

无人机开发指南

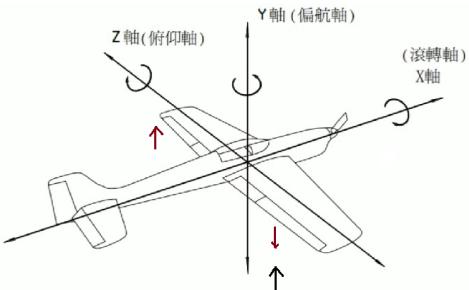
作者：向仔州

目录

固定翼飞机空气动力学.....	2
浆叶气流情况.....	4
飞机螺旋桨介绍.....	5
四旋翼飞机运动控制简介.....	9
直升机旋翼控制简介.....	11
锂电池特征.....	12
锂电池充电三种方式.....	13
无人机螺旋桨电机的选择.....	15
空心杯有刷电机.....	15
空心杯有刷原理介绍.....	17
BLDC 无刷电机	19
BLDC 无刷电机原理介绍	20
无人机天线选择.....	22
四旋翼无人机飞控系统设计.....	25
油门曲线分段线性方法.....	26
无人机控制逻辑.....	27
无人机姿态控制.....	27
四元数使用(第 1 部分)	34
无人机飞控算法实现.....	43
无人机飞控算法代码实现.....	47
遥控器油门数据，角度数据获取处理.....	47
无人机控制系统代码设计.....	60

固定翼飞机空气动力学

1. 固定翼飞机空气动力学



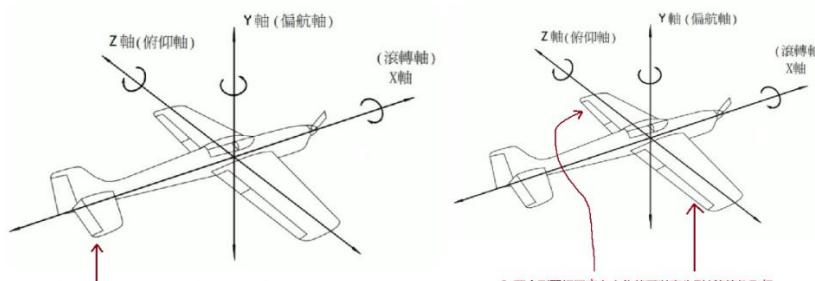
3. 固定翼飞机主要靠装在机头的螺旋桨产生推力

4. 推力 > 阻力，飞机向前飞

5. 对于固定翼飞机来说，一旦速度为0，飞机就没有升力了。

因为固定翼飞机的升力就是靠推力的产生，出现向后走的气流，这个气流在机翼上面产生的速度差，导致升力产生。

6. 固定翼飞机，有抬升，下降，左右拐弯，横滚这几种飞行方式。



10. 伯努利定律

伯努利定律是空气动力最重要的公式，简单的说流体的速度越大，静压力越小，速度越小，静压力越大，这里说的流体一般是指空气或水，在这里当然是指空气，设法使机翼上部空气流速较快，静压力则较小，机翼下部空气流速较慢，静压力较大，两边互相较力（如图1-3），于是机翼就被往上推去，然后飞机就飞起来，以前的理论认为两个相邻的空气质点同时由机翼的前端往后走，一个流经机翼的上缘，另一个流经机翼的下缘，两个质点应在机翼的后端相会合（如图1-4），经过仔细的计算后发觉依上述理论，上缘的流速不够大，机翼应该无法产生那么大的升力，现在经风洞实验已证实，两个相邻空气的质点流经机翼上缘的质点会比流经机翼的下缘质点先到达后缘（如图1-5）。

这句话怎么理解？

比如我开车，车速很低的时候，车子感觉很重，很稳，

当车子速度很快的时候，车子容易飘，抓地力不稳。

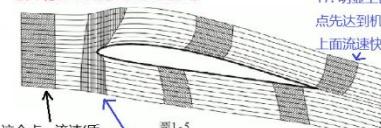
如果我车子停着不动，这时候去打方向盘，发现方向盘很重，很费力很难转动。

所以当速度静止的时候，压力是最大的。

11. 这是空气经过飞机翅膀，被切割成了上下两股流体。



12. 这个飞机翅膀的结构决定了，下面的空气流动慢一些。下面的空气流动慢，那么下面的空气跟翅膀下部分结合的压力就比较大。



15. 这个点，流速(质点)是一样的。

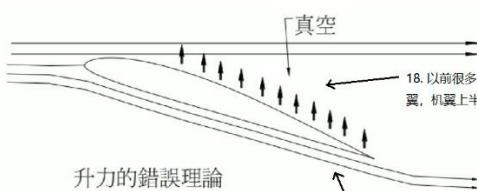
16. 质点从这里分开

13. 上面的空气流动快，翅膀与空气压力小。



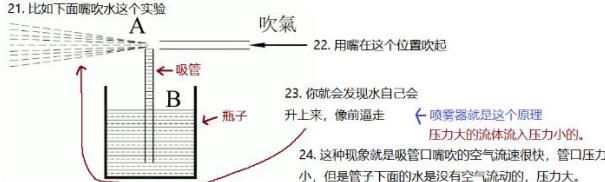
14. 翼下面压力大，上面压力小，根据力的合成，合力还是向上的。

所以：下面压力 - 上面压力 > 0 就是升力



20. 实际上，并不是因为机翼上边真空造成升力，而是机翼上边的压力 < 机翼下边的压力造成升力。

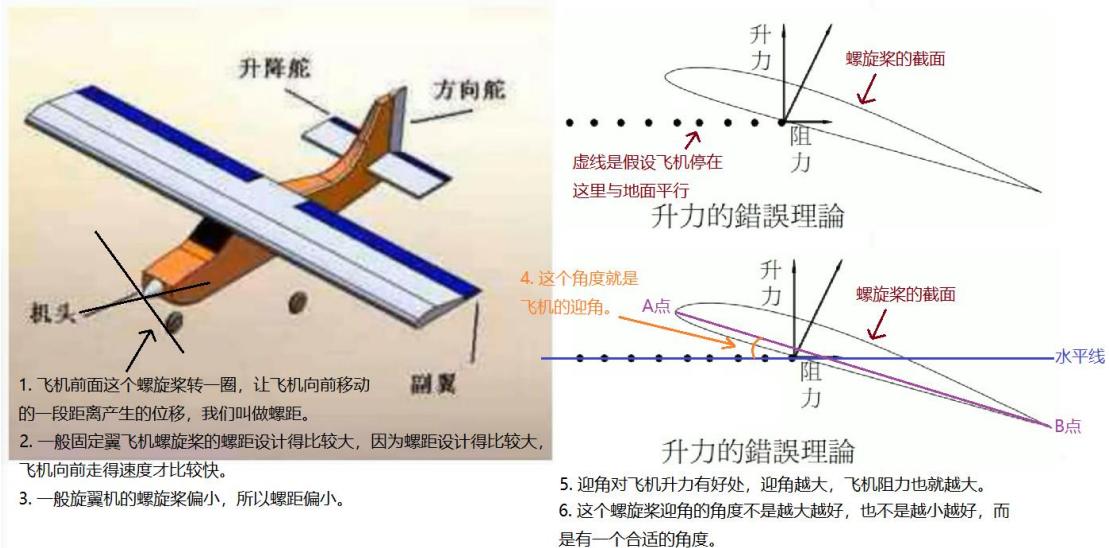
21. 比如下面嘴吹水这个实验



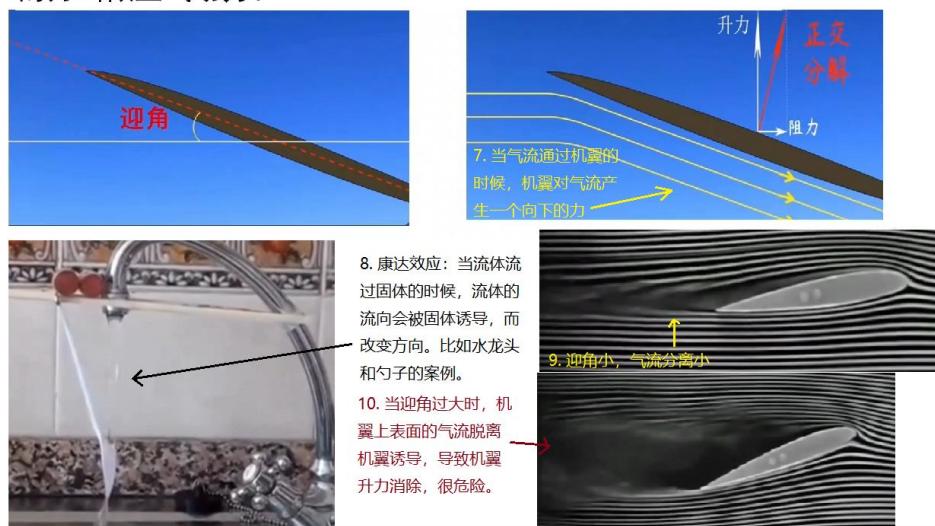


25. 如果副翼, 方向舵, 升降舵复位保持平行之后, 这时候飞机应该不会去转弯的, 如果方向舵, 升降舵, 副翼我都没有动它, 它们处于复位状态, 这时候飞机下降到地面, 在地面滑行的时候出现偏向现象, 那么这个飞机的空气动力设计是有问题的。

所以为什么飞机起飞的时候要在地面高速滑行一段时间, 就是看飞机各个翅膀有没有缺陷。

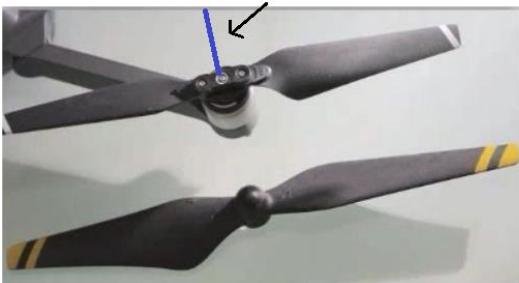


当一个物体在空气中运动, 或者空气从物体表面流过得时候, 空气对物体都会产生作用力, 我们把空气这种作用在物体上的力叫做空气动力。

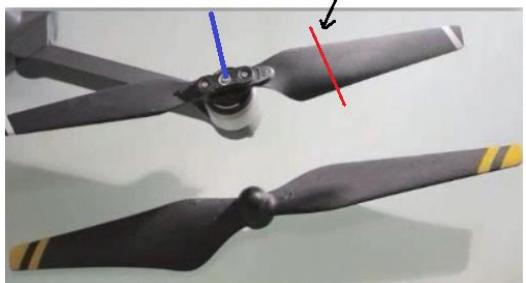


浆叶气流情况

1. 假设这是螺旋桨电机轴



2. 如果我将翅膀这里剪开, 得到这点刨面



3. A面和B面的角速度是相等的, 什么是角速度?



4. 角速度就是A面和B面每分钟旋转的角度



5. A面和B面线速度不相等, 什么是线速度?

线速度就是翅膀刨面旋转的周长。线速度与半径有关。



B面旋转半径大



6. 所以B点线速度 > A点线速度。

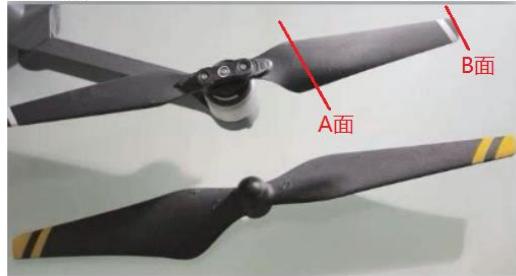
7. 所以之前我们固定翼飞机的两遍机翼线速度是一样的。

因为机翼是按照固定方向向前走。AB两点线速度相等。

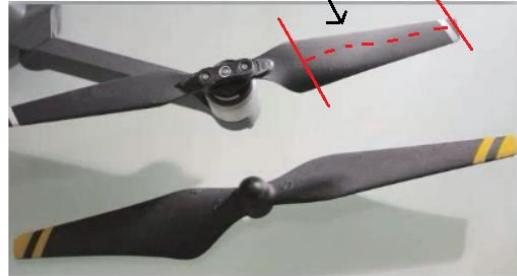
8.但是对于螺旋桨飞机, 线速度就不一样了。

因为螺旋桨飞机(如四轴飞行无人机), A 面和 B 面的线速度不一样,

9. 因为A面和B面线速度不一样，我们在AB两个面之间取无数个点。

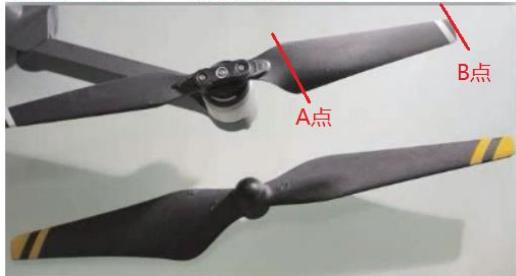


10. 这就是AB两面之间取无数个点，每个点的线速度都不一样。

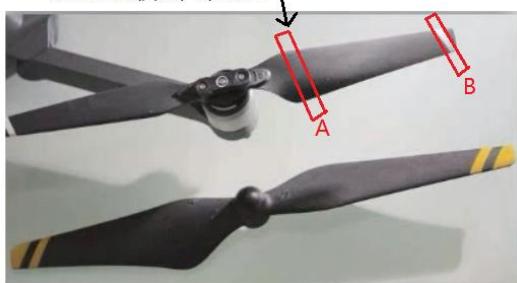


11. 所以说螺旋桨前进方向产生的升力，一定等于AB面之间所有点的线速度积分之和。

现在将AB面改成AB点来讨论方便些



12. 我们现在将A点和B点各取一块面积，记住是面积哦，来讨论



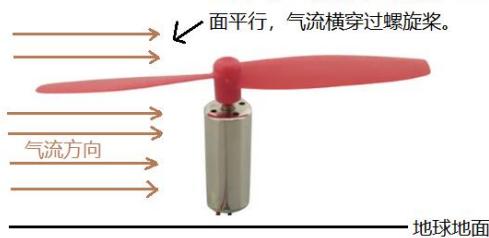
飞机螺旋桨介绍

轴向速度的理解

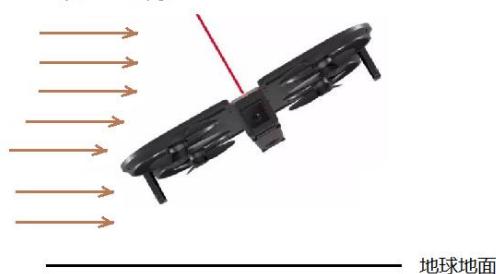


气流角的理解

1. 这是第一种情况，螺旋桨与地



3. 螺旋桨左倾斜，与地球地面也是没处于平行状态，与气流产生夹角。



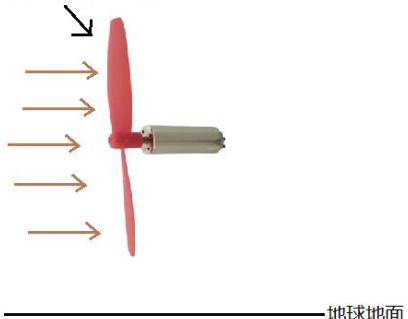
5. 气流流向是水平面，螺旋桨旋转面也是水平面，那么它们之间的气流角 φ 为0度



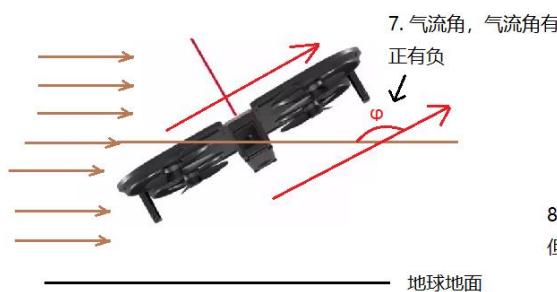
2. 无人机在飞行过程中，飞机在调整方向时会出现倾斜，这时候螺旋桨与地面也就形成了倾斜。那么气流角也会出现倾斜现象。



4. 这是最恶劣的情况，螺旋桨与地面垂直了，气流直接打在螺旋桨上。

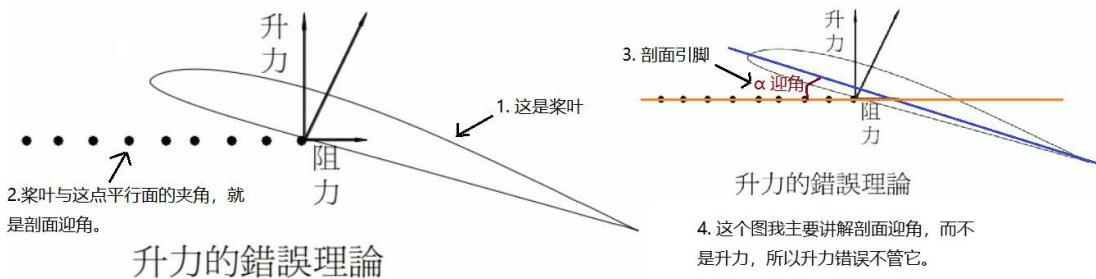


7. 气流角，气流角有正有负



8. 还有情况是，螺旋桨是平行的，但是气流进的方向不是平行的。

桨叶的剖面迎角



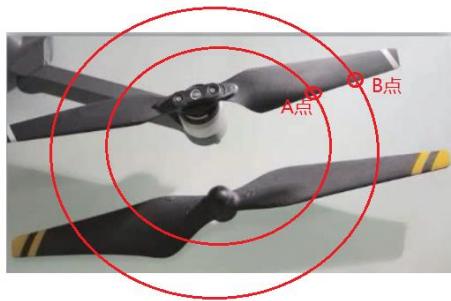
桨叶角

$$\text{桨叶角} \beta = \alpha(\text{剖面迎角}) + \varphi(\text{气流角})$$

φ (气流角): 根据飞机倾斜方向不一样, φ 可能正, 可能负。

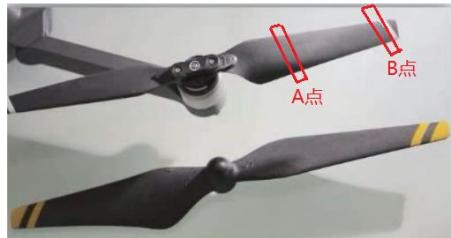
升力, 阻力

1. 空气流过桨叶各小段时产生的气动力为阻力 ΔD , 升力 ΔL , 合成后总空气动力为 ΔR .

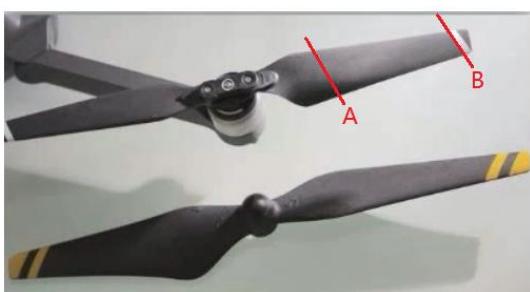


A点和B点线速度是不一样的。

线速度不一样, 那边
A点和B点的阻力就不
一样。



2. 如果A点和B点迎角是同一个角度, 那么
桨叶在高速旋转的时候, B点阻力绝对会比
A点大。因为B点半径大, 速度快阻力偏
大。A点半径小, 所以阻力偏小。



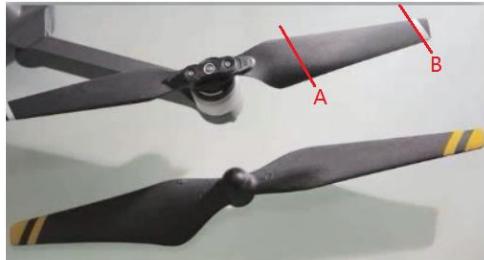
3. 如何减小B点的阻力呢?

减小B点的迎角, A点因为线速度最低,
所以增大A点迎角。这样转起来B点的阻
力就降低了。整个螺旋桨效率就提高了。



4. 发现没有, 整个桨叶这一片, 每一段扭
曲得都不一样。

5. 注意: 固定翼飞机不存在这个问题, 翼膀的迎
角每一段都是一样的, 因为固定翼飞机整个翅膀
的阻力是一样的。只有旋转的桨叶才有阻力不均匀的情况。

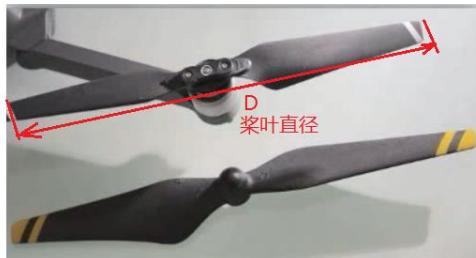


3. 如何减小B点的阻力呢?
减小B点的迎角，A点因为线速度最低，所以增大A点迎角。这样转起来B点的阻力就降低了。整个螺旋桨效率就提高了。



4. 发现没有，整个桨叶这一片，每一段扭曲得都不一样。
5. 注意：固定翼飞机不存在这个问题，翅膀的迎角每一段都是一样的，因为固定翼飞机整个翅膀的阻力是一样的。只有旋转的桨叶才有阻力不均匀的情况。
6. 所以螺旋桨设计桨叶各剖面在升阻力比较大的迎角工作，才能获得较大拉力，较小的阻力距。

6. 螺旋桨的功率，它与转速的平方成正比，转速越快，桨的功率越大，升力越快。



7. 螺旋桨功率与桨叶直径D的立方成正比关系。
所以桨直径的旋转与转速有很大的关系。

8. A点线速度小，升力小，B点线速度大，升力大。
但是B点因为靠近桨边，会遇到乱流，所以B点升力也就不高了。只有B点边上某一个点升力很大。

9. 所以我们用从桨叶根部到边上70%的位置，在这个位置的桨叶迎角度表示桨的型号，比如10度浆，20度浆，30度浆，等等...，桨叶设计有专门的人来设计。

浆的效率高不高，决定了无人机飞行时间和飞行距离。

10. 因为浆的根部迎角比较大，边上的迎角比较小。



11. 因为根部迎角大，边上迎角小，所以在设计浆的过程中，桨叶迎角也就设计到十几度，不要超过20度，因为角度超过20度，阻力就开始变大。浆边上设计迎角角度就是十几度到几度渐变。

12. 气流角实际上反映的是前进速度和切线速度的比值，用进矩比“J”来反应螺旋桨，浆尖处气流角大小。

$$J = V / nD$$

J：浆尖处气流角

V：浆轴向上速度

D：桨叶整个直径

n：桨叶转速

13. 螺旋桨的拉力(T)

克服螺旋桨阻力矩所需的功率(P)

效率(η)

$$(拉力)T = C_t * p * n * 2 * D * 4$$

C_t: 拉力系数

p: 空气密度

n: 螺旋桨转速

D: 螺旋桨直径

$$(阻力矩功率)P = C_p * p * n * 3 * D * 5$$

C_p: 功率系数(浆不一样，功率系数不一样)

14. 螺旋桨直径是性能重要参数之一

浆的直径增大，拉力(T)也随之增大，效率提高。

效率提高是什么意思？

效率提高的意思就是，浆直径大，提供相同的升力情况下，浆转速可以下降一些。浆转速下降浆的阻力也会减小，效率提高。

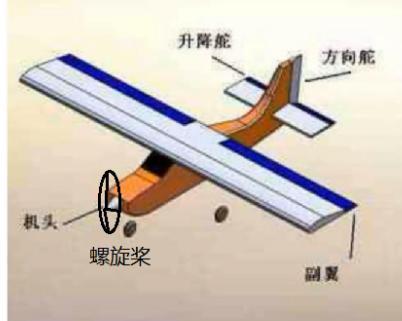
15. 所以在结构允许的情况下，尽量选择直径大的浆，但是直径大的浆实际上扭力也大了，虽然直径大的浆在同样升力的情况下，转速相对于直径小的浆可以变得很低。但是因为大浆转速低，扭矩就会增大，导致电机直径也会变大。

18. 实度(σ): 浆叶面积与螺旋桨旋转面积(πR^2)的比值，它的影响与浆叶数目影响相似。

一般经验认为实度取0.1比较合适。投影的面积 / $\pi R^2 = 0.1$ 比较合适

19. 螺距：它是桨叶的另外一种的表示方式，螺旋桨旋转一周飞机前进的距离。

螺距在有些情况会对飞机产生阻力，以固定翼飞机为例，比如我螺旋桨每分钟转1圈，飞机就向前飞1米，如果螺旋桨每分钟转600圈，那么飞机每分钟就飞600米。如果飞机处于滑行的时候，飞机飞行速度突然变成了每分钟1000米，但是我飞机螺旋桨实际还是按照每分钟600圈(600米)这样旋转。那么这时候螺旋桨对于飞机来说就是阻力。

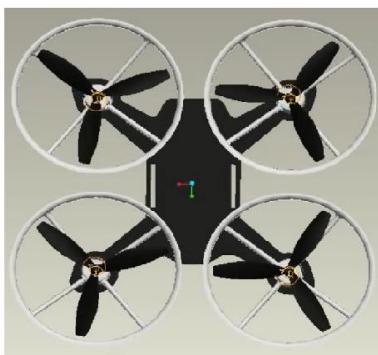


那么，螺距和飞机实际飞行速度不匹配是什么时候发生的呢？

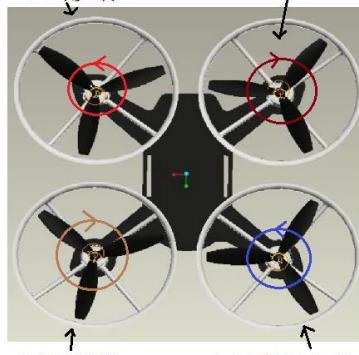
飞机在停飞，起飞阶段都不会出现螺距与飞行速度不匹配，只有在飞机下降滑行的时候会出现不匹配。飞机俯冲也是不匹配。

四旋翼飞机运动控制简介

1. 四旋翼无人机要实现平衡，怎么做？



2. 左上角正转



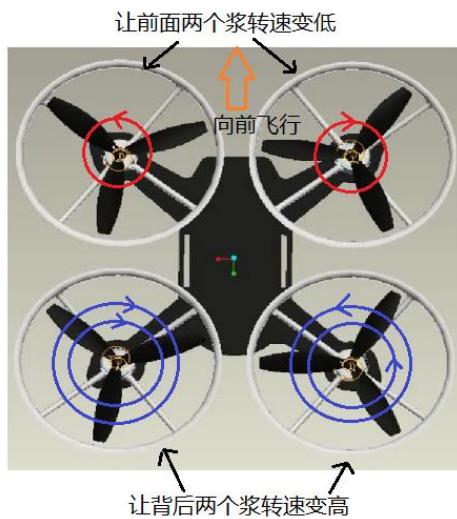
4. 右上角反转



5. 左下角反转

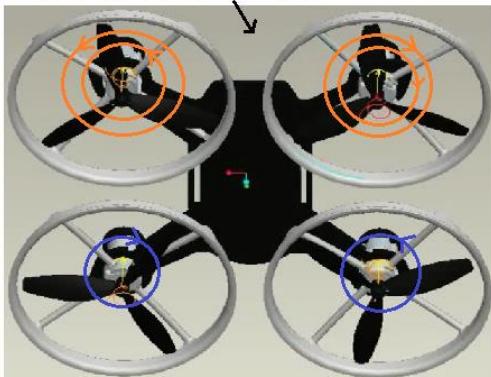
6. 这样旋翼对角转动，飞机才能保持平衡，力平衡

7. 如何让飞机向前运动。



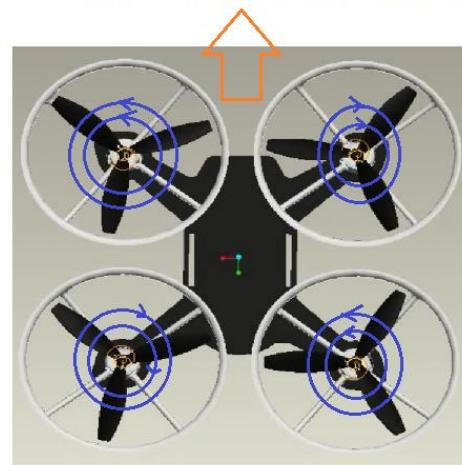
8. 前螺旋桨转速低，后螺旋桨转速高，飞机向前飞行，但是注意：螺旋桨旋转方向还是对角对称。对称相邻螺旋桨是反向旋转。
螺旋桨方向和前面讲的没变。

