$$
$$\mathbf{q}_z = \sin(\alpha/2) \cdot 0$$

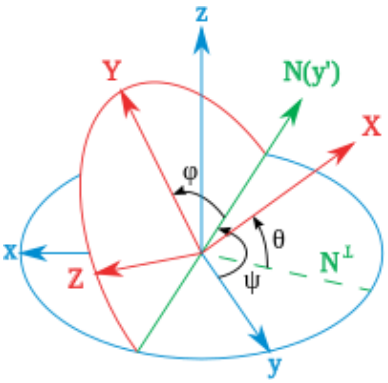
Therefore, the *x* and *y* axes "share" influence over the new axis of rotation.

Tait–Bryan angles

Similarly for Euler angles, we use the Tait Bryan angles (in terms of flight dynamics):

- Heading – ψ: rotation about the Z-axis
- Pitch – θ: rotation about the new Y-axis
- Bank – ϕ: rotation about the new X-axis

where the X-axis points forward, Y-axis to the right and Z-axis downward. In the conversion example above the rotation occurs in the order heading, pitch, bank.



Tait–Bryan angles. *z-y'-x"* sequence (intrinsic rotations; *N* coincides with *y'*). The angle rotation sequence is *ψ*, *θ*, *φ*. Note that in this case *ψ* > *90°* and *θ* is a negative angle.

Rotation matrices

The orthogonal matrix (post-multiplying a column vector) corresponding to a clockwise/left-handed (looking along positive axis to origin) rotation by the unit quaternion $q = q_w + iq_x + jq_y + kq_z$ is given by the inhomogeneous expression:

$$R = \begin{bmatrix} 1 - 2(q_y^2 + q_z^2) & 2(q_x q_y - q_w q_z) & 2(q_w q_y + q_x q_z) \\ 2(q_x q_y + q_w q_z) & 1 - 2(q_x^2 + q_z^2) & 2(q_y q_z - q_w q_x) \\ 2(q_x q_z - q_w q_y) & 2(q_w q_x + q_y q_z) & 1 - 2(q_x^2 + q_y^2) \end{bmatrix}$$

or equivalently, by the homogeneous expression:

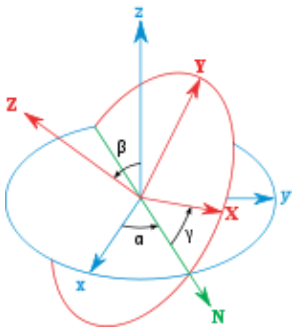
$$R = \begin{bmatrix} q_w^2 + q_x^2 - q_y^2 - q_z^2 & 2(q_x q_y - q_w q_z) & 2(q_w q_y + q_x q_z) \\ 2(q_x q_y + q_w q_z) & q_w^2 - q_x^2 + q_y^2 - q_z^2 & 2(q_y q_z - q_w q_x) \\ 2(q_x q_z - q_w q_y) & 2(q_w q_x + q_y q_z) & q_w^2 - q_x^2 - q_y^2 + q_z^2 \end{bmatrix}$$

If $q_w + iq_x + jq_y + kq_z$ is not a unit quaternion then the homogeneous form is still a scalar multiple of a rotation matrix, while the inhomogeneous form is in general no longer an orthogonal matrix. This is why in numerical work the homogeneous form is to be preferred if distortion is to be avoided.

The direction cosine matrix (from the rotated Body XYZ coordinates to the original Lab xyz coordinates for a clockwise/lefthand rotation) corresponding to a post-multiply **Body 3-2-1** sequence with Euler angles (ψ, θ, φ) is given by:[1]

$$\begin{aligned} \begin{bmatrix} x \\ y \\ z \end{bmatrix} &= R_z(\psi) R_y(\theta) R_x(\phi) \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \\ &= \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi & \cos\phi \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \\ &= \begin{bmatrix} \cos\theta\cos\psi & -\cos\phi\sin\psi + \sin\phi\sin\theta\cos\psi & \sin\phi\sin\psi + \cos\phi\sin\theta\cos\psi \\ \cos\theta\sin\psi & \cos\phi\cos\psi + \sin\phi\sin\theta\sin\psi & -\sin\phi\cos\psi + \cos\phi\sin\theta\sin\psi \\ -\sin\theta & \sin\phi\cos\theta & \cos\phi\cos\theta \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \end{aligned}$$

