四元数完全解析及资料汇总

本文原帖出自匿名四轴论坛，附件里的资源请到匿名论坛下载：

<http://www.anobbs.com/forum.php>

感谢匿名的开源分享，感谢群友的热心帮助。

说什么四元数完全解析其实都是前辈们的解析，小弟真心是一个搬砖的，搬得不好希望大神们给以批评和指正，在此谢过了。因为本人是小菜鸟一枚，对，最菜的那种菜鸟······所以对四元数求解姿态角这么一个在大神眼里简单的算法，小弟我还是费了很大劲才稍微理解了那么一点点，小弟搬砖整理时也是基于小弟的理解和智商的，有些太基础，有些可能错了，大牛们发现了再骂过我后希望能够给与指正哈。

好，废话到此为止，开始说主体。四元数和姿态角怎么说呢？先得给和我一样的小菜鸟们理一理思路，小鸟我在此画了一个“思维导图”（我承认我画的丑），四元数解算姿态首先分为两部分理解：第一部分先理解什么是四元数，四元数与姿态角间的关系；第二部分要理解怎么由惯性单元测出的加速度和角速度求出四元数，再由四元数求出欧拉角。



图1 渣渣思维导图

在讲解什么是四元数时，小弟的思维是顺着说的，先由四元数的定义说起，说到四元数与姿态角间的关系。但在讲解姿态解算时，小弟的思维是逆向的，就是反推回来的，从欧拉角一步步反推回到惯性器件的测量数据，这样逆向说是因为便于理解，因为实际在工程应用时和理论推导有很大差别。

实际应用时正确的求解顺序应该为图1中序号顺序，即1->2->3->…….

但在笔者讲解姿态求解时思路是如图2的。



图2 逆向讲解思路

大家在看四元数时最好结合着代码一块看，小弟看的是匿名四轴的代码，感觉写的非常好也非常清晰，粘出来大家一块观摩。红色部分是核心代码，总共分为八个步骤，和图1中的八个步骤是一一对应的。讲解介绍时也是和代码对比起来讲解的。代码可以去匿名官网上下载，都是开源的，不是小弟的，所以小弟不方便加在附件中。

|  |
| --- |
| //四元数更新姿态  #define Kp 2.0f //加速度权重，越大则向加速度测量值收敛越快  #define Ki 0.001f //误差积分增益  void ANO\_IMU::Quaternion\_CF(Vector3f gyro,Vector3f acc, float deltaT)  {  Vector3f V\_gravity, V\_error, V\_error\_I;  //1.重力加速度归一化  acc.normalize();  //2.提取四元数的等效余弦矩阵中的重力分量  Q.vector\_gravity(V\_gravity);  //3.向量叉积得出姿态误差  V\_error = acc % V\_gravity;  //4.对误差进行积分  V\_error\_I += V\_error \* Ki;  //5.互补滤波，姿态误差补偿到角速度上，修正角速度积分漂移  Gyro += V\_error \* Kp + V\_error\_I;  //6.一阶龙格库塔法更新四元数  Q.Runge\_Kutta\_1st(Gyro, deltaT);  //7.四元数归一化  Q.normalize();  //8.四元数转欧拉角  Q.to\_euler(&angle.x, &angle.y, &angle.z);  } |

好的，下面搬砖开始！。。。。。。。。嘿咻嘿咻！！！！

# 什么是四元数？

关于四元数的定义摘自秦永元的《惯性导航》，里面有非常好的讲解，大家可以直接看绪论和第九章就可以。下面我粘贴了部分原文，粘贴的比较多比较详细，应为本人比较笨还爱较真，所以按本人的风格就要详尽一点，大牛们都可以自动忽略。

