



欧拉角速度推导与不建议使用欧拉角的原因

原创 找不到服务器zhn 已于 2024-06-11 12:48:37 修改 阅读量1k 收藏 21 点赞数 18

分类专栏: 控制 文章标签: 欧拉角 微分方程 控制



控制 专栏收录该内容

4 订阅 20 篇文章

摘要 欧拉角速度指欧拉角对时间的微分与角速度的关系，用于刚体姿态运动学建模，本文从一种复杂但严谨（应该）的角度推导这一关系，建议使用欧拉角来描述姿态。

欧拉角微分方程

$$\vec{w} = \begin{bmatrix} 1 & 0 & -\sin\theta \\ 0 & \cos\phi & \sin\phi\cos\theta \\ 0 & -\sin\phi & \cos\phi\cos\theta \end{bmatrix} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} \dot{\phi} - \sin\theta\dot{\psi} \\ \cos\phi\dot{\theta} + \sin\phi\cos\theta\dot{\psi} \\ -\sin\phi\dot{\theta} + \cos\phi\cos\theta\dot{\psi} \end{bmatrix}$$
$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 1 & \sin\phi\tan\theta & \cos\phi\tan\theta \\ 0 & \cos\phi & -\sin\phi \\ 0 & \frac{\sin\phi}{\cos\theta} & \frac{\cos\phi}{\cos\theta} \end{bmatrix} \vec{w}$$

其中 ϕ, θ, ψ 分别为横滚角、俯仰角、偏航角。

欧拉角微分方程推导

推导欧拉角微分方程有3种方法。第一种方法比较简单，好多教材里也都这么解释，可以参考

[欧拉角导数和角速度之间的转换关系推导 -CSDN博客](#)

另外两种方法用的都是根据旋转矩阵微分方程和旋转矩阵与欧拉角的关系来推导，可以参考

[Angular Velocity expressed via Euler Angles -Physics Stack Exchange](#)

[【刚体运动】欧拉角时间导数与角速度 -知乎](#)

下面写一下我对第一种方法的理解。把世界系和本体系分别记作 I, B ，世界系 I 绕 Z 轴旋转 ψ 角后的坐标系记作 Z ， Z 系绕 Y 轴旋转 θ 角记作 Y ， Y 系绕 X 轴旋转 ϕ 角后的坐标系记作 B 。从一个坐标系 A 到另一个坐标系 B 的旋转矩阵记作 R_{BA} ，即 $v^B = R_{BA}v^A, v^A = R_{AB}v^B$ ， A 系与 B 系之间的角速度向量记作 w_{AB} ，并系下的坐标是一样的（有人不理解为什么一样的话我再补充）。首先可以直观地得到

$$\begin{aligned} w_{ZI}^Z &= w_{ZI}^I = [0 \ 0 \ \dot{\psi}]^T \\ w_{YZ}^Y &= w_{YZ}^Z = [0 \ \dot{\theta} \ 0]^T \\ w_{BY}^B &= w_{BY}^Y = [\dot{\phi} \ 0 \ 0]^T \end{aligned}$$

三次旋转的旋转矩阵分别为

$$R_{ZI} = R_z(-\psi), R_{YZ} = R_y(-\theta), R_{BY} = R_x(-\phi)$$

于是



找不到服务器zhn

关注

18



21

0

$$\begin{aligned}
 w_{ZI}^B &= R_{BY} R_{YZ} w_{ZI}^Z \\
 w_{YZ}^B &= R_{BY} w_{YZ}^Y \\
 w_{BI}^B &= w_{BY}^B + w_{YZ}^B + w_{ZI}^B \\
 &= \begin{bmatrix} \dot{\phi} \\ 0 \\ 0 \end{bmatrix} + R_x(-\phi) \begin{bmatrix} 0 \\ \dot{\theta} \\ 0 \end{bmatrix} + R_x(-\phi) R_y(-\theta) \begin{bmatrix} 0 \\ 0 \\ \dot{\psi} \end{bmatrix} \\
 &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \phi & \sin \phi \\ 0 & -\sin \phi & \cos \phi \end{bmatrix} \left(\begin{bmatrix} \dot{\phi} \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} \cos \theta & 0 & -\sin \theta \\ 0 & 1 & 0 \\ \sin \theta & 0 & \cos \theta \end{bmatrix} \begin{bmatrix} 0 \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \right) \\
 &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \phi & \sin \phi \\ 0 & -\sin \phi & \cos \phi \end{bmatrix} \begin{bmatrix} \dot{\phi} - \sin \theta \dot{\psi} \\ \dot{\theta} \\ \cos \theta \dot{\psi} \end{bmatrix} \\
 &= \begin{bmatrix} \dot{\phi} - \sin \theta \dot{\psi} \\ \cos \phi \dot{\theta} + \sin \phi \cos \theta \dot{\psi} \\ -\sin \phi \dot{\theta} + \cos \phi \cos \theta \dot{\psi} \end{bmatrix} \\
 &= R_x(-\phi) \left(\begin{bmatrix} \dot{\phi} \\ 0 \\ 0 \end{bmatrix} + R_y(-\theta) \left(\begin{bmatrix} 0 \\ \dot{\theta} \\ 0 \end{bmatrix} + R_z(-\psi) \begin{bmatrix} 0 \\ 0 \\ \dot{\psi} \end{bmatrix} \right) \right)
 \end{aligned}$$

