深蓝学院
shenlanxueyuan.com

# 移动机器人运动规划
# 第五章作业分享

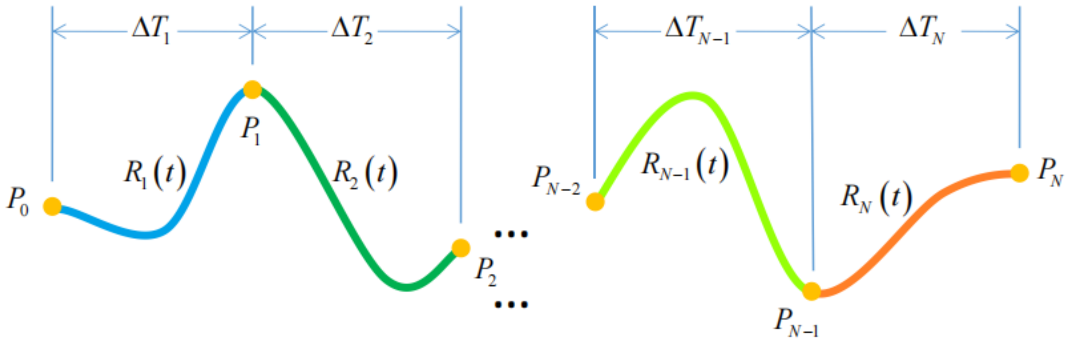
主讲人 CHENXXX

# 问题描述

**目标：**补全代码，使得用户可以在rviz中给定任意waypoints生成平滑轨迹

## Homework

### 3D Minimum Jerk Trajectory Generation




Input:

3-d pos, vel, and acc at start and terminal stamp;
(M-1) 3-d waypoint pos;
M durations for each trajectory pieces.

Output:

Coefficients of 3-d minimum jerk trajectory.

Requirement:

Use optimality conditions.

**Theorem (Optimality Conditions).** *A trajectory, denoted by $z^*(t) : [t_0, t_M] \mapsto \mathbb{R}^m$, is optimal, if and only if the following conditions are satisfied:*

- *The map $z^*(t) : [t_{i-1}, t_i] \mapsto \mathbb{R}^m$ is parameterized as a $2s-1$ degree polynomial for any $1 \le i \le M$;*
- *The boundary and intermediate conditions all hold;*
- *$z^*(t)$ is $2s - d_i - 1$ times continuously differentiable at $t_i$ for any $1 \le i < M$.*

*Moreover, a unique trajectory exists for these conditions.*

# 思路

本章的一个重点是最优性条件。设$m$是平坦输出空间的总维数，M为轨迹的段数，s为目标函数中导数的阶数，一条轨迹$z^*(t):[t_0,t_M]\mapsto\mathbb{R}^m$是最优轨迹，当且仅当同时满足下面四个条件：

1. 对于任意一段轨迹轨迹$1\le i\le M$，轨迹$z^*(t):[t_{i-1},t_i]\mapsto\mathbb{R}^m$被建模为$(2s-1)$阶的多项式
2. 满足边界条件：$z^{[s-1]}(t_0)=\bar{z}_0, z^{[s-1]}(t_M)=\bar{z}_f$。其中，$z^{[s-1]}$表示轨迹的0~s-1阶导数，$\bar{z}_0$表示用户给定的起始条件，$\bar{z}_f$表示用户给定的终止条件
3. 满足中间条件：$z^{[d_i-1]}(t_i)=\bar{z}_i, 1\le i\le M$。其中，$\bar{z}_i$表示用户给定的中间条件，如waypoints
4. 轨迹$z^*(t)$在任意$t_i$处是$\bar{d}_i-1$次连续可微的，其中$\bar{d}_i=2s-d_i$

最优性条件提供了一种直接求解最优路径的方法，即通过构建线性方程Ax=b即可求出最优路径，这种方法在求解时具有线性复杂度，同时不需要显示或隐式地给出目标函数和梯度。下面通过一种general的形式给出直接求解最优路径的方法。