|  |
| --- |
| 四元数定义、表达方式及运算方法——摘自《惯性导航》-秦永元P289-292 |

好，关于四元数定义就搬这么多，其他的大家去附件下载《惯性导航》的pdf自己看吧。

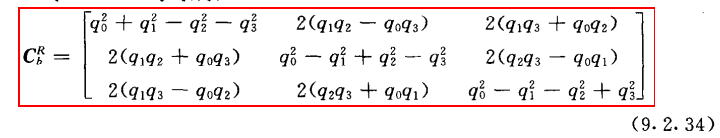
下面开始搬四元数与姿态解算关系的。。。。。。嘿咻嘿咻~~~~

# 二、四元数与姿态阵间的关系

从上面我们知道了四元数的定义，可这四个数和我们要求的三个姿态角有什么关系呢？下面是详细的推导，同样摘自《惯性导航》-秦永元P292-297。

|  |
| --- |
| 四元数与姿态阵间的关系——摘自《惯性导航》-秦永元P292-297 |

呃，粘了这么多其实就是为了想知道推导过程小伙伴好理解，真正有用的就是（9.2.34）这个公式。▲这个公式把四元数转换成了方向余弦矩阵中的几个元素，再用这几个元素转换为欧拉角。就求解除了姿态！



先从四元数q0~q3转成方向余弦矩阵：

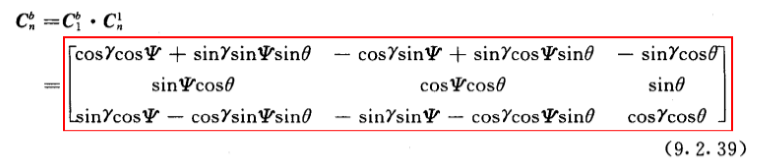
再从方向余弦矩阵转换为欧拉角：

好的，公式原理都讲清楚了，下面来看一下匿名的代码：

|  |
| --- |
| //四元数转欧拉角，这里四元数是q1~q4 和公式里q0~q3相对应。  void Quaternion::to\_euler(float \*roll, float \*pitch, float \*yaw)  {  if (roll) {  \*roll = degrees(atan2f(2.0f\*(q1\*q2 + q3\*q4),1 - 2.0f\*(q2\*q2 + q3\*q3)));  //\*roll = degrees(atan2f(-2.0f\*(q2\*q4 - q1\*q3),1 - 2.0f\*(q2\*q2 + q3\*q3)));  }  if (pitch) {  // 使用safe\_asin()来处理pitch接近90/-90时的奇点  \*pitch = degrees(safe\_asin(2.0f\*(q1\*q3 - q2\*q4)));  //\*pitch = degrees(safe\_asin(2.0f\*(q3\*q4 + q1\*q2)));  }  if (yaw) {  \*yaw = degrees(atan2f(2.0f\*(q2\*q3 - q1\*q4), 2.0f\*(q1\*q1 + q2\*q2) - 1));  //\*yaw = degrees(atan2f(2.0f\*(q2\*q3 - q1\*q4), 2.0f\*(q1\*q1 + q3\*q3) - 1));  }  } |

对比余弦矩阵转换为欧拉角的公式很容易理解了吧，注意一下，红色是匿名原版的代码，和公式还是有出入的，自己细心观察一下吧。被注释了的黑色代码是我根据上面的公式写的。但黑色的求解出来的欧拉角反映出来的姿态是不对的，具体表现为俯仰（pitch）和横滚（roll）是相反的，为啥根据公式写的是不对的？群里的小伙伴给与了我热心的解答。

这个错误主要是由于方向余弦矩阵的旋转顺序不一样，也就是公式（9.2.39）不一样，这是由于旋转的先后顺序不同引起的，具体大家参考《惯性导航》绪论来看就能明白，因为这一点小弟还有点混乱，就点到这为止。



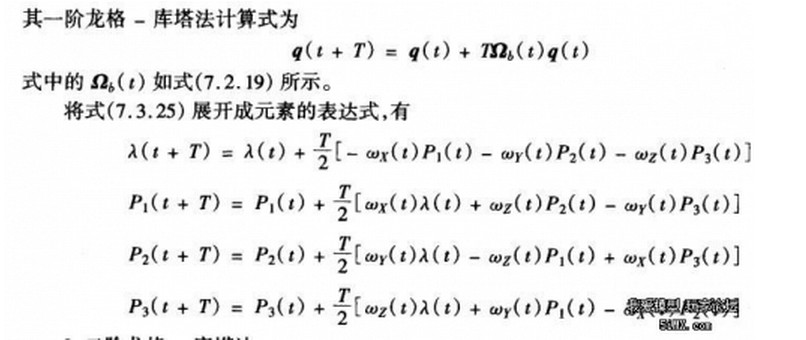
以上就是四元数求解欧拉角的方法。通过公式可以看到，要求欧拉角需要求得四元数的方向余弦矩阵；要求得四元数方向余弦矩阵，需要求得四元数q0~q3，那么如何求得q0~q3？接下来详细介绍。

