樹莓派, 樹莓派之學習, 樹莓派之教育

STEM 隨筆：古典力學：運動學【二・四E】

2018-04-02 | 懸鉤子 | 發表迴響

閱讀古今科技歷史往往有很多驚訝！！

On Apollo 11

A well-known gimbal lock incident happened in the Apollo 11 Moon mission. On this spacecraft, a set of gimbals was used on an inertial measurement unit (IMU). The engineers were aware of the gimbal lock problem but had declined to use a fourth gimbal.[5] Some of the reasoning behind this decision is apparent from the following quote:

"The advantages of the redundant gimbal seem to be outweighed by the equipment simplicity, size advantages, and corresponding implied reliability of the direct three degree of freedom unit."

— David Hoag, Apollo Lunar Surface Journal

They preferred an alternate solution using an indicator that would be triggered when near to 85 degrees pitch.

"Near that point, in a closed stabilization loop, the torque motors could theoretically be commanded to flip the gimbal 180 degrees instantaneously. Instead, in the LM, the computer flashed a 'gimbal lock' warning at 70 degrees and froze the IMU at 85 degrees"

— Paul Fjeld, Apollo Lunar Surface Journal

Rather than try to drive the gimbals faster than they could go, the system simply gave up and

froze the platform. From this point, the spacecraft would have to be manually moved away from the gimbal lock position, and the platform would have to be manually realigned using the stars as a reference.[6]

After the Lunar Module had landed, Mike Collins aboard the Command Module joked “How about sending me a fourth gimbal for Christmas?”

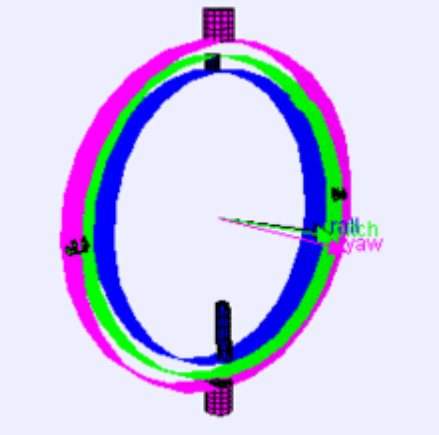
比方好奇『萬向鎖』到底是鎖住了什麼？？

想來有時以普通話『解釋名詞』實在也並非容易：

Gimbal lock

Gimbal lock is the loss of one degree of freedom in a three-dimensional, three-gimbal mechanism that occurs when the axes of two of the three gimbals are driven into a parallel configuration, “locking” the system into rotation in a degenerate two-dimensional space.

The word *lock* is misleading: no gimbal is restrained. All three gimbals can still rotate freely about their respective axes of suspension. Nevertheless, because of the parallel orientation of two of the gimbals’ axes there is no gimbal available to accommodate rotation along one axis.



Gimbal with 3 axes of rotation. A set of three gimbals mounted together to allow three degrees of freedom: roll, pitch and yaw. When two gimbals rotate around the same axis, the system loses one degree of freedom.

……

In three dimensions

Consider a case of a level sensing platform on an aircraft flying due north with its three gimbal axes mutually perpendicular (i.e., roll, pitch and yaw angles each zero). If the aircraft pitches up 90 degrees, the aircraft and platform's yaw axis gimbal becomes parallel to the roll axis gimbal, and changes about yaw can no longer be compensated for.

Solutions

This problem may be overcome by use of a fourth gimbal, intelligently driven by a motor so as to maintain a large angle between roll and yaw gimbal axes. Another solution is to rotate one or more of the gimbals to an arbitrary position when gimbal lock is detected and thus reset the device.

Modern practice is to avoid the use of gimbals entirely. In the context of inertial navigation systems, that can be done by mounting the inertial sensors directly to the body of the vehicle (this is called a strapdown system)[3] and integrating sensed rotation and acceleration digitally using quaternion methods to derive vehicle orientation and velocity. Another way to replace gimbals is to use fluid bearings or a flotation chamber.[4]

Gimbal locked airplane. When the pitch (green) and yaw (magenta) gimbals become aligned, changes to roll (blue) and yaw apply the same rotation to the airplane.

