Minimum Snap轨迹规划详解（3）闭式求解

轨迹规划

如果QP问题只有等式约束没有不等式约束，那么是可以闭式求解（close form）的。闭式求解效率要快很多，而且只需要用到矩阵运算，不需要QPsolver。
这里介绍Nicholas Roy文章中闭式求解的方法。

1. QP等式约束构建

闭式法中的Q矩阵计算和之前一样（参照文章一），但约束的形式与之前略为不同，在之前的方法中，等式约束只要构造成$[\dots]p = b$的形式就可以了，而闭式法中，每段poly都构造成

$$A_i p_i = d_i,\ A_i = [A_0\ A_t]_i^T,\ d_i = [d_0, d_T]_i$$

其中d_0, d_T为第i段poly的起点和终点的各阶导数组成的向量，比如只考虑PVA：$d_0 = [p_0, v_0, a_0]^T$，当然也可以把jerk，snap等加入到向量。注意：这里是不管每段端点的PVA是否已知，都写进来。
块合并各段轨迹的约束方程得到

$$\underbrace{A_{total}}_{k(n+1)\times 6k}\begin{bmatrix} p_1 \\ \vdots \\ p_k \end{bmatrix} = \begin{bmatrix} d_1 \\ \vdots \\ d_k \end{bmatrix} = \underbrace{\begin{bmatrix} p_1(t_0) \\ v_1(t_0) \\ a_1(t_0) \\ p_1(t_1) \\ v_1(t_1) \\ a_1(t_1) \\ \vdots \\ p_k(t_{k-1}) \\ v_k(t_{k-1}) \\ a_k(t_{k-1}) \\ p_k(t_k) \\ v_k(t_k) \\ a_k(t_k) \end{bmatrix}}_{6k\times 1}$$

k为轨迹段数，n为轨迹的阶数，设只考虑pva，A_{total}的size为$(n_{order}+1)k \times 6k$。这里为了简化，没有把每段poly的timestamp都改成从0开始，一般，为了避免timestamp太大引起数值问题，每段poly的timestamp都成0开始。
由上式可以看到，A_{total}是已知的（怎么构造可参见文章一种的等式约束构造方法），而d中只有少部分（起点、终点的pva等）是已知的，其他大部分是未知的。如果能够求出d，那么轨迹参数可以通过$p = A^{-1}d$很容易求得。

2. 如何求d?

闭式法的思路是：将d向量中的变量分成两部分：”d中所有已知量组成的Fix部分d_F”和”所有未知量组成的Free部分d_P”。然后通过推导，根据d_F求得d_P，从而得到d，最后求得p。
下面介绍整个推导过程，

2.1. 消除重复变量（连续性约束）

可以会发现，上面构造等式约束时，并没有加入连续性约束，连续性约束并不是直接加到等式约束中。
考虑到连续性（这里假设PVA连续），d向量中很多变量其实重复了，即

$$p_i(t_i) = p_{i+1}(t_i),\quad v_i(t_i) = v_{i+1}(t_i),\quad a_i(t_i) = a_{i+1}(t_i)$$

因此需要一个映射矩阵将一个变量映射到两个重复的变量上，怎么映射?

- 如$\begin{bmatrix} a \\ a \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \end{bmatrix} a$，将变量$a$映射到左边向量中的两个变量。

所以构造映射矩阵$M_{6k\times 3(k+1)}$：

$$\underbrace{\begin{bmatrix} d_1 \\ \vdots \\ d_k \end{bmatrix}}_{6k\times 1} = \underbrace{\begin{bmatrix} 1 &&&&&&&&& \\ & 1 &&&&&&&& \\ && 1 &&&&&&& \\ &&& 1 &&&&&& \\ &&&& 1 &&&&& \\ &&&&& 1 &&&& \\ &&& 1 &&&&&& \\ &&&& 1 &&&&& \\ &&&&& 1 &&&& \\ &&&&&& 1 &&& \\ &&&&&&& 1 && \\ &&&&&&&& 1 & \\ &&&&&&&&& \ddots \end{bmatrix}}_{M} \underbrace{\begin{bmatrix} p(t_0) \\ v(t_0) \\ a(t_0) \\ p(t_1) \\ v(t_1) \\ a(t_1) \\ p(t_2) \\ v(t_2) \\ a(t_2) \\ \vdots \\ p(t_k) \\ v(t_k) \\ a(t_k) \end{bmatrix}}_{3(k+1)\times 1}$$