11. 如果前两个旋翼转速提升，后两个旋翼转速降低，无人机就是向后飞行。



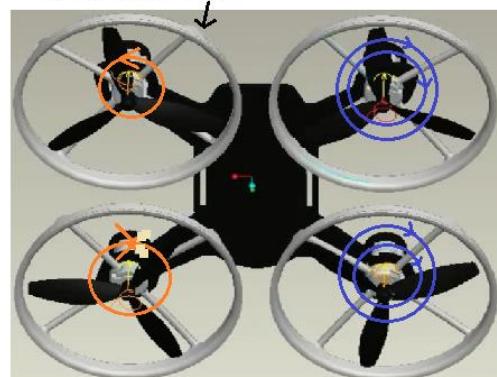
12. 因为无人机是根据陀螺仪来决定螺旋桨转速和前后螺旋桨转速差异的，所以需要用到一些滤波算法，如卡尔曼滤波，PI滤波等。

9. 在飞机的前进过程中，可能飞机会出现高度下降。



10. 遇到飞机前进下降的时候，提高四个螺旋桨的转速，这样可以补偿飞机前进时下降的影响。但是背后两个浆转速同时也要保持高于前面两个浆的转速。一般使用陀螺仪来测量飞机是否下降，然后自动调整螺旋桨转速。

13. 如果向左移动，左边旋翼转速变慢，右边旋翼转速变快。



14. 如果无人机向右边移动，那么左边两个浆转速快，右边两个浆转速慢。

直升机旋翼控制简介

1. 如果直升机只有顶部螺旋桨在旋转



2. 直升机没有尾部螺旋桨，只有顶部螺旋桨，那么直升机就会原地打转。



因为顶部螺旋桨转动，空气也会产生阻力阻止顶部螺旋桨转动，导致直升机原地打转。

3. 我在直升机尾部加一个垂直的旋转翼，记住一定是垂直的，与顶部螺旋桨正交垂直的尾部旋转翼。



这个尾翼旋转方向是根据顶部螺旋桨旋转方向来的，如果顶部螺旋桨顺时针转，那么直升机在顺时针打转。尾部翼就要逆时针旋转，抵消掉顶部螺旋桨的力。

4. 直升机螺旋桨上面还有一个平衡锤。



5. 这个平衡锤是以前，在直升机没有陀螺仪的情况下，来探测直升机是否平衡的。



6. 直升机顶部桨叶高速旋转的时候，平衡锤也会高速旋转，平衡锤高速旋转就会产生一个和下的力。这时候如果飞机倾斜一点点，平衡锤可以自动帮你修正飞机平衡。



7. 还有一种就是这种双桨共轴直升机，顶部有两个桨。



8. 这种双桨直升机主要就是解决了不需要尾翼桨，也能正常飞行的问题。



9. 因为直升机只有一个顶部桨旋转，在没有尾翼浆的时候会出现原地打转，所以这里采用双浆，下浆转动方向与上浆转动方向相反，来克服直升机原地打转问题。



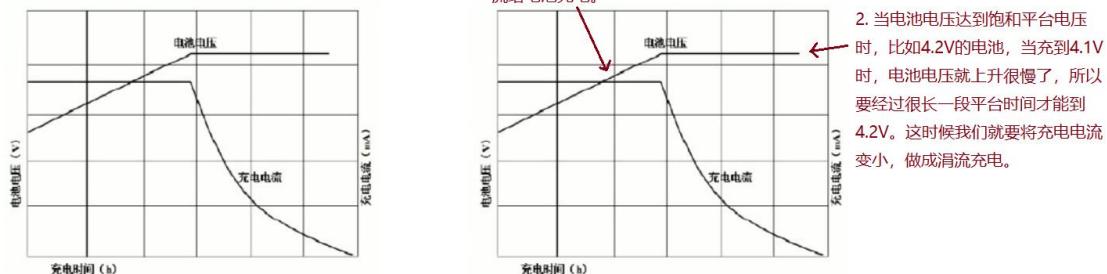
锂电池特征

- 1) 锂电池能量密度比较大。
- 2) 锂电池使用寿命长，2023年之后手机锂电池使用寿命能达到2年以上，性能几乎无衰减。也就是电池充满电的时候电压下降比较少。容量衰减小。
- 3) 锂电池本身自放电很慢，一块锂电池充满电4.2V，然后将充满电的锂电池单独放一个月，还剩4V左右的电压。
- 4) 锂电池充电和放电速度是很快的。
- 5) 锂电池对温度要求比较宽松，但是温度低一点的地方放电能力受影响。
- 6) 锂电池放电肯定是有热量的，电池发热高了会造成锂电池内部元素衰减影响电池寿命，损坏，爆炸。所以使用锂电池的时候我们必须控制住电流，所以锂电池都有一个管理电路板，这个管理电路板是限制锂电池最大输出电流，让电池不至于那么热。
- 7) 锂电池欠压保护，就是锂电池低于3V之后就不能让电池继续放电了，保护电池寿命。如果电池低于3V还在过度放电，电池会被损坏。
- 8) 镍电池每次充电之后必须把之前的电放光，放光之后才可以给镍电池重新充电，但是锂电池就不存在这种情况，任何时候都可以充电放电。
- 9) 锂电池多次充放电之后，容量会变小。比如我们用手机的时候最先给电池充电，电量满格可以用很久，过几年，电量满格就用不到很久了。
- 10) 石墨烯电池，可以实现充放电次数更多，充放电电流更大。
- 11) 为了提高锂电池寿命，充电方式有三种，分段式充电法，限流恒压充电发法，分级定电流充电法。
- 12) 分段式充电法：就是在电池没电的时候怎么充，在电池有电的时候怎么充，在电池充满的时候怎么充。
- 13) 限流恒压充电法：就是把充电电流限制住，让电池碳结构不会崩溃，用恒定电压4.2V去充电。
- 14) 分级定电流充电法：把分段式和限流恒压式的缺点优化就是分级式。

锂电池充电三种方式

分段式充电法

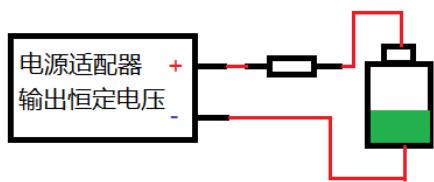
3.2.1 分段式充电法



这种恒流充电方式，可以比较方便的计算电池当前充电容量，使用充电时间 \times 充电电流 = 电荷 Q ，所以恒流可以基本计算出电池容量，在电池电压达到 4.1V 时，用积分方式去计算累计容量。所以分段式充电好计算。

电池充电为什么要将恒压充电改成恒流充电

1. 如果我们用恒压充电会出现什么？

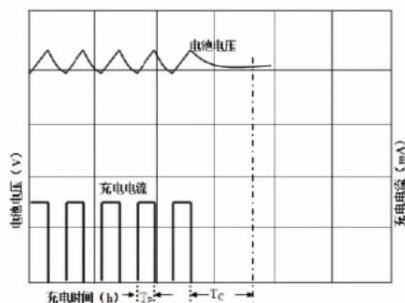


电源输出恒定电压经过电阻给电池充电，因为电池没电的时候电压很低，那么电源适配器和电池的压差就很大，那么最开始时间段因为压差大，电阻固定值，充电电流就很大。如果我在充电回路上取消电阻，那么电池充电电流就更大。随着电池电压升高，适配器和电池压差降低，充电电流降低。但是这种最先充电电流很大然后电流慢慢减小的恒压充电方式，是最容易损伤电池的方法。

压差大的时候电阻发热也严重，一般低廉价充电器就是恒压充电法。

所以电池充电采用分段充电法就是这个原因，前期电池低压时，充电用恒流，电池电压变高之后用恒压。

限压恒流充电法



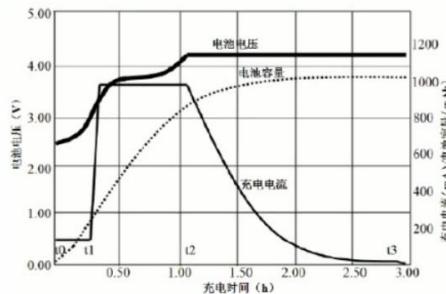
限流恒压法是恒定电流充电法的一种改进形式。从图中可以看到，限流恒压充电方法与恒定电流充电方法最大的不同就是限流恒压充电方法的充电电流不是连续的，而是时断时续的，从这个角度来看，限流恒压充电方法又有点像脉冲式充电方法。

脉冲充电法可以避免锂电池一直处于充电状态，这样可以控制电池的温升。就算电池温度有点上升，但是脉冲的低电平时间段也可以让电池散热。其次，充电间隙存在能够加速带电离子的扩散，增大电池充电电流。

3.2.3 分级定电流充电法

分级定电流充电法实际上是一种更为复杂的分段式充电法，从而可以更好的贴近锂电池的物理特性。分级定电流充电法是目前运用最广泛的充电方法。为了方便描述，可以人为的把这种充电方法按时间划分为不同的时期：

1. 预充电时期；
2. 恒流充电时期；
3. 恒压充电时期；
4. 充电终止时期。



国外的充电器都是用这种分级定电流充电法，但是这种方法成本很高。

1. 预充电时期：针对电池放在库房里面很久没用，电池自身掉电很严重了，这个时候是不能用大电流去充电的，不然电池内部结构会损坏。用几十mA到100mA去充电池。
2. 恒流充电时期：当电池被几十mA充电电流激活之后，可以进行恒流大电流充电。**(注意蓄电池没有预充电这种说法，只有锂电池有这种要求)**
3. 恒压充电时期：当电池被恒流大电流充到接近4.2V的时候，电池充电电流自动变小了，可以进行涓流充电。
4. 最后电池达到完整的4.2V，停止充电。

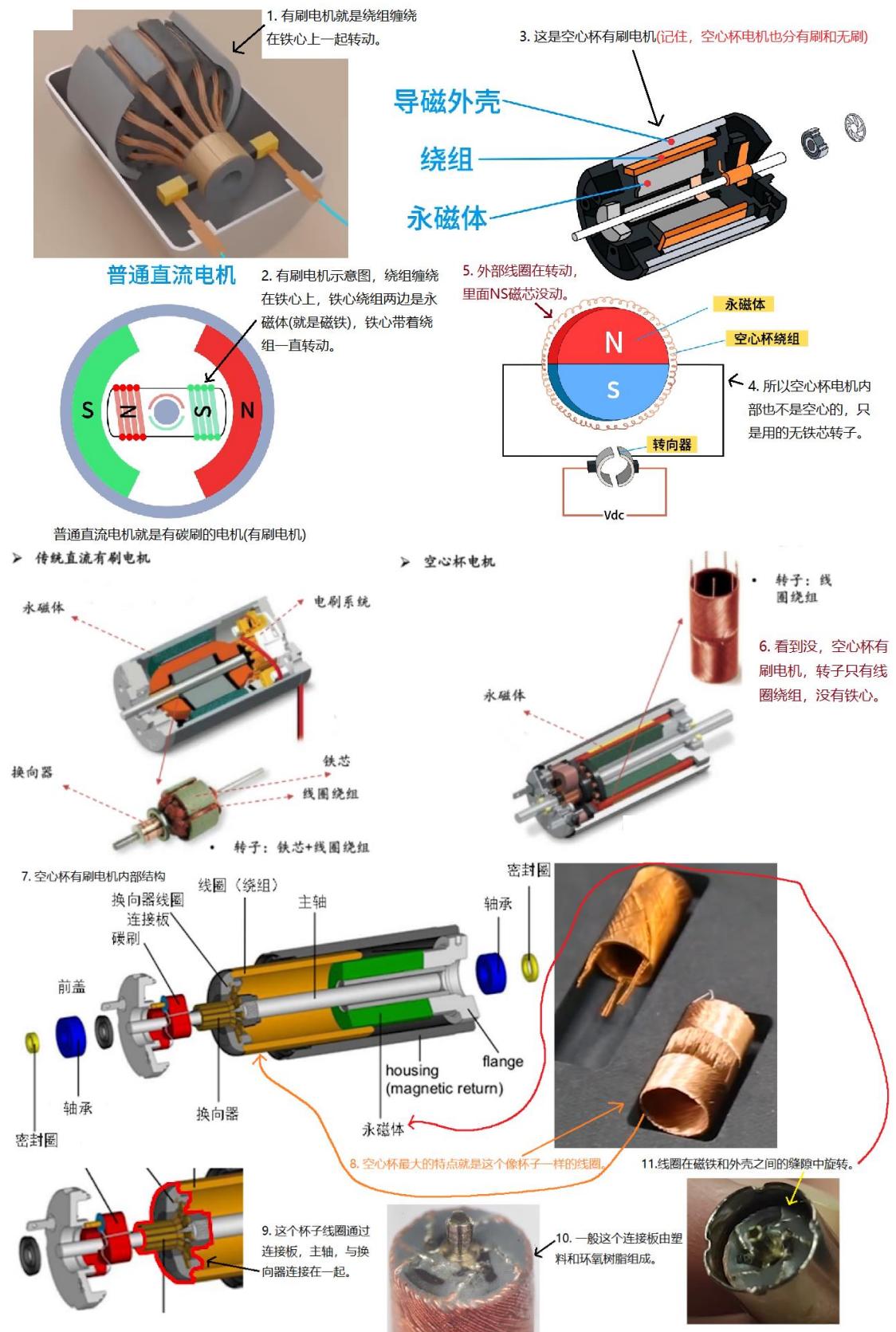
涓流充电：也就是电池预充电的说法。然后再恒流充电，最后恒压充电，充电结束。

注意：两节锂电池并联，用镍片焊接速度要快，如果焊接时间过长锂电池会容易坏，所以一般用激光电焊。锂电池燃烧起来，用水是无法浇灭的，只有眼睁睁看它烧完，二氧化碳灭火器应该可以。

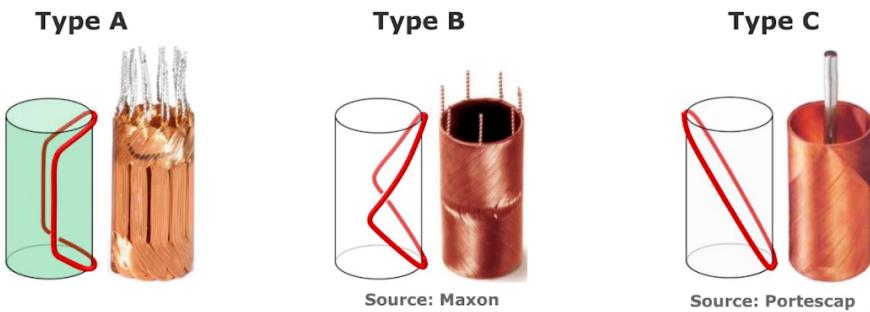
锂电池使用过程中，千万不要让锂电池内部电量完全放空，如果放空造成电池馈电会消耗电池寿命，最好是做到浅冲浅放。就是电池的电在没有放完的时候就给电池充电。

无人机螺旋桨电机的选择

空心杯有刷电机



12. 空心杯有刷电机因为是磁芯，而不是铁心，所以消除了直流有刷电机铁心形成涡流造成的电能损耗。传统普通直流有刷电机效率只有50%.



13. 空心杯由于转子重量大幅度降低，所以转动惯量变小。相比普通有刷直流电机，大扭矩加速，减速，反应快。比如在转动过程中，突然我要停止转动。如果是普通直流有刷电机，肯定会因为转子重量惯性转动一段时间才会停止。但是空心杯可以很快停止转动。

14. 空心杯电机，体积可以做很小，效率可以做很高，功率密度高，可控性高，噪音小，散热好。

15. 空心杯有刷电机缺点：因为线圈没有铁心支撑，所以线圈只能做的很薄，导致线圈和输出轴连接强度有限，所以整体功率不可能做的太大。一般空心杯电机最大功率也就几百瓦。

16. 所以空心杯电机适合用于快速响应的系统中，如导弹飞行方向快速调节，相机快速自动调焦，工业机器人，仿身翅膀，无人机。

空心杯和传统电机对比

	空心杯电机	传统直流电机
能量转换效率	70%-90%	小于70%
响应速度	10毫秒级别 启动停止性能好	大于100毫秒
能量密度	体积、重量减少1/3~一半	体积大、重量大

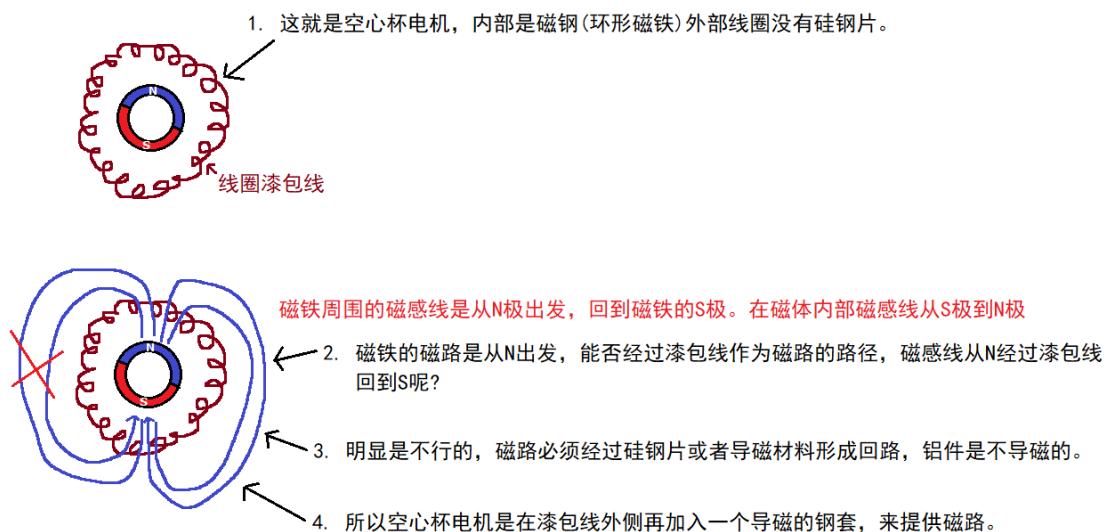
空心杯有刷原理介绍

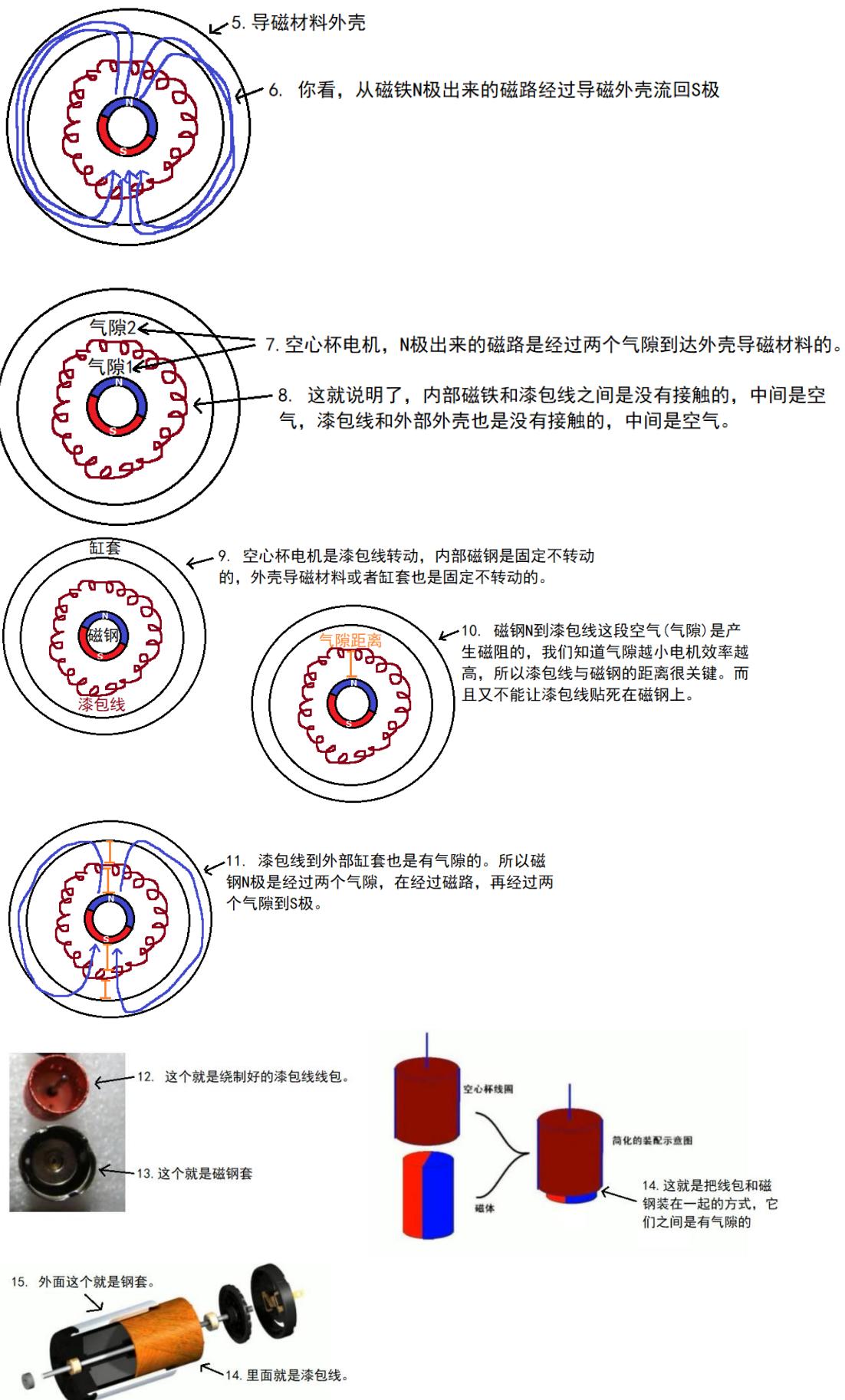
1. 空心杯电机是没有硅钢片的，记住哦，没有硅钢片。而BLDC无刷电机是有硅钢片的。
2. 空心杯电机因为没有硅钢片，所以很轻，因为硅钢片本来就很重。
3. 空心杯电机因为没有硅钢片，所以没有因为重量造成惯性很大，那么启动空心杯电机转动就很快，空心杯电机停止转动也停止得很快。不像带硅钢片得电机，启动的时候转的很慢，要慢慢转起来，停止得时候，断电了，都还要转一阵。



5. 空心杯电机只剩下漆包线圈。
4. 空心杯电机就是把硅钢片取消掉了。

磁路讲解，可以先看看后面的无刷电机<硅钢片具有导磁特性，磁路在硅钢片上面走>的基本内容





BLDC 无刷电机

1. 从结构来看，无刷直流电动机可分为两种类型：内转子和外转子。从根本上说，它们的工作原理相同，但优点和局限性却不同。



2. 内转子无刷电机，转动轴在内部，定子在外面 3. 外转子无刷电机，转动轴在外部，内部是定子线圈是不转动的。

相对而言，内转子无刷电机直径小，转速更快，更适用于高速应用；而外转子无刷电机则有着更大的直径和更好的散热性，利于低速转矩的使用，适用于低速应用。

内转子和外转子BLDC电机的区别

特性	内转子电机	外转子电机
长度	更长	较短
直径	更小	较大
功率密度	更高	较低
转速	相对高速	相对低速
稳定性	较低	更高
成本	相对更高	相对更低
散热性	较差	更好

4. 有些无人机螺旋桨用的是内转子电机，你看，螺旋桨接的内部转子。



6. 内转子电机转速高，扭力小。

7. 外转子电机转速低，扭力大。

8. 在电机输出相同功率的情况下，外转子无刷电机体积可以做更小。正因为这个提交小的特性，外转子在无人机领域应用非常普遍。

9. 内转子在结构强度远远大于外转子，所以在工控领域，军用，内转子用的比较多。

10. 无刷电机又有带位置检测霍尔和不带位置检测霍尔两种类型
11. 无刷电机是三相电流驱动。
12. 因为是三相驱动的无刷电机，所以无刷电机是要装3个位置传感器(霍尔传感)
13. 又因为无人机是用高转速电机(**这和前面外转子低速有冲突哈，其实我这里是用外转子高速电机**)，无人机提供升力的是螺旋桨，螺旋桨转速越快，螺旋桨提供的升力越大，那么电机输出功率也越大，转速越高。
14. 所以外转子无刷电机，在无人机领域都是用的无霍尔位置检测的方案。
15. 因为高转速电机，反电动势大，位置比较好检测，所以不用加霍尔传感器。
16. 无人机有个最低转速，但是这个最低转速也是很高的转速，所以无需霍尔。
17. **如果是低转速无刷电机，我们一般都要加霍尔位置检测传感器，因为低转速，电机反电动势很小，不容易检测到位置，所以要加霍尔。**

BLDC 无刷电机原理介绍

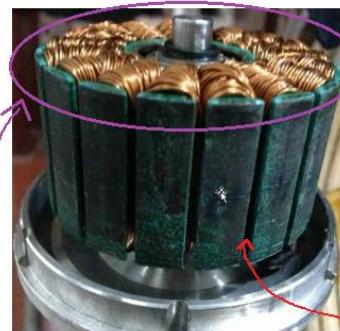




8. 但是气隙越小，对电机安装要求就更严格，因为气隙与内部这个硅钢片得空小很容易让硅钢片与外壳磁气间距有多少，也就是钢接触，造成摩擦，从而外气隙多大，气隙越小，壳磁钢转动不起来。
电机效率越高。因为空气是最大得磁阻。

国内电机气隙一般做在40个丝间距。国外做在5个丝间距。

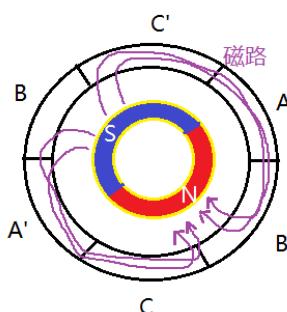
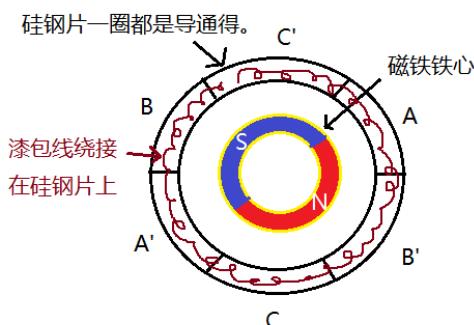
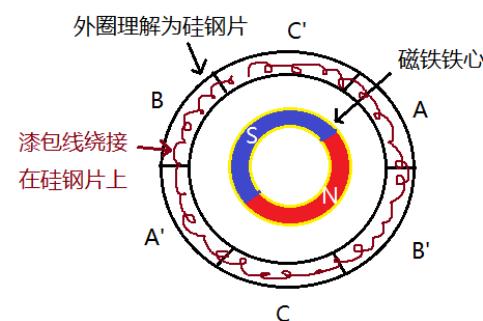
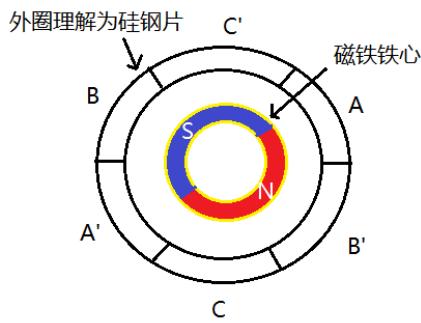
7. 这种结构的无刷电机，我们要看漆包线绕得多不多，漆包线越多，硅钢片得槽也就被占用得更多，槽满率就越高。槽满率越高，电机得效率也就越高。
所以看电机质量，第1看电机硅钢片是不是用得很薄，硅钢片是进口还是国产得。
第2就是看槽满率是不是更高。