故而仍舊回歸其本的好呀！！？？

In applied mathematics

The problem of gimbal lock appears when one uses Euler angles in applied mathematics; developers of 3D computer programs, such as 3D modeling, embedded navigation systems, and video games must take care to avoid it.

In formal language, gimbal lock occurs because the map from Euler angles to rotations (topologically, from the 3-torus T^3 to the real projective space $\mathbf{RP}^3$) is not a covering map – it is not a local homeomorphism at every point, and thus at some points the rank (degrees of freedom) must drop below 3, at which point gimbal lock occurs. Euler angles provide a means for giving a numerical description of any rotation in three-dimensional space using three numbers, but not only is this description not unique, but there are some points where not every change in the target space (rotations) can be realized by a change in the source space (Euler angles). This is a topological constraint – there is no covering map from the 3-torus to the 3-dimensional real projective space; the only (non-trivial) covering map is from the 3-sphere, as in the use of quaternions.

To make a comparison, all the translations can be described using three numbers x , y , and z , as the succession of three consecutive linear movements along three perpendicular axes X , Y and Z axes. The same holds true for rotations: all the rotations can be described using three numbers α , β , and γ , as the succession of three rotational movements around three axes that are perpendicular one to the next. This similarity between linear coordinates and angular coordinates makes Euler angles very intuitive, but unfortunately they suffer from the gimbal lock problem.

Loss of a degree of freedom with Euler angles

A rotation in 3D space can be represented numerically with matrices in several ways. One of these representations is:

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

An example worth examining happens when $\beta = \frac{\pi}{2}$. Knowing that $\cos\frac{\pi}{2} = 0$ and $\sin\frac{\pi}{2} = 1$,

the above expression becomes equal to:

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

Carrying out matrix multiplication:

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

And finally using the trigonometry formulas:

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

Changing the values of α and γ in the above matrix has the same effects: the rotation angle α+γ changes, but the rotation axis remains in the Z direction: the last column and the first row in the matrix won't change. The only solution for α and γ to recover different roles is to change β.

It is possible to imagine an airplane rotated by the above-mentioned Euler angles using the **X-Y-Z** convention. In this case, the first angle – α is the pitch. Yaw is then set to $\frac{\pi}{2}$ and the final rotation – by γ – is again the airplane's pitch. Because of gimbal lock, it has lost one of the degrees of freedom – in this case the ability to roll.

It is also possible to choose another convention for representing a rotation with a matrix using Euler angles than the **X-Y-Z** convention above, and also choose other variation intervals for the angles, but in the end there is always at least one value for which a degree of freedom is lost.

The gimbal lock problem does not make Euler angles "invalid" (they always serve as a well-defined coordinate system), but it makes them unsuited for some practical applications.

Alternate orientation representation

The cause of gimbal lock is representing an orientation as three axial rotations with Euler angles. A potential solution therefore is to represent the orientation in some other way. This could be as a rotation matrix, a quaternion (see quaternions and spatial rotation), or a similar orientation representation that treats the orientation as a value rather than three separate and related values. Given such a representation, the user stores the orientation as a value. To apply angular changes, the orientation is modified by a delta angle/axis rotation. The resulting orientation must be re-normalized to prevent floating-point error from successive transformations from accumulating. For matrices, re-normalizing the result requires converting the matrix into its nearest orthonormal representation. For quaternions, re-normalization requires performing quaternion normalization.