参考文献：《Geometrically Constrained Trajectory Optimization for Multicopters》

# 思路

深蓝学院
shenlanxueyuan.com

考虑m维的平坦输出空间轨迹，其第i段定义为$(N=2s-1)$维的多项式

$$p_i(t)=\mathbf{c}_i^\top\beta(t-t_{i-1}),t\in[t_{i-1},t_i]$$

其中，$\beta(x)=(1,x,\ldots,x^N)^\top$是多项式的基，$\mathbf{c}_i\in\mathbb{R}^{2s\times m}$是多项式的系数。注意这里使用的是相对时间，起始时间$t_0=0$。整段路径可以通过系数矩阵$\mathbf{c}\in\mathbb{R}^{2Ms\times m}$和时间向量$\mathbf{T}\in\mathbb{R}_{>0}^M$（$\mathbb{R}_{>0}$表示里面的元素都是大于0的数）来描述

$$\mathbf{c}=(c_1^\top,\ldots,c_M^\top),\mathbf{T}=(T_1,\ldots,T_M)^\top$$

其中，T_i表示为第i段的持续时间。然后可以定义时间戳$t_i=\sum_{j=1}^i T_j$和总持续时间$T=||\mathbf{T}||_1$。那么M段轨迹$p:[0,T]\mapsto\mathbb{R}^m$可以定义为

$$p(t)=p_i(t),\forall t\in[t_{i-1},t_i),\forall i\in 1,\ldots M$$

参考文献：《Geometrically Constrained Trajectory Optimization  for Multicopters》

# 思路

我们可以将边界条件约束和中间条件约束加在系数矩阵$c$中，进而写成线性方程的形式。定义在起始、终止和中间时间戳$t_i$处特定的导数分别为$\mathbf{D}_0, \mathbf{D}_m \in \mathbb{R}^{s\times m}$和$\mathbf{D}_i \in \mathbb{R}^{d_i\times m}$。其中，$\mathbf{D}_i$中的一列表示一个维度。然后，在$t_i$处的约束条件可以隐藏在矩阵$\mathbf{E}_i, \mathbf{F}_i \in \mathbb{R}^{2s\times 2s}$

$$\begin{bmatrix}\mathbf{E}_i & \mathbf{F}_i\end{bmatrix}\begin{bmatrix}\mathbf{c}_i \\ \mathbf{c}_{i-1}\end{bmatrix} = \begin{bmatrix}\mathbf{D}_i^{d_i\times m} \\ \mathbf{0}_{\bar{d}_i\times m}\end{bmatrix}$$

$$\mathbf{E}_i = \left(\beta(T_i), \ldots, \beta^{(d_i-1)}(T_i), \beta(T_i), \ldots, \beta^{(\bar{d}_i-1)}(T_i)\right)^\top$$

$$\mathbf{F}_i = \left(\mathbf{0}, -\beta(0), \ldots, -\beta^{(\bar{d}_i-1)}(0)\right)^\top$$

对于起始时刻和终止时刻，$\mathbf{F}_0, \mathbf{E}_M \in \mathbb{R}^{s\times 2s}$形式有所不同

$$\mathbf{F}_0 = \left(\beta(0), \ldots, \beta^{(s-1)}(0)\right)^\top$$

$$\mathbf{E}_M = \left(\beta(T_M), \ldots, \beta^{(s-1)}(T_M)\right)^\top$$

参考文献：《Geometrically Constrained Trajectory Optimization  for Multicopters》

# 思路

深蓝学院
shenlanxueyuan.com

线性方程组$\mathbf{Mc} = \mathbf{b}$描述为

$$\mathbf{M} = \begin{bmatrix} \mathbf{F}_0 & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{E}_1 & \mathbf{F}_1 & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{0} & \mathbf{E}_2 & \mathbf{F}_2 & \dots & \mathbf{0} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{F}_{M-1} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{E}_M \end{bmatrix}_{2Ms \times 2Ms}$$

$$\mathbf{b}_{2Ms \times m} = (\mathbf{D}_0^\top, \mathbf{D}_1^\top, \mathbf{0}_{m \times \bar{d}_1}, \dots, \mathbf{D}_{M-1}^\top, \mathbf{0}_{m \times \bar{d}_1}, \mathbf{D}_M^\top)^\top$$

通过最优条件件，可以保证矩阵M是非奇异的，同时我们注意到矩阵M是一个带状矩阵，可以通过PLU分解来进行求解，其复杂度为O(M)