9. 假如这是外部磁钢外壳，外壳在转动的时候NS南北极不一定完全都对准硅钢片，有可能会对准缝隙。这点就是缝隙——黑色的缝隙

10. 所以外转子磁钢在外面的时候，NS对准磁钢的部分磁通量大，NS对准黑色缝隙的部分磁通量就小。电机转动起来就有‘嘎嘎嘎嘎’的声音。

硅钢片具有导磁特性，磁路在硅钢片上面走



1. S产生得磁场是经过磁路进行流通得。
2. 我们发现，S产生得磁场经过硅钢片，流向N极。

3. 从这就能看出来，磁场是有磁路的，而且空气的磁阻很大，所以磁场没有向空气扩散，而是在硅钢片路径在走。也许是被硅钢片包裹的原因，大部分磁场都是走的硅钢片磁路。
所以磁路是一个闭环，和电路一样的，磁场从 S 走出来，必须有地方走回去，那么经过硅钢片这条路走回 N 极就是个闭环。

无人机天线选择

1. 什么是棒状天线？

棒状天线也叫做偶极子天线

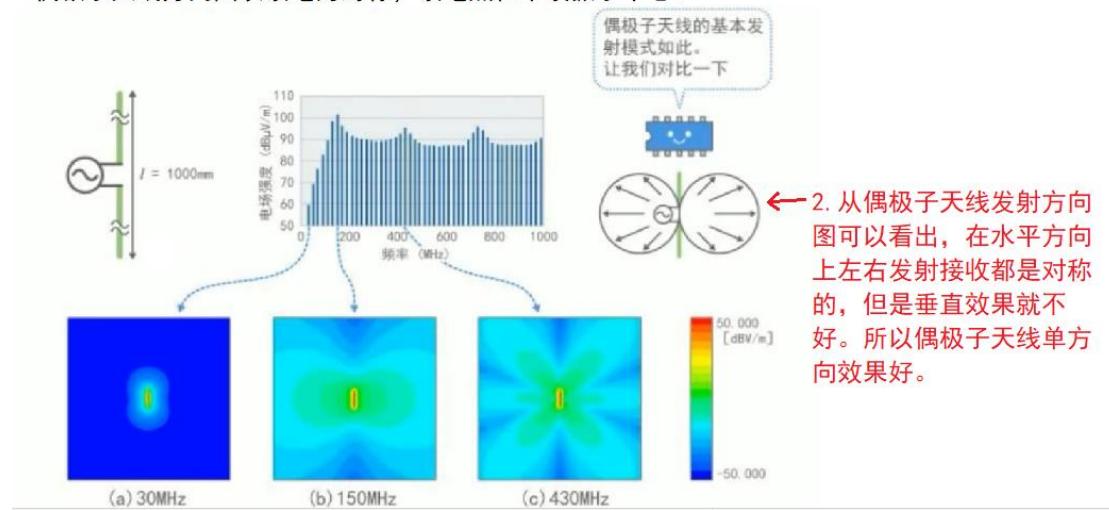
偶极子天线用来发射和接受固定频率的信号。虽然平时测量都是使用宽带天线，但在场地衰减和天线系数测量中都需要使用偶极子天线。

偶极子天线的频率范围由30Mhz ~ 4Ghz。

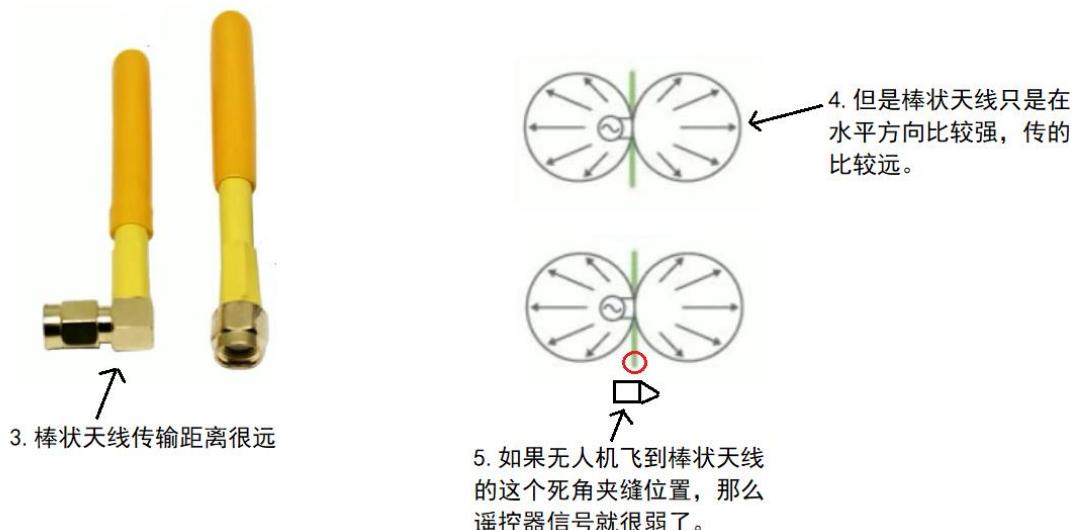
垂直天线实际上也是偶极子天线的一种。偶极子天线由两根导体组成，每根导体为1/4波长，即天线总长度为半波长。

所以偶极子天线叫做半波振子，偶极子天线的振子可以水平位置，也可垂直位置。

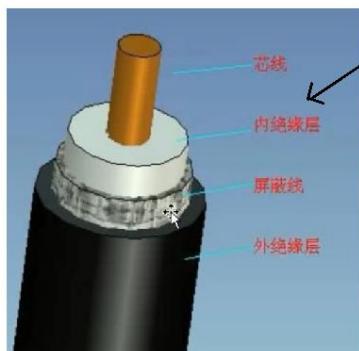
偶极子天线方向图以馈电为对称，馈电点在半波振子中心。



2. 从偶极子天线发射方向图可以看出，在水平方向上左右发射接收都是对称的，但是垂直效果就不好。所以偶极子天线单方向效果好。



6. 为了解决棒状天线在垂直方向上信号很弱的问题，我觉得另外再加一根垂直方向信号很强的天线。



9. 这就是棒子天线的电
缆，由四层组成。

10. 这是三叶草天线 →
(也就是蘑菇天线)
这种天线盲区比较少。



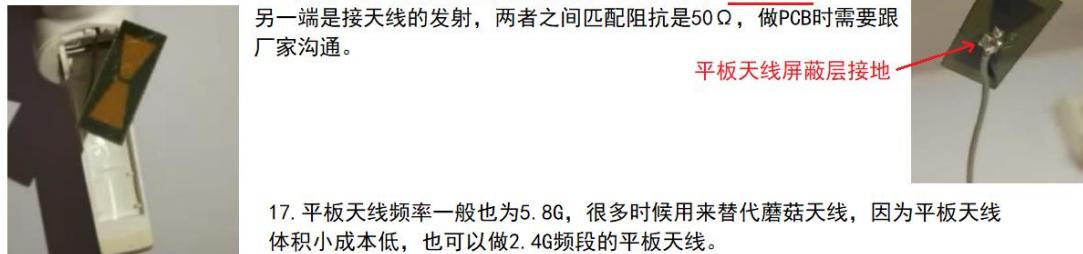
11. 三叶草天
线一般用做
5.8Ghz频率来
做图传。

14. 这是装一个三叶草
的无人机。

15. 一般三叶草天线装两个，防止无
人机自身遮挡。

16. 平板天线两端是喇叭状覆铜，有一定的距离，一端接地。

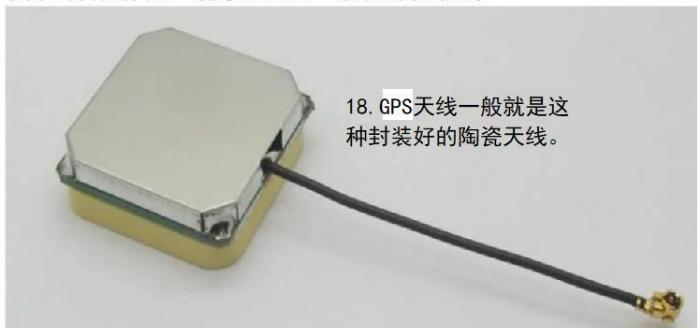
另一端是接天线的发射，两者之间匹配阻抗是 50Ω ，做PCB时需要跟
厂家沟通。



平板天线屏蔽层接地

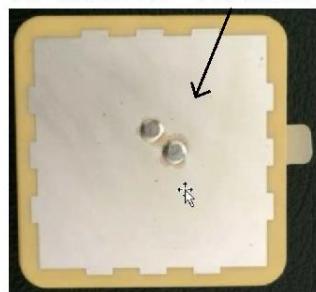
17. 平板天线频率一般也为5.8G，很多时候用来替代蘑菇天线，因为平板天线
体积小成本低，也可以做2.4G频段的平板天线。

机身上方背的圆盘一般是 GPS 天线，一般采用陶瓷天线。



18. GPS 天线一般就是这种封装好的陶瓷天线。

19. 这种双馈点 GPS 天线对增益有提高，也可以将有一个馈电接地。



陶瓷天线，现有与未来的数个 GPS 波段为：

1, L1 波段 - 1.57542GHz。

20. GPS 天线一般是放在无人机上面，方便接收卫星信号。因为 GPS 信号一般都比较弱。

2, L2 波段 - 1.22760GHz。

3, L3 波段 - 1.38105GHz。

4, L4 波段 - 1.84140GHz (预计将在 2017 年开始使用)。

GPS 跟卫星通信是采用无线通信，与单片机连接属于 UART 通信。

天线的信号强度，都与哪些因素有关

1, 天线的方向和极化，接收方向和天线极化都正确的情况下，信号才会正常。

2, 天线自身的增益和驻波，通俗一点就是天线的质量，天线质量好信号才强。

3, 天气，雨雪天的时候信号会有衰弱现象。

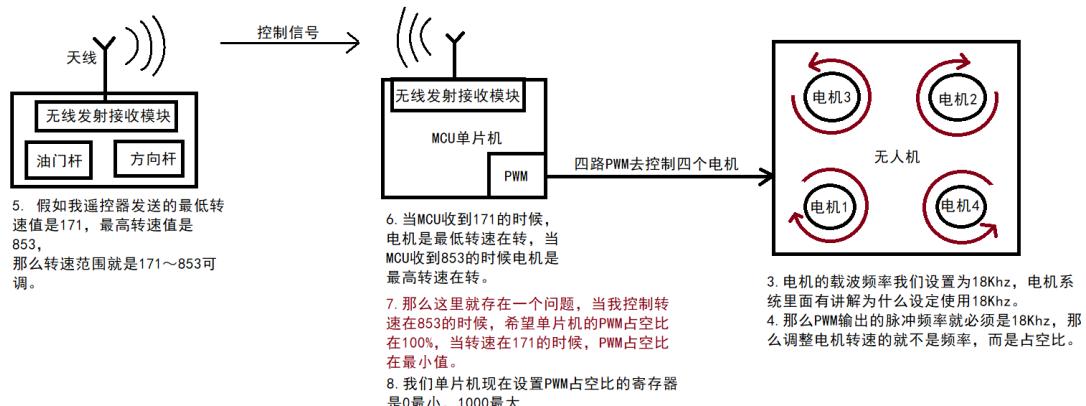
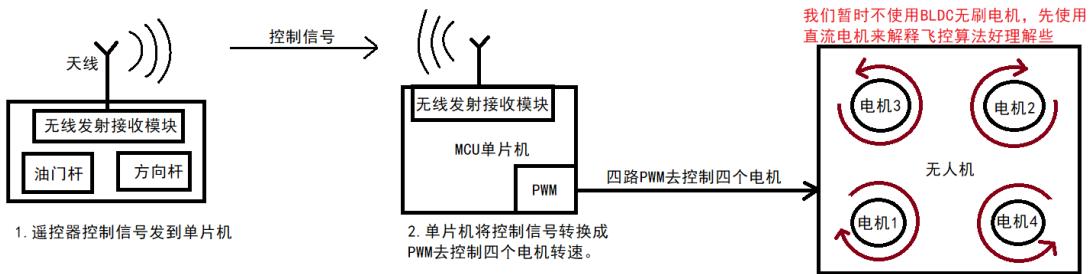
4, 连接电缆的长度和质量，如果电缆太长或是电缆本身质量不好，会对信号造成严重的衰竭。

5, 接收天线安装的高度，如果高度不够，信号会被周围的建筑或是树木遮挡。

6, 发射台本身的信号发射功率和发射高度。

天线增益功率一般就在 650mW 左右。

四旋翼无人机飞控系统设计



9. 所以油门转速与占空比对应关系如下：

油门转速	PWM占空比寄存器
171~853	0~1000
171	0
853	1000

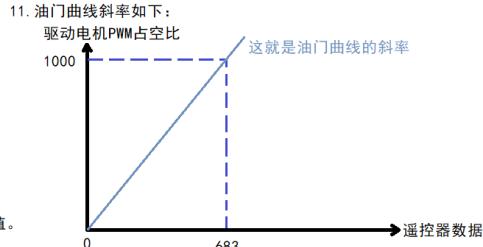
你看，这就存在一个问题，油门和PWM占空比的值无法一一对应。

10. 所以我们需要对油门数据和PWM占空比之间实现一个归一化拟合

归一化就是把很大的数值范围归一化成0~1的数
我们这里要把171~853归一化成0~1000

第1步骤： 171~854 减掉最低数171
0 ~ 683 两边都减171， 得到0~683

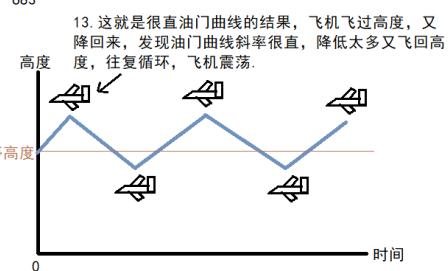
第2步骤： 0~683 归一到 0~1000
我们将 $1000/683 = 1.4641$ 系数
0~683 任何值乘1.4641系数， 得到0~1000对应值。



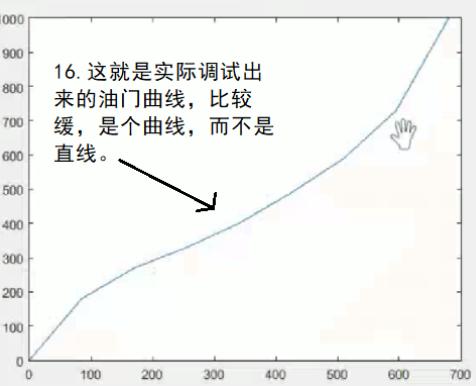
12. 我们发现一条很直很直斜率的油门曲线在飞机上是不合适的。

比如我们占空比在500，那么假如飞机的螺旋桨这时候升力>飞机重力，飞机会一直上升。

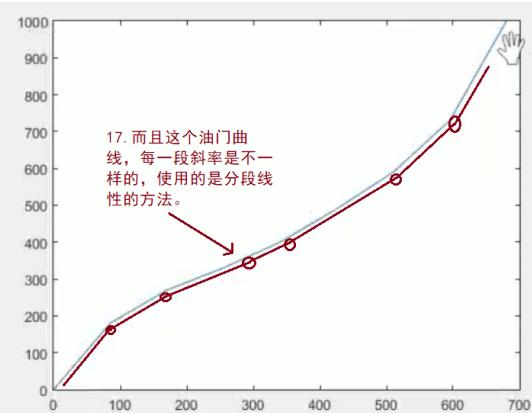
假如我们占空比在400的时候，飞机螺旋桨升力<飞机重力，飞机会一直下降。
但是我们要求飞机有时候可以空中悬停，定高，那么这种很直的油门曲线会出现什么情况呢？其实也能想的出来，很直的油门曲线会造成飞机悬停的时候停不到指定高度，导致飞机震荡。



14. 所以我们希望飞机悬停定高的时候，油门曲线要缓慢



17. 而且这个油门曲线，每一段斜率是不一样的，使用的是分段线性的方法。



18. 尤其是飞机才起飞阶段，要求升力很大，这样飞机起飞快，油门曲线很陡峭。

19. 飞机上升到一定高度油门曲线比较缓，方便悬停。

20. 飞机下降的时候油门曲线要陡，这是一个保护，当飞机落地之后，因为油门曲线很陡，来判断飞机是否完整落地。

油门曲线是根据自己实测测出来的，一般是先定义一个曲线，然后不停的修改调试，得到最终曲线。

油门曲线分段线性方法

1. 油门曲线，遥控器输出值和PWM驱动电机的值进行分段线性补偿
遥控器值 0 ~ 683
PWM占空比 0 ~ 1000

2. 我们遥控器值之间的间隔是等分的，都是 85



4. 分段线性函数一般是 $y = kx + b$

x: 是读出来的值如遥控器速度值 0~683

y: 是最后算出来的值，如 PWM 占空比 0~1000

5. 计算实例：

$$\text{第1段线斜率: } k = \frac{y}{x} = \frac{180}{85} = 2.12, \text{ 对应输出的 PWM 占空比}$$

遥控器第1个点值 第1段直线是从0开始，所以 $b = 0$

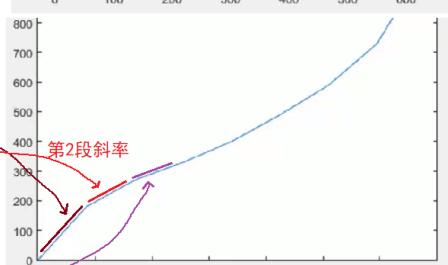
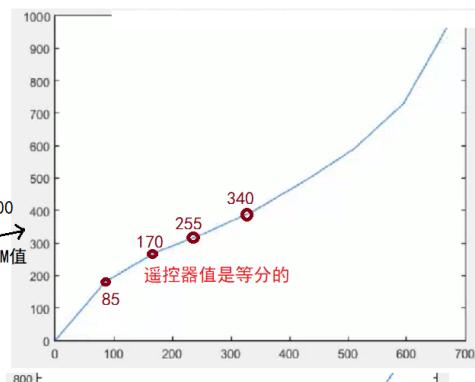
$$\text{第2段线斜率: } k = \frac{y}{x} = \frac{90}{85} = 1.058, \text{ 斜率算是差值之比}$$

$$\text{第2段直线是 } b \text{ 是 } 270 = 1.0588 * 170 + b \text{ 截距 } b \text{ 算的是原始值}$$

$$b = 90 \quad (\text{公式 } y = kx + b)$$

$$\text{第3段斜率: } k = \frac{y}{x} = \frac{60}{85} = 0.705.$$

$$\text{第3段直线 } b: 330 = 0.705 * 255 + b \quad b = 150.225$$

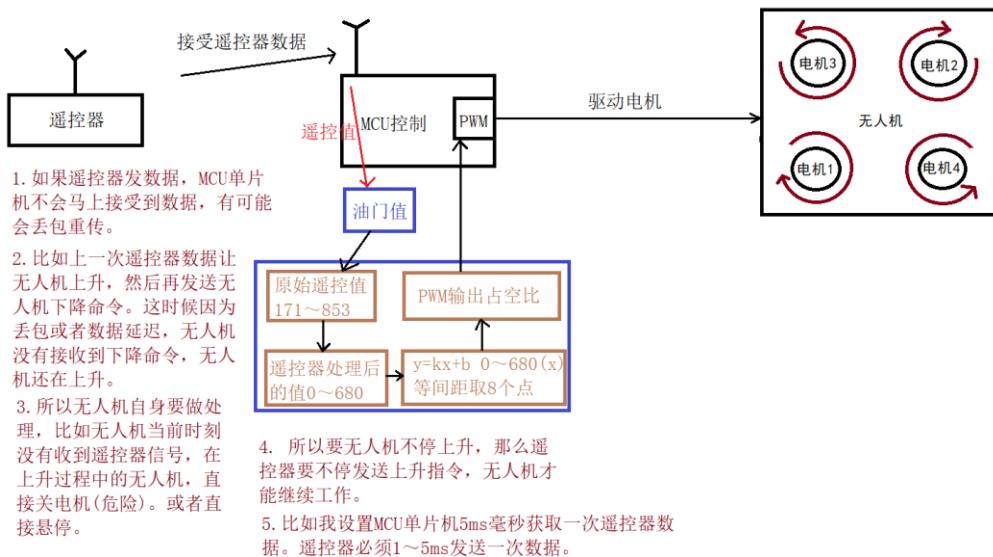


6. 将剩下的5个点斜率k和截距b求出来之后，就可以在使用过程中执行分段函数了
怎么执行呢？

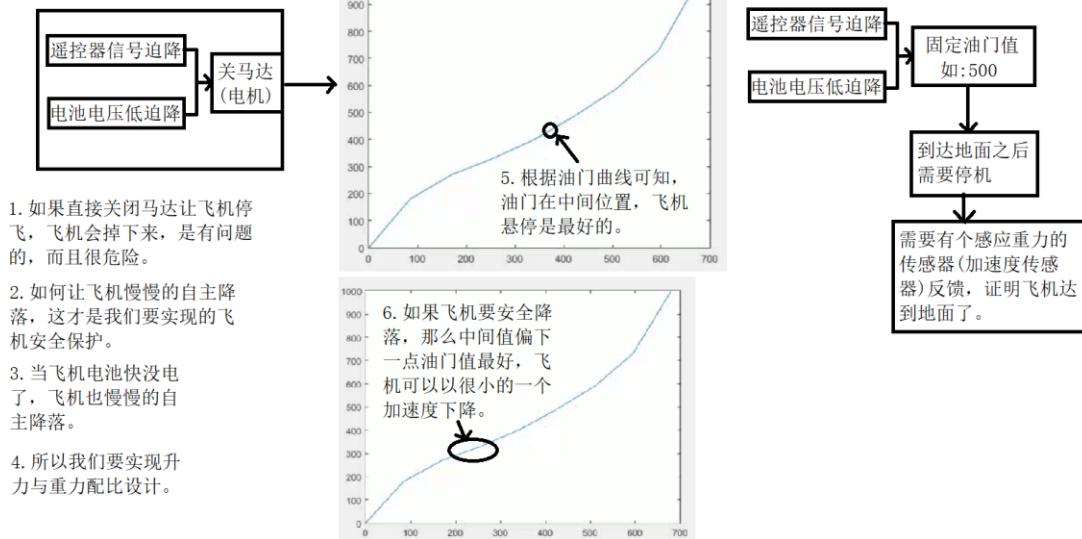
0 85 170 255 340 425 510 595 680

7. 比如这时候遥控器值是 170，然后将 170 的值传入 x，经过 x 值判断已经写好的 k 和 b
将 x, k, b 代入 $y = kx + b$ ，得到 y，y 就是要传入 PWM 寄存器的占空比值。

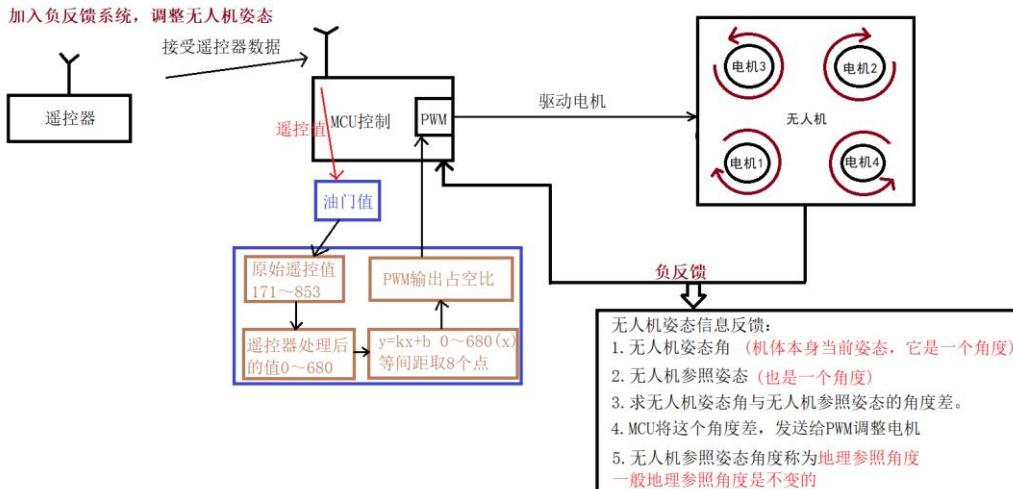
无人机控制逻辑



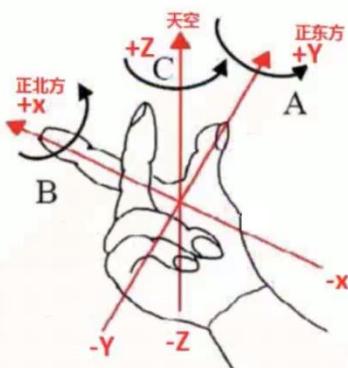
无人机安全保护



无人机姿态控制

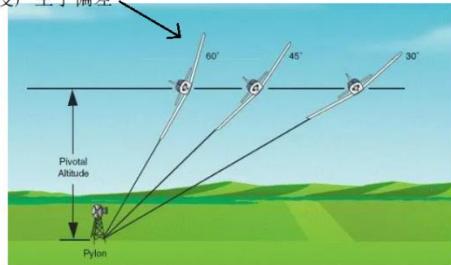


6. 地理参照角度如下：



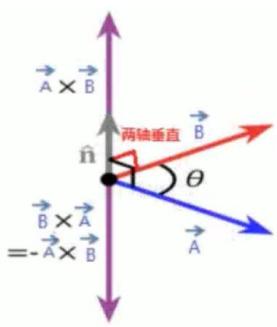
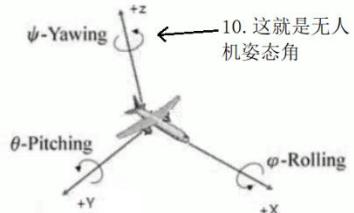
7. 右手定律，定义地理参照角度，中指对天空，大拇指方向为正东方。

8. 因为地理角度是固定的，飞机移动旋转后与地面的X, Y, Z角度产生了偏差

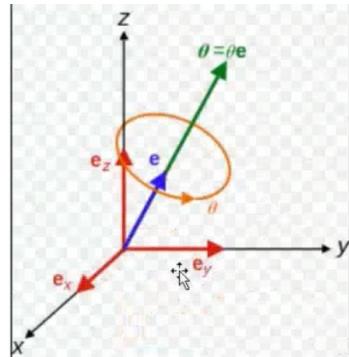


9. 所以我们主要是求飞机自身旋转后XYZ和地理固定XYZ之间的关系。

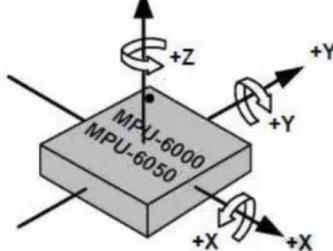
无人机姿态角矢量 b
参照的地理姿态角 矢量 e



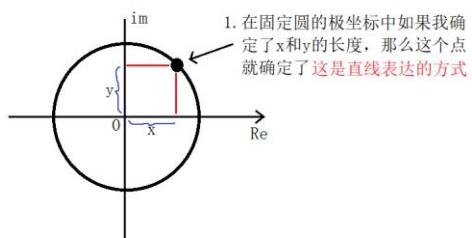
这两个AB向量，它们可以用 θ 角来表示这两个向量在某一个平面角的关系。



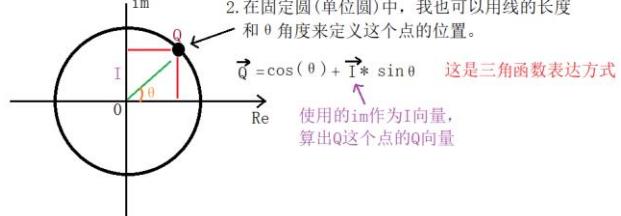
e 向量与 e_y , e_z , e_x 都有一个角度，所以这个 e 向量跟xyz的投影夹角来反应 e 的情况。



