WIKIPEDIA

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".

Contents

Definition

For the rest of this article, the JPL quaternion convention[1] shall be used. A unit quaternion can be described as:

$$\mathbf{q} = \begin{bmatrix} q_0 & q_1 & q_2 & q_3 \end{bmatrix}^T = \begin{bmatrix} q_w & q_x & q_y & q_z \end{bmatrix}^T$$
$$|\mathbf{q}|^2 = q_0^2 + q_1^2 + q_2^2 + q_3^2 = q_w^2 + q_x^2 + q_y^2 + q_z^2 = 1$$

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

$$\begin{aligned}\mathbf{q}_0 &= \mathbf{q}_w = \cos(\alpha/2)\\ \mathbf{q}_1 &= \mathbf{q}_x = \sin(\alpha/2)\cos(\beta_x)\\ \mathbf{q}_2 &= \mathbf{q}_y = \sin(\alpha/2)\cos(\beta_y)\\ \mathbf{q}_3 &= \mathbf{q}_z = \sin(\alpha/2)\cos(\beta_z)\end{aligned}$$

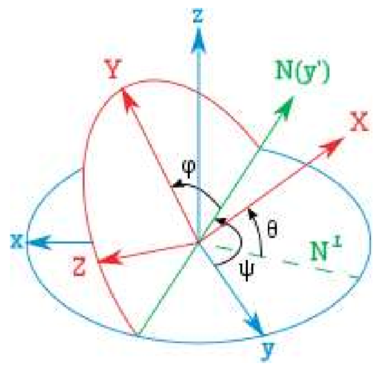
where α is a simple rotation angle (the value in radians of the angle of rotation) and cos(β_x), cos(β_y) and cos(β_z) are the "direction cosines" locating the axis of rotation (Euler's Rotation Theorem).

Tait–Bryan angles

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

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

where the X-axis points forward, Y-axis to the right and Z-axis downward with angles defined for clockwise/lefthand rotation. In the conversion example above the rotation occurs in the order heading, attitude, bank (about intrinsic axes).



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 $q = q_0 + iq_1 + jq_2 + kq_3$ is given by the inhomogeneous expression:

$$R = \begin{bmatrix} 1-2(q_2^2+q_3^2) & 2(q_1q_2-q_0q_3) & 2(q_0q_2+q_1q_3) \\ 2(q_1q_2+q_0q_3) & 1-2(q_1^2+q_3^2) & 2(q_2q_3-q_0q_1) \\ 2(q_1q_3-q_0q_2) & 2(q_0q_1+q_2q_3) & 1-2(q_1^2+q_2^2) \end{bmatrix}$$

or equivalently, by the homogeneous expression:

$$R = \begin{bmatrix} q_0^2+q_1^2-q_2^2-q_3^2 & 2(q_1q_2-q_0q_3) & 2(q_0q_2+q_1q_3) \\ 2(q_1q_2+q_0q_3) & q_0^2-q_1^2+q_2^2-q_3^2 & 2(q_2q_3-q_0q_1) \\ 2(q_1q_3-q_0q_2) & 2(q_0q_1+q_2q_3) & q_0^2-q_1^2-q_2^2+q_3^2 \end{bmatrix}$$

If $q_0 + iq_1 + jq_2 + kq_3$ 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:[2]

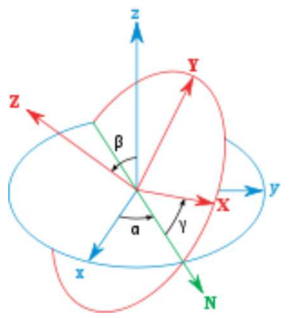
$$\begin{aligned} \begin{bmatrix} x \\ y \\ z \end{bmatrix} &= R_z(\psi)R_y(\theta)R_x(\phi) \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \\ &= \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta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

Euler Angles to Quaternion Conversion

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**):

$$\mathbf{q}_{\mathbf{IB}} = \begin{bmatrix} \cos(\psi/2) \\ 0 \\ 0 \\ \sin(\psi/2) \end{bmatrix} \begin{bmatrix} \cos(\theta/2) \\ 0 \\ \sin(\theta/2) \\ 0 \end{bmatrix} \begin{bmatrix} \cos(\phi/2) \\ \sin(\phi/2) \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} \cos(\phi/2)\cos(\theta/2)\cos(\psi/2) + \sin(\phi/2)\sin(\theta/2)\sin(\psi/2) \\ \sin(\phi/2)\cos(\theta/2)\cos(\psi/2) - \cos(\phi/2)\sin(\theta/2)\sin(\psi/2) \\ \cos(\phi/2)\sin(\theta/2)\cos(\psi/2) + \sin(\phi/2)\cos(\theta/2)\sin(\psi/2) \\ \cos(\phi/2)\cos(\theta/2)\sin(\psi/2) - \sin(\phi/2)\sin(\theta/2)\cos(\psi/2) \end{bmatrix}$$

Other rotation sequences use different conventions.[2]



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.