# 三、四元数微分及龙格库塔求q0~q3

首先我们先来看一下在程序里如何求解的q0~q3：

|  |
| --- |
| //一阶龙格库塔法更新四元数  void Quaternion::Runge\_Kutta\_1st(Vector3f &g, float deltaT)  {  q1 += 0.5 \* (-q2\*g.x - q3\*g.y - q4\*g.z)\* deltaT;  q2 += 0.5 \* (q1\*g.x + q3\*g.z - q4\*g.y)\* deltaT;  q3 += 0.5 \* (q1\*g.y - q2\*g.z + q4\*g.x)\* deltaT;  q4 += 0.5 \* (q1\*g.z + q2\*g.y - q3\*g.x)\* deltaT;  } |

这就是一阶龙格库塔法求解q的微分方程，传入参数只需要这个周期的角速度g.x、g.y、g.z和周期时间deltaT。下面一张是从某位大神的贴吧上盗的图，描绘的是一阶龙格库塔的计算式。



相信很多人和我一样，单看上图很难理解其中的意思和其由来，于是我又找了很多帖子，感谢前人做出的贡献，小弟在这里再次整理大神的四元数微分方程推导公式，便于大家理解。摘自附件中《推導\_四元數.pdf》

虽然在下也不是很懂，不过粘出来还是能起到理解的作用，这样大家就不会觉得这是凭空变出来的，本人数学功底薄弱，没有对推导进行过验证，如果有不对的地方欢迎指正。

接着使用一阶龙格库塔（Runge-Kutta）发求出q0~q3，这一点很多人不知道一阶龙格库塔怎么推导的，下面也是这位网友的推导，大家参考着理解吧。

|  |
| --- |
|  |
|  |

这里的角速度是由捷联陀螺的输出（对机械转子陀螺必须经过误差补偿，将在下面介绍）。

对比着匿名四轴的代码看一看（g.x、g.y、g.z是捷联陀螺的输出），代码的意思就比较清楚了。在往上一步步推，我们就要求陀螺输出了，并且还要对数据进行互补滤波处理。

# 四、惯性单元测量值融合

这部分看似很简单，但是也有让笔者难以理解的地方，希望后人能补充修正进行更好的讲解。有了上一步的龙格库塔方程，我们现在需要的就是角速度的测量值。

在四轴上安装陀螺仪，可以测量四轴倾斜的角速度，由于陀螺仪输出的是四轴的角速度，不会受到四轴振动影响。因此该信号中噪声很小。四轴的角度又是通过对角速度积分而得，这可进一步平滑信号，从而使得角度信号更加稳定。因此四轴控制所需要的角度和角速度可以使用陀螺仪所得到的信号。由于从陀螺仪的角速度获得角度信息，需要经过积分运算。如果角速度信号存在微小的偏差，经过积分运算之后，变化形成积累误差。这个误差会随着时间延长逐步增加，最终导致电路饱和，无法形成正确的角度信号。

如何消除这个累积误差呢？可以通过上面的加速度传感器获得的角度信息对此进行校正。利用加速度计所获得的角度信息 θg 与陀螺仪积分后的角度θ 进行比较，将比较的误差信号经过比例Tg 放大之后与陀螺仪输出的角速度信号叠加之后再进行积分。对于加速度计给定的角度θg ，经过比例、积分环节之后产生的角度θ必然最终等于θg 。由于加速度计获得的角度信息不会存在积累误差，所以最终将输出角度θ中的积累误差消除了。加速度计所产生的角度信息θg 中会叠加很强的有四轴运动加速度噪声信号。为了避免该信号对于角度θ 的影响，因此比例系数 Tg 应该非常小。这样，加速度的噪声信号经过比例、积分后，在输出角度信息中就会非常小了。由于存在积分环节，所以无论比例Tg多么小，最终输出角度θ必然与加速度计测量的角度θg相等，只是这个调节过程会随着Tg 的减小而延长。