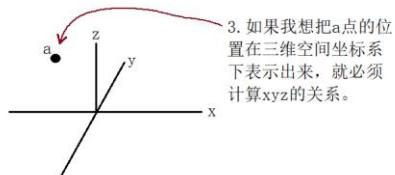
飞机上使用的关键陀螺仪



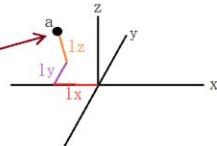
2. 在固定圆(单位圆)中，我也可以用线的长度和 θ 角度来定义这个点的位置。



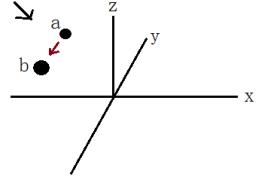
使用的im作为I向量，算出Q这个点的Q向量



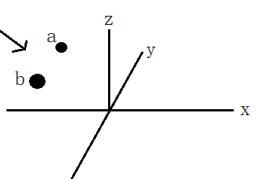
4. a点位置传统的方式就是先计算 $1x$ 长度，然后计算 $1y$ 长度，然后计算 $1z$ 高度，确定 a 点位置。



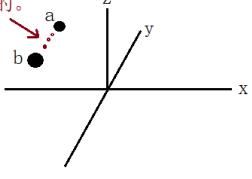
5. 如果无人机想从a点移动到b点，该怎么理解？



6. 我们知道，无人机不可能一下子从a点跳跃到b点，而是从a点慢慢移动到b点。



7. 所以无人机是经过很多个点，从a点移动到b点的。

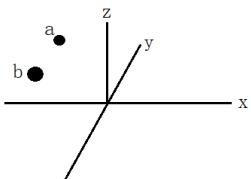


8. 所以我们知道无人机可能会绕在y旋转，z和x就是一个平面。

9. 无人机绕在z旋转，x和y就是一个平面。

10. 无人机绕在x旋转，y和z是一个平面。

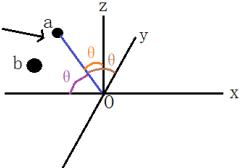
11. 所以这个三坐标系统就有三个虚数。



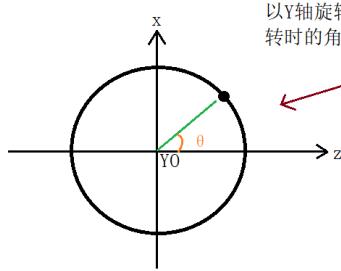
$$Q = \cos \theta + [\vec{i} + \vec{j} + \vec{k}] \sin \theta$$

三维坐标系就是三个虚数相加。

12. 如果我们将a点使用矢量表示法表示，我发现a点这条线在x, y, z三个坐标线上各有一个对应的角度 θ 。

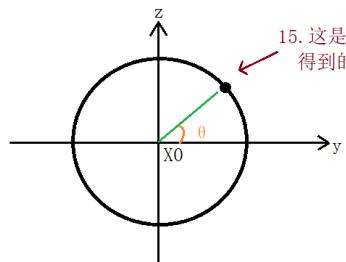


13. 那么换句话说，我只要知道a点的xyz的三个角度和向量，我就知道a点在空间中的位置。

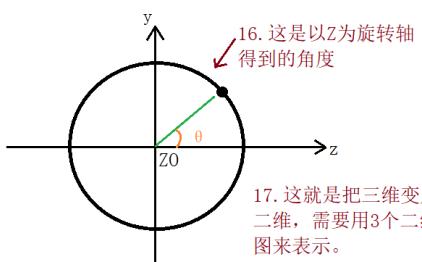


14. 我们现在将a点表示为0a, 0a以Y轴旋转的时候，分解出Y轴旋转时的角度用二维表示。

这就a点以y轴旋转的时候，得到的夹角

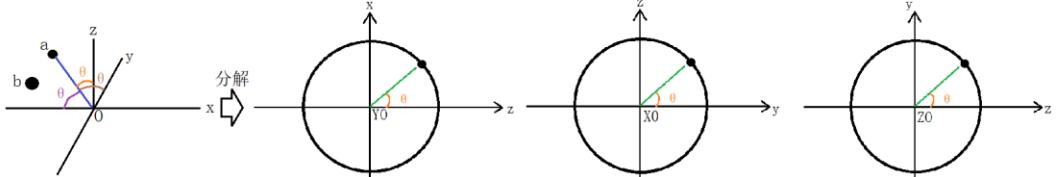


15. 这是以X轴为旋转轴得到的角度



16. 这是以Z为旋转轴得到的角度

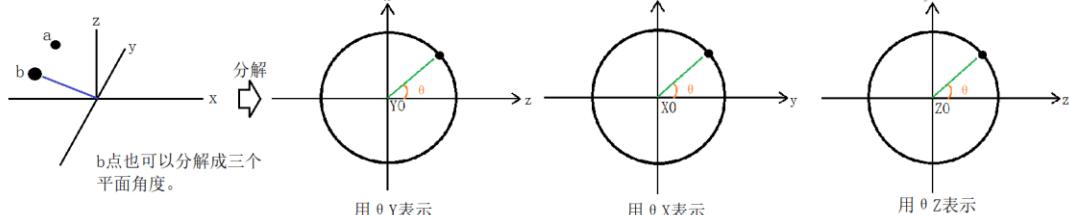
17. 这就是把三维变成二维，需要用3个二维图来表示。



用 θ_Y 表示

用 θ_X 表示

用 θ_Z 表示



用 θ_Y 表示

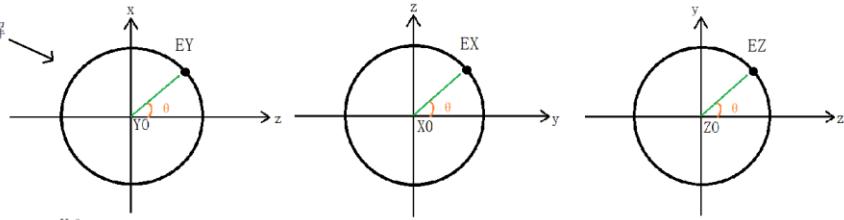
用 θ_X 表示

用 θ_Z 表示

b点也可以分解成三个平面角度。

18. 地理参照坐标系是不变的，假如地理系为E
 19. 飞机坐标系是在不断变化的
 20. 地理系和机体系是重合的
 21. 机体系相对地理系旋转了一个角度，假如为角度 \vec{a} 当前角度
 22. 期望机体系旋转到b角度，假如为角度 \vec{b} 旋转角度

23. 如果将地理系参照坐标分解



24. 将 \vec{a} 投影到地理系坐标里面去
 a在xyz三个轴的角度

25. 将 b 投影到地理系坐标里面去
 b在xyz三个轴的角度

$a_x \theta$

$a_y \theta$

$a_z \theta$

$b_x \theta$

$b_y \theta$

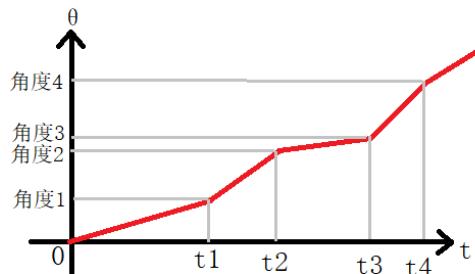
$b_z \theta$

26. 这两个组角度的差值，就是我们需要的值。

27. 要实现a向量到b向量角度的变换，必须使用加速度计和陀螺仪这两个东西组成6轴传感器。

陀螺仪是三个轴，实现的是角速度而不是角度哦。
 只是陀螺仪也分xyz三个轴的角速度。

角速度的采用速率，比如1毫秒采样一次



28. 最终a点旋转到b点的角度是经过t1, t2, t3, t4... 等等时间之后，积分出来的一个角度

29. 比如a点在t4时刻得到的角度，就是t3时刻到t4时刻积分的量再加上t1, t2, t3角度的量。

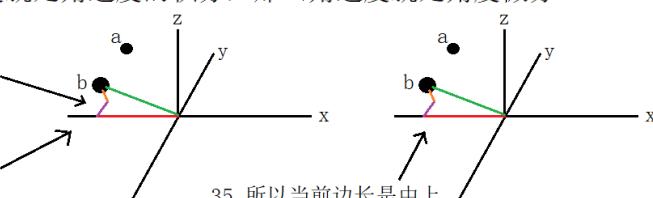
30. 比如最终角度4是由角速度*时间积分出来的一个量。

31. 如果对角度4进行微分，就是它的角速度。

32. 那么角度就是角速度的积分，那么角速度就是角度微分。

33. 我们来看，红紫橙这个边线和角速度有什么关系呢？

34. 其实这个边线长度也是a到b点之间经过无数个点慢慢移动之后形成的边线。



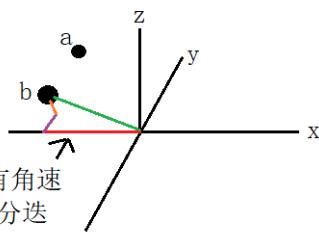
35. 所以当前边长是由上一次边长*时间形成的。
 所以边长里面包含了角速度。

36. 最终求解就是用陀螺仪测出角速度*时间，然后进行迭代积分，就求出当前角度的边长。

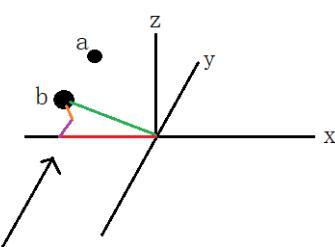
37. 也可以用角度对xyz三轴用三角函数来对边进行求解。

38. 所以b点我们要先求b向量在xyz轴上角度的分解。

39. 还要用边的形式来进行表达角速度的信息



40. 边长使用含有角速度的值，进行积分迭代的方式求得。



43. a点到b点向量同时分到三个坐标系中的角度也要求解。这个角度采用“欧拉角”求解。

求解欧拉角介绍

1. 一般欧拉角也就三个参数，也就是求三角函数。**但是欧拉角在求90度(俯仰角)有一个奇点。**这就造成欧拉角无法在全坐标里面变换。

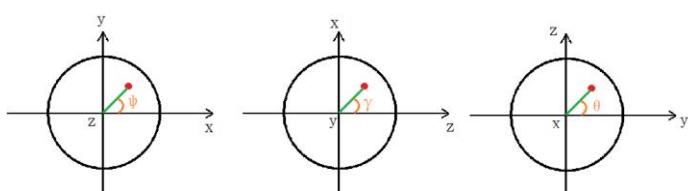
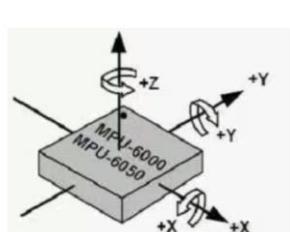
解决欧拉角奇点的方法有一个是方向余弦法

2. 但是方向余弦法有9个参数，计算量太大。
3. 三角函数法有6个参数，计算量也挺大的。

Rodrigus法

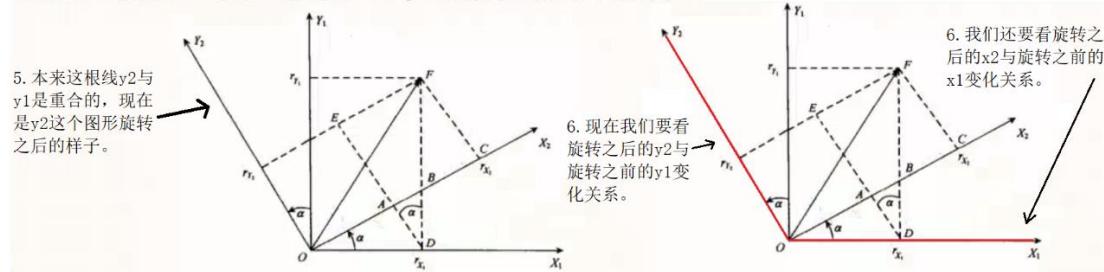
4. Rodrigus这种方法其实比四元素方法还要简单，但是它需要修正校0，比较麻烦。
5. 所以最后选择了“四元数”法
6. 还有等效旋转适量法，这个方法也存在奇点。

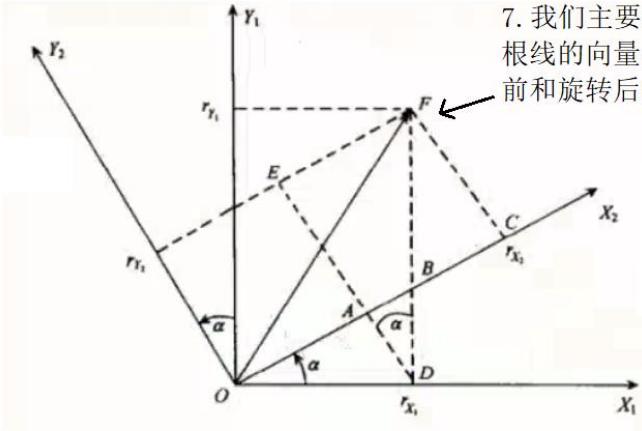
欧拉角使用



1. 我们使用陀螺仪求物体姿态角的时候，我们要建立旋转前和旋转之后的角度变量。这里旋转之后的角度变量用 $\psi \gamma \theta$ 三个变量来表示。

物体绕 z 轴旋转，可知 z 轴不变，x、y 轴的变换关系如下图所示：





7. 我们主要是求OF这根线的向量，在旋转前和旋转后的投影。

$$\begin{aligned}
 r_{x2} &= OA + AB + BC \\
 &= OD \cos \alpha + BD \sin \alpha + BF \sin \alpha \\
 &= r_{x1} \cos \alpha + r_{y1} \sin \alpha
 \end{aligned}$$

r_{x2} 就是OC这条线
OA就是OD旋转 α
AB就是BD对边旋转 $\sin \alpha$
OD是 r_{x1}

$$\begin{aligned}
 r_{y2} &= DE - AD \\
 &= DF \cos \alpha - OD \sin \alpha \\
 &= r_{y1} \cos \alpha - r_{x1} \sin \alpha
 \end{aligned} \tag{26}$$

$$r_{z2} = r_{z1}$$

因为线段绕着Z轴在旋转，所以旋转前后的Z是相等的。

将以上三式写为矩阵形式则得到绕 Z 轴旋转 α 角度后的坐标关系，

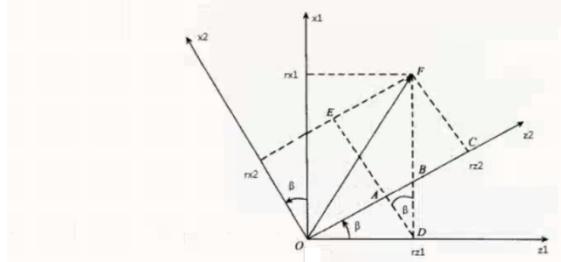
如下所示： 9. 旋转参数在这一项
↓ 8. 意思就是我们绕在这一列数据在旋转

$$\begin{bmatrix} r_{x2} \\ r_{y2} \\ r_{z2} \end{bmatrix} = \begin{bmatrix} \cos \alpha & \sin \alpha & 0 \\ -\sin \alpha & \cos \alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{x1} \\ r_{y1} \\ r_{z1} \end{bmatrix}$$

矩阵 $\begin{bmatrix} \cos \alpha & \sin \alpha & 0 \\ -\sin \alpha & \cos \alpha & 0 \\ 0 & 0 & 1 \end{bmatrix}$ 即为变换矩阵。

物体绕 Y 轴旋转，可知 Y 轴不变，x、z 轴的变换关系如下图所示：

由图可知



$$\begin{aligned}
 r_{x2} &= DE - AD \\
 &= DF \cos \beta - OD \sin \beta \\
 &= r_{x1} \cos \beta - r_{z1} \sin \beta
 \end{aligned} \tag{28}$$

$$r_{z2} = r_{z1} \tag{29}$$

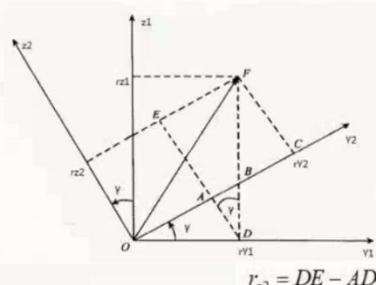
$$\begin{aligned}
 r_{z2} &= OA + AB + BC \\
 &= OD \cos \beta + BD \sin \beta + BF \sin \beta \\
 &= r_{z1} \cos \beta + r_{x1} \sin \beta
 \end{aligned} \tag{30}$$

将以上三式写为矩阵形式则得到物体绕 Y 轴旋转 β 角度后的坐标信息，如下所示：

$$\begin{bmatrix} r_{x2} \\ r_{y2} \\ r_{z2} \end{bmatrix} = \begin{bmatrix} \cos \beta & 0 & -\sin \beta \\ 0 & 1 & 0 \\ \sin \beta & 0 & \cos \beta \end{bmatrix} \begin{bmatrix} r_{x1} \\ r_{y1} \\ r_{z1} \end{bmatrix}$$

矩阵 $\begin{bmatrix} \cos \beta & 0 & -\sin \beta \\ 0 & 1 & 0 \\ \sin \beta & 0 & \cos \beta \end{bmatrix}$ 即为变换矩阵

物体绕 x 轴旋转，可知 x 轴不变，y、z 轴的变换关系如下图所示：



由图可知

$$r_{x2} = r_{x1}$$

(31)

$$= DF \cos \gamma - OD \sin \gamma$$

$$r_{y2} = OA + AB + BC$$

$$= r_{z2} \cos \gamma - r_{y1} \sin \gamma$$

因为是绕 x 旋转，所以 x 为 1

$$\begin{bmatrix} r_{x2} \\ r_{y2} \\ r_{z2} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \gamma & \sin \gamma \\ 0 & \sin \gamma & \cos \gamma \end{bmatrix} \begin{bmatrix} r_{x1} \\ r_{y1} \\ r_{z1} \end{bmatrix}$$

矩阵 $\begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \gamma & \sin \gamma \\ 0 & \sin \gamma & \cos \gamma \end{bmatrix}$ 即为变换矩阵

$$= OD \cos \gamma + BD \sin \gamma + BF \sin \gamma$$

$$= r_{y1} \cos \gamma + r_{z1} \sin \gamma$$

(32)

(33)

上面我们推导了物体分别绕三个轴旋转的变换矩阵，求一个物体的姿态角时，可以等效为物体绕三个轴旋转的复合，规定绕 z 轴旋转称物体的航向角 (ψ)、绕 y 轴旋转称物体的俯仰角 (γ)、绕 x 轴旋转称物体的翻滚角 (θ)，则复合后的姿态矩阵为 c_E^b ，在旋转时，可以有多种旋转方法（如 Z→X→Y、X→Z→Y、Y→Z→X 等），这里我们取计算姿态矩阵的旋转顺序为 Z→Y→X，可求得旋转矩阵如下：

$$c_E^b = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \theta & \sin \theta \\ 0 & -\sin \theta & \cos \theta \end{bmatrix} \begin{bmatrix} \cos \gamma & 0 & -\sin \gamma \\ 0 & 1 & 0 \\ \sin \gamma & 0 & \cos \gamma \end{bmatrix} \begin{bmatrix} \cos \psi & \sin \psi & 0 \\ -\sin \psi & \cos \psi & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

3. 再转 x 轴 2. 再转 z 轴 1. 先转 y 轴

4. 三个矩阵相乘计算

先旋转的放在后面，后旋转的放在前面

对(34)化简得

$$c_E^b = \begin{bmatrix} \cos \gamma & 0 & -\sin \gamma \\ \sin \theta \sin \gamma & \cos \theta & \sin \theta \cos \gamma \\ \cos \theta \sin \gamma & -\sin \theta & \cos \theta \cos \gamma \end{bmatrix} \begin{bmatrix} \cos \psi & \sin \psi & 0 \\ -\sin \psi & \cos \psi & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

5. 这个坐标相当是由地理坐标到机体坐标。

$$= \begin{bmatrix} \cos \gamma \cos \psi & \cos \gamma \sin \psi & -\sin \gamma \\ \sin \theta \sin \gamma \cos \psi - \cos \theta \sin \psi & \sin \theta \sin \gamma \sin \psi + \cos \theta \cos \psi & \sin \theta \cos \gamma \\ \cos \theta \sin \gamma \cos \psi + \sin \theta \sin \psi & \cos \theta \sin \gamma \sin \psi - \sin \theta \cos \psi & \cos \theta \cos \gamma \end{bmatrix}$$

6. 机体系坐标到地理系坐标如下：

$$c_E^b = (c_b^E)^T$$
 (符号“T”为矩阵的转置)

$$c_b^E = \begin{bmatrix} \cos \gamma \cos \psi & \sin \theta \sin \gamma \cos \psi - \cos \theta \sin \psi & \cos \theta \sin \gamma \cos \psi + \sin \theta \sin \psi \\ \cos \gamma \sin \psi & \sin \theta \sin \gamma \sin \psi + \cos \theta \cos \psi & \cos \theta \sin \gamma \sin \psi - \sin \theta \cos \psi \\ -\sin \gamma & \sin \theta \cos \gamma & \cos \theta \cos \gamma \end{bmatrix}$$

四元数使用(第1部分)

四元数有多种表现形式，有矢量式、复数式、三角式、矩阵式、指数式，下面我们简单看一下他们各自的定义式。

A. 矢量式

$$\vec{Q} = q_0 + \vec{q} \quad (\text{其中 } \vec{q} = q_1 \vec{i} + q_2 \vec{j} + q_3 \vec{k})$$

2. 三个轴的向量

q_0 为标量部分， \vec{q} 为矢量部分。

1. \vec{q} 是一个虚数，它的平方 = -1
在三坐标里面 \vec{q} 这个虚数包含了三个轴的信息

B. 复数式

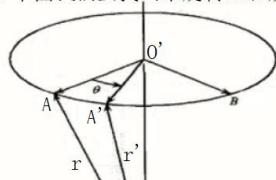
$$\vec{Q} = q_0 + q_1 \vec{i} + q_2 \vec{j} + q_3 \vec{k}$$

3. 也可以写成复数形式

前已述及四元数可看作一个超复数，所以其共轭复数为

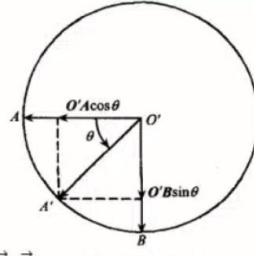
$$\vec{Q}^* = q_0 - q_1 \vec{i} - q_2 \vec{j} - q_3 \vec{k}$$

4. 下面我们要找出未旋转OA和旋转之后OA'的关系。



$$\vec{OO'} = (\vec{r} \bullet \vec{u}) \vec{u}$$

\vec{u} 是单位向量也就是 OO'



(1) ($\vec{r} \bullet \vec{u}$ 为两向量的点乘)

$$\vec{O'A} = \vec{OA} - \vec{OO'}$$

(两向量相减, 所得向量指向被减向量)

$$= \vec{r} - (\vec{r} \bullet \vec{u}) \vec{u}$$

$$\vec{O'B} = \vec{u} \times \vec{O'A}$$

(符号“ \times ”表示向量的叉乘符号, 两向量的外积还是一个向量)

$$= \vec{u} \times [\vec{r} - (\vec{r} \bullet \vec{u}) \vec{u}]$$

(向量 $\vec{u} \times \vec{u} = \vec{0}$)

$$= \vec{u} \times \vec{r} - (\vec{r} \bullet \vec{u}) \vec{u} \times \vec{u}$$

(3)

$$\text{又 } \vec{O'A}' = \vec{O'A} \cos \theta + \vec{O'B} \sin \theta$$

(4) (两向量相加)

将(2)(3)两式代入(4)式得

$$= \vec{r} \cos \theta - (\vec{r} \bullet \vec{u}) \vec{u} \cos \theta + \vec{u} \times \vec{r} \sin \theta$$

(5)

所以 $\vec{r}' = \vec{O'A}' + \vec{OO'}$ (由图中向量关系得到)

将(1)式(5)式两式代入, 得

$$= \vec{r} \cos \theta - (\vec{r} \bullet \vec{u}) \vec{u} \cos \theta + \vec{u} \times \vec{r} \sin \theta + (\vec{r} \bullet \vec{u}) \vec{u}$$

整理得

$$= \vec{r} \cos \theta + (1 - \cos \theta)(\vec{r} \bullet \vec{u}) \vec{u} + \vec{u} \times \vec{r} \sin \theta$$

(6)

接下来化简 $(\vec{r} \bullet \vec{u}) \vec{u}$

$$c_b^E = \begin{bmatrix} \cos \gamma \cos \psi & \sin \theta \sin \gamma \cos \psi - \cos \theta \sin \psi & \cos \theta \sin \gamma \cos \psi + \sin \theta \sin \psi \\ \cos \gamma \sin \psi & \sin \theta \sin \gamma \sin \psi + \cos \theta \cos \psi & \cos \theta \sin \gamma \sin \psi - \sin \theta \cos \psi \\ -\sin \gamma & \sin \theta \cos \gamma & \cos \theta \cos \gamma \end{bmatrix} \quad \text{欧拉角矩阵}$$

$$c_b^E = \begin{bmatrix} q0^2 + q1^2 - q2^2 - q3^2 & 2(q1q2 - q0q3) & 2(q1q3 + q0q2) \\ 2(q1q2 + q0q3) & q0^2 - q1^2 + q2^2 - q3^2 & 2(q2q3 - q0q1) \\ 2(q1q3 - q0q2) & 2(q2q3 + q0q1) & q0^2 - q1^2 - q2^2 + q3^2 \end{bmatrix} \quad \text{四元数矩阵}$$

$$g1 = 2(q1q3 - q0q2)$$

$$2(q1q3 - q0q2) = -\sin \gamma = g1$$

$$g2 = 2(q2q3 + q0q1)$$

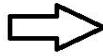
四元数和欧拉角之间的转换关系

$$2(q2q3 + q0q1) = \sin \theta \cos \gamma = g2$$

$$g3 = q0^2 - q1^2 - q2^2 + q3^2$$

$$q0^2 - q1^2 - q2^2 + q3^2 = \cos \theta \cos \gamma = g3$$

$$g4 = 2(q1q2 + q0q3)$$

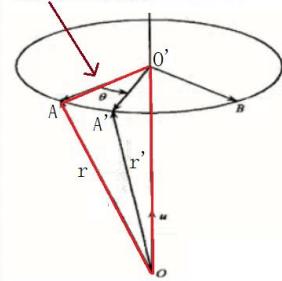


$$2(q1q2 + q0q3) = \cos \gamma \sin \psi = g4$$

$$g5 = q0^2 + q1^2 - q2^2 - q3^2$$

$$q0^2 + q1^2 - q2^2 - q3^2 = \cos \gamma \cos \psi = g5$$

$$\vec{OA} = \vec{OA}' - \vec{OO}', \text{ 三角形式}$$



$$\gamma = -\arcsin(g_1)$$

$$\theta = \arctan(g_2 / g_3)$$

$$\psi = \arctan(g_4 / g_5)$$

最后经过三角函数，
计算得到姿态角

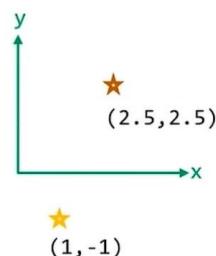
- 四元数具体 q_1, q_2, q_3 的值我们是不知道的
- 我们现在只知道陀螺仪的角速度，和获取角速度的时间周期我定义为1ms
- q_0, q_1, q_2, q_3 的值其实是角速度根据时间去积分出来的角度值。所以四元数 q_0, q_1, q_2, q_3 的值是积分出来的。
- 每次变化的角速度值，是根据每1ms变化一次角速度值来积分。
- 所以为了让四元数 q_0, q_1, q_2, q_3 包含角速度信息，那么就必须把已经积分算出来的 q_0, q_1, q_2, q_3 四元数进行微分。
- 所以需要求解四元数的微分方程。