下面写一下我想的另一种方法。由旋转矩阵微分方程可以得到角速度关于旋转矩阵的表达式
(旋转矩阵微分方程的推导见 [角速度变化时四元数和旋转矩阵微分方程的证明](#))

$$\begin{aligned}
 \dot{R}^T &= -\vec{w} \times R^T \\
 \vec{w}^\times &= -\dot{R}^T R
 \end{aligned}$$

或者

$$\begin{aligned}
 R^T R &= I \\
 \dot{R}^T R + R^T \dot{R} &= 0 \\
 \vec{w}^\times &= R^T \dot{R}
 \end{aligned}$$

因为 R 可以看作是多元复合函数 $R(\phi(t), \theta(t), \psi(t))$, 所以

$$\frac{dR}{dt} = \frac{\partial R}{\partial \phi} \dot{\phi} + \frac{\partial R}{\partial \theta} \dot{\theta} + \frac{\partial R}{\partial \psi} \dot{\psi}$$

于是

$$\vec{w}^\times = R^T \dot{R} = R^T \frac{\partial R}{\partial \phi} \dot{\phi} + R^T \frac{\partial R}{\partial \theta} \dot{\theta} + R^T \frac{\partial R}{\partial \psi} \dot{\psi}$$

其中 $R^T \frac{\partial R}{\partial \phi}$, $R^T \frac{\partial R}{\partial \theta}$, $R^T \frac{\partial R}{\partial \psi}$ 是3个反对称矩阵, 简单想想可以理解, 因为 w^\times 是反对称矩阵, 而 $\dot{\phi}, \dot{\theta}, \dot{\psi}$ 都是标量并且可以是任意值, 那么时, $R^T \frac{\partial R}{\partial \phi}$ 一定是反对称矩阵, 同理, 3个矩阵都是反对称矩阵, 并且分别对应角速度和欧拉角时间导数之间转换矩阵 T 的3列元素



找不到服务器zhn

关注

18



21



0

$$\begin{aligned}\vec{w} &= T \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \\ \vec{w}^\times &= \begin{bmatrix} R^T \frac{\partial R}{\partial \phi} & R^T \frac{\partial R}{\partial \theta} & R^T \frac{\partial R}{\partial \psi} \end{bmatrix} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \\ T &= \begin{bmatrix} \left(R^T \frac{\partial R}{\partial \phi}\right)_\times & \left(R^T \frac{\partial R}{\partial \theta}\right)_\times & \left(R^T \frac{\partial R}{\partial \psi}\right)_\times \end{bmatrix} \\ &= \begin{bmatrix} \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix}_\times & \begin{bmatrix} 0 & \sin(\phi) & \cos(\phi) \\ -\sin(\phi) & 0 & 0 \\ -\cos(\phi) & 0 & 0 \end{bmatrix}_\times & \begin{bmatrix} 0 & -\cos(\phi)\cos(\theta) & \sin(\phi)\cos(\theta) \\ \cos(\phi)\cos(\theta) & 0 & \sin(\theta) \\ -\sin(\phi)\cos(\theta) & -\sin(\theta) & 0 \end{bmatrix} \end{bmatrix} \\ &= \begin{bmatrix} 1 & 0 & -\sin\theta \\ 0 & \cos\phi & -\sin\phi\cos\theta \\ 0 & \sin\phi & \cos\phi\cos\theta \end{bmatrix}\end{aligned}$$

