学习四轴飞行器编程所需的数学知识

1. 写在前面
   1. 一直以来，为自己的四轴飞行器编程是我一直追求的，但是由于网上关于这方面的很多，却没有一个良好的整合文章。
   2. 四轴飞行器的主要程序流程

输入：输入数据，包括传感器数据和控制参数。

卡尔曼滤波：传感器数据通常有较大的误差，直接进行计算会导致整个系统的稳定性较差，卡尔曼滤波可以滤去输入数据中的大部分误差且计算速度较快。

姿态结算：传感器数据通常无法直观地体现四轴飞行器当前所处的状态，通常需要进行运算才可以得到飞行器的姿态数据。

PID控制：得出姿态数据后，可以算出与期望值（控制系数）的差值，由于该差值与理论输出值的关系是非线性的，所以需要PID计算相应的输出值。

输出:通常是四轴飞行器的四个电机的功率。

1. PID

1.1 PID由三部分构成，他们分别是：

P：Proportion比例

I：Integral积分

D：Derivative微分

1.2 PID控制器的模式图为：

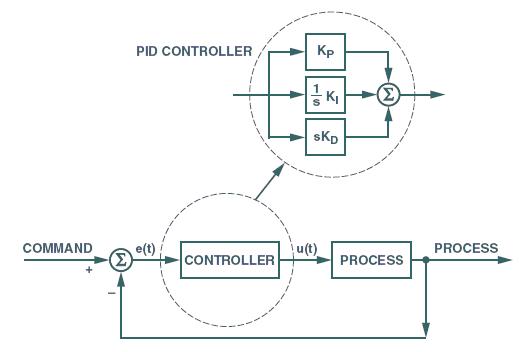


图1.1

1.3 控制参数U(t)为

式中、、分别为PID算法的比例、积分、微分系数，为控制参数的偏置系数，为实际值与理想值之差（即误差，所以PID算法为误差控制算法）。

上面这个公式大家可能看不太懂，没事，我们慢慢来。

第一部分：比例控制

上面式子，就是PID的P部分，这一部分就是把误差乘以一个值，这个值是你事先给定的，为什么要这样呢？因为我们的这个轴飞行器是可以近似地看成一个线性系统（就是理想的控制值与误差成比例），于是把误差乘以一个值之后，就可以作为控制系数来控制我们的电机的转速了。

第二部分：微分控制

这是PID的I部分，是误差的微分（由式子不难看出，就是改变的误差除以改变的时间，可以理解成误差改变的速度），我们为什么要用到微分这个玩意呢？因为我们前面讲过，我们是近似地把这个系统看成线性的，然而这个系统不是线性的（可能离线性差得很远），于是我们要根据误差改变的速度来调整控制系数以弥补比例控制的不足。

第三部分积分控制

这是PID的D部分，是误差的积分（可以看成之前所有的误差的和），前面讲了微分控制，但是我们的微分控制还不是尽善尽美，有时候还是会导致控制系数不是最优的，而且系统有滞后性（你改变控制值后要过一会误差才会有相应的改变），我们要用之前所有的误差的和来调整。

在程序中，的积分一般使用累加器进行计算，可定义一变量e\_INT，在每一次计算时该变量加上该时刻E的值即可得出积分（由于该值可能过大，可以定义为浮点量；但在低性能单片机中，应定义为整形量以减小计算）。同理，可定义一变量e\_DER,用于计算的微分，可以用逐差法计算。

1.4 在四轴飞行器中，有三变量需要用PID控制，他们是飞行器姿态欧拉角的三个角度。

1. 姿态解算

3.1 四元数

四元数是简单的超复数。 复数是由实数加上虚数单位 i 组成，其中。 相似地，四元数都是由实数加上三个虚数单位 i、j、k 组成，而且它们有如下的关系： ， , 每个四元数都是 1、i、j 和 k 的线性组合，即是四元数一般可表示为

,其中a、b、c 、d是实数。[1]

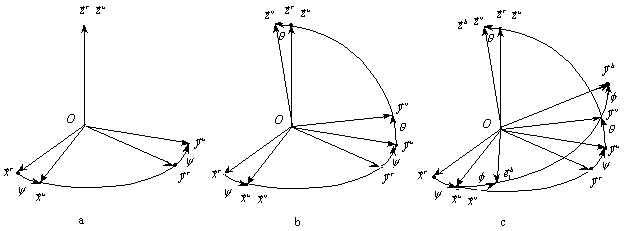
两四元数相乘可以表示为旋转，结果为

http://f.hiphotos.baidu.com/baike/s%3D216/sign=812d3bc0f31f3a295ec8d2cfaf24bce3/7e3e6709c93d70cf36be3013fbdcd100baa12b63.jpg

http://e.hiphotos.baidu.com/baike/s%3D572/sign=84c33f35357adab439d01b44b9d5b36b/54fbb2fb43166d2233909989452309f79152d2cc.jpg