四元数的理解(介绍旋转矩阵的缺陷)

1. 先来复习下旋转矩阵，看旋转矩阵遇到的问题

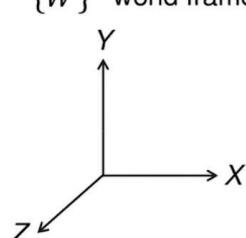


2. 如果是在二维平面，只需要x和y就可以描述这个质点的位置



4. 如果是黄色星星，在二维平面中看原点是在什么位置？

{W} world frame

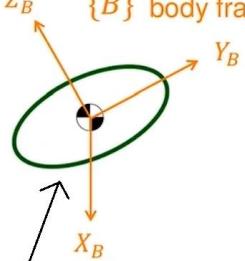


3. 如果在三维空间中，就用xyz来描述这个质点的位置

5. 黄色星从当前走到原点，相当于x向负方向移动1格，y向上移动1格，才能走到坐标原点，向量就是 $(-1, 1)$

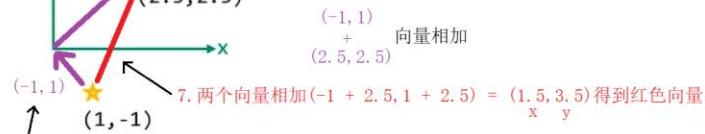


{B} body frame



6. 从原点出发到红星就是 $(2.2, 2.5)$

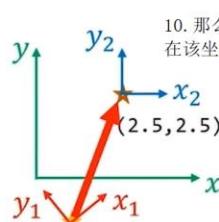
(1, -1)



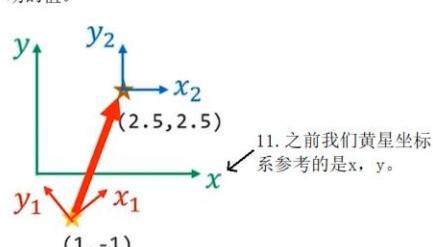
$$\begin{matrix} (-1, 1) \\ + \\ (2.5, 2.5) \end{matrix}$$

向量相加

10. 那么红星位置向量在该坐标系表示为？

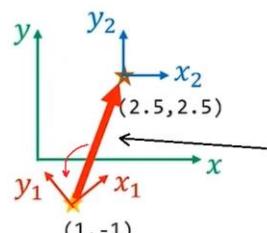


8. 所以这里为什么是 $(-1, 1)$ ，因为这是向量的值，而不是坐标的值。向量指的是移动的值。

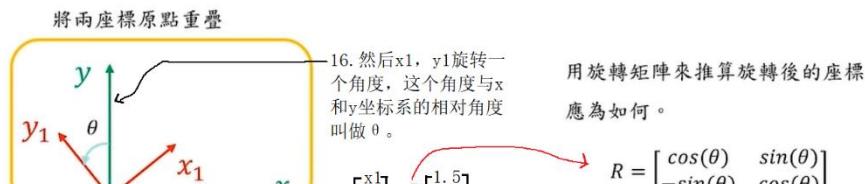
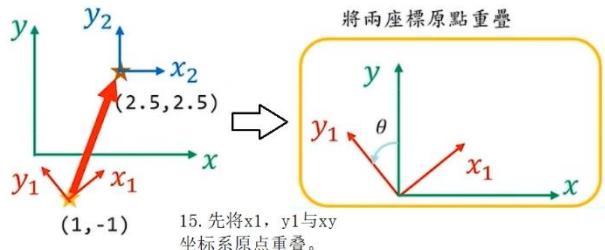
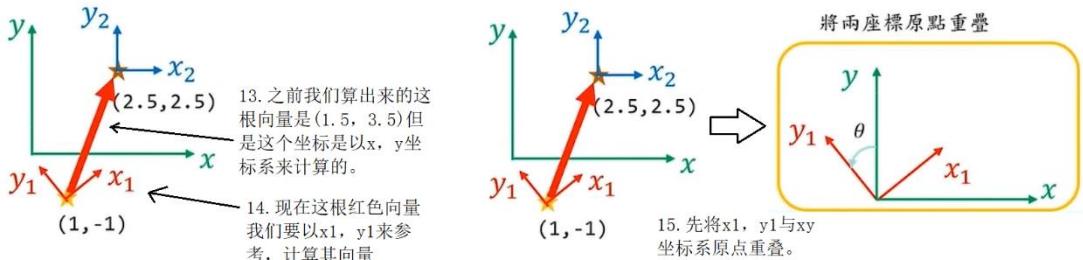


11. 之前我们黄星坐标系参考的是x, y。

12. 现在取消用x, y坐标系做参考。那么红星会在黄星 x_1, y_1 坐标系的什么方向呢？

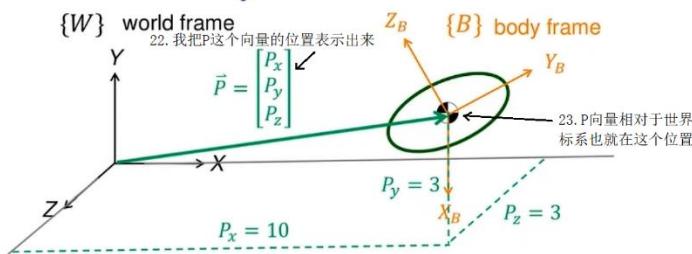
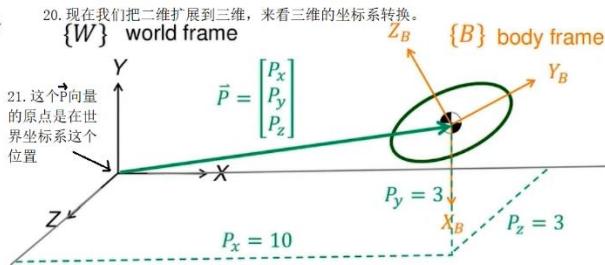
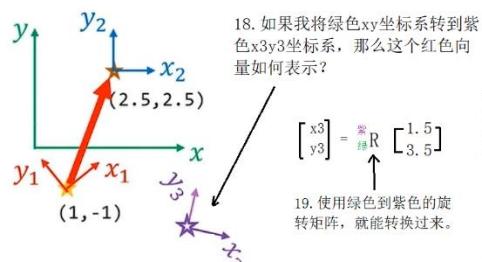


13. 也可以理解成红色这个向量投影到 x_1, y_1 这个坐标系。



17. 所以将xy坐标系的红色向量乘旋转矩阵之后, 得到x1, y1的向量坐标值。

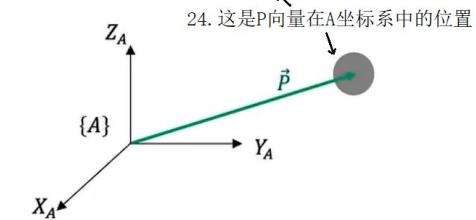
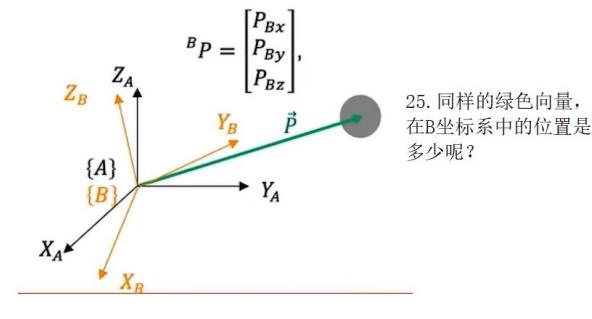
所以同一个向量, 在不同的坐标系下面, 表示方式是不一样的, 向量是向量, 坐标系是坐标系。



$$\vec{p} = \begin{pmatrix} P_X \\ P_Y \\ P_Z \end{pmatrix} = \begin{pmatrix} 10 \\ 3 \\ 3 \end{pmatrix}$$

假设这个是P向量的值

e.g., ${}^B P = \begin{bmatrix} P_{Bx} \\ P_{By} \\ P_{Bz} \end{bmatrix}$, ${}^A P = \begin{bmatrix} P_{Ax} \\ P_{Ay} \\ P_{Az} \end{bmatrix}$



$${}^A P = {}^B R {}^B P$$

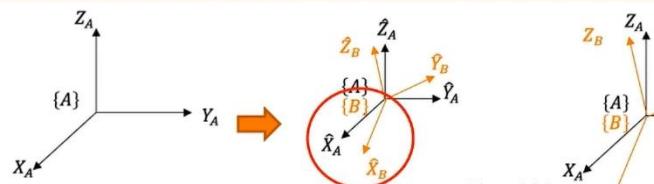
27. 旋转矩阵怎么计算呢?

The three columns of ${}^A_B R$ describe the projection from frame B to frame A:

$\hat{X}_A, \hat{Y}_A, \hat{Z}_A$: the unit basis of frame A (i.e., $\hat{\cdot}^A_A = \frac{(\cdot)_A}{\|\cdot\|}$) 这是xyz在A坐标系的单位向量

28. 假设我 $\hat{X}_B, \hat{Y}_B, \hat{Z}_B$: the unit basis of frame B (i.e., $\hat{\cdot}^B_B = \frac{(\cdot)_B}{\|\cdot\|}$) 这是xyz在B坐标系的单位向量

$${}^A_B R = \begin{pmatrix} | & | & | \\ {}^A X_B & {}^A Y_B & {}^A Z_B \\ | & | & | \end{pmatrix} = \begin{pmatrix} \hat{X}_B \cdot \hat{X}_A & \hat{Y}_B \cdot \hat{X}_A & \hat{Z}_B \cdot \hat{X}_A \\ \hat{X}_B \cdot \hat{Y}_A & \hat{Y}_B \cdot \hat{Y}_A & \hat{Z}_B \cdot \hat{Y}_A \\ \hat{X}_B \cdot \hat{Z}_A & \hat{Y}_B \cdot \hat{Z}_A & \hat{Z}_B \cdot \hat{Z}_A \end{pmatrix}$$



29. 我们可以先把 \hat{X}_B 这个向量投影到 X_A, Y_A, Z_A

30. X_B 投影方式就是与 X_A, Y_A, Z_A 做内积。

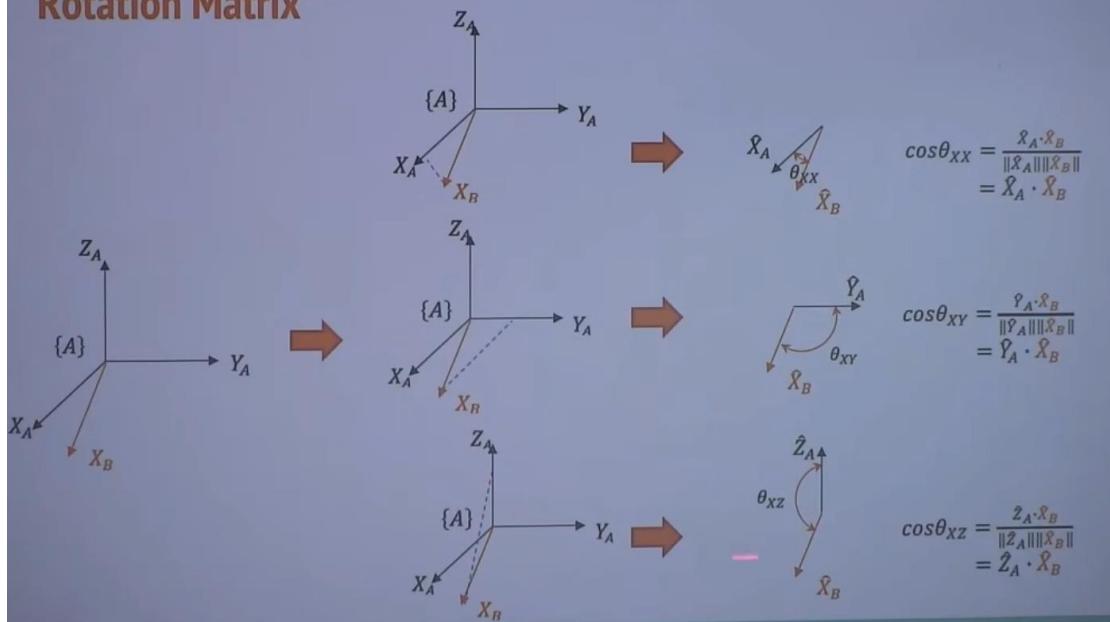
$$\begin{pmatrix} \hat{X}_B \cdot \hat{X}_A & \hat{Y}_B \cdot \hat{X}_A & \hat{Z}_B \cdot \hat{X}_A \\ \hat{X}_B \cdot \hat{Y}_A & \hat{Y}_B \cdot \hat{Y}_A & \hat{Z}_B \cdot \hat{Y}_A \\ \hat{X}_B \cdot \hat{Z}_A & \hat{Y}_B \cdot \hat{Z}_A & \hat{Z}_B \cdot \hat{Z}_A \end{pmatrix}$$

${}^A X_B$ 完成第1个旋转
转矩阵的计算

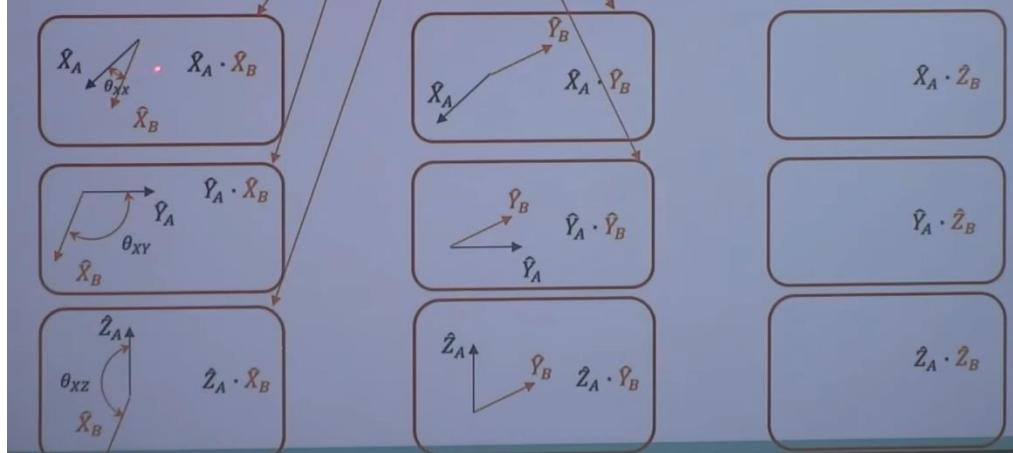
$$\cos \theta_{XX} = \frac{\hat{X}_A \cdot \hat{X}_B}{\|\hat{X}_A\| \|\hat{X}_B\|} = \hat{X}_A \cdot \hat{X}_B$$

31. 投影就是做内积，
也就是根据它们的夹角来计算

Rotation Matrix



$${}^A_B R = \begin{pmatrix} | & | & | \\ {}^A X_B & {}^A Y_B & {}^A Z_B \\ | & | & | \end{pmatrix} = \begin{pmatrix} \hat{X}_B \cdot \hat{X}_A & \hat{Y}_B \cdot \hat{X}_A & \hat{Z}_B \cdot \hat{X}_A \\ \hat{X}_B \cdot \hat{Y}_A & \hat{Y}_B \cdot \hat{Y}_A & \hat{Z}_B \cdot \hat{Y}_A \\ \hat{X}_B \cdot \hat{Z}_A & \hat{Y}_B \cdot \hat{Z}_A & \hat{Z}_B \cdot \hat{Z}_A \end{pmatrix}$$



$$\begin{bmatrix} P_{Ax} \\ P_{Ay} \\ P_{Az} \end{bmatrix} = {}^A_B R \begin{bmatrix} P_{Bx} \\ P_{By} \\ P_{Bz} \end{bmatrix}$$

整理之后

$$\begin{bmatrix} P_{Ax} \\ P_{Ay} \\ P_{Az} \end{bmatrix} = \begin{pmatrix} \hat{X}_B \cdot \hat{X}_A & \hat{Y}_B \cdot \hat{X}_A & \hat{Z}_B \cdot \hat{X}_A \\ \hat{X}_B \cdot \hat{Y}_A & Y_B \cdot Y_A & Z_B \cdot Y_A \\ \hat{X}_B \cdot \hat{Z}_A & \hat{Y}_B \cdot \hat{Z}_A & \hat{Z}_B \cdot \hat{Z}_A \end{pmatrix} \begin{bmatrix} P_{Bx} \\ P_{By} \\ P_{Bz} \end{bmatrix}$$

在B坐标系看到的P向量投影到A坐标系x轴

$$\begin{bmatrix} P_{Ay} \\ P_{Az} \end{bmatrix} = \begin{pmatrix} \hat{X}_B \cdot \hat{X}_A & \hat{Y}_B \cdot \hat{X}_A & \hat{Z}_B \cdot \hat{X}_A \\ \hat{X}_B \cdot \hat{Y}_A & \hat{Y}_B \cdot \hat{Y}_A & \hat{Z}_B \cdot \hat{Y}_A \\ \hat{X}_B \cdot \hat{Z}_A & \hat{Y}_B \cdot \hat{Z}_A & \hat{Z}_B \cdot \hat{Z}_A \end{pmatrix} \begin{bmatrix} P_{Bx} \\ P_{By} \\ P_{Bz} \end{bmatrix}$$

在B坐标系看到的P向量投影到A坐标系y轴

$$\begin{bmatrix} P_{Az} \end{bmatrix} = \begin{pmatrix} \hat{X}_B \cdot \hat{X}_A & \hat{Y}_B \cdot \hat{X}_A & \hat{Z}_B \cdot \hat{X}_A \\ \hat{X}_B \cdot \hat{Y}_A & \hat{Y}_B \cdot \hat{Y}_A & \hat{Z}_B \cdot \hat{Y}_A \\ \hat{X}_B \cdot \hat{Z}_A & \hat{Y}_B \cdot \hat{Z}_A & \hat{Z}_B \cdot \hat{Z}_A \end{pmatrix} \begin{bmatrix} P_{Bx} \\ P_{By} \\ P_{Bz} \end{bmatrix}$$

在B坐标系看到的P向量投影到A坐标系z轴

四元数的理解(弥补旋转矩阵的不足)

四元数表示法如下:

$$q = w + xi + yj + zk$$

w: 表示纯量(实数)

$\vec{i}, \vec{j}, \vec{k}$: 表示三维的向量

x, y, z: 就是ijk方向的复数

$$q^* = w - xi - yj - zk$$

$$i^2 = j^2 = k^2 = -1$$

.	1	i	j	k
1	1	i	j	k
i	i	-1	k	-j
j	j	-k	-1	i
k	k	j	-i	-1

$$i \times j = k = -j \times i$$

$$j \times k = i = -k \times j$$

$$k \times i = j = -i \times k$$

1. 四元数加法和向量加法是一样的
Suppose there are two quaternions

2. q1+q2实数和实数相加

$$q_1 = w_1 + x_1 \vec{i} + y_1 \vec{j} + z_1 \vec{k}$$

3. 复数的部分向量

$$q_2 = w_2 + x_2 \vec{i} + y_2 \vec{j} + z_2 \vec{k}$$

加向量

We define addition as below

$$q_1 + q_2 = (w_1 + w_2) + (x_1 + x_2) \vec{i} + (y_1 + y_2) \vec{j} + (z_1 + z_2) \vec{k}$$

4. 四元数减法也是, 实数-实数, 负数部分向量-向量

$$q_1 - q_2 = (w_1 - w_2) + (x_1 - x_2) \vec{i} + (y_1 - y_2) \vec{j} + (z_1 - z_2) \vec{k}$$

5. 四元数两个向量相乘

$$q_1 = w_1 + x_1 \vec{i} + y_1 \vec{j} + z_1 \vec{k} \quad q_2 = w_2 + x_2 \vec{i} + y_2 \vec{j} + z_2 \vec{k}$$

$$q_1 \otimes q_2 =$$

$$\text{实数部分 } (w_1 w_2 - x_1 x_2 - y_1 y_2 - z_1 z_2) +$$

$$\text{向量部分 } (w_1 x_2 + x_1 w_2 + y_1 z_2 - z_1 y_2) \vec{i} +$$

$$(w_1 y_2 - x_1 z_2 + y_1 w_2 + z_1 x_2) \vec{j} +$$

$$(w_1 z_2 + x_1 y_2 - y_1 x_2 + z_1 w_2) \vec{k}$$

6. 四元数乘法的另一种写法

$$q = w + xi + yj + zk \quad S = \vec{w}$$

$$\vec{V} = xi + yj + zk$$

$$q = (S, \vec{V})$$

S是纯量(实数)部分, \vec{V} 是向量部分

$$q_1 \otimes q_2 = (S_1 + \vec{V}_1) * (S_2 + \vec{V}_2) = (\underline{S_1 S_2 - \vec{V}_1 \cdot \vec{V}_2}, \underline{\vec{S}_1 \vec{V}_2 + S_2 \vec{V}_1 + \vec{V}_1 \times \vec{V}_2})$$

实数部分 两个向量做内积

$$q_1 \otimes q_2 =$$

$$(w_1 w_2 - x_1 x_2 - y_1 y_2 - z_1 z_2) +$$

$$(w_1 x_2 + x_1 w_2 + y_1 z_2 - z_1 y_2) \vec{i} +$$

$$(w_1 y_2 - x_1 z_2 + y_1 w_2 + z_1 x_2) \vec{j} +$$

$$(w_1 z_2 + x_1 y_2 - y_1 x_2 + z_1 w_2) \vec{k}$$

8. 四元数共轭计算

$$q^* = w - xi - yj - zk = [S - \vec{V}]$$

7. 四元数模的计算方法

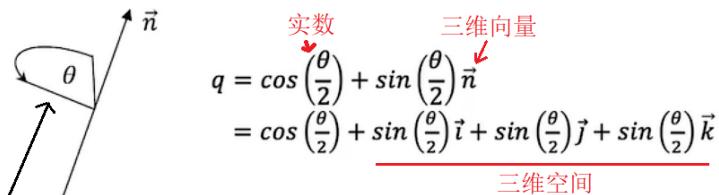
$$\|q\| = \sqrt{qq^*} = \sqrt{q^*q} = \sqrt{w^2 + x^2 + y^2 + z^2}$$

$$qq^* = q^*q$$

9. 四元数倒数计算

$$q^{-1} = \frac{q^*}{qq^*} = \frac{q^*}{\|q\|^2}, \text{ and } q^{-1} = q^* \text{ if } \|q\| = 1$$

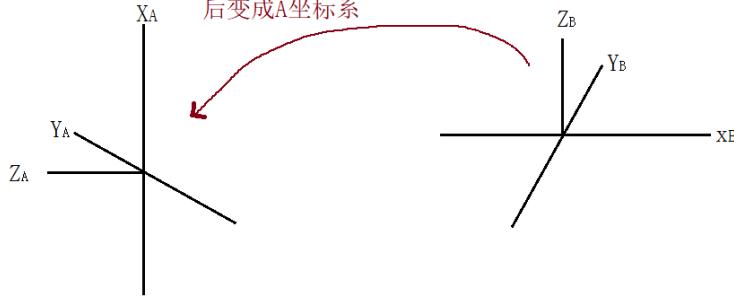
10. 四元数有实数和向量两部分



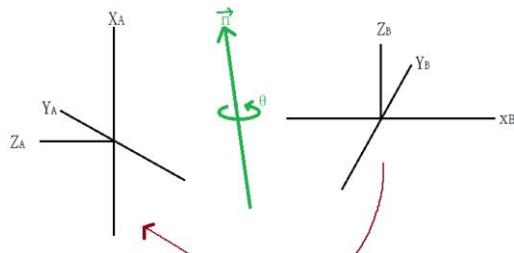
12. \vec{n} 向量绕着 θ 转了一个角度

11. \vec{n} 是三维空间中的一根向量。

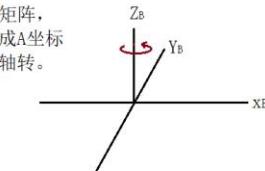
13. 之前讲旋转矩阵，就算用R讲B坐标系旋转之后变成A坐标系



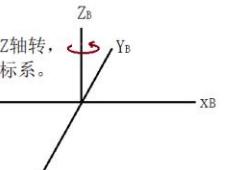
15. 如果是旋转矩阵，要把B坐标系转成A坐标系，我先要对Z轴转。



16. 然后对Y轴转



17. 然后又对Z轴转，才能达到A坐标系。



14. 我们可以使用四元数方式，将B坐标系统绕着 \vec{n} 向量旋转一个 θ 角度，形成A坐标系。

18. 但是四元数就不用像旋转矩阵那么麻烦。只需转动一次，就能把B坐标系转到A坐标系。所以四元数计算量要少很多。

四元数主要是通过转轴实现空间向量的转换

Given a vector $p = x_1\vec{i} + x_2\vec{j} + x_3\vec{k}$, it can be converted into a quaternion from as

1. 假如空间中有一个向量

$$p = \begin{bmatrix} 0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix}, \quad \text{2. 把空间向量写成四元数，就可以是这样}$$

where the scalar part is 0.

因为是三维向量，所以实数可以不写，写0

Now an arbitrary axis of rotation \vec{n} expressed by a quaternion is given by

$$q = \cos\left(\frac{\theta}{2}\right) + \sin\left(\frac{\theta}{2}\right)\vec{n}.$$

3. 找到一个 n 的转动向量

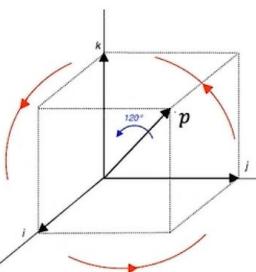
$$p' = qpq^{-1} \quad (4.q1)$$

4. p' 向量就是转动之后的向量

5. 也可以用旋转矩阵来做转换

$$p' = Rp,$$

$$\text{where } \mathbf{R} = \begin{bmatrix} 1 - 2s(q_j^2 + q_k^2) & 2s(q_i q_j - q_k q_r) & 2s(q_i q_k + q_j q_r) \\ 2s(q_i q_j + q_k q_r) & 1 - 2s(q_i^2 + q_k^2) & 2s(q_j q_k - q_i q_r) \\ 2s(q_i q_k - q_j q_r) & 2s(q_j q_k + q_i q_r) & 1 - 2s(q_i^2 + q_j^2) \end{bmatrix}$$



$$p' = qpq^{-1} \quad (4.q2)$$

两种旋转方式都可以

1. 比如我有一个向量 $p = (0, 2, 0, 0)$ 我想将这个向量沿着Z轴旋转90度

Z轴旋转= 90度 90度

$$q = \cos\left(\frac{\theta}{2}\right) + \sin\left(\frac{\theta}{2}\right)\vec{n} \quad 4. 因为是沿着Z轴转动, 所以这个n必须是Z轴。也就是Z轴为1 (0, 0, 1)$$

$$= \cos\left(\frac{\pi/2}{2}\right) + \sin\left(\frac{\pi/2}{2}\right)(0, 0, \vec{k})$$

$$(0, 0, 1)$$

$$q = (\underline{0.707}, \underline{0, 0, 0.707})$$

$$q^{-1} = \text{向量部分加负号 } (0.707, 0, 0, -0.707)$$

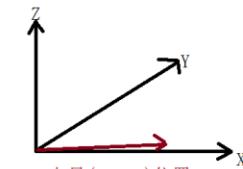
$$p' = qpq^{-1} \quad p \text{ 向量经过转轴旋转}$$

$$= (0.707, 0, 0, 0.707)(0, 2, 0, 0)(0.707, 0, 0, -0.707)$$

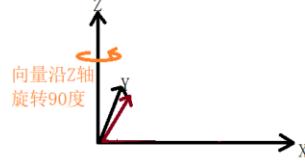
$$= (0, 0, 2, 0)$$

$$p' = 2\vec{j} \quad ijk, j在中间, 所以说2j也就是(0, 2, 0)$$

这就是转动p向量案例



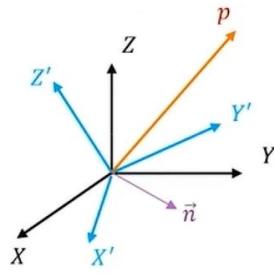
2. p 向量(2, 0, 0)位置



3. 绕Z轴旋转90度应该跑到Y轴(0, 2, 0)

你看, 向量长度没变还是2, 但是值的位置变了

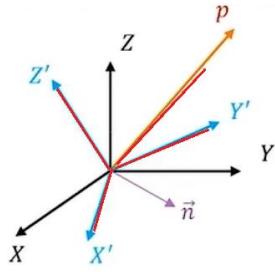
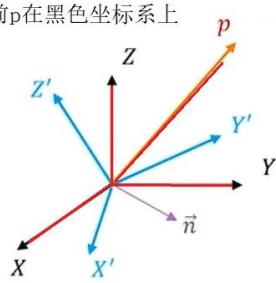
上一节是1个坐标系, 向量的转动。这一节是多个坐标系, 1个向量的转动



$$p = x_1\vec{i} + x_2\vec{j} + x_3\vec{k} \quad \text{向量 } p$$

$$p' = q^{-1}pq \quad (4.q3) \quad q^{-1} \text{位置变了}$$

1. 当前p在黑色坐标系上



2. 当黑色坐标系经过旋转之后, 跑到了蓝色坐标系上。那么同样的p向量, 在蓝色坐标系上, 写法是什么样的。

1. $p = 2\vec{j} (2, 0, 0)$, 现在将XYZ坐标系转动一个90度, 形成X'Y'Z'坐标系, 请问p向量在新坐标系是多少?