推导代码

下面的代码使用 [sympy](#) 推导转换矩阵 T

```
1 import sympy as sp
2 phi, theta, psi = sp.symbols('\phi, \theta, \psi')
3 Rx = sp.Matrix([
4     [1, 0, 0],
5     [0, sp.cos(phi), -sp.sin(phi)],
6     [0, sp.sin(phi), sp.cos(phi)],
7 ])
8 Ry = sp.Matrix([
9     [sp.cos(theta), 0, sp.sin(theta)],
10    [0, 1, 0],
11    [-sp.sin(theta), 0, sp.cos(theta)],
12 ])
13 Rz = sp.Matrix([
14     [sp.cos(psi), -sp.sin(psi), 0],
15     [sp.sin(psi), sp.cos(psi), 0],
```

另外使用这一代码时，只需要改代码第18行处3次旋转的顺序就可以方便地计算出气角角微分方程（见 [攻角、侧滑角、倾侧角与欧拉角的关系](#)）

$$\begin{aligned}\vec{w} &= \begin{bmatrix} \cos\beta\cos\alpha & 0 & -\sin\alpha \\ -\sin\beta & 1 & 0 \\ \sin\alpha\cos\beta & 0 & \cos\alpha \end{bmatrix} \begin{bmatrix} \dot{\sigma} \\ \dot{\alpha} \\ \dot{\beta} \end{bmatrix} \\ \begin{bmatrix} \dot{\sigma} \\ \dot{\alpha} \\ \dot{\beta} \end{bmatrix} &= \begin{bmatrix} \frac{\cos\alpha}{\cos\beta} & 0 & \frac{\sin\alpha}{\cos\beta} \\ \cos\alpha\tan\beta & 1 & \sin\alpha\tan\beta \\ -\sin\alpha & 0 & \cos\alpha \end{bmatrix} \vec{w}\end{aligned}$$

仿真

仿真一下来验证sympy推导出的结果正确，将欧拉角微分方程的仿真结果与四元数微分方程的仿真结果比较。
使用欧拉角微分方程建模的被控对象代码如下

```
1 # rigidbodyeuler.py
2 # 四阶龙格库塔法根据欧拉角微分方程建立刚体的运动学和动力学模型
3 import numpy as np
4 # from quaternions import *
5
6 # 刚体
7 # J: 刚体的转动惯量矩阵
8 # initEuler: 刚体的初始欧拉角
9 # initOmega: 刚体的初始角速度向量
10 class RigidBody:
11     def __init__(self, J, initEuler):
12         self.J = J
```



找不到服务器znh

关注

18 21 0

```
13     e1 = initEuler
14     w1 = initOmega
15     self._t = 0
16     self._Jinv = np.linalg.inv(self._J)
17     self._states = np.concatenate((e1, w1), dtype=float)
18     self.ctrlLaw = lambda x: 0 # 控制律
19
20     def Simulate_OneStep(self):
21         h = 0.01
22         K1 = self._ODE4Function(self._t, self._states)
23         K2 = self._ODE4Function(self._t+h/2, self._states + h/2*K1)
24         K3 = self._ODE4Function(self._t+h/2, self._states + h/2*K2)
25         K4 = self._ODE4Function(self._t+h, self._states + h*K3)
26         dx = h/6*(K1 + 2*K2 + 2*K3 + K4)
27         self._states += dx
28
29     def Get_State(self):
30         return self._states
31     def Get_EulerVec(self):
32         # q1 = Euler_To_Quaternion(self._states[0:3])
33         # return Quaternion_to_Euler(q1)
34         return self._states[0:3]
35
36     def _ODE4Function(self, t, x):
37         E, W = x[0:3], x[3:6]
38         matTR = np.matrix([
39             [1, 0, -np.sin(E[1])],
40             [0, np.cos(E[0]), np.sin(E[0])*np.cos(E[1])],
41             [0, -np.sin(E[0]), np.cos(E[0])*np.cos(E[1])],
42         ])
43         # matTR = np.matrix([
44         #     [1, np.sin(E[0])*np.tan(E[1]), np.cos(E[0])*np.tan(E[1])],
45         #     [0, np.cos(E[0]), - np.sin(E[0])],
46         #     [0, np.sin(E[0]) / np.cos(E[1]), np.cos(E[0]) / np.cos(E[1])],
47         # ])
48         dE = np.linalg.inv(matTR) @ W
49         # dE = matTR @ W
50         torque = self.ctrlLaw(x)
51         dW = self._Jinv @ (torque - np.cross(W, self._J @ W))[0]
52         return np.concatenate((np.array(dE)[0], np.array(dW)[0]))
```