但因『形式語』常常難以領會，且此『借例說例』吧？？！！

如果已知一『旋轉矩陣』R，將如何求解『歐拉角』呢？

假使依循前述文本之『約定』，可推導如下也：

```
%matplotlib inline
from sympy import *
from sympy.physics.mechanics import *
init_printing(use_latex='mathjax', pretty_print=False)
```

```
from sympy.abc import a, b, c, alpha, beta, gamma, theta
```

```
A = ReferenceFrame('A')
```

```
Zγ = Matrix([[cos(gamma), - sin(gamma), 0],[sin(gamma), cos(gamma), 0],[0,0,1]])
Zγ
```

$$\begin{bmatrix} \cos(\gamma) & -\sin(\gamma) & 0 \\ \sin(\gamma) & \cos(\gamma) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

```
Yβ = Matrix([[cos(beta), 0, sin(beta)],[0,1,0],[- sin(beta), 0, cos(beta)]])
Yβ
```

$$\begin{bmatrix} \cos(\beta) & 0 & \sin(\beta) \\ 0 & 1 & 0 \\ -\sin(\beta) & 0 & \cos(\beta) \end{bmatrix}$$

```
Xα = Matrix([[1,0,0],[0, cos(alpha), - sin(alpha)],[0, sin(alpha), cos(alpha)]])
Xα
```

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\alpha) & -\sin(\alpha) \\ 0 & \sin(\alpha) & \cos(\alpha) \end{bmatrix}$$

```
R = Xα * Yβ * Zγ
R
```

$$\begin{bmatrix} \cos(\beta)\cos(\gamma) & -\sin(\gamma)\cos(\beta) & \sin(\beta) \\ \sin(\alpha)\sin(\beta)\cos(\gamma)+\sin(\gamma)\cos(\alpha) & -\sin(\alpha)\sin(\beta)\sin(\gamma)+\cos(\alpha)\cos(\gamma) & -\sin(\alpha)\cos(\beta) \\ \sin(\alpha)\sin(\gamma)-\sin(\beta)\cos(\alpha)\cos(\gamma) & \sin(\alpha)\cos(\gamma)+\sin(\beta)\sin(\gamma)\cos(\alpha) & \cos(\alpha)\cos(\beta) \end{bmatrix}$$

```
R[0,2]
```

$$\sin(\beta)$$

```
print("R[0,1]=",R[0,1], "R[0,0]=",R[0,0])
R[0,1] / R[0,0]
```

```
R[0,1]= -sin(gamma)*cos(beta) R[0,0]= cos(beta)*cos(gamma)
```

$$-\frac{\sin(\gamma)}{\cos(\gamma)}$$

```
print("R[1,2]=",R[1,2], "R[2,2]=",R[2,2])
R[1,2]/R[2,2]
```

```
R[1,2]= -sin(alpha)*cos(beta) R[2,2]= cos(alpha)*cos(beta)
```

$$-\frac{\sin(\alpha)}{\cos(\alpha)}$$

※ 注意不同的『慣例』：

Euler angles

Complexity of conversion escalates with Euler angles (used here in the broad sense). The first difficulty is to establish which of the twenty-four variations of Cartesian axis order we will use. Suppose the three angles are $\theta_1, \theta_2, \theta_3$; physics and chemistry may interpret these as

$$Q(\theta_1, \theta_2, \theta_3) = Q_{\mathbf{z}}(\theta_1) Q_{\mathbf{y}}(\theta_2) Q_{\mathbf{z}}(\theta_3),$$

while aircraft dynamics may use

$$Q(\theta_1, \theta_2, \theta_3) = Q_{\mathbf{z}}(\theta_3) Q_{\mathbf{y}}(\theta_2) Q_{\mathbf{x}}(\theta_1).$$

One systematic approach begins with choosing the rightmost axis. Among all permutations of

(*x*,*y*,*z*), only two place that axis first; one is an even permutation and the other odd. Choosing parity thus establishes the middle axis. That leaves two choices for the left-most axis, either duplicating the first or not. These three choices gives us 3 × 2 × 2 = 12 variations; we double that to 24 by choosing static or rotating axes.

This is enough to construct a matrix from angles, but triples differing in many ways can give the same rotation matrix. For example, suppose we use the **zyz** convention above; then we have the following equivalent pairs:

(90°,	45°,	−105°)	≡	(−270°,	−315°,	255°)	*multiples of 360°*
(72°,	0°,	0°)	≡	(40°,	0°,	32°)	*singular alignment*
(45°,	60°,	−30°)	≡	(−135°,	−60°,	150°)	*bistable flip*

Angles for any order can be found using a concise common routine (Herter & Lott 1993; Shoemake 1994).