3.2欧拉角

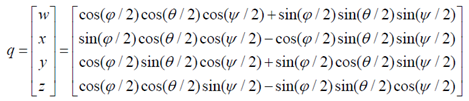
用来确定定点转动刚体位置的3个一组独立角参量，由章动角θ、旋进角（即进动角）ψ和自转角j组成，为欧拉首先提出而得名。[2]

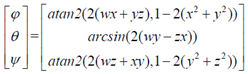
[](http://photo.blog.sina.com.cn/showpic.html#blogid=557d254601018ddv&url=http://s11.sinaimg.cn/orignal/557d2546gd4d10f498c2a)

我们可以把欧拉角理解成一个物体在三维空间分别沿三个维度旋转的角度，上图很好地描述了这个旋转。

欧拉角有一个问题就是万向节死锁，先Yaw 45度，再Pitch 90度(45,90,0)，与先Pitch 90度，再Roll 45度(0,90,45)的结果完全相同。也就是说同一个欧拉角有不同的表示方法，所以欧拉角不能直接旋转，而需要四元数来进行旋转的计算。

3.3四元数于欧拉角的转换：





3.4过程：

先得出线速度gx,gy,gz.除以测量间隔t可以得到当前的旋转欧拉角，将这个欧拉角转换成四元数q1，将q1乘以原姿态欧拉角的相应四元数q0，得到q2，对这个四元数取模：

http://d.hiphotos.baidu.com/baike/s%3D273/sign=095a9701e9c4b7453094b011fcfc1e78/f3d3572c11dfa9ec891c473f60d0f703918fc1e3.jpg

然后使

以减少运算，就可解算出当前欧拉角，再使q0=q3以进行下一次运算。

得到的数据需要与加速度算出的姿态进行融合。（可以直接加权取得较近似值）（建议使用融合后的数据对q0进行矫正）

3.5编程事项：

四元数可以用数组表示，内部四个元素，也可以定义一个type，里面分别有变量a,b,c,d。

3.6写在后面

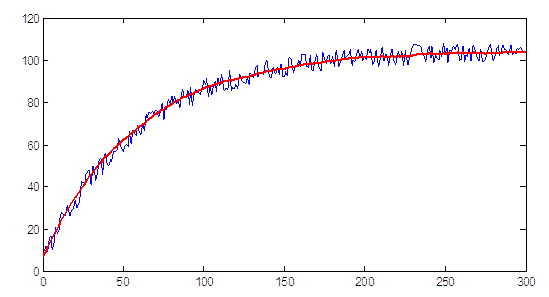
MPU6050的确很方便，但是误差不容小觑，建议使用相对来说精度更高的传感器来做，有时MPU6050偏差比较大。若要获得更高精度，可以采用双传感器进行数据融合。

1. 卡尔曼滤波

式中为系统测量值，为测量误差，可以取一个比传感器精度较大的值。

大家可能不明白括号里的前后两个值得含义，前一个值表示的是当前的系统的时间，后一个值表示的是当前的系统的值的时间，因为不同时间有不同的系统，所以就这样表示。比如表示的是当前时间的系统的当前时间的值，而就是当前时间的系统的前一时间的值，可能听起来有点绕，但数学就是这样地美妙。

这个滤波的主要原理是信任度，如果这个值改变的速度太快，我们会增大传感器值的信任度，如果这个值改变的速度太慢（以测量误差为界限），这很可能是测量噪声（测量的时候的误差），于是传感器值的信任度变小，这样我们就可以得到一个比较平缓的信号，在使用过程中可以极大减少四轴飞行器的不稳定性，而就是这个信任度。



上图是卡尔曼滤波的效果图，蓝色是带噪音的输入信号，红色是输出结果。

为了令卡尔曼滤波器开始工作，我们需要告诉卡尔曼两个零时刻的初始值，即X(0|0)和P(0|0)。对于P，不要取0，因为这样可能会令卡尔曼相信所有系数是系统最优的，而X(0|0)可以是传感器的第一次测量值。

卡尔曼滤波虽然方便，但是在其开始运算的前30秒，由于未完全收敛，所得出的结果一般可信度不大。

参考资料：

[1]四元数\_百度百科 , http://baike.baidu.com/view/319754.htm

[2]欧拉角\_百度百科,http://baike.baidu.com/view/319754.htm