四元数微分方程代码见 [刚体四元数姿态控制](#),
主程序如下

```
1 # main.py
2 import numpy as np
3 import matplotlib.pyplot as plt
4 # from rigidbodyquaternion import RigidBody
5 from rigidbodyeuler import RigidBody
6 J = np.matrix([
7     [1, 0, 0],
8     [0, 10, 0],
9     [0, 0, 3],
10 ])
11 usv1 = RigidBody(J, np.array([1, -0.2, 0.3]), np.array([-0.1, 0.2, -0.3]))
12 t = 0
13 plottime, plotdata = [], []
14 while t < 10:
15     for n in range(10):
```

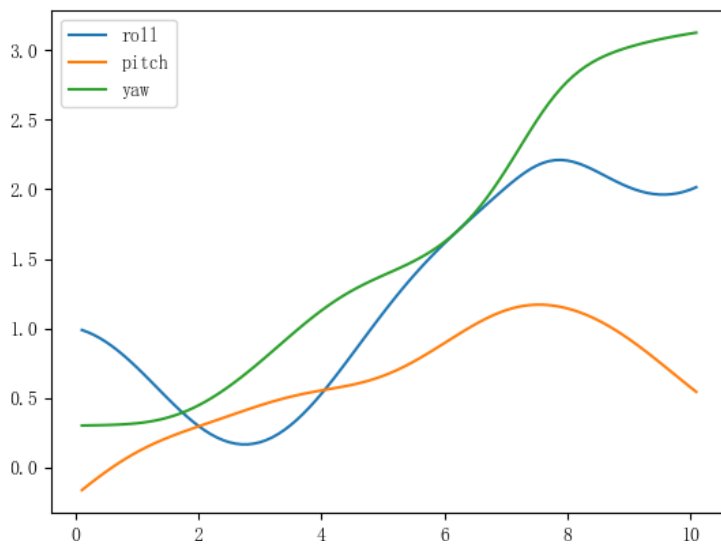
四元数和欧拉角微分方程的仿真结果相同，如图所示。



找不到服务器zhn

关注

18 21 0

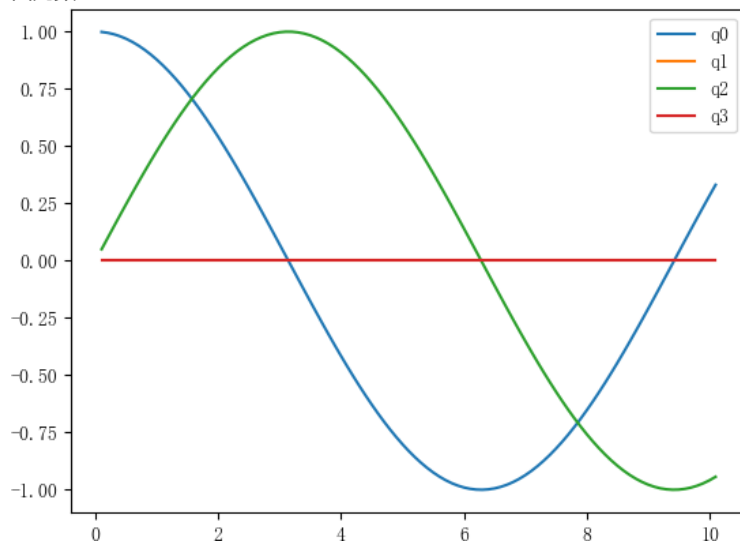


欧拉角的缺点

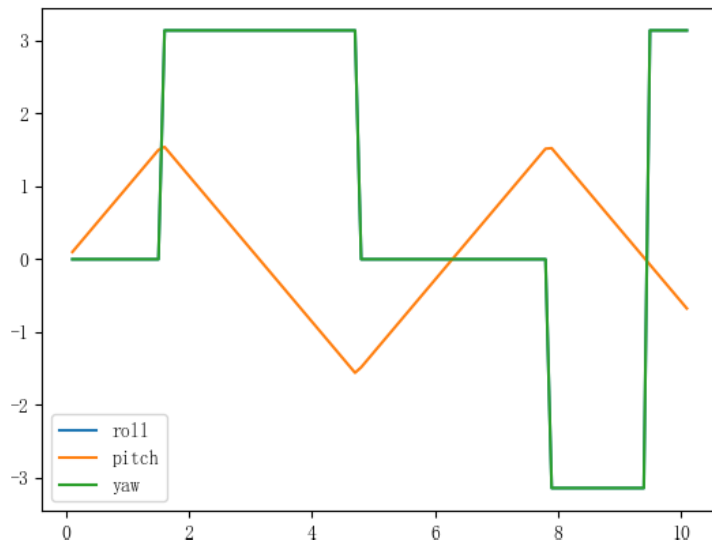
可以说欧拉角除了看上去更直观以外几乎没有任何优点了。本文从欧拉角微分方程的角度谈谈欧拉角的其中一个缺点：在限定的范围以外知道的是，四元数、欧拉角、旋转矩阵这三种描述姿态的方法之间可以互相换算，也几乎可以一一对应，之所以说“几乎”就是因为，如果旋转角应关系就出了一些问题。

先看看当绕俯仰轴(Y轴)旋转一圈时，3种姿态表示方法的表现。

四元数：



欧拉角：



旋转矩阵：



找不到服务器zhu

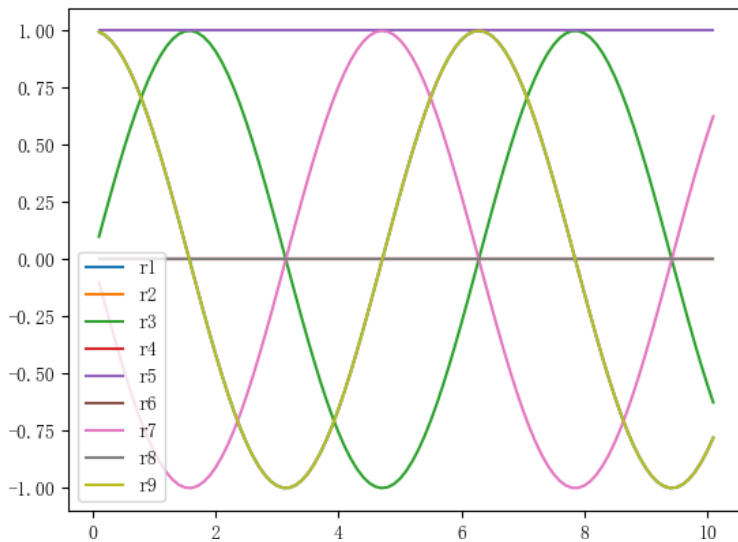
关注

18



21

0



由此可见，在大角度机动时欧拉角存在严重的不连续问题，应尽量避免使用。

画这幅图使用四元数微分方程的代码，把控制律设为0，初始欧拉角为全0，初始角速度为 $[0, 1, 0]$ 。如果使用欧拉角微分方程，则需要把俯仰角限制在 $[-\pi/2, \pi/2]$ 以内。也可以直接简单地使用欧拉角转一圈，同时转四元数和旋转矩阵出图，但此时我就发现了欧拉角转四元数的问题。

