

## 题目 1

### 计算思路

题目已给出：

$${}^B R_D(t) = \begin{bmatrix} \cos \omega t & -\sin \omega t \cos \alpha & \sin \omega t \sin \alpha \\ \sin \omega t & \cos \omega t \cos \alpha & -\cos \omega t \sin \alpha \\ 0 & \sin \alpha & \cos \alpha \end{bmatrix}$$

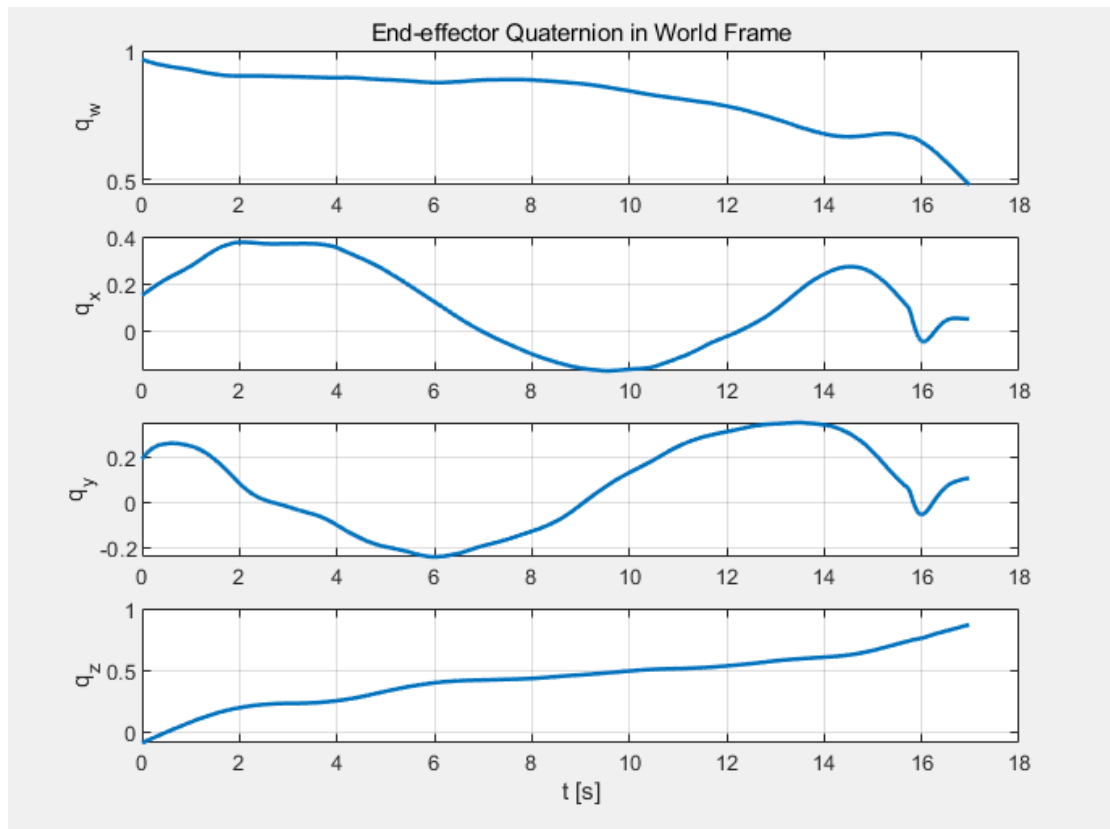
将矩阵转换为四元数

末端执行器在世界系下的姿态由两段旋转相乘得到：

$$q_{WD}(t) = q_{WB}(t) \otimes q_{BD}(t)$$

计算得到的四元数归一化，并强制保证  $q_w > 0$ ，避免符号跳变导致曲线不连续。

根据上述思路写出 matlab 代码，随后将其四个分量随时间绘制，即得到末端执行器在世界坐标系下的四元数变化曲线。



## 题目 2

### 一、A\* 轨迹规划的启发式与路径简化

#### 1. 现有 tie\_breaker 对拓展顺序和路径平滑性的影响

##### 对开集拓展顺序

当两个节点的  $g$  相近时，离目标直线更近、启发更小的节点会更快被取出拓展。这会减少在起点附近和局部区域的横向试探，更多地沿着“朝向终点”的方向扩展，扩展区域更瘦、搜索节点数减少。

##### 对路径平滑性的影响

在普通、未放大的启发下，很多  $f$  值相近的候选节点会被交替拓展，容易走出“一个格向东、一个格向北”的“之”字形 路径。

稍微放大启发后，靠近“直线”方向的节点在  $f$  值上更有优势，于是搜索更偏向直线，得到的离散路径“折线角度更小”，弯来弯去的“走之”现象减弱，路径更平滑、更接近直线。

#### 2. 想让无人机更偏好“平面飞行”该怎么改？

少做不必要的高度变化，优先在同一高度层上飞行，可以在 `AstarGetSucc` 里对含有垂直分量的移动额外加权。

#### 3. pathSimplify 的 path\_resolution 太大或太小会怎样？

##### path\_resolution 过大

很多中间 waypoint 被删掉，只剩下少数关键点，得到长直线 + 大转角的粗路径。

这些直线有可能贴着障碍边缘走，甚至直线本身穿过障碍附近的狭窄区域。理论上算法保证离原离散路径偏差不超过 `path_resolution`，但原路径本身就贴得很近时，这个偏差足以越界。

##### path\_resolution 过小

Douglas-Peucker 很少删除点，路径几乎保持原始 A\* 栅格路径，节点很密、折线很多。

路径虽然完全在空闲栅格里，但几乎每一两个栅格就要拐一次方向。

节点过多也增加轨迹优化/跟踪算法的计算负担，实时性变差，间接影响安全

## 二、SO(3)位置控制器的力姿态生成

### 1.各项在飞行中的作用

$k_x$  · 位置误差是主要的位置刚度来源。

$k_v$  · 速度误差是速度阻尼。

$mass \cdot des\_acc$  使得理想情况下准确跟踪参考轨迹

$mass \cdot k_a \cdot (des\_acc - acc)$  在中等误差区间，多给一点内环加速度反馈，快速补偿外界扰动、模型误差；在误差很小且  $totalError \approx 0$  时， $k_a$  很小，避免把加速度传感器的噪声直接放大到推力上。

### 2. 为什么在大误差或噪声大时要截断/限幅 $k_a$ ?

加速度测量噪声和瞬态不可靠

大位置/速度误差时，外环的  $k_x/k_v$  已经给出很大控制力

### 3. 换更重的机体，怎样调质量 $mass$ 与 $k_x/k_v$ ?

$mass$  改为真实质量， $k_x$  与  $k_v$  都按质量同比例放大

### 4. 对推力大小、姿态角度和稳定性的影响

推力绝对值增大，加速度水平基本不变。

姿态角度不变

线性稳定性和响应速度与原机体基本一致

## 三、动力学建模与约束

### 1. 质量、惯量与气动阻尼建模对控制分配的影响

质量估计偏差

过小：系统动作过猛，产生过冲与振荡

过大：系统反应迟钝，跟踪滞后，有稳态误差

惯量估计偏差

主要影响姿态响应，导致姿态滞后或振荡

忽略阻力或阻力建模简单

前飞时出现稳态偏差、滞后

减速时出现过冲或“甩尾”

## 2. 加入气动阻尼应在控制层还是规划层实现？

### 控制层补偿

优点：高精度、高性能跟踪

缺点：模型依赖强，易因建模误差引发振荡，调参难度高

### 规划层处理

优点：最稳定、最鲁棒、实现简单

缺点：速度/加速度上限会变得保守，利用不了全部性能