先把这个过程的代码粘出来，看着代码一步步理解：

|  |
| --- |
| #define Kp 2.0f //加速度权重，越大则向加速度测量值收敛越快  #define Ki 0.001f //误差积分增益  //1.重力加速度归一化  acc.normalize();  //2.提取四元数的等效余弦矩阵中的重力分量  Q.vector\_gravity(V\_gravity);  //3.向量叉积得出姿态误差  V\_error = acc % V\_gravity;  //4.对误差进行积分  V\_error\_I += V\_error \* Ki;  //5.互补滤波，姿态误差补偿到角速度上，修正角速度积分漂移  Gyro += V\_error \* Kp + V\_error\_I; |

1.重力加速度归一化：加速度计数据归一化，把加速度计的三维向量转换为单位向量， 因为是单位矢量到参考性的投影，所以要把加速度计数据单位化，其实归一化改变的只是这三个向量的长度，也就是只改变了相同的倍数，方向并没有改变，也是为了与单位四元数对应。

2.提取四元数的等效余弦矩阵中的重力分量：

|  |
| --- |
| // 返回该四元数的等效余弦矩阵中的重力分量  void Quaternion::vector\_gravity(Vector3f &v)  {  v.x = 2\*(q2\*q4 - q1\*q3);  v.y = 2\*(q1\*q2 + q3\*q4);  v.z = 1-2\*(q2\*q2 + q3\*q3);  } |

将当前姿态的重力在三个轴上的分量分离出来,把四元数换算成方向余弦中的第三行的三个元素，根据余弦矩阵和欧拉角的定义，就是地理坐标系(参考坐标系)的Z轴的重力向量。当我读完这句话脑子挺懵的，闹不明白啊，于是又找到了下面的资料，可以进行解释了。

|  |
| --- |
|  |

别忘了这是个正交矩阵哦！这样就知道代码怎么来的了吧？好继续。

3.向量叉积得出姿态误差:

哎呀，又来棘手问题了，这个我也不太明白怎么讲啊，还是把大神的讲解粘过来吧，大家看看是不是这么回事：

|  |
| --- |
| acc是机体坐标参照系上，加速度计测出来的重力向量，也就是实际测出来的重力向量。  acc是测量得到的重力向量，V\_gravity是陀螺积分后的姿态来推算出的重力向量，它们都是机体坐标参照系上的重力向量。  那它们之间的误差向量，就是陀螺积分后的姿态和加计测出来的姿态之间的误差。  向量间的误差，可以用向量叉积（也叫向量外积、叉乘）来表示，V\_error就是两个重力向量的叉积。  这个叉积向量仍旧是位于机体坐标系上的，而陀螺积分误差也是在机体坐标系，而且叉积的大小与陀螺积分误差成正比，正好拿来纠正陀螺。（你可以自己拿东西想象一下）由于陀螺是对机体直接积分，所以对陀螺的纠正量会直接体现在对机体坐标系的纠正。 |

看了上面的话，小弟一直对这个误差向量感到莫名其妙，后来又找到大神的一下一段话：

|  |
| --- |
| 这里误差没说清楚，不是指向量差。这个叉积误差是指将带有误差的加计向量转动到与重力向量重合，需要绕什么轴，转多少角度。逆向推理一下，这个叉积在机体三轴上的投影，就是加计和重力之间的角度误差。也就是说，如果陀螺按这个叉积误差的轴，转动叉积误差的角度（也就是转动三轴投影的角度）那就能把加计和重力向量的误差消除掉。（具体可看向量叉积的定义）如果完全按叉积误差转过去，那就是完全信任加计。如果一点也不转，那就是完全信任陀螺。那么把这个叉积的三轴乘以x%，加到陀螺的积分角度上去，就是这个x%互补系数的互补算法了。 |