下面的代码是最常见的欧拉角转四元数的代码

```
1 def Euler_To_Quaternion(euler):
2     from numpy import sin, cos
3     cr = cos(euler[0] * 0.5)
4     sr = sin(euler[0] * 0.5)
5     cp = cos(euler[1] * 0.5)
6     sp = sin(euler[1] * 0.5)
7     cy = cos(euler[2] * 0.5)
8     sy = sin(euler[2] * 0.5)
9     q0 = cy * cp * cr + sy * sp * sr
10    q1 = cy * cp * sr - sy * sp * cr
11    q2 = sy * cp * sr + cy * sp * cr
12    q3 = sy * cp * cr - cy * sp * sr
13    return np.array([q0, q1, q2, q3])
```

这一代码出自下面这个公式

$$Q = \begin{bmatrix} \cos(\psi/2) \\ 0 \\ 0 \\ \sin(\psi/2) \end{bmatrix} \begin{bmatrix} \cos(\theta/2) \\ 0 \\ \sin(\theta/2) \\ 0 \end{bmatrix} \begin{bmatrix} \cos(\phi/2) \\ \sin(\phi/2) \\ 0 \\ 0 \end{bmatrix}$$

$$= \begin{bmatrix} \cos(\phi/2)\cos(\theta/2)\cos(\psi/2) + \sin(\phi/2)\sin(\theta/2)\sin(\psi/2) \\ \sin(\phi/2)\cos(\theta/2)\cos(\psi/2) - \cos(\phi/2)\sin(\theta/2)\sin(\psi/2) \\ \cos(\phi/2)\sin(\theta/2)\cos(\psi/2) + \sin(\phi/2)\cos(\theta/2)\sin(\psi/2) \\ \cos(\phi/2)\cos(\theta/2)\sin(\psi/2) - \sin(\phi/2)\sin(\theta/2)\cos(\psi/2) \end{bmatrix}$$