Answer: according the definition of quaternion

$$q = \cos\left(\frac{\theta}{2}\right) + \sin\left(\frac{\theta}{2}\right)\vec{n}$$

$$= \cos\left(\frac{\pi/2}{2}\right) + \sin\left(\frac{\pi/2}{2}\right)(0, 0, \vec{k})$$

We can apply the relation from (4.q3)

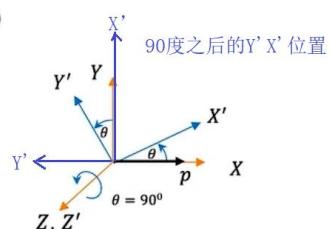
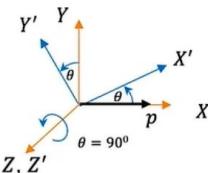
$$p' = q^{-1}pq$$

$$= (0.707, 0, 0, -0.707)(0, 2, 0, 0)(0.707, 0, 0, 0.707)$$

$$= (0, 0, -2, 0) \quad P \text{向量}$$

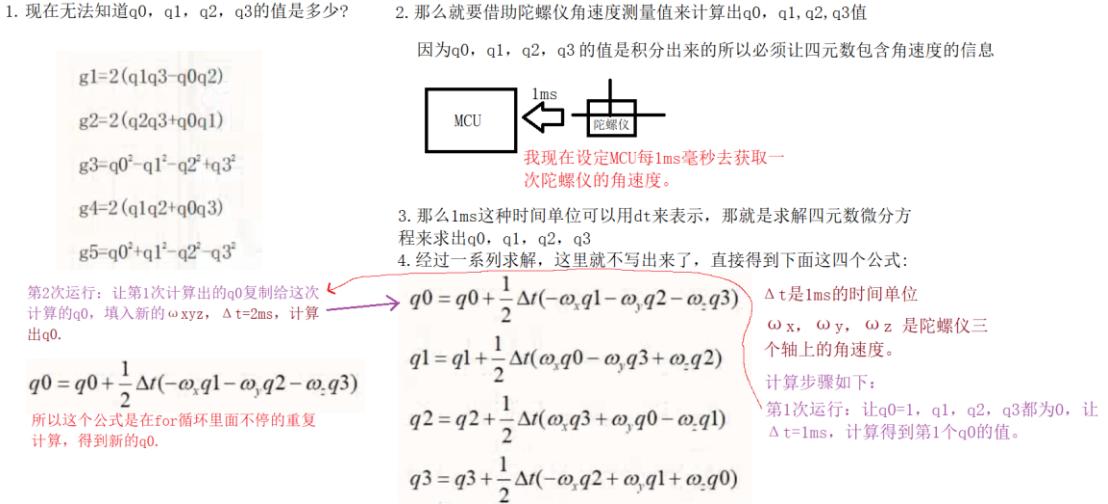
$$= -2\vec{j}$$

旋转坐标系之后新的p向量 = (0, -2, 0)



四元数使用(第2部分)

我们来完成四元数使用(第1部分)未完成的内容



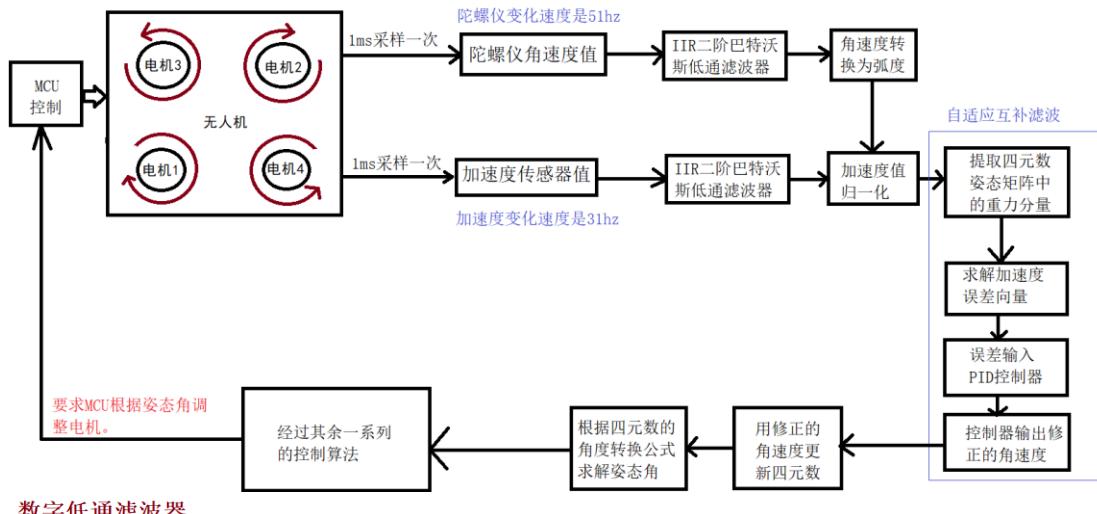
四元数归一化方法

$$ax = \frac{ax}{\sqrt{ax^2 + ay^2 + az^2}}$$

$$ay = \frac{ay}{\sqrt{ax^2 + ay^2 + az^2}}$$

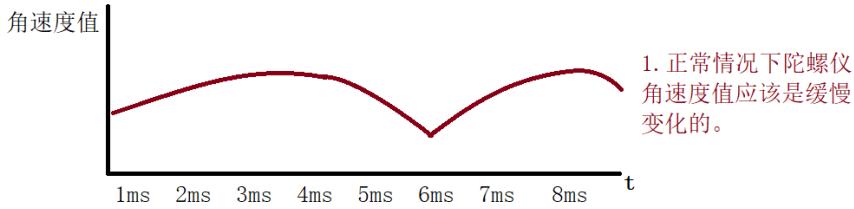
$$az = \frac{az}{\sqrt{ax^2 + ay^2 + az^2}}$$

无人机飞控算法实现

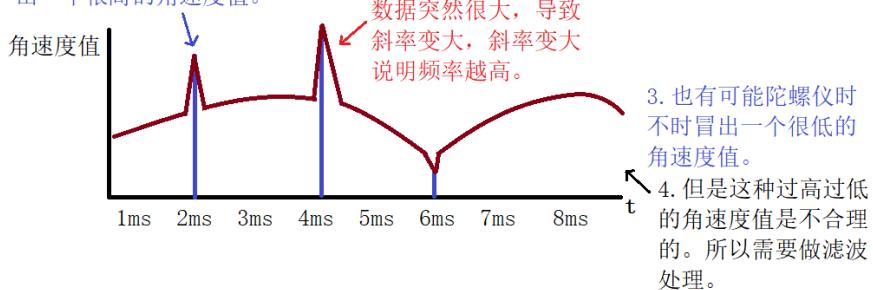


数字低通滤波器

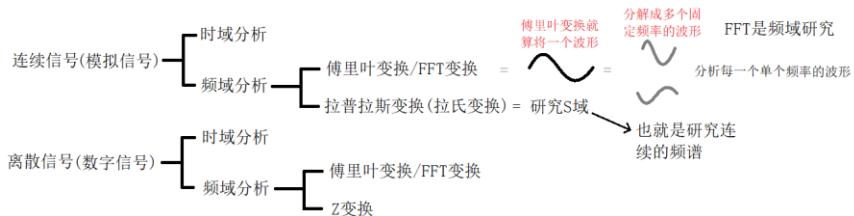
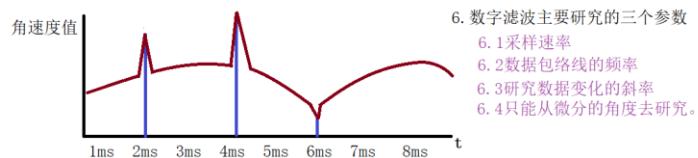
如：我单片机1ms毫秒采集一次陀螺仪角速度



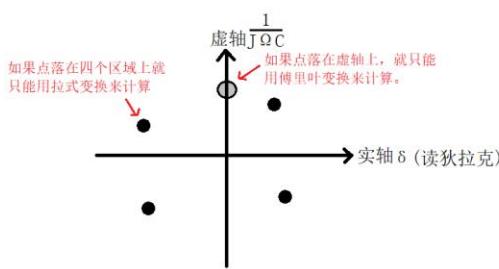
2. 但是因为陀螺仪有干扰，就造成了时不时冒出一个很高的角速度值。



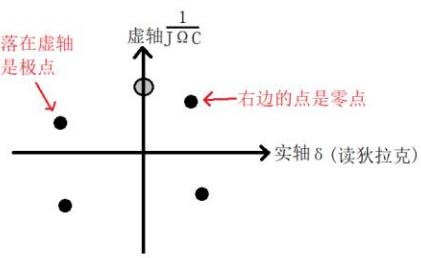
5. 因为陀螺仪角速度变化最高频率也就51hz。所以我们用低通滤波的方式过滤掉高频干扰。



拉氏变换 S域研究

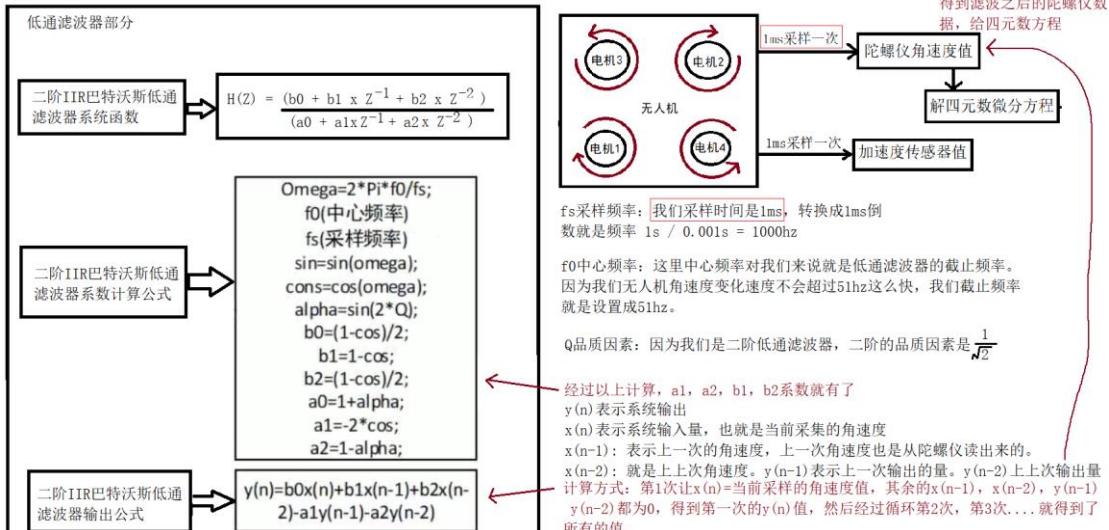


零极点含义



所有的‘点’只有落在虚轴左边，也就是极点位置，系统才稳定。
如果不落在虚轴左边，系统就具有发散性，不稳定。

$$H(s) = \frac{[1+2s]}{[1+s]} \cdot \frac{[1+3s]}{[1+5s]} = 0$$



加速度计误差处理

利用陀螺仪去更新的四元数需要进行归一化，所以加速度计获得的值也需要归一化。这样陀螺仪和加速度计两者才能对应。

获取的加速度值 ax, ay, az 分别对应 x, y, z 轴的值，其归一化方法如下：

$$ax = \frac{ax}{\sqrt{ax^2 + ay^2 + az^2}}$$

$$ay = \frac{ay}{\sqrt{ax^2 + ay^2 + az^2}}$$

$$az = \frac{az}{\sqrt{ax^2 + ay^2 + az^2}}$$

1. 获取陀螺仪算出的姿态矩阵中的重力分量（因为加速度计是测得的物体坐标系下的值，所以我们也要提取利用角速度算出的姿态矩阵中的物体坐标系下的重力分量）。重力分量记为 V_x, V_y, V_z ，具体计算方法如下：

通过四元数计算的从 E 系（地理坐标系）变换到 b 系（物体坐标系）姿态矩阵为：

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

则提取 b 系(物体坐标系)下重力分量为

$$\begin{bmatrix} V_x \\ V_y \\ V_z \end{bmatrix} = \begin{bmatrix} 1-2(q_2^2+q_3^2) & 2(q_1q_2+q_0q_3) & 2(q_1q_3-q_0q_2) \\ 2(q_1q_2-q_0q_3) & 1-2(q_1^2+q_3^2) & 2(q_2q_3+q_0q_1) \\ 2(q_1q_3+q_0q_2) & 2(q_2q_3-q_0q_1) & 1-2(q_1^2+q_2^2) \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

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

我们知道加速度计的值是
G是重力

但是加速度值归一化之后，G的大小我就不再关注了。G默认取1

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

1. 将加速度计获得的重力向量归一化后的值与提取的姿态矩阵的重力向量叉乘获取姿态误差，根据叉乘定义

$\vec{A} \times \vec{B} = (y_1z_2-y_2z_1, x_1z_2-x_2z_1, x_1y_2-x_2y_1)$, 得出姿态误差向量

$$\begin{bmatrix} ax \\ ay \\ az \end{bmatrix} \times \begin{bmatrix} V_x \\ V_y \\ V_z \end{bmatrix} = \begin{bmatrix} ay * V_z - az * V_y \\ az * V_x - ax * V_z \\ ax * V_y - ay * V_x \end{bmatrix}$$

记 ex、ey、ez 为误差向量对应的 x、y、z 的三个元素，即有

$$ex = ay * V_z - az * V_y$$

$$ey = az * V_x - ax * V_z$$

$$ez = ax * V_y - ay * V_x$$

有了这个偏差值，也不能马上进行调整，需要有个缓慢的调整过程。
这个缓慢的过程叫积分
所以下面会用到PID算法

PID调整误差方法

P的调整作用最大，最直接，但是过冲也会很大。

I 积分，更多是针对直流量进行积分。

D 微分，更多是对交流变化量很大的情况进行微分。

我们用角速度为例，来解释PID用法

例如：弧度 1度 = 0.0174弧度

gx = gx * 0.0174f (角速度 * 0.0174 = 弧度)，这时候gx就是角度

gy = gy * 0.0174f (gy角度)

gz = gz * 0.0174f (gz角度)

PID公式如下：

角速度弧度 gx = gx + Kp x Ex + Ki x Ex x dt + $\frac{Kd x Ex}{dt}$ ← 这是修正公式

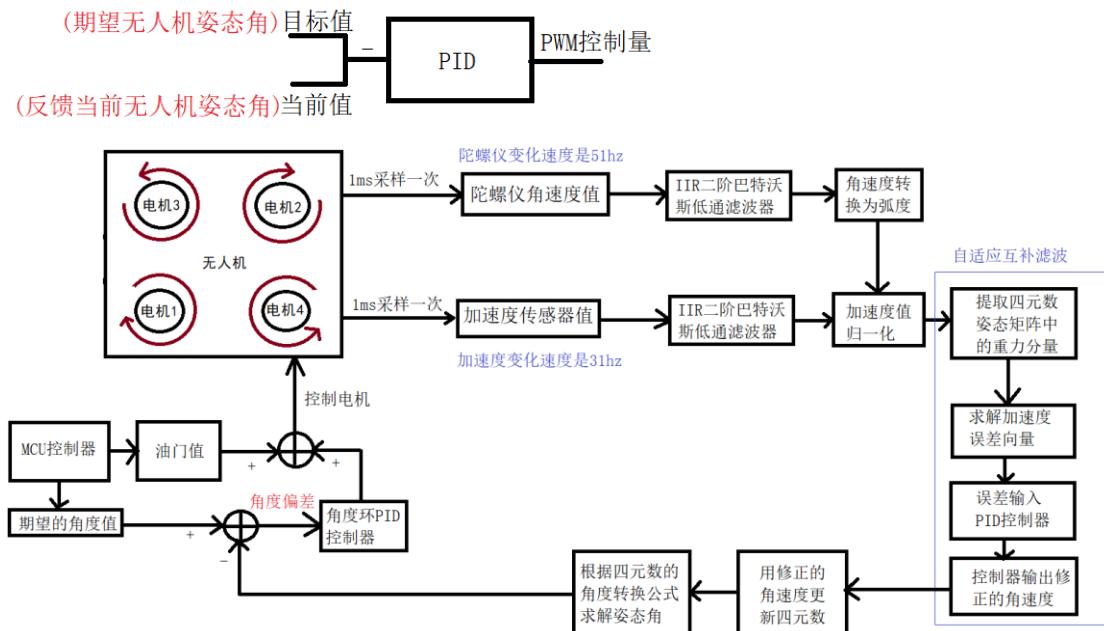
角速度弧度 gy = gy + Kp x Ey + Ki x Ey x dt + $\frac{Kd x Ey}{dt}$ ← 这是修正公式

角速度弧度 gz = gz + Kp x Ez + Ki x Ez x dt + $\frac{Kd x Ez}{dt}$ ← 这是修正公式

本次弧度 = 上一次角速度弧度 + 误差量

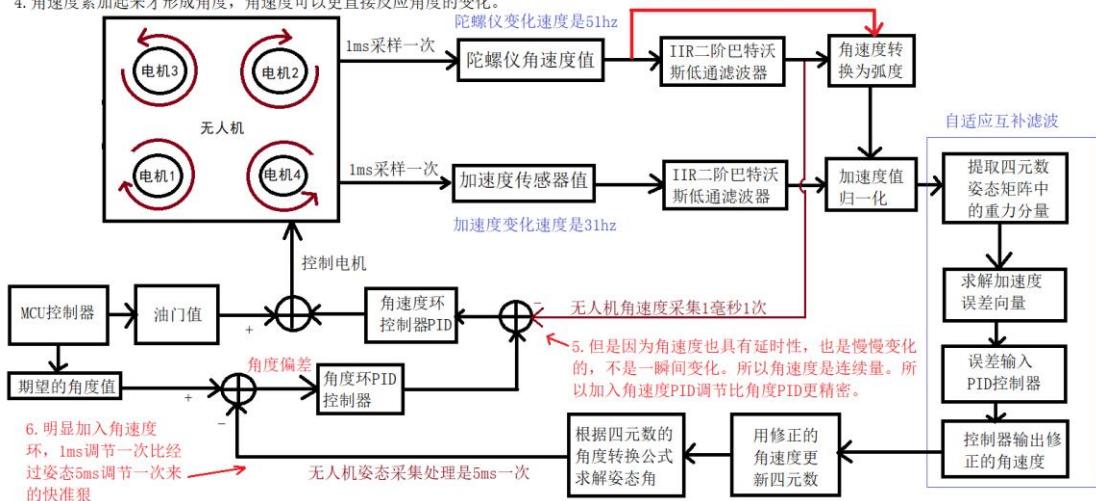
自适应互补滤波算法

无人机修正误差飞行基本思路



1. 我们知道角度是角速度的积分值，所以角度的变化是需要一段时间的。
2. 如果只控制角度变化，那么控制的精密度是否能达到要求呢？
3. 我们说角度的微分就是角速度，也就是角度是由很多个角速度细分点形成的。
4. 角速度累加起来才形成角度，角速度可以更直接反应角度的变化。

为了加快响应速度，角速度可以不用滤波。滤波会造成延时。



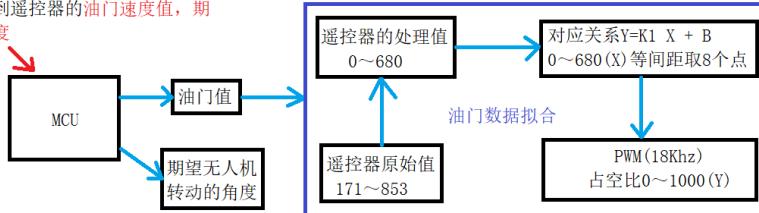
无人机飞控算法代码实现

遥控器油门数据，角度数据获取处理

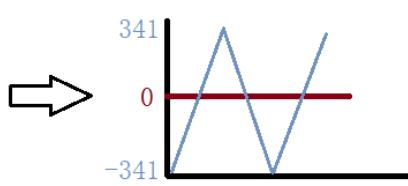
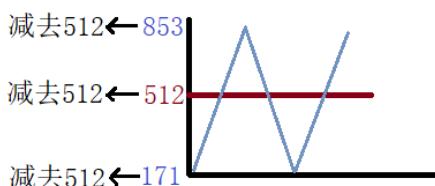
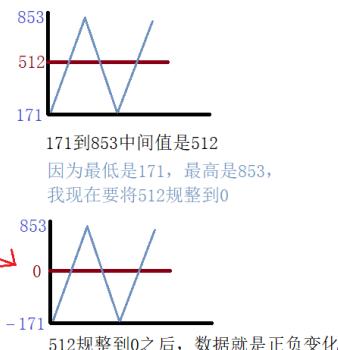
```
typedef struct
{
    油门, 方向舵变量定义
    float pitch;
    float roll;
    float yaw;
}float_PRY;

float_PRY temp_Expect;
float_PRY Expect;
```

无人机获取到遥控器的油门速度值，期望旋转的角度



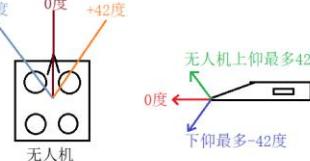
```
float temp_throttle = 0.0f; //油門值
/*获取油门数据和角度数据*/
void nrf51822_get_data(void)
{
    temp_throttle = 获取无线传输过来的油门值
    //遥控器的油门值,初始为171~853.减去171后变为0~680
    //roll和pitch就是期望的角度值
    temp_Expect.roll = 得到期望的roll值 - 512.0f;
    temp_Expect.pitch = 得到期望的pitch值 - 512.0f;
}
```



当中心点512数据规整为0，那么期望的最大两个方向角度值也会规整成正负对称的值，如341，-341

但是341和-341都是数字量，我们要将其转换成角度值

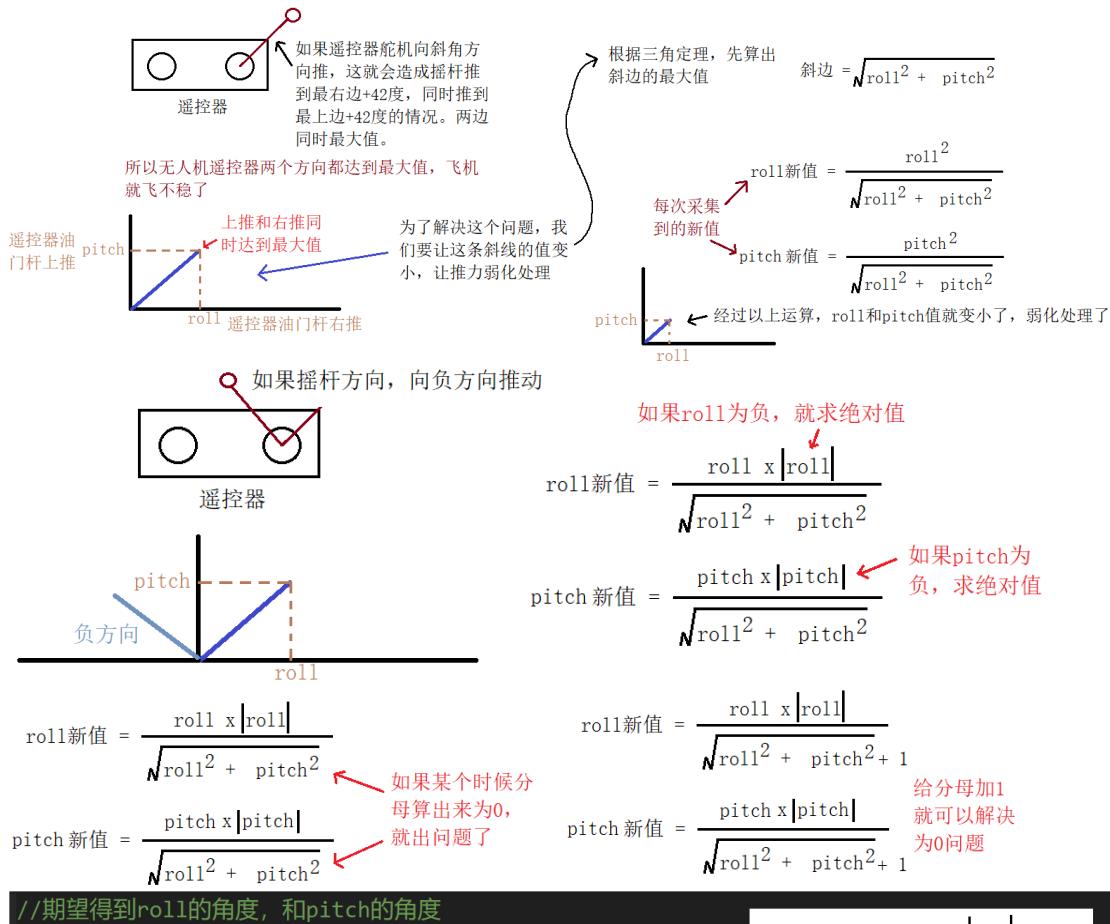
遥控器推到最左边是-42度，推到最右边是+42度



因为遥控器数字量是+341 ~ -341，那么要转成-42度 ~ +42度。就只有将341/42 = 8.11. 所以需要进行除运算。

```
float temp_throttle = 0.0f; //油門值必须是全局变量
int main(void)
{
    temp_throttle = 获取无线传输过来的油门值
    //遥控器的油门值,初始为171~853,减去171后变成0~680
    //roll 和 pitch就是期望的角度值. (roll左右控制, pitch前后控制)
    temp_Expect.roll = 得到期望值roll值 - 512.0f;
    temp_Expect.pitch = 得到期望的pitch值 - 512.0f;

    //期望得到roll的角度, 和pitch的角度
    temp_Expect.roll = temp_Expect.roll / 8.0f;
    temp_Expect.pitch = temp_Expect.pitch / 8.0f;
}
```

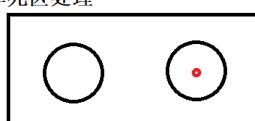


```
//期望得到roll的角度，和pitch的角度
temp_Expect.roll = temp_Expect.roll / 8.0f;
temp_Expect.pitch = temp_Expect.pitch / 8.0f;

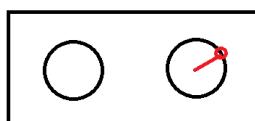
//得到期望的舵值
temp_Expect.yaw = 得到期望的舵值

float expect_len = temp_Expect.roll * temp_Expect.roll +
                   temp_Expect.pitch * temp_Expect.pitch;
Expect.roll = (temp_Expect.roll * fabs(temp_Expect.roll)) / sqrtf(expect_len + 1.0f);
Expect.pitch = temp_Expect.pitch * fabs(temp_Expect.pitch) / sqrtf(expect_len + 1.0f);
```