参考文献：《Geometrically Constrained Trajectory Optimization  for Multicopters》

# 代码实现

深蓝学院 shenlanxueyuan.com

补全minimumJerkTrajGen()函数代码。代码已全部贴出，复现即可！

```
void minimumJerkTrajGen(
    // Inputs:
    const int pieceNum, // num of trajectories
    const Eigen::Vector3d &initialPos,
    const Eigen::Vector3d &initialVel,
    const Eigen::Vector3d &initialAcc,
    const Eigen::Vector3d &terminalPos,
    const Eigen::Vector3d &terminalVel,
    const Eigen::Vector3d &terminalAcc,
    const Eigen::Matrix3Xd &intermediatePositions,
    const Eigen::VectorXd &timeAllocationVector,
    // Outputs:
    Eigen::MatrixX3d &coefficientMatrix)
{
    Eigen::MatrixXd M = Eigen::MatrixXd::Zero(6*pieceNum, 6*pieceNum);
    Eigen::MatrixX3d b = Eigen::MatrixX3d::Zero(6*pieceNum,3);

    Eigen::VectorXd T1 = timeAllocationVector;
    Eigen::VectorXd T2 = T1.cwiseProduct(T1);
    Eigen::VectorXd T3 = T2.cwiseProduct(T1);
    Eigen::VectorXd T4 = T3.cwiseProduct(T1);
    Eigen::VectorXd T5 = T4.cwiseProduct(T1);

    M(0,0) = 1.0;
    M(1,1) = 1.0;
    M(2,2) = 2.0;
    b.row(0) = initialPos.transpose();
    b.row(1) = initialVel.transpose();
    b.row(2) = initialAcc.transpose();
    for(int i=0; i<pieceNum-1; i++){
        // E
        M(3+i*6,i*6) = M(4+i*6,i*6) = 1.0;
        M(3+i*6,i*6+1) = M(4+i*6,i*6+1) =T1(i);
        M(3+i*6,i*6+2) = M(4+i*6,i*6+2) =T2(i);
        M(3+i*6,i*6+3) = M(4+i*6,i*6+3) =T3(i);
        M(3+i*6,i*6+4) = M(4+i*6,i*6+4) =T4(i);
        M(3+i*6,i*6+5) = M(4+i*6,i*6+5) =T5(i);
```

```
        M(5+i*6,i*6+1) = 1.0;
        M(5+i*6,i*6+2) = 2.0*T1(i);
        M(5+i*6,i*6+3) = 3.0*T2(i);
        M(5+i*6,i*6+4) = 4.0*T3(i);
        M(5+i*6,i*6+5) = 5.0*T4(i);

        M(6+i*6,i*6+2) = 2.0;
        M(6+i*6,i*6+3) = 6.0*T1(i);
        M(6+i*6,i*6+4) = 12.0*T2(i);
        M(6+i*6,i*6+5) = 20.0*T3(i);

        M(7+i*6,i*6+3) = 6.0;
        M(7+i*6,i*6+4) = 24.0*T1(i);
        M(7+i*6,i*6+5) = 60.0*T2(i);

        M(8+i*6,i*6+4) = 24.0;
        M(8+i*6,i*6+5) = 120.0*T1(i);

        // F
        M(4+i*6,(i+1)*6) = -1.0;
        M(5+i*6,(i+1)*6+1) = -1.0;
        M(6+i*6,(i+1)*6+2) = -2.0;
        M(7+i*6,(i+1)*6+3) = -6.0;
        M(8+i*6,(i+1)*6+4) = -24.0;

        b.row(3+i*6) = intermediatePositions.col(i).transpose();
    }

    int end_row = 6*(pieceNum) - 3;
    int end_col = 6*(pieceNum-1);
    M(end_row,end_col) = 1.0;
    M(end_row,end_col+1) = T1(pieceNum-1);
    M(end_row,end_col+2) = T2(pieceNum-1);
    M(end_row,end_col+3) = T3(pieceNum-1);
    M(end_row,end_col+4) = T4(pieceNum-1);
    M(end_row,end_col+5) = T5(pieceNum-1);
```

```
    M(end_row+1,end_col+1) = 1.0;
    M(end_row+1,end_col+2) = 2.0*T1(pieceNum-1);
    M(end_row+1,end_col+3) = 3.0*T2(pieceNum-1);
    M(end_row+1,end_col+4) = 4.0*T3(pieceNum-1);
    M(end_row+1,end_col+5) = 5.0*T4(pieceNum-1);

    M(end_row+2,end_col+2) = 2.0;
    M(end_row+2,end_col+3) = 6.0*T1(pieceNum-1);
    M(end_row+2,end_col+4) = 12.0*T2(pieceNum-1);
    M(end_row+2,end_col+5) = 20.0*T3(pieceNum-1);

    b.row(end_row) = terminalPos.transpose();
    b.row(end_row+1) = terminalVel.transpose();
    b.row(end_row+2) = terminalAcc.transpose();

    std::cout<<"M:"<<std::endl<<M<<std::endl;
    std::cout<<"b:"<<std::endl<<b<<std::endl;

    coefficientMatrix = M.colPivHouseholderQr().solve(b);
    std::cout<<"result:"<<std::endl<<coefficientMatrix<<std::endl;
```