Source Code

Below code in C++ illustrates above conversion:

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

Quaternion ToQuaternion(double yaw, double pitch, double roll) // yaw (Z), pitch
(Y), roll (X)
{
    // Abbreviations for the various angular functions
    double cy = cos(yaw * 0.5);
    double sy = sin(yaw * 0.5);
    double cp = cos(pitch * 0.5);
    double sp = sin(pitch * 0.5);
    double cr = cos(roll * 0.5);
    double sr = sin(roll * 0.5);

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

    return q;
}
```

Quaternion to Euler Angles Conversion

The Euler angles can be obtained from the quaternions via the relations:[3]

$$\begin{bmatrix} \phi \\ \theta \\ \psi \end{bmatrix} = \begin{bmatrix} \arctan \frac{2(q_0 q_1 + q_2 q_3)}{1 - 2(q_1^2 + q_2^2)} \\ \arcsin(2(q_0 q_2 - q_3 q_1)) \\ \arctan \frac{2(q_0 q_3 + q_1 q_2)}{1 - 2(q_2^2 + q_3^2)} \end{bmatrix}$$

Note, however, that the arctan and arcsin functions implemented in computer languages only produce results between $-\pi/2$ and $\pi/2$, and for three rotations between $-\pi/2$ and $\pi/2$ one does not obtain all possible orientations. To generate all the orientations one needs to replace the arctan functions in computer code by atan2:

$$\begin{bmatrix} \phi \\ \theta \\ \psi \end{bmatrix} = \begin{bmatrix} \operatorname{atan2}(2(q_0 q_1 + q_2 q_3), 1 - 2(q_1^2 + q_2^2)) \\ \operatorname{asin}(2(q_0 q_2 - q_3 q_1)) \\ \operatorname{atan2}(2(q_0 q_3 + q_1 q_2), 1 - 2(q_2^2 + q_3^2)) \end{bmatrix}$$

Source Code

The following C++ program illustrates conversion above:

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

EulerAngles ToEulerAngles(Quaternion q)
{
    EulerAngles angles;

    // roll (x-axis rotation)
    double sinr_cosp = +2.0 * (q.w * q.x + q.y * q.z);
    double cosr_cosp = +1.0 - 2.0 * (q.x * q.x + q.y * q.y);
    angles.roll = atan2(sinr_cosp, cosr_cosp);

    // pitch (y-axis rotation)
    double sinp = +2.0 * (q.w * q.y - q.z * q.x);
    if (fabs(sinp) >= 1)
        angles.pitch = copysign(M_PI / 2, sinp); // use 90 degrees if out of range
    else
        angles.pitch = asin(sinp);

    // yaw (z-axis rotation)
    double siny_cosp = +2.0 * (q.w * q.z + q.x * q.y);
    double cosy_cosp = +1.0 - 2.0 * (q.y * q.y + q.z * q.z);
    angles.yaw = 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_0 and vector $\vec{q}$ such that $\mathbf{q} = (q_0, \vec{q}) = q_0 + iq_1 + jq_2 + kq_3$.

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

$$\mathbf{p}' = \mathbf{qpq}^*$$

where $\mathbf{p} = (0, \vec{v}) = 0 + iv_1 + jv_2 + kv_3$ is a quaternion containing the embedded vector $\vec{v}$, $\mathbf{q}^* = (q_0, -\vec{q})$ is a conjugate quaternion, and $\mathbf{p}' = (0, \vec{v}\,')$ is the rotated vector $\vec{v}\,'$. In computational implementations this requires two quaternion multiplications. An alternative approach is to apply the pair of relations

$$\begin{aligned}\vec{t} &= 2\vec{q} \times \vec{v} \\ \vec{v}\,' &= \vec{v} + q_0\vec{t} + \vec{q} \times \vec{t}\end{aligned}$$

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

$$\begin{aligned}\mathbf{q_1 q_2} &= (r_1, \vec{v}_1)(r_2, \vec{v}_2) \\ &= (r_1 r_2 - \vec{v}_1 \cdot \vec{v}_2, r_1\vec{v}_2 + r_2\vec{v}_1 + \vec{v}_1 \times \vec{v}_2)\end{aligned}$$

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

$$\begin{aligned}\mathbf{pq}^* &= (0,\vec{v})(q_0,-\vec{q}) \\ &= (\vec{v}\cdot\vec{q}, q_0\vec{v}-\vec{v}\times\vec{q})\end{aligned}$$

and upon substitution for the triple product

$$\begin{aligned}\mathbf{qpq}^* &= (q_0,\vec{q})(\vec{v}\cdot\vec{q}, q_0\vec{v}-\vec{v}\times\vec{q}) \\ &= (0, q_0^2\vec{v}+q_0\vec{q}\times\vec{v}+(\vec{v}\cdot\vec{q})\vec{q}+q_0\vec{q}\times\vec{v}+\vec{q}\times(\vec{q}\times\vec{v}))\end{aligned}$$

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 $q_0^2 = 1-\vec{q}\cdot\vec{q}$, 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

$$\mathbf{p}' = \mathbf{qpq}^* = (0, \vec{v}+2q_0\vec{q}\times\vec{v}+2\vec{q}\times(\vec{q}\times\vec{v}))$$

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

$$(0,\vec{v}\,') = (0, \vec{v}+q_0\vec{t}+\vec{q}\times\vec{t}).$$

See also

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

References

1. W. G. Breckenridge, "Quaternions proposed standard conventions," NASA Jet Propulsion Laboratory, Technical Report, Oct. 1979.
2. NASA Mission Planning and Analysis Division. "Euler Angles, Quaternions, and Transformation Matrices" (https://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/19770024290_1977024290.pdf) (PDF). NASA. Retrieved 12 January 2013.
3. Blanco, Jose-Luis (2010). "A tutorial on se (3) transformation parameterizations and on-manifold optimization" (http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.468.5407&rep=rep1&type=pdf). *University of Malaga, Tech. Rep*.
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. doi:10.3390/s150307016 (https://doi.org/10.3390%2Fs150307016). PMC 4435132 (https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4435132). PMID 25806874 (https://www.ncbi.nlm.nih.gov/pubmed/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