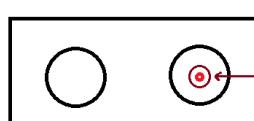
摇杆回弹死区处理



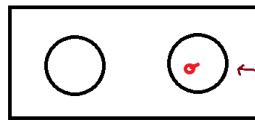
1. 手柄摇杆本来在中心位置



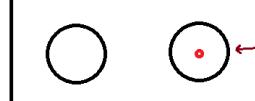
2. 当我把摇杆搬到最大位置时，飞机飞行方向发生偏移。



5. 所以我将摇杆回弹的这块区域都定义为0



3. 当我不想控制无人机方向了，手松开摇杆，发现摇杆并没有第1时间回到手柄中心位置。



4. 摆杆有点像弹簧，经过一个回弹的过程回到摇杆中心。

42度
0度
-42度
42度
-42度

6. 也就是摇杆不管从哪个方向松手，回弹的时候，都不会超过这块区域，那么这块区域本来是有角度的，我就让这个角度都变成0度。使其不会因为摇杆回弹，让无人机出现方向上的抖动。

```

/*
 * 死区控制
 * value当前输入的摇杆, deadband死区大小(传入我要设置为0的实际角度)
 * 返回当前摇杆值
 */
float applyDeadband(float value, float deadband)
{
    if (fabs(value) <= deadband) //如果摇杆值(绝对值, 不管正方向还是负方向) < 死区值
    {
        value = 0; //让遥感值为0
    }
    if (value > deadband) //如果摇杆值 > 死区值
    {
        value = value - deadband; //减去死区值, 得到摇杆方向值
    }
    if (value < -deadband) //如果摇杆值是负方向扳动
    {
        value = value + deadband; //负数的时候还是把死去减去掉了, 得到负方向值
    }
    return value;
}

```

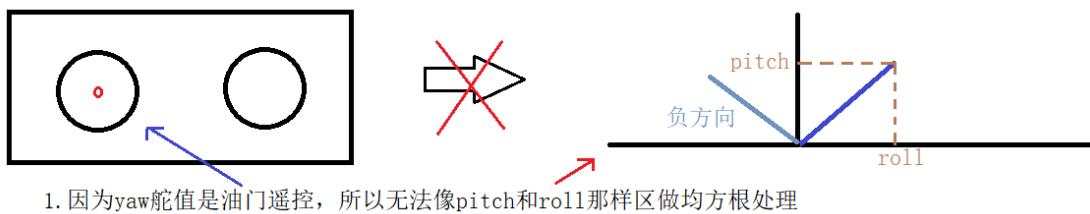
```

int main(void)
{
    temp_throttle = 获取无线传输过来的油门值
    //遥控器的油门值, 初始为171~853, 减去171后变成0~680
    //roll 和 pitch就是期望的角度值. (roll左右控制, pitch前后控制)
    temp_Expect.roll = 得到期望值roll值 - 512.0f;
    temp_Expect.pitch = 得到期望的pitch值 - 512.0f;
    //期望得到roll的角度, 和pitch的角度
    temp_Expect.roll = temp_Expect.roll / 8.0f;
    temp_Expect.pitch = temp_Expect.pitch / 8.0f;
    //得到期望的舵值
    temp_Expect.yaw = 得到期望的舵值
    float expect_len = temp_Expect.roll * temp_Expect.roll +
                       temp_Expect.pitch * temp_Expect.pitch;
    Expect.roll = (temp_Expect.roll * fabs(temp_Expect.roll)) / sqrtf(expect_len + 1.0f);
    Expect.pitch = temp_Expect.pitch * fabs(temp_Expect.pitch) / sqrtf(expect_len + 1.0f);

    //做死区处理
    Expect.roll = applyDeadband(Expect.roll, 5.0f); //死区设置为5.0
    Expect.pitch = applyDeadband(Expect.pitch, 5.0f); //死区设置为5.0
}

```

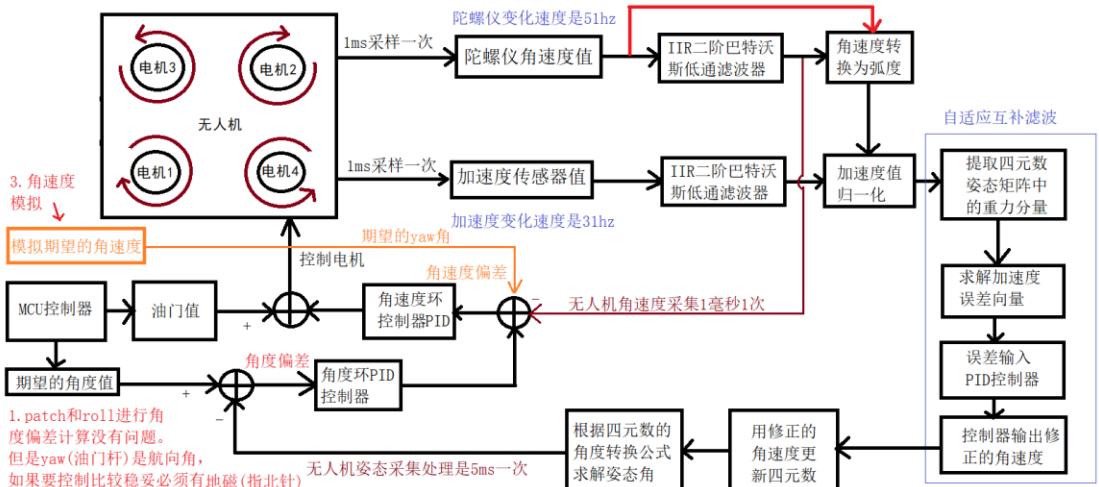
舵值处理



$$yaw = \frac{roll * |roll|}{350}$$

由于遥控器最大值我们处理减去512之后是300多, 本来是171~853. 但是减去512就变成330左右, 所以舵机弱化处理要比原来小一点, 我取350。

2. 但是我现在这个无人机项目没有用到地磁，那么就没有指北针，所以航向角yaw就不准。既然航向角不准，就无法读出一个角度。既然没有角度，就无法做角度环，所以只有做角速度环。



```
/****************************************************************************
 * 限定最大最小输出
 * 参数: amt当前值, low最小值, high最大值
 *****/
float constrain_float(float amt, float low, float high)
{
    if (isnan(amt)) //如果amt经过isnan之后变成很大的值, 结果为真。
    {
        return (low + high) * 0.5f; //最大值+最小值 /2 返回
    }
    //先比小, 再比大。如果amt<low返回low, 如果amt>high返回high。如何两个不成立, 返回当前值amt
    return ((amt) < (low)? (low) : ((amt) > (high)? (high) : (amt)));
}
```

```
//得到期望的舵值
temp_Expect.yaw = 得到期望的舵值
isnan(..)就是计算最大值, 这个最大值是根据编译器来确定的, keil
编译器有自己的最大值要求, 我只需要用isnan就可以了

/*防斜边处理*/
float expect_len = temp_Expect.roll * temp_Expect.roll +
    temp_Expect.pitch * temp_Expect.pitch;
Expect.roll = (temp_Expect.roll * fabs(temp_Expect.roll)) /sqrtf(expect_len + 1.0f);
Expect.pitch = temp_Expect.pitch * fabs(temp_Expect.pitch)/sqrtf(expect_len + 1.0f);

//做死区处理
Expect.roll = applyDeadband(Expect.roll,5.0f); //死区设置为5.0
Expect.pitch = applyDeadband(Expect.pitch,5.0f); //死区设置为5.0
yaw = yaw * |yaw| 油门摇杆弱化处理
油门数据处理

//yaw(油门杆)防斜边处理, 弱化处理
float expect_len_raw = temp_Expect.yaw * fabs(temp_Expect.yaw) / 350;
//yaw(油门数据)比大小取值
Expect.yaw = constrain_float(expect_len_raw,-330.0f,330.0f); //限定油门最大最小输出
Expect.yaw = applyDeadband(Expect.yaw, 35.0f); //油门杆死区处理

油门也需要死区控制, 而且油门是用的角速度测量, 所以死区要比一般角度测量的5度大很多, 这里取的35度
```

飞行模式设置

```
typedef enum{
    FLYMODE_FLP = 0, //翻跟斗模式
    FLYMODE_6AXIE = 1 //6轴模式
}FLY_MODE; //飞行模式
FLY_MODE Fly_Mode;

typedef enum
{
    FlyStop = 0, //停止模式
    FlyIdle = 1, //怠速模式
    FlyStart = 2, //起飞模式
}FLY_STATE;
FLY_STATE Fly_State;

unsigned char last_rfdata_9; //遥控器值
unsigned char last_rfdata_10; //遥控器值
bool start_input_once_flag = false;

int main(void)
{
    Expect.yaw = constrain_float(expect_len_raw, -330.0f, 330.0f); //限定油门最大最小输出
    Expect.yaw = applyDeadband(Expect.yaw, 35.0f); //油门杆死区处理

    if(start_input_once_flag == false)
    {
        start_input_once_flag = true; //系统启动之后，这个状态值就是唯一了
        last_rfdata_9 = RFdata[9]; //翻滚进入开关，遥控器按键值
        last_rfdata_10 = RFdata[10]; //飞机启停开关，遥控器按键值
    }
    if(RFdata[9] != last_rfdata_9) //如果第二次传输来的RFdata[9]不等于之前值，证明按键按下
    {
        if(Fly_Mode == FLYMODE_FLP) //如果现在是翻跟斗模式
        {
            Fly_Mode = FLYMODE_6AXIE; //我将其变成6轴模式
        }
        else
        {
            Fly_Mode = FLYMODE_FLP; //如果是6轴模式，就变成翻跟头模式
        }
        last_rfdata_9 = RFdata[9]; //更新这次按键的值
    }
    if(RFdata[10] != last_rfdata_10) //怠速/停止模式按键切换
    {
        if(Fly_State == FlyStop)
            Fly_State = FlyIdle;
        else
            Fly_State = FlyStop;
        last_rfdata_10 = RFdata[10];
    }
}
```

油门曲线拟合

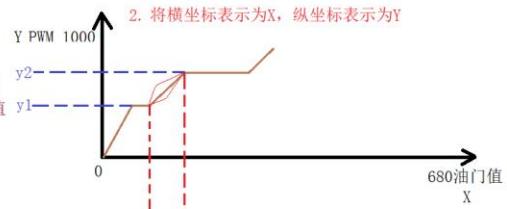
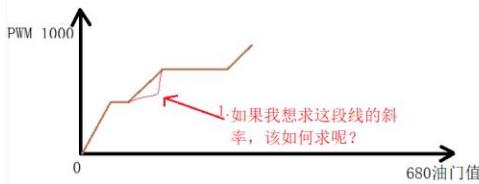
差值X	85	85	85	85	85	85	85	85
0	85	170	255	340	425	510	595	680
0	180	270	330	400	490	590	730	1000

分段线性函数是 $y = kx + b$

求k和b的值

$$k = \frac{y_2 - y_1}{x_2 - x_1}$$

$$b = y_1 - kx_1$$



3. 用 $x_2 - x_1 = \text{值1}$

用 $y_2 - y_1 = \text{值2}$

值2 / 值1 = K

5. 假设我油门值X读出来是100，那对应的Y是多少呢？

0	85	170	255	340	425	510	595	680
---	----	-----	-----	-----	-----	-----	-----	-----

6. 我油门值100，落在这个区间

0	180	270	330	400	490	590	730	1000
---	-----	-----	-----	-----	-----	-----	-----	------

7. 那么y就落在这个区间

```
/*
 * 油门曲线拟合, 获取Y = KX + B 中K值
 * 参数: 两坐标点的坐标, Ax , Ay , Bx , By
 * 返回值: K
 */
float Get_K(float Ax, float Ay, float Bx , float By)
{
    float Tmp1 , Tmp2;
    Tmp1 = By - Ay;
    Tmp2 = Bx - Ax;
    return (Tmp1 / Tmp2);
}
```

用 $x_2 - x_1 = \text{值1}$
用 $y_2 - y_1 = \text{值2}$
值2 / 值1 = K

```
/*
 * 油门曲线拟合, 获取Y = KX + B 中B值
 * 参数: 两坐标点的坐标, Ax , Ay , Bx , By
 * 返回值: K
 */
float Get_B(float Ax, float Ay, float Bx , float By)
{
    float Tmp1 , Tmp2, Tmpk;
    Tmp1 = By - Ay;
    Tmp2 = Bx - Ax;
    Tmpk = (Tmp1 / Tmp2);
    return (Ay - Ax * Tmpk);
}
```

用 $y_1 - x_1 * k = b$

```
const unsigned short PointX[] = {0,85,170,255,340,425,510,595,680};
const unsigned short PointY[] = {0,180,270,330,400,490,590,730,1000};
float Curve_K[8] = {0};
float Curve_B[8] = {0};
```

差值X	85	85	85	85	85	85	85	
0	85	170	255	340	425	510	595	680
0	180	270	330	400	490	590	730	1000

遥控器油门值
PWM油门输出值

差值Y 180 90 60 70 90 100 140 270

```
/*
 * 油门曲线, 分段曲线拟合K和B
 */
void Calc_Curve(void)
{
    unsigned char i;
    for (i = 0; i < 8; i++)
    {
        Curve_K[i] = Get_k(PointX[i],PointY[i],PointX[i+1],PointY[i+1]);
        Curve_B[i] = Get_k(PointX[i],PointY[i],PointX[i+1],PointY[i+1]);
    }
}
```

每一段的K和B 放入数组

差值X	85	85	85	85	85	85	85	
0	85	170	255	340	425	510	595	680
0	180	270	330	400	490	590	730	1000

遥控器油门值
PWM油门输出值

每一段的K和B

1. 我们前面讲过，这个拟合数据有8个区间段

0	85	170	255	340	425	510	595	680
1	100	3	4	5	6	7	8	八个区间段
0	180	270	330	400	490	590	730	1000

2. 但是我计算油门K和B的时候只有7个区间段，因为i=8退出循环。

```
void Calc_Curve(void)
{
    unsigned char i;
    for (i = 0; i < 8; i++)
    {
        Curve_K[i] = Get_k(PointX[i], PointY[i], PointX[i+1], PointY[i+1]);
        Curve_B[i] = Get_b(PointX[i], PointY[i], PointX[i+1], PointY[i+1]);
    }
}
```

3. 因为我们数组下标只有[0~7]，所以这个100的油门值要怎么取区间位置呢？

100 拟合到数组[0~7]里面，取对应的系数。

比如我 $\frac{100}{110} = 0.9$ ，就只有取数组[0]的系数

比如我油门值是650 $\frac{650}{100} = 6.5$ 就取数组[6]的系数

4. 所以到底是选择除以110，还是除以100，主要看自己实际调试的情况，没有固定的计算值。

我们可以称它为M值

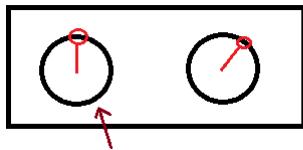
5. 无人机的惯性要求大，马达功率小，M值就需要偏小。
无人机的惯性要求大，马达功率越大，M值就可以偏大。

```
float gCurve_Y = 0;
/***********************/
/* 油门曲线拟合处理函数
 * 参数: Curve_X 处理前的数据
 * 返回值: Curve_Y 拟合后的数据
 */
float Curve_Ctrl(float Curve_X)
{
    unsigned char Curve_index; // 这个就是M值
    Curve_index = Curve_X / 118.0f;
    if(Curve_index > 7)
        Curve_index = 7;
    else
    {}
    gCurve_Y = Curve_X * Curve_K[Curve_index] + Curve_B[Curve_index]; //y = kx + b
    return gCurve_Y; // 返回实际给油门的PWM值
}
```

油门数据获取

```
float temp_throttle = 0.0f; //油门值必须是全局变量
float throttle;
int main(void)
{
    Calc_Curve(); //在循环之前先初始化油门曲线表

    //以下的代码可以放入死循环
    temp_throttle = 获取无线传输过来的油门值
    //遥控器的油门值，初始为171~853，减去171后变成0~680
    throttle = Curve_Ctrl(temp_throttle); //将拟合之后0~1000的油门数据输出
    if(throttle <= 0) //油门值超出范围保护措施
        throttle = 0;
    else if(throttle > 1000)
        throttle = 1000;
    else {}
}
```



1. 当遥控器和无人机连接成功的时候，因为有些遥控器比较劣质，油门杆无法自动弹回中间，导致油门杆一直处于启动飞行状态。

```

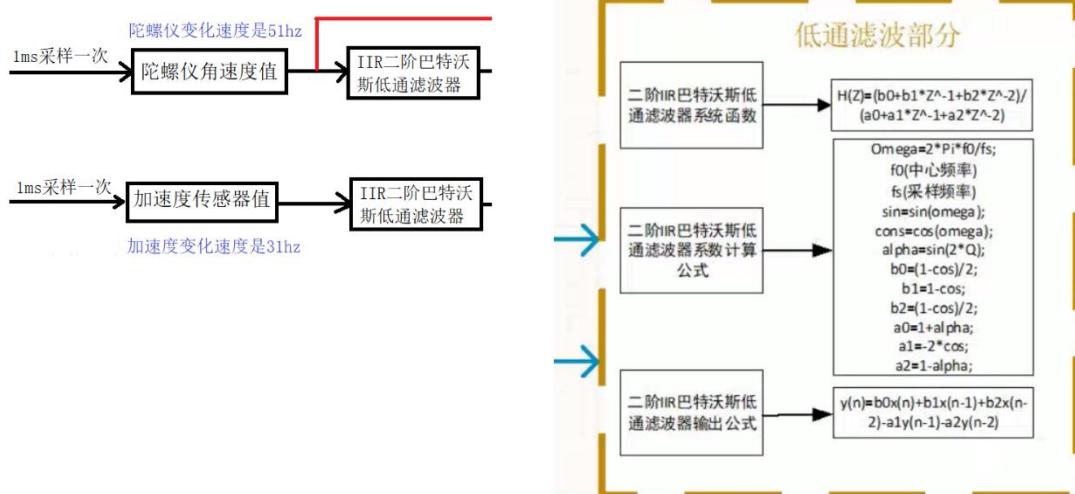
float temp_throttle = 0.0f; //油门值必须是全局变量
float throttle; //拟合后的油门
unsigned char thr_lowest_flag = 0; //默认状态为假(飞行状态全局标志)

if(RFdata[10] != last_rfdata_10) //怠速/停止模式按键切换 死循环内容
{
    if(Fly_State == FlyStop)
        Fly_State = FlyIdle;
    else
        Fly_State = FlyStop;
    last_rfdata_10 = RFdata[10];
}

//遥控器开机问题预处理，停机模式FlyStop，怠速模式FlyIdle，油门值throttle
if(Fly_State == FlyIdle && throttle < 30 && thr_lowest_flag == 0)//怠速，油门<30, false
    thr_lowest_flag = 1; //标志位设置为真
if(Fly_State == FlyStop)
    thr_lowest_flag = 0; //如果飞机处于停机阶段，标志为false
if(Fly_State == FlyIdle && thr_lowest_flag == 1 && throttle >= 30) //满足3个条件
    Fly_State = FlyStart; //进入起飞模式

```

陀螺仪滤波，二阶滤波器实现



```

typedef struct
{
    float b0,b1,b2,a1,a2;
    float d1,d2;
}biguadFilter_T;

```

```

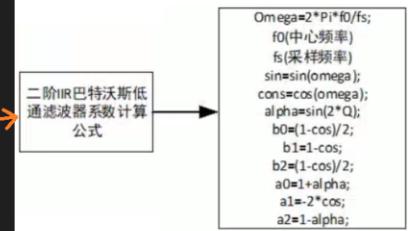
/*
 * 二阶IIR低通滤波器,求滤波系数函数
 * 二阶低通滤波器表达式: H(Z) = (b0+b1*z^-1+b2*z^-2)/(a0+a1*z^-1+a2*z^-2)
 * 参数: pfilter滤波器, filterFreq截止频率, refreshRate刷新频率(采样频率), Q品质因数
 */
void biguadFilterInit(biguadFilter_T *pfilter,
                      unsigned int filterFreq,
                      unsigned int refreshRate,
                      float Q)
{
    //我们之前定义的采样频率是1ms, 转换成频率hz
    const float SampleRate = 1/((float)refreshRate * 0.000001f);
    const float omega = 2 * 3.14 * filterFreq / SampleRate;
    const float sn = sinf(omega); //sin(omega) 正弦
    const float cs = cosf(omega); //cos(omega)余弦
    // const float alpha = sinf(2 * Q); //未优化参数阿尔法
    const float alpha = sn/(2 * Q); //二阶低通滤波参数进行优化
    float b0 = 0, b1 = 0, b2 = 0, a0 = 0, a1 = 0, a2 = 0;
    b0 = (1-cs) / 2;
    b1 = 1 - cs;
    b2 = (1 - cs) / 2;
    a0 = 1 + alpha;
    a1 = -2 * cs;
    a2 = 1 - alpha;
    pfilter->b0 = b0 / a0;
    pfilter->b1 = b1 / a0;
    pfilter->b2 = b2 / a0;
    pfilter->a1 = a1 / a0;
    pfilter->a2 = a2 / a0;
    pfilter->d1 = 0;
    pfilter->d2 = 0;
}

/*
 * 滤波器数据更新
 */
void filterValuation(biguadFilter_T *pfilterInput , biguadFilter_T *pfilterOutput)
{
    pfilterOutput->a1 = pfilterInput->a1;
    pfilterOutput->a2 = pfilterInput->a2;
    pfilterOutput->b0 = pfilterInput->b0;
    pfilterOutput->b1 = pfilterInput->b1;
    pfilterOutput->b2 = pfilterInput->b2;
    pfilterOutput->d1 = pfilterInput->d1;
    pfilterOutput->d2 = pfilterInput->d2;
}

//定义陀螺仪滤波参数
biguadFilter_T gryo_51hz_parameterX;
biguadFilter_T gryo_51hz_parameterY;
biguadFilter_T gryo_51hz_parameterZ;

//定义加速度计滤波参数
biguadFilter_T acc_30hz_parameterX;
biguadFilter_T acc_30hz_parameterY;
biguadFilter_T acc_30hz_parameterZ;

```



```

/*
 * 巴特沃斯低通滤波器系数求解初始化
 */
void butterworth_flier_parameter(void)
{
    //陀螺仪低通滤波器系数计算
    //截止频率51hz 采样率1000hz, 品质因数根号1/2
    biguadFilterInit(&gryo_51hz_parameterX, 51, 1000, 1.0f/sqrif(2.0f));
    filterValuation(&gryo_51hz_parameterX, &gryo_51hz_parameterY); //X轴的系数赋值给Y轴
    filterValuation(&gryo_51hz_parameterX, &gryo_51hz_parameterZ); //X轴的系数赋值给Z轴

    //加速度计低通滤波器系数计算
    //截止频率30hz 采样率1000hz, 品质因数根号1/2
    biguadFilterInit(&acc_30hz_parameterX, 30, 1000, 1.0f/sqrif(2.0f)); // 根号1/2
    filterValuation(&acc_30hz_parameterX, &acc_30hz_parameterY); //X轴的系数赋值给Y轴
    filterValuation(&acc_30hz_parameterX, &acc_30hz_parameterZ); //X轴的系数赋值给Z轴
}

/*
 * 低通滤波器运算
 * 滤波器输出公式: Y(n)=b0x(n) + b1x(n-1) + b2x(n-2) - a1y(n-1) - a2y(n-2)
 * n当前算出的值, n-1上次算出的值, n-2上上次算出的值。
 * 下面函数实现, 用d1表示上次算出来的值, d2表是上上次算出来的值。
 * 如果没有进行第一次运算, 可以先让d1,d2初始为0
 * x(n): 就是这次输入的input值
 * 每次执行该函数, 迭代循环的时候n-1放入d1, n-2放入d2
 * 返回值: 滤波之后的值
 */
float biquadFilterApply(biquadFilter_T *pfilter, float input)
{
    const float result = pfilter->b0 * input + pfilter->d1;
    pfilter->d1 = pfilter->b1 * input - pfilter->a1 * result + pfilter->d2;
    pfilter->d2 = pfilter->b2 * input - pfilter->a2 * result;
    return result;
}



二阶IIR巴特沃斯低通  
滤波器输出公式



$$y(n) = b0x(n) + b1x(n-1) + b2x(n-2) - a1y(n-1) - a2y(n-2)$$



//作为陀螺仪滤波, 陀螺仪有三个轴XYZ, 加速度有三个XYZ, 总共6个轴的数据都是需要滤波的。
    //所以需要调用6次低通滤波运算, 每调用一次, 滤波器的值回发生变化。由于系数全部封装在一个结构体中
    //每一次d1,d2都发生变化, 结果造成结构体就只能调用一次。
    //为了解决结构体只能调用一次的矛盾。
    //需要再建立新的结构体, 让其值在每一个周期里面保持一样。那么我们需要实现filterValuation函数


/*
 * 滤波器数据更新
 */
void filterValuation(biquadFilter_T *pfilterInput, biquadFilter_T *pfilterOutput)
{
    pfilterOutput->a1 = pfilterInput->a1;
    pfilterOutput->a2 = pfilterInput->a2;
    pfilterOutput->b0 = pfilterInput->b0;
    pfilterOutput->b1 = pfilterInput->b1;
    pfilterOutput->b2 = pfilterInput->b2;
    pfilterOutput->d1 = pfilterInput->d1;
    pfilterOutput->d2 = pfilterInput->d2;
}