同样地绕y轴转一圈看看欧拉角转四元数式(1)的表现。首先可以求出四元数随时间变化的解析解。设角速度为 $\vec{\omega} = [0, 1, 0]^T$ ，四元数初值为 $[1, 0, 0, 0]^T$ ，代入四元数微分方程



找不到服务器zhn

关注

18



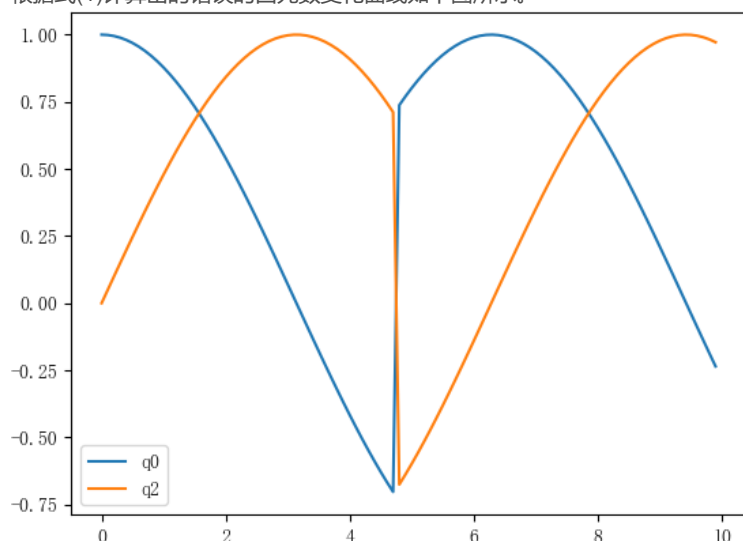
21

0

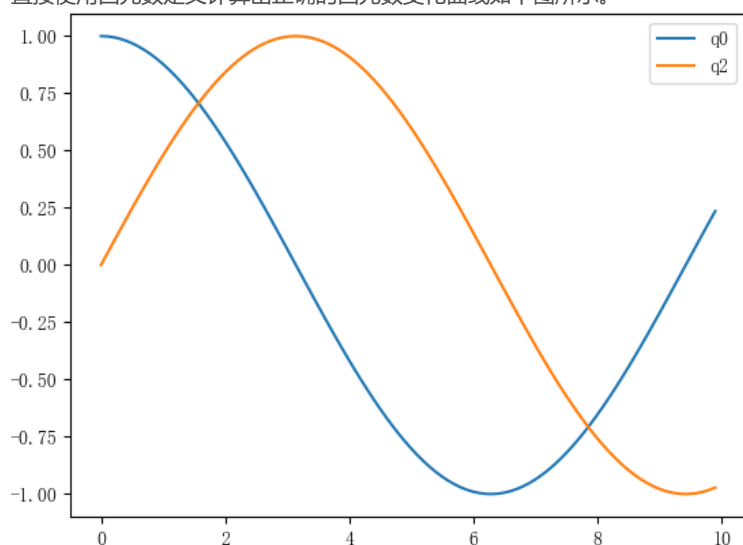
$$\begin{aligned} \begin{bmatrix} \dot{q}_0 \\ \dot{q}_1 \\ \dot{q}_2 \\ \dot{q}_3 \end{bmatrix} &= \frac{1}{2} \begin{bmatrix} 0 & -w_x & -w_y & -w_z \\ w_x & 0 & w_z & -w_y \\ w_y & -w_z & 0 & w_x \\ w_z & w_y & -w_x & 0 \end{bmatrix} \begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{bmatrix} \\ &= \frac{1}{2} \begin{bmatrix} 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & -1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{bmatrix} \\ \dot{q}_1 &= -0.5q_3 = 0 \\ \dot{q}_3 &= 0.5q_1 = 0 \\ \dot{q}_2 &= 0.5q_0 \\ \ddot{q}_0 &= -0.5\dot{q}_2 = -0.25q_0 \\ q_0(t) &= C_1 \cos \frac{t}{2} + C_2 \sin \frac{t}{2} = \cos \frac{t}{2} \\ q_2(t) &= \sin \frac{t}{2} \end{aligned}$$