Euler angles (in 3-2-1 sequence) to quaternion conversion



Euler angles for Body 3-1-3 Sequence – The xyz (original fixed Lab) system is shown in blue, the XYZ (rotated final Body) system is shown in red. The line of nodes, labelled N and shown in green, is the intermediate Body X-axis around which the second rotation occurs.

By combining the quaternion representations of the Euler rotations we get for the **Body 3-2-1** sequence, where the airplane first does yaw (Body-Z) turn during taxiing onto the runway, then pitches (Body-Y) during take-off, and finally rolls (Body-X) in the air. The resulting orientation of Body 3-2-1 sequence (around the capitalized axis in the illustration of Tait–Bryan angles) is equivalent to that of lab 1-2-3 sequence (around the lower-cased axis), where the airplane is rolled first (lab-x axis), and then nosed up around the horizontal lab-y axis, and finally rotated around the vertical lab-z axis (**lB** = **lab2Body**):

$$\begin{aligned} \mathbf{q_{lB}} &= \begin{bmatrix} \cos(\psi/2) \\ 0 \\ 0 \\ \sin(\psi/2) \end{bmatrix} \begin{bmatrix} \cos(\theta/2) \\ 0 \\ \sin(\theta/2) \\ 0 \end{bmatrix} \begin{bmatrix} \cos(\phi/2) \\ \sin(\phi/2) \\ 0 \\ 0 \end{bmatrix} \\ &= \begin{bmatrix} \cos(\phi/2)\cos(\theta/2)\cos(\psi/2) + \sin(\phi/2)\sin(\theta/2)\sin(\psi/2) \\ \sin(\phi/2)\cos(\theta/2)\cos(\psi/2) - \cos(\phi/2)\sin(\theta/2)\sin(\psi/2) \\ \cos(\phi/2)\sin(\theta/2)\cos(\psi/2) + \sin(\phi/2)\cos(\theta/2)\sin(\psi/2) \\ \cos(\phi/2)\cos(\theta/2)\sin(\psi/2) - \sin(\phi/2)\sin(\theta/2)\cos(\psi/2) \end{bmatrix} \end{aligned}$$

Other rotation sequences use different conventions.[1]

Source code

Below code in C++ illustrates above conversion:

```
struct Quaternion
{
    double w, x, y, z;
};

Quaternion ToQuaternion(double roll, double pitch, double yaw) // roll (x), pitch (y), yaw (z)
{
    // Abbreviations for the various angular functions

    double cr = cos(roll * 0.5);
    double sr = sin(roll * 0.5);
    double cp = cos(pitch * 0.5);
    double sp = sin(pitch * 0.5);
    double cy = cos(yaw * 0.5);
    double sy = sin(yaw * 0.5);

    Quaternion q;
    q.w = cr * cp * cy + sr * sp * sy;
    q.x = sr * cp * cy - cr * sp * sy;
    q.y = cr * sp * cy + sr * cp * sy;
    q.z = cr * cp * sy - sr * sp * cy;

    return q;
}
```

Quaternion to Euler angles (in 3-2-1 sequence) conversion

A direct formula for the conversion from a quaternion to Euler angles in any of the 12 possible sequences exists.[2] For the rest of this section, the formula for the sequence **Body 3-2-1** will be shown. If the quaternion is properly **normalized**, the Euler angles can be obtained from the quaternions via the relations:

$$\begin{bmatrix} \phi \\ \theta \\ \psi \end{bmatrix} = \begin{bmatrix} \operatorname{atan2}\left(2(q_w q_x + q_y q_z), 1 - 2(q_x^2 + q_y^2)\right) \\ -\pi/2 + 2\operatorname{atan2}\left(\sqrt{1 + 2(q_w q_y - q_x q_z)}, \sqrt{1 - 2(q_w q_y - q_x q_z)}\right) \\ \operatorname{atan2}\left(2(q_w q_z + q_x q_y), 1 - 2(q_y^2 + q_z^2)\right) \end{bmatrix}$$

Note that the arctan functions implemented in computer languages only produce results between −π/2 and π/2, which is why atan2 is used to generate all the correct orientations. Moreover, typical implementations of arctan also might have some numerical disadvantages near zero and one.