```

前面实现过该函数, 只是没有说明使用含义, 这里说明一下

下面使用二阶低通滤波器处理陀螺仪和加速度计的数据

```
typedef struct
{
    float X;
    float Y;
    float Z;
}float_XYZ;

//定义临时变量
float_XYZ Temp_Gry_F;
float_XYZ Temp_ACC_F;

//定义输出变量
float_XYZ Gry_F;           //不反馈的陀螺仪值不需要滤波
float_XYZ Gry_FeedBack_F; //反馈的陀螺仪值是需要滤波的
float_XYZ ACC_F;

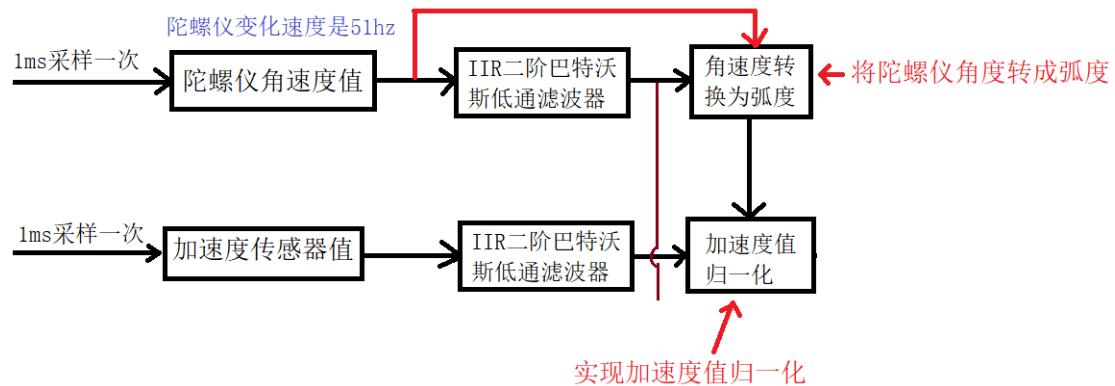
/******************
 * Mpu6500陀螺仪数据读取
 *****/
void Cal_Mpu_Data(void)
{
    Temp_Gry_F.X = 读取6500,X轴数据
    Temp_Gry_F.Y = 读取6500,Y轴数据
    Temp_Gry_F.Z = 读取6500,Z轴数据

    //陀螺仪，角速度无需滤波
    Gry_F.X = Temp_Gry_F.X; //这是无需滤波的
    Gry_F.Y = Temp_Gry_F.Y;
    Gry_F.Z = Temp_Gry_F.Z;

    //陀螺仪，角速度进行低通滤波
    //需要滤波的如下
    Gry_FeedBack_F.X = biquadFilterApply(&gryo_51hz_parameterX,Temp_Gry_F.X);
    Gry_FeedBack_F.Y = biquadFilterApply(&gryo_51hz_parameterY,Temp_Gry_F.Y);
    Gry_FeedBack_F.Z = biquadFilterApply(&gryo_51hz_parameterZ,Temp_Gry_F.Z);

    //加速度值获取
    Temp_ACC_F.X = 读取加速度传感器,X轴数据 (有些要乘以 ACC_gain转成1g值,看自己需要)
    Temp_ACC_F.Y = 读取加速度传感器,Y轴数据
    Temp_ACC_F.Z = 读取加速度传感器,Z轴数据

    //加速度进行低通滤波
    ACC_F.X = biquadFilterApply(&acc_30hz_parameterX,Temp_ACC_F.X);
    ACC_F.Y = biquadFilterApply(&acc_30hz_parameterY,Temp_ACC_F.Y);
    ACC_F.Z = biquadFilterApply(&acc_30hz_parameterZ,Temp_ACC_F.Z);
```



```

/*
 * 更新四元数，获取欧拉角
 * 参数：传入陀螺仪值*gryg，传入加速度计值*accel，更新周期
 */
void UpdateQ_GetEulerAngle(float_XYZ *gryg, float_XYZ *accel, float dt)
{
    float ax, ay, az; // 加速度
    float gx, gy, gz; // 陀螺仪
    float recipNorm;

    // 获取加速度值
    ax = accel->X * 100; // 需要 米^2/秒 转化成 厘米^2/秒，用乘100来做。用来做定高很准。
    ay = accel->Y * 100;
    az = accel->Z * 100;

    // 获取陀螺仪值
    gx = gryg->X;
    gy = gryg->Y;
    gz = gryg->Z;

    // 将陀螺仪值转换成弧度(1角速度 = 0.01745f)
    gx = gx * 0.01745f;
    gy = gy * 0.01745f;
    gz = gz * 0.01745f;

    // 将加速度得值归一化
    recipNorm = invSqrt(ax*ax + ay*ay + az*az);
    ax = ax * recipNorm;
    ay = ay * recipNorm;
    az = az * recipNorm;

    // 提取四元数中的重力分量
    // 还未写完
}

// 快速求解平方根得倒数(如果觉得单片机ST官方求解平方根浪费CPU导致效率低，就用这个)
// 参数：求解的数
// 返回值：求解结果
static float invSqrt(float x)
{
    float halfx = 0.5f * x;
    float y = x;
    long i = *(long *)&y;
    i = 0x5f3759df - (i >> 1);
    y = *(float *)&i;
    y = y * (1.5f - (halfx * y*y));
    return y;
}

```

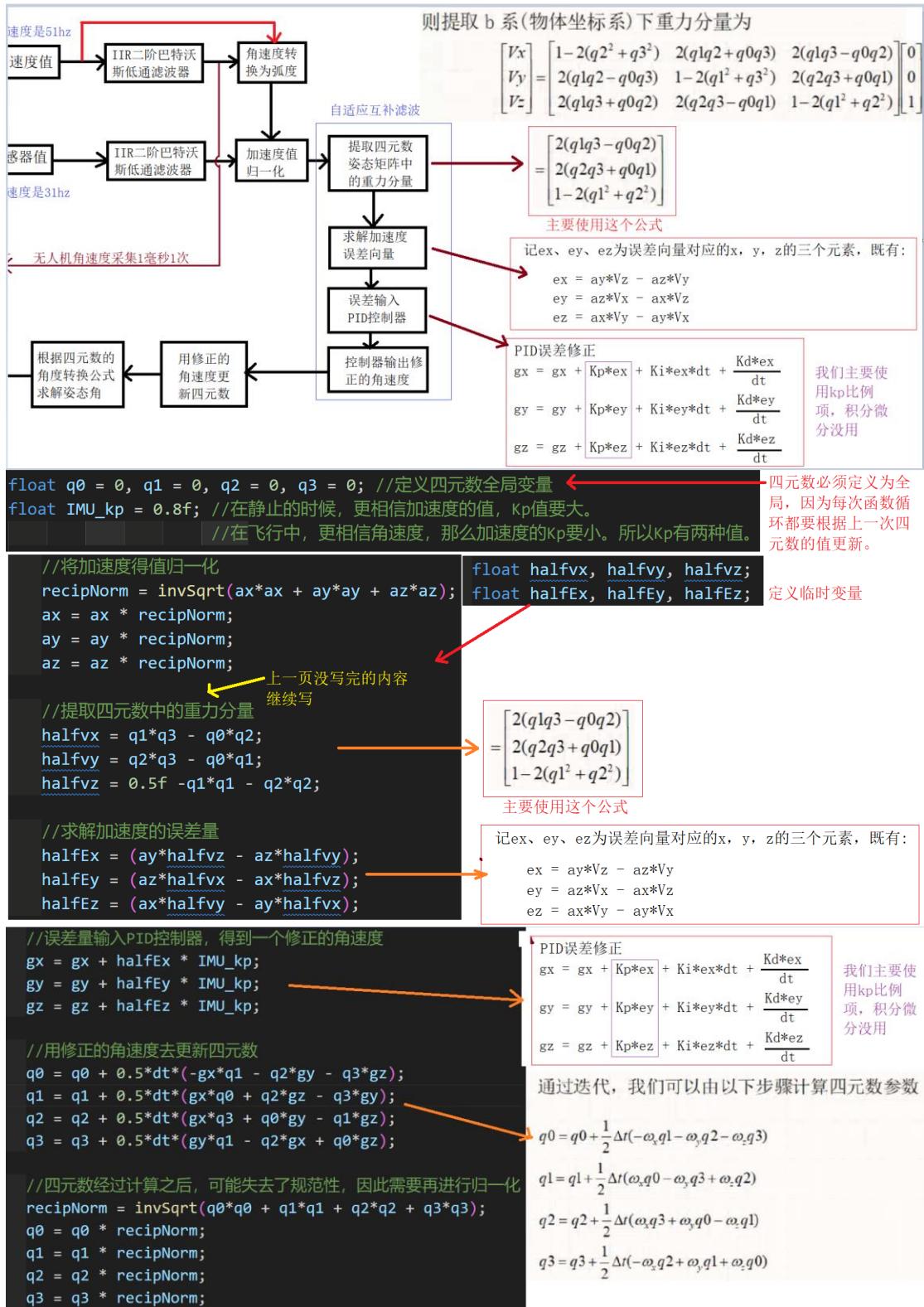
实现加速度值归一化

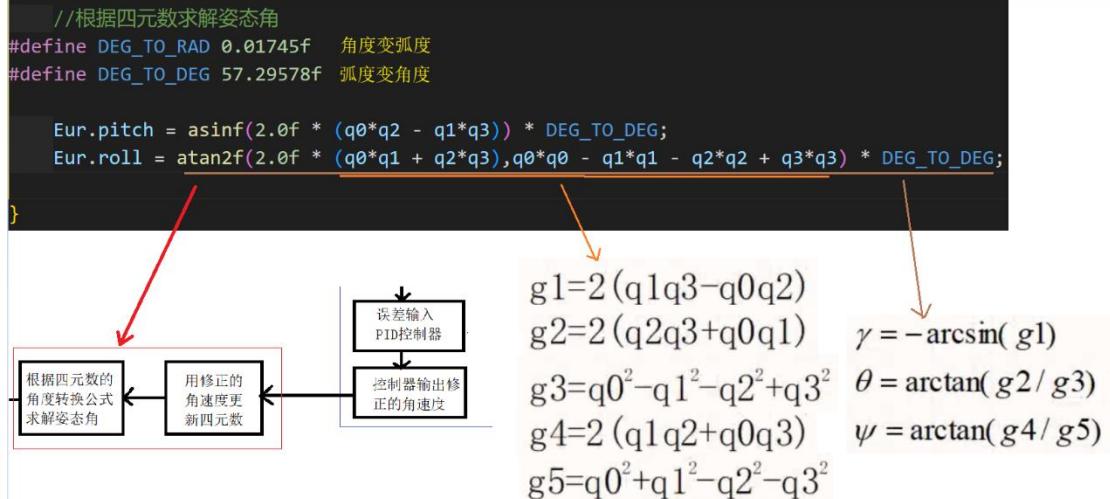
四元数归一化方法

$$ax = \frac{ax}{\sqrt{ax^2 + ay^2 + az^2}}$$

$$ay = \frac{ay}{\sqrt{ax^2 + ay^2 + az^2}}$$

$$az = \frac{az}{\sqrt{ax^2 + ay^2 + az^2}}$$





$\omega_x, \omega_y, \omega_z$ 为陀螺仪测得的角速度, 由(47)可知, 当我们知道前一时刻四元数的值, 便可以通过迭代法去更新下一时刻四元数的值, 以此一步更新各个时刻的四元数值, 也就是说, 在初始时刻给定一个四元数的初值(假设零时刻给定 $q_0=1, q_1=0, q_2=0, q_3=0$), 通过陀螺仪不断测得的角速度去更新四元数, 通过四元数就可求得姿态变换矩阵, 求出姿态角, 也通过四元数不断的更新, 这样也就实现了物体姿态的实时更新。

所以要修改四元数全局变量初始值 $q_0=1$

```

float q0 = 1, q1 = 0, q2 = 0, q3 = 0; //定义四元数全局变量
float IMU_kp = 0.8f; //在静止的时候, 更相信加速度的值, Kp值要大。
                                //在飞行中, 更相信角速度, 那么加速度的Kp要小。所以Kp有两种值。

```

无人机控制系统代码设计

在以上设计好滤波, 遥控器功能, 四元数算法的基础上, 在主函数实现无人机整体控制。

```

//系统初始化硬件, 初始化四元数滤波系数等.....
while (1)
{
    //5毫秒获取一次遥控器数据
    获取RFdata[9], RFdata[10]
    //1毫秒获取一次陀螺仪, 加速度的值
    Cal_Mpu_Data(); //读取陀螺仪, 加速度计经过滤波转化的数据
    //5毫秒更新四元数, 获取欧拉角
    if(5毫秒时间到)
    {

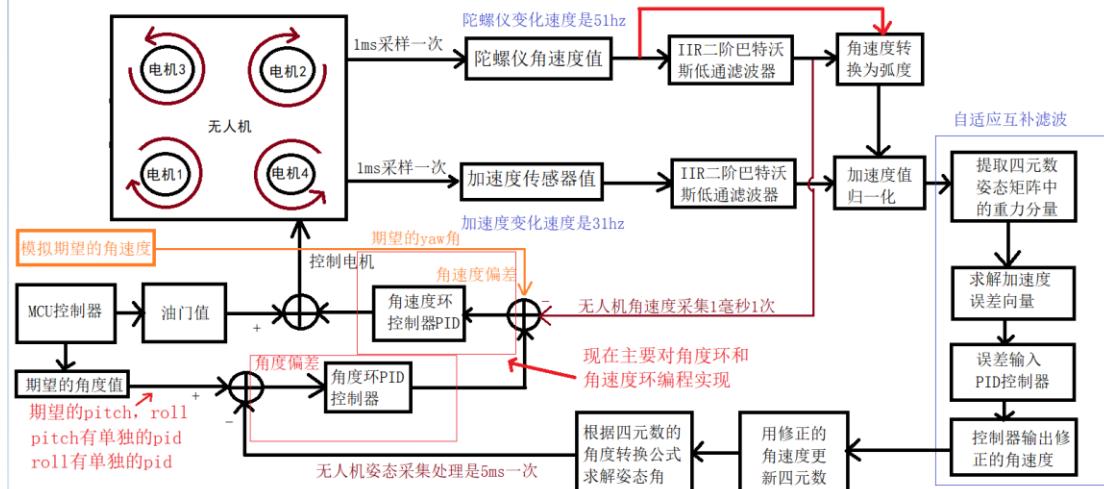
```

```

        //如果飞机不在启动模式下，同时处于静止状态
        if(Fly_State != Fly_State && GRYState == true)
        {
            IMU_kp = 10.0f; //PID的比例系数设置大
        }
        else //如果飞机在飞行状态下
        {
            IMU_kp = 0.6f; //PID的比例系数设置小
            //陀螺仪无需滤波参数，加速度滤波后参数，5毫秒执行一次该函数，所以天0.005
            UpdateQ_GetEurAngle(&Gry_F,&ACC_F,0.005f);
        }
    }
}

```

角度环，角速度环PID算法实现



```

typedef struct
{
    float kp,ki,kd; //系数
    float Diff,Integral; //微分, 积分
    float Integral_Max; //积分最大值
    float Err,PreErr; //当前误差, 上一次误差
    float Pout,Iout,Dout; //Kp输出, Ki输出, Kd输出
    float dt; //时间
}PID_T; //PID参数集

```

```

PID_T Pitch,Roll; //角度PID
PID_T PitchRate,RollRate,YawRate; //角速度PID

```

```

/*****************
 * PID 角度环, 角速度环参数初始化
 *****************/
void pid_parameter_init(void)
{
    //角度环偏差PID控制器
    Pitch.kp = 0.0f;
    Pitch.ki = 0.0f;
    Pitch.kd = 0.0f;
    Pitch.dt = 0.005f; //角度环是5毫秒

    Roll.kp = 0.0f;
    Roll.ki = 0.0f;
    Roll.kd = 0.0f;
    Roll.dt = 0.005f; //角度环是5毫秒
}

```

```

//角速度环初始化
PitchRate.kp = 0.0f;
PitchRate.ki = 0.0f;
PitchRate.kd = 0.0f;
PitchRate.dt = 0.001f; //角速度环是1毫秒
PitchRate.Integral_Max = 200; //角速度环要设置积分最大值，根据飞机特性调试出来的

RollRate.kp = 0.0f;
RollRate.ki = 0.0f;
RollRate.kd = 0.0f;
RollRate.dt = 0.001f; //角速度环是1毫秒
RollRate.Integral_Max = 200; //角速度环要设置积分最大值，根据飞机特性调试出来的

YawRate.kp = 0.0f;
YawRate.ki = 0.0f;
YawRate.kd = 0.0f;
YawRate.dt = 0.001f; //角速度环是1毫秒
YawRate.Integral_Max = 200; //角速度环要设置积分最大值，根据飞机特性调试出来的
}

/****************************************
* PID角度环控制
* *参数：期望值expect，反馈值feedback，PID系数(pitch, roll)
* 返回值：PID控制后的总输出
****************************************/
float pid_angle_control(PID_T *pid, float expect, float feedback)
{
    float output = 0;//总输出
    //算偏差值,微分项系数计算
    pid->Err = expect - feedback; //当前误差
    //获得微分
    pid->Diff = (pid->Err - pid->PreErr)/pid->dt; //((当前误差 - 上一次误差) / 时间dt
    pid->PreErr = pid->Err; //当前误差给上次误差，方便迭代
    //比例项输出
    pid->Pout = pid->kp * pid->Err;
    //微分项输出
    pid->Dout = pid->kd * pid->Diff;
    //控制器总输出
    output = pid->Pout + pid->Dout;
    return output;
}

/****************************************
* PID角速度环控制
* *参数：期望值expect，反馈值feedback，PID角速度环系数(pitch, roll)
* 返回值：PID控制后的总输出
****************************************/
float pid_gyro_control(PID_T *pid, float expect, float feedback)
{
    float output = 0;//总输出
    //算偏差值,微分项系数计算
    pid->Err = expect - feedback; //当前误差
    //获得微分
    pid->Diff = (pid->Err - pid->PreErr)/pid->dt; //((当前误差 - 上一次误差) / 时间dt
    pid->PreErr = pid->Err; //当前误差给上次误差，方便迭代
    //比例项输出
    pid->Pout = pid->kp * pid->Err;
    //积分项输出，需要有一个保护，有一个最小值，有一个最大值，超过最大值不积分。
}

```

角速度偏差

角速度环
控制器PID

角度偏差

角度环PID
控制器

PID角度环

$$gx = gx + K_p \cdot ex + K_i \cdot ex \cdot dt + \frac{K_d \cdot ex}{dt}$$

$$gy = gy + K_p \cdot ey + K_i \cdot ey \cdot dt + \frac{K_d \cdot ey}{dt}$$

$$gz = gz + K_p \cdot ez + K_i \cdot ez \cdot dt + \frac{K_d \cdot ez}{dt}$$

PID角度环
只需要比例和微分

PID角速度环

$$gx = gx + K_p \cdot ex + K_i \cdot ex \cdot dt + \frac{K_d \cdot ex}{dt}$$

$$gy = gy + K_p \cdot ey + K_i \cdot ey \cdot dt + \frac{K_d \cdot ey}{dt}$$

$$gz = gz + K_p \cdot ez + K_i \cdot ez \cdot dt + \frac{K_d \cdot ez}{dt}$$

PID角速度环，加入了
比例积分微分

```

//积分项输出，需要有一个保护，有一个最小值，有一个最大值，超过最大值不积分。
if(fabs((pid->Err)) > 0.5f && (pid->Err) < 800) //保护判断
{
    //条件满足，开始积分
    pid->Integral = pid->Integral + (pid->Err * pid->dt);
    if (pid->Integral > pid->Integral_Max) //大于最大值
        pid->Integral = pid->Integral_Max;
    if (pid->Integral < -pid->Integral_Max) //小于最小值
        pid->Integral = -pid->Integral_Max;
}
pid->Iout = pid->ki * pid->Integral;
//微分项输出
pid->Dout = pid->kd * pid->Diff;
//控制器总输出
output = pid->Pout + pid->Iout + pid->Dout;
return output;
}

```

```

float pitch_out = 0; //姿态外环(角度)控制后的俯仰角输出，全局变量
float roll_out = 0; //姿态外环(角度)控制后的翻滚角输出，全局变量
*****

```

```

* 姿态外环(角度)控制
*****
void attitude_Outer_loop_control(void)
{
    //pitch 俯仰角PID控制
    //传入pitch的PID参数，遥控器输出pitch期望值，姿态角解算出的反馈值Eur.pitch
    pitch_out = pid_angle_control(&Pitch, Expect.pitch, Eur.pitch);

    //roll 翻滚角PID控制
    //传入roll的PID参数，遥控器输出roll期望值，姿态角解算出的反馈值Eur.roll
    roll_out = pid_angle_control(&Roll, Expect.roll, Eur.roll);
}

```

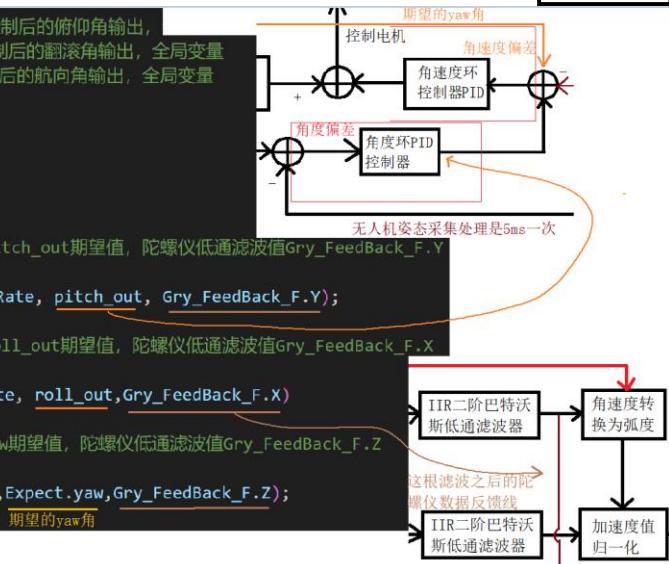
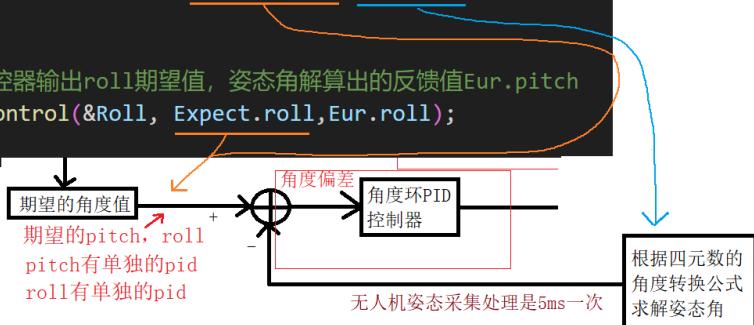
```

float pitchRate_out = 0; //姿态内环(角速度)控制后的俯仰角输出,
float rollRate_out = 0; //姿态内环(角速度)控制后的翻滚角输出，全局变量
float YawRate_out = 0; //姿态内环(角速度)控制后的航向角输出，全局变量
*****
* 姿态内环(角速度)控制
*****
void attitude_Inner_loop_control(void)
{
    //pitchRate 俯仰角速度PID控制
    //传入PitchRate的角速度PID参数，角度输出pitch_out期望值，陀螺仪低通滤波值Gry_FeedBack_F.Y
    //俯仰角是按照Y方向来的
    pitchRate_out = pid_gyro_control(&PitchRate, pitch_out, Gry_FeedBack_F.Y);

    //rollRate 翻滚角速度PID控制
    //传入PitchRate的角速度PID参数，角度输出roll_out期望值，陀螺仪低通滤波值Gry_FeedBack_F.X
    //翻滚角是按照X方向来的
    rollRate_out = pid_gyro_control(&RollRate, roll_out, Gry_FeedBack_F.X);

    //YawRate 航向角速度PID控制
    //传入PitchRate的角速度PID参数，Expect yaw期望值，陀螺仪低通滤波值Gry_FeedBack_F.Z
    //航向角是按照Z方向来的
    YawRate_out = pid_gyro_control(&YawRate, Expect.yaw, Gry_FeedBack_F.Z);
}

```

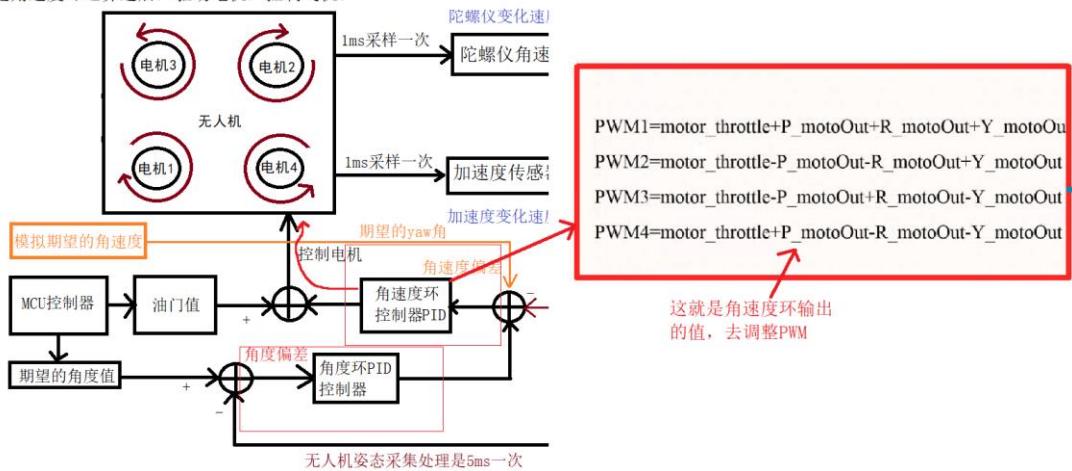


```

if(5毫秒时间到)
{
    atitude_Outer_loop_control(); // (外环)角度PID 5毫秒执行一次
}
if(1毫秒时间到)
{
    atitude_Inner_loop_control(); // (内环)角速度PID 1毫秒执行一次
}

```

经过角速度环运算之后，驱动电机，控制飞机。



```

volatile unsigned short PWM1 = 0, PWM2 = 0, PWM3 = 0, PWM4 = 0;
float P_motoOut = 0;
float R_motoOut = 0;
float Y_motoOut = 0;

/*****************
 * 四电机PWM控制
 *****************/
void motor_control_output(void)
{
    P_motoOut = pitchRate_out; // 将姿态内环的角速度赋值给PWM的参数
    R_motoOut = rollRate_out; // 将姿态内环的角速度赋值给PWM的参数
    Y_motoOut = YawRate_out; // 将姿态内环的角速度赋值给PWM的参数
    // 初始化PWM值，自己定义，然后进行补偿
    motor_throttle = throttle * (4.2 / bat_voltage); // 为了不让电机转速不变，需要进行电压补偿
    // 计算PWM的输出
    PWM1 = motor_throttle + P_motoOut + R_motoOut + Y_motoOut;
    PWM2 = motor_throttle - P_motoOut - R_motoOut + Y_motoOut;
    PWM3 = motor_throttle - P_motoOut + R_motoOut - Y_motoOut;
    PWM4 = motor_throttle + P_motoOut - R_motoOut - Y_motoOut;

    if(待机模式)
    {
        // 待机模式下输出的PWM值
        motor_pwm_set(...);
    }
    else if (起飞模式)
    {
        // 起飞模式输出PWM值
        motor_pwm_set(PWM3, PWM1, PWM4, PWM2);
    }
    else // 停机模式
    {
        motor_pwm_set(0, 0, 0, 0);
    }
}

```

```

if(5毫秒时间到)
{
    atititude_Outer_loop_control(); // (外环)角度PID 5毫秒执行一次

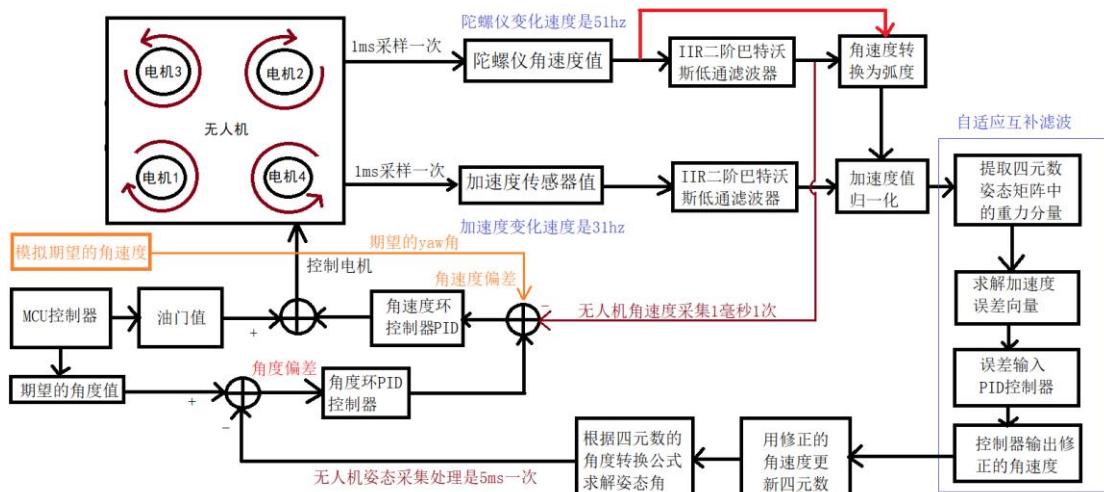
}

if(1毫秒时间到)
{
    atititude_Inner_loop_control(); // (内环)角速度PID 1毫秒执行一次
    主函数加入 motor_control_output(); // 驱动4个螺旋桨电机也是1毫秒执行一次
}

```

到此无人机的飞控算法就算完成了，具体细节还需要实际调试和修改代码中的不足。

最后再次贴上系统框图



剩下的主要是实现无人机的信号掉线保护，无人机个别电机损坏如何安全降落。这些由自己自行摸索，本文档就不在赘述。