这个解也很直观地表现出四元数的变化过程，但是这个结果跟式(1)的不同之处在于少了横滚角 ϕ 和偏航角 ψ ，因此在大角度机动时根据式(1)连续了，这不是四元数的问题，而是式(1)的问题。

根据式(1)计算出的错误的四元数变化曲线如下图所示。



直接使用四元数定义计算出正确的四元数变化曲线如下图所示。



出图代码如下，其中 `quaternions.py` 的代码见 [自用的四元数、欧拉角、旋转矩阵转换代码](#)。

```
1 import numpy as np
2 import matplotlib.pyplot as plt
3 from quaternions import *
```



找不到服务器zhn

关注

18

21

0

```
4 plottime, plotq0, plotq2 = [], [], []
5 for n1 in range(100):
6     t = n1 * 0.1 # 角度从0到10弧度(4.7124弧度是-y轴)
7     vecx = np.array([np.cos(t), 0, -np.sin(t)]) # 组成旋转矩阵的x轴坐标
8     vecy = np.array([0, 1, 0]) # 组成旋转矩阵的y轴坐标
9     vecz = np.array([np.sin(t), 0, np.cos(t)]) # 组成旋转矩阵的z轴坐标
10    R = np.vstack([vecx, vecy, vecz]).T # 旋转矩阵
11    E = Rotation_to_Euler(R) # 由旋转矩阵计算欧拉角
12    # Q = Euler_To_Quaternion(E) # 由欧拉角计算四元数 (有问题的转换公式)
13    Q = [np.cos(t/2), 0, np.sin(t/2), 0] # 直接使用四元数定义 (正确的结果)
14    plottime.append(t)
```

欧拉角速率和机体角速度转换

欧拉角速率和机体角速度转换的详细推导

机器人旋转矩阵与欧拉角转换公式

六轴机器人空间旋转矩阵与欧拉角之间转换公式，算法在实际应用中得到了验证。

TensorView for MATLAB：使用欧拉角解码可视化张量

TensorView for MATLAB 是一个基于 GUI 的可视化工具，用于将二阶笛卡尔张量描述为三维分子模型上的表面。支持椭圆体和椭圆张量显示格式，该软件允许从张量中提取特征值、特征向量和特征平面。

基于欧拉角的方向联轴节速比关系推导

基于欧拉角的方向联轴节速比关系推导，李春明，，根据欧拉角在右手坐标系中推导的方向联轴节两轴之间转速关系，可修正教材中的公式。以主动轴为研究对象

方向余弦矩阵求欧拉角解算

方向余弦矩阵求欧拉角

欧拉角与四元数相互转换.docx

欧拉角与四元数相互转换知识点总结 一、四元数的定义和性质 * 四元数是一种数学概念，用于描述三维空间中的旋转和运动。 * 四元数可以用 $q=[w, x, y, z]^T$ 表示

人脸识别欧拉角及位移矩阵计算及求解

接下来，我们可以推导出欧拉角计算的公式： $R = R_1 * R_2 * R_3$ 其中， R 是总的旋转矩阵。三、位移矩阵求解 位移矩阵是人脸识别中另一个关键步骤。位移矩阵：

欧拉角的概念理解和欧拉角旋转矩阵推导

qq_4332

欧拉角用来计算空间中刚体的旋转位置，目的是改变刚体的朝向. 具体来说,空间中有一个点p和一根轴k, 点p绕轴k旋转 θ 角度到p',求p'的坐标.这就是欧拉角要解决的问题。

一步步推导由欧拉角到旋转矩阵的计算过程

Du

文章目录为了便于理解，首先进行二维坐标系中的公式推导。如图坐标系OX1Y1OX_1Y_1OX1Y1经过逆时针旋转 θ 角变换为坐标系OX2Y2OX_2Y_2OX2Y2:

欧拉角速率与机体角速度转换详细推导

weixin_396f

根据旋转矩阵及绕各个轴旋转的角速度，推导机体角速度 旋转矩阵旋转矩阵还不清楚的同学去看我的另一篇博客，这里咱们废话不多说，旋转矩阵已知 欧拉角 大

四元数与欧拉角的转换

想想0

两种大地坐标系下的欧拉角 东北地坐标系下四元数与欧拉角的转换东北天坐标系下四元数与欧拉角的转换四元数计算不同坐标系下的欧拉角

文本(2024-06-23 161043).txt

文本(2024-06-23 161043).txt

PSO_VMD_MCKD 基于PSO_VMD_MCKD方法的风机轴承微弱函数.rar

PSO_VMD_MCKD 基于PSO_VMD_MCKD方法的风机轴承微弱故障诊断。为实现 VMD 和 MCKD 的参数自适应选择，采用粒子群优化算法对两种算法中的参数进行优化。

计算机软考高级真题2012年上半年 系统分析师 综合知识.docx

考试资料，计算机软考，系统分析师高级，历年真题资料，WORD版本，无水印，下载。

THE CACHE MEMORY BOOK

THE CACHE MEMORY BOOK

IMG_20240623_224516.jpg 最新发布

IMG_20240623_224516.jpg

sxs-win11.zip 是一个与 Windows 11 相关的压缩文件

bootstrap卡片排版这是Windows 11系统中sxs文件包的镜像，它适用于.NET 3.5的安装过程。。内容来源于网络分享，如有侵权请联系我删除。另外如果没有积分

paramiko-3.2.0-py3-none-any.zip

paramiko-3.2.0-py3-none-any.zip

欧拉角顺序为xyz,欧拉角速度和机体角速度的关系

欧拉角是一种表示刚体运动状态的方式，欧拉角顺序为xyz



找不到服务器zhn

关注

18



21

0

“相关推荐”对你有帮助？

非常没帮助

没帮助

一般

有帮助

非常有帮助

关于我们 招贤纳士 商务合作 寻求报道 400-660-0108 kefu@csdn.net 在线客服 工作时间 8:30-22:00

公安备案号11010502030143 京ICP备19004658号 京网文〔2020〕1039-165号 经营性网站备案信息 北京互联网违法和不良信息举报中心 家长监护 网络110报警服务 中国互联网举报中心 Chrome商店下载 账号管理规范 版权与免责声明 版权申诉 出版物许可证 营业执照

©1999-2024北京创新乐知网络技术有限公司



找不到服务器zhn

码龄8年

暂无认证

98

6万+

6万+

14万+



原创

周排名

总排名

访问

等级

1425

203

228

73

1077

积分

粉丝

获赞

评论

收藏



私信

关注

大额流量券送不停

多发多得，流量翻倍！

去查看

搜博主文章



热门文章

latex多行公式加大括号、整体编号及多行编号及不同方法的区别 26405

stm32用定时器通过驱动器控制多个步进电机 11349

pytorch加载预训练模型与自己模型不匹配的解决方法 6740

演化博弈、复制动态方程与仿真 6713

降维状态观测器的直观理解与仿真 3952

分类专栏



自用代码

8篇



控制

20篇



算法博弈论

4篇



自用的数学笔记

13篇



基于特征模型的全系数自...

4篇



论文相关

10篇



最新评论

非线性系统的反馈线性化



找不到服务器zhn

关注

18



21

0

m0_45895842: 其中 $\eta = \eta(x)$ 需要满足抵消条件, 请问一下满足这个抵消条件是为...

特征模型(3)证明全系数之和等于1

找不到服务器zhn: 我直接在原文里补充上了

特征模型(3)证明全系数之和等于1

qq_34892665: 请问微分方程系数与极点 and 一阶系统时间常数的关系是怎么来的呢

角速度变化时四元数和旋转矩阵微分方程...

找不到服务器zhn: 对, 我认为四元数、欧拉角、旋转矩阵等价, 只差一个转换关系...

角速度变化时四元数和旋转矩阵微分方程...

Apologize-AN: 求教一下, 您导出角速度在世界系下表示的四元数微分方程 \dot{Q} 点 = ...

您愿意向朋友推荐“博客详情页”吗?



强烈不推荐



不推荐



一般般



推荐



强烈推荐

最新文章

LATEX中优化问题如何排列 max—s.t. 格式

什么是P问题、NP问题和NPC问题

证明四元数乘法与旋转矩阵乘法等价

2024年 3篇

2023年 28篇

2022年 29篇

2021年 47篇

目录

欧拉角微分方程推导

推导代码

仿真

欧拉角的缺点



找不到服务器zhn

关注

18



21

0