这个看了好像终于理解点了，再看下代码：

|  |
| --- |
| //3.向量叉积得出姿态误差  V\_error = acc % V\_gravity; |

这里“ % ”已经重定义为叉乘的算法了。

4.对误差进行积分：

积分求误差,关于当前姿态分离出的重力分量,与当前加速度计测得的重力分量的差值进行积分消除误差

|  |
| --- |
| V\_error\_I += V\_error \* Ki; |

5.互补滤波，姿态误差补偿到角速度上，修正角速度积分漂移

系数不停地被陀螺积分更新，也不停地被误差修正，它和公式所代表的姿态也在不断更新。 将积分误差反馈到陀螺仪上，修正陀螺仪的值。将该误差V\_error输入 PI 控制器后与本次姿态更新周期中陀螺仪测得的角速度相加，最终得到一个修正的角速度值,将其输入四元数微分方程，更新四元数。

|  |
| --- |
| Gyro += V\_error \* Kp + V\_error\_I; |

Gyro就是得到的修正角速度值，可以用于求解四元数q0~q3了。

到这里回顾一下八个步骤还漏了一个第七步：

7.四元数归一化：

规范化四元数作用：

1.表征旋转的四元数应该是规范化的四元数，但是由于计算误差等因素，计算过程中四元数会逐渐失去规范化特性，因此必须对四元数做规范化处理。

2.意义在于单位化四元数在空间旋转时是不会拉伸的,仅有旋转角度.这类似与线性代数里面的正交变换。

3.由于误差的引入，使得计算的变换四元数的模不再等于1，变换四元数失去规范性，因此再次更新四元数。

计算欧拉角时候必须要对四元数归一化处理。

|  |
| --- |
| Q.normalize(); |

呃，关于四元数求解姿态的砖好像搬完了。为什么要用四元数法求解姿态呢？再搬一点关于欧拉角法和旋转矢量法的介绍的。

|  |
| --- |
|  |

搬砖搬得好累啊，不过搬得差不多了，感觉挺乱的？呃，主要是由于比较多吧，那我再串一遍？拉倒吧，你看得都累，我写着不累？没闹明白再自己串一遍吧，相信第二遍就能明白了。

哎~对于我这样的渣渣而言也就能理解到这一步了，这也是我好几天的心血整理了一下，也许有和我一样的菜鸟呢，对他们也许能有点帮助，做得不好希望大神们能耐心给与指正，而不是嗤之以鼻，或者喷我一顿就走。。。毕竟整理了两天呢（我还以为一中午就能搞定呢）。渣渣的学习之路也是挺不容易的，因为基础渣渣，学校渣渣，所以难以得到有效地帮助和指导，有时在群里寻求帮助，无聊的群友会告诉你看书去，呵呵。。。我也知道看书啊。。。哪怕你能告诉我我的问题在那本书的那部分能有相似吧？一句看书去，上网查啊，等于没回答，如果一直这样自己看下去可能半年也解决不了，因为渣渣的学习环境是有局限性的。

不过好在有更多很热心的群友能提供给我帮助，把他们收集的好贴发给我，或者干脆手写一个公式推导，一个电路图，然后拍照发给我，还有的帮我下载照片，分类命名给我，艾玛！热泪盈眶啊有木有！！！再次感谢这些热心帮助我的小伙伴@奇点，@杜掌柜，@廉价物品，@忘记名字的小伙伴······

下面附上被我搬砖的几个好贴，谢谢大神们的乐于分享：

对四元数解算姿态的理解(基于匿名六轴)，感谢社区: <http://www.playuav.com/article/79>

软件姿态解算: <http://www.crazepony.com/wiki/software-algorithm.html>

捷联惯导算法心得: <http://www.amobbs.com/forum.php?mod=viewthread&tid=5492189&highlight=>

附件：匿名姿态解算代码

《惯性导航》秦永元

《推導\_四元數.pdf》