即$d = Md'$。

2.2 向量元素置换

消除掉重复变量之后，需要调整d'中的变量，把fix部分和free部分分开排列，可以左成一个置换矩阵C，使得

$$d' = C\begin{bmatrix} d_F \\ d_P \end{bmatrix}$$

。
C矩阵怎么构造？

- 举个例子，设$d' = \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix}$，其中$a, c, d$是已知($d_F$)，$b$未知($d_P$)，构造一个$4 \times 4$的单位阵，取$d_F$所在的(1,3,4)列放到左边，再取$d_P$所在的(2)列放到右边，就构造出置换矩阵$C$：

$$\begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}}_{C} \begin{bmatrix} a \\ c \\ d \\ b \end{bmatrix}$$

2.3 转成无约束优化问题

由上面两步可得

$$\begin{gathered} d = MC \begin{bmatrix} d_F \\ d_P \end{bmatrix} \\ p = A^{-1} d = \underbrace{A^{-1} MC}_{K} \begin{bmatrix} d_F \\ d_P \end{bmatrix} = K \begin{bmatrix} d_F \\ d_P \end{bmatrix} \end{gathered} \tag{1}$$

代入优化函数

$$\begin{aligned} \min J &= p^T Qp \\ J &= \begin{bmatrix} d_F \\ d_P \end{bmatrix}^T \underbrace{K^T QK}_{R} \begin{bmatrix} d_F \\ d_P \end{bmatrix} \\ &= \begin{bmatrix} d_F \\ d_P \end{bmatrix}^T \begin{bmatrix} R_{FF} & R_{FP} \\ R_{PF} & R_{PP} \end{bmatrix} \begin{bmatrix} d_F \\ d_P \end{bmatrix} \\ &= d_F^T R_{FF} d_F + d_F^T R_{FP} d_P + d_P^T R_{PF} d_F + d_P^T R_{PP} d_P \\ Q\text{对称} \Rightarrow R\text{对称} &\Rightarrow = d_F^T R_{FF} d_F + 2 d_F^T R_{FP} d_P + d_P^T R_{PP} d_P \end{aligned} \tag{2}$$

令J对d_P的导数$\frac{\partial J}{\partial d_P} = 0$求极值点：

$$\begin{gathered} \Rightarrow 2 d_F^T R_{FP} + 2 d_P^T R_{PP} d_P = 0 \ \ (\text{注意} R_{PP}^T = R_{PP}) \\ \Rightarrow d_p = -R_{PP}^{-1} R_{FP}^T d_F \end{gathered} \tag{3}$$

至此求得d_P，从而求出p。

3. 闭式法步骤

总结一下整个闭式法的步骤：

1. 先确定轨迹阶数（比如5阶），再确定d向量中的约束量（pva），进而根据各段的时间分配求得A_{total}。
2. 根据连续性约束构造映射矩阵M，并确定d向量中哪些量是Fix(比如起点终点pva，中间点的p等)，哪些量是Free，进而构造置换矩阵C，并求得$K = A^{-1}MC$。
3. 计算QP目标函数中的Q（$\min Jerk/Snap$）并计算$R = K^T QK$，根据fix变量的长度将R拆分成$R_{FF}, R_{FP}, R_{PF}, R_{PP}$四块。
4. 填入已知变量得到d_F，并根据$d_p = -R_{PP}^{-1} R_{FP}^T d_F$计算得到d_P。
5. 根据公式 $p = K \begin{bmatrix} d_F \\ d_P \end{bmatrix}$ 计算得到轨迹参数p。

闭式法主要计算量就在A矩阵的求逆，其他计算基本上是矩阵构造，所以效率比较高，但由于没有不等式约束，所以在中间点只能加强约束，corridor不能直接加到QP问题中，只能是通过压点来实现corridor。
在对计算效率要求比较高或者不想用QPsolver时，可以使用闭式法求解。

代码见这里，由于效果和文章一中的效果一样，这里就不再贴图。

参考文献

1. Richter C, Bry A, Roy N. Polynomial trajectory planning for aggressive quadrotor flight in dense indoor environments[M]//Robotics Research. Springer International Publishing, 2016: 649-666.