The problem of singular alignment, the mathematical analog of physical gimbal lock, occurs when the middle rotation aligns the axes of the first and last rotations. It afflicts every axis order at either even or odd multiples of 90°. These singularities are not characteristic of the rotation matrix as such, and only occur with the usage of Euler angles.

The singularities are avoided when considering and manipulating the rotation matrix as orthonormal row vectors (in 3D applications often named the right-vector, up-vector and out-vector) instead of as angles. The singularities are also avoided when working with quaternions.

這裡 $\sin(\beta) = R[0,2]$ 眼見垂手可得。不過

$$\sin(\beta) = \sin(\pi - \beta)$$

，將有『兩解』哩！

要是 $\cos(\beta) \neq 0$ ，得出

$$\tan(\gamma) = \frac{R[0,1]}{R[0,0]} = -\frac{\sin(\gamma)\cdot\cos(\beta)}{\cos(\gamma)\cdot\cos(\beta)}$$

以及

$$\tan(\alpha) = \frac{R[1,2]}{R[2,2]} = -\frac{\sin(\alpha)\cdot\cos(\beta)}{\cos(\alpha)\cdot\cos(\beta)}$$

亦無困擾乎？★

※ 小心適用『角度範圍』：

Atan2

在三角函數中，兩個參數的函數**atan2**是正切函數tan的一個變種。對於任意不同時等於0的實參數x和y，atan2(y,x)所表達的意思是坐標原點為起點，指向(x,y)的射線在坐標平面上與x軸正方向之間的角的角度。當y>0時，射線與x軸正方向的所得的角的角度指的是x軸正方向繞逆時針方向到達射線旋轉的角的角度；而當y<0時，射線與x軸正方向所得的角的角度指的是x軸正方向繞順時針方向達到射線旋轉的角的角度。

在幾何意義上，atan2(y, x) 等價於 atan(y/x)，但 atan2 的最大優勢是可以正確處理 x=0 而 y≠0 的情況，而不必進行會引發除零異常的 y/x 操作。

atan2函數最初在計算機程式語言中被引入，但是現在它的應用在科學和工程等其他多個領域十分常見。他的出現最早可以追溯到FORTRAN語言[1]，並且可以在C語言的數學標準庫的math.h文件中找到，此外在Java數學庫、.NET的System.Math（可應用於C#、VB.NET等語言）、Python的數學模塊以及其他地方都可以找到atan2的身影。許多腳本語言，比如Perl，也包含了C語言風格的atan2函數[2]。

函數定義

基於值域為 $\left(-\frac{\pi}{2}, \frac{\pi}{2}\right)$ 的反正切函數，該函數定義如下：

$$\operatorname{atan2}(y,x) = \begin{cases} \arctan\left(\frac{y}{x}\right) & x > 0 \\ \arctan\left(\frac{y}{x}\right) + \pi & y \geq 0, x < 0 \\ \arctan\left(\frac{y}{x}\right) - \pi & y < 0, x < 0 \\ +\frac{\pi}{2} & y > 0, x = 0 \\ -\frac{\pi}{2} & y < 0, x = 0 \\ \text{undefined} & y = 0, x = 0 \end{cases}$$

說明:

- 該函數的值域為 $(-\pi, \pi]$，可以通過對負數結果加 2π 的方法，將函數的結果映射到 $[0, 2\pi)$ 範圍內。

其餘所謂 □□○○ 可自得之的勒？☆

if $\sin(\beta) = 1$:

```
Yβ1 = Matrix([[0, 0, 1],[0,1,0],[-1, 0, 0]])
Yβ1
```

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

```
simplify(Xα * Yβ1 * Zγ)
```

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

if $\sin(\beta) = -1$:

```
YβN1 = Matrix([[0, 0, -1],[0,1,0],[1, 0, 0]])
YβN1
```

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

```
simplify(Xα * YβN1 * Zγ)
```

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