Some implementations use the equivalent expression:[3]

$$\theta = \arcsin(2(q_w q_y - q_x q_z))$$

Source code

The following C++ program illustrates conversion above:

```
#define _USE_MATH_DEFINES
#include <cmath>

struct Quaternion {
    double w, x, y, z;
};

struct EulerAngles {
    double roll, pitch, yaw;
};

// this implementation assumes normalized quaternion
// converts to Euler angles in 3-2-1 sequence
EulerAngles ToEulerAngles(Quaternion q) {
    EulerAngles angles;
```

```
    // roll (x-axis rotation)
    double sinr_cosp = 2 * (q.w * q.x + q.y * q.z);
    double cosr_cosp = 1 - 2 * (q.x * q.x + q.y * q.y);
    angles.roll = std::atan2(sinr_cosp, cosr_cosp);

    // pitch (y-axis rotation)
    double sinp = std::sqrt(1 + 2 * (q.w * q.y - q.x * q.z));
    double cosp = std::sqrt(1 - 2 * (q.w * q.y - q.x * q.z));
    angles.pitch = 2 * std::atan2(sinp, cosp) - M_PI / 2;

    // yaw (z-axis rotation)
    double siny_cosp = 2 * (q.w * q.z + q.x * q.y);
    double cosy_cosp = 1 - 2 * (q.y * q.y + q.z * q.z);
    angles.yaw = std::atan2(siny_cosp, cosy_cosp);

    return angles;
}
```

Singularities

One must be aware of singularities in the Euler angle parametrization when the pitch approaches ±90° (north/south pole). These cases must be handled specially. The common name for this situation is gimbal lock.

Code to handle the singularities is derived on this site: www.euclideanspace.com (http://www.euclideanspace.com/maths/geometry/rotations/conversions/quaternionToEuler/)

Vector rotation

Let us define scalar q_w and vector $\vec{q}$ such that quaternion $\mathbf{q} = (q_w, \vec{q})$.

Note that the canonical way to rotate a three-dimensional vector $\vec{v}$ by a quaternion q defining an Euler rotation is via the formula

$$\mathbf{v}' = \mathbf{q}\mathbf{v}\mathbf{q}^*$$

where $\mathbf{v} = (0, \vec{v})$ is a quaternion containing the embedded vector $\vec{v}$, $\mathbf{q}^* = (q_w, -\vec{q})$ is a conjugate quaternion, and $\mathbf{v}' = (0, \vec{v}')$ is the rotated vector $\vec{v}'$. In computational implementations this requires two quaternion multiplications. An alternative approach is to apply the pair of relations

$$\vec{t} = 2\vec{q} \times \vec{v}$$
$$\vec{v}' = \vec{v} + q_w\vec{t} + \vec{q} \times \vec{t}$$

where $\times$ indicates a three-dimensional vector cross product. This involves fewer multiplications and is therefore computationally faster. Numerical tests indicate this latter approach may be up to 30% [4] faster than the original for vector rotation.

Proof

The general rule for quaternion multiplication involving scalar and vector parts is given by

$$\begin{aligned}\mathbf{pq} &= (p_w, \vec{p})(q_w, \vec{q}) \\ &= (p_w q_w - \vec{p} \cdot \vec{q}, p_w\vec{q} + q_w\vec{p} + \vec{p} \times \vec{q})\end{aligned}$$

Using this relation one finds for $\mathbf{v} = (0, \vec{v})$ that

$$\begin{aligned}\mathbf{vq}^* &= (0, \vec{v})(q_w, -\vec{q}) \\ &= (\vec{v} \cdot \vec{q}, q_w\vec{v} - \vec{v} \times \vec{q})\end{aligned}$$

and upon substitution for the triple product

$$\begin{aligned}\mathbf{qvq}^* &= (q_w, \vec{q})(\vec{v} \cdot \vec{q}, q_w\vec{v} - \vec{v} \times \vec{q}) \\ &= (0, q_w^2\vec{v} + q_w\vec{q} \times \vec{v} + (\vec{v} \cdot \vec{q})\vec{q} + q_w\vec{q} \times \vec{v} + \vec{q} \times (\vec{q} \times \vec{v}))\end{aligned}$$

where anti-commutivity of cross product and $\vec{q} \cdot \vec{v} \times \vec{q} = 0$ has been applied. By next exploiting the property that $\mathbf{q}$ is a unit quaternion so that $q_w^2 = 1 - \vec{q} \cdot \vec{q}$, along with the standard vector identity

$$\vec{q} \times (\vec{q} \times \vec{v}) = (\vec{q} \cdot \vec{v})\vec{q} - (\vec{q} \cdot \vec{q})\vec{v}$$

one obtains

$$\mathbf{v}' = \mathbf{qvq}^* = (0, \vec{v} + 2q_w\vec{q} \times \vec{v} + 2\vec{q} \times (\vec{q} \times \vec{v}))$$

which upon defining $\vec{t} = 2\vec{q} \times \vec{v}$ can be written in terms of scalar and vector parts as

$$(0, \vec{v}\,') = (0, \vec{v} + q_w\vec{t} + \vec{q} \times \vec{t}).$$

See also

- Rotation operator (vector space)
- Quaternions and spatial rotation
- Euler Angles
- Rotation matrix
- Rotation formalisms in three dimensions

References

1. NASA Mission Planning and Analysis Division (July 1977). "Euler Angles, Quaternions, and Transformation Matrices" (https://ntrs.nasa.gov/citations/19770024290). NASA. Retrieved 24 May 2021.
2. Bernardes, Evandro; Viollet, Stéphane (10 November 2022). "Quaternion to Euler angles conversion: A direct, general and computationally efficient method" (https://www.ncbi.nlm.nih.gov/pmc/articles/PMC9648712). *PLOS ONE*. **17** (11): e0276302. Bibcode:2022PLoSO..1776302B (https://ui.adsabs.harvard.edu/abs/2022PLoSO..1776302B). doi:10.1371/journal.pone.0276302 (https://doi.org/10.1371%2Fjournal.pone.0276302). ISSN 1932-6203 (https://www.worldcat.org/issn/1932-6203). PMC 9648712 (https://www.ncbi.nlm.nih.gov/pmc/articles/PMC9648712). PMID 36355707 (https://pubmed.ncbi.nlm.nih.gov/36355707).
3. Blanco, Jose-Luis (2010). "A tutorial on se (3) transformation parameterizations and on-manifold optimization". *University of Malaga, Tech. Rep*. CiteSeerX 10.1.1.468.5407 (https://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.468.5407).
4. Janota, A; Šimák, V; Nemec, D; Hrbček, J (2015). "Improving the Precision and Speed of Euler Angles Computation from Low-Cost Rotation Sensor Data" (https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4435132). *Sensors*. **15** (3): 7016–7039. Bibcode:2015Senso..15.7016J (https://ui.adsabs.harvard.edu/abs/2015Senso..15.7016J). doi:10.3390/s150307016 (https://doi.org/10.3390%2Fs

150307016). PMC 4435132 (https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4435132). PMID 25806874 (https://pubmed.ncbi.nlm.nih.gov/25806874).

External links

- Q60. How do I convert Euler rotation angles to a quaternion? (http://www.j3d.org/matrix_faq/matrfaq_latest.html#Q60) and related questions at The Matrix and Quaternions FAQ