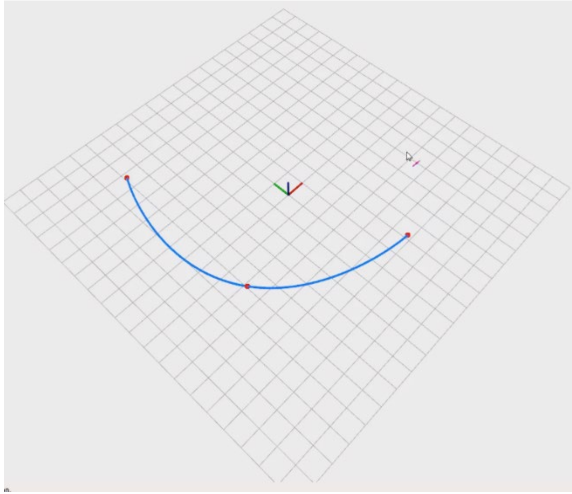
# 实验结果

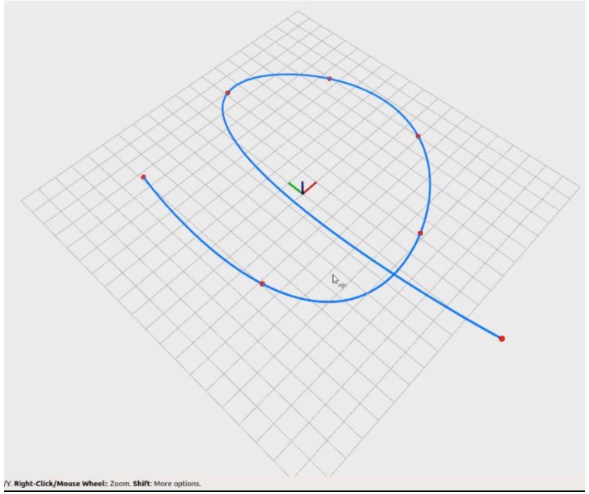
深蓝学院
shenlanxueyuan.com

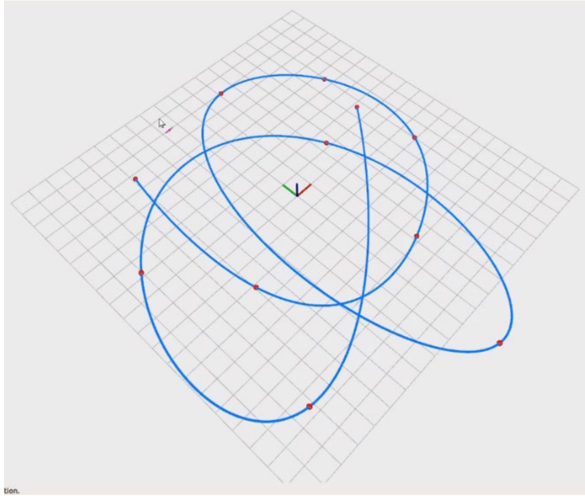
生成得M矩阵和b矩阵如下图所示，结构与论文中相同

```
M:
       1        0        0        0        0        0        0        0        0        0        0        0
       0        1        0        0        0        0        0        0        0        0        0        0
       0        0        2        0        0        0        0        0        0        0        0        0
       1  13.2962  176.788  2350.61  31254.1   415560        0        0        0        0        0        0
       1  13.2962  176.788  2350.61  31254.1   415560       -1        0        0        0        0        0
       0        1  26.5923  530.365  9402.43   156270        0       -1        0        0        0        0
       0        0        2   79.777  2121.46  47012.1        0        0       -2        0        0        0
       0        0        0        6  319.108  10607.3        0        0        0       -6        0        0
       0        0        0        0       24  1595.54        0        0        0        0      -24        0
       0        0        0        0        0        0        1  7.37169  54.3419  400.592  2953.04  21768.9
       0        0        0        0        0        0        0        1  14.7434  163.026  1602.37  14765.2
       0        0        0        0        0        0        0        0        2  44.2302  652.103  8011.84
b:
-1.56789   9.15566   1.49707
       0         0         0
       0         0         0
-1.79905  -3.09971  0.523322
       0         0         0
       0         0         0
       0         0         0
       0         0         0
       0         0         0
 4.12099  -5.42224  0.126424
       0         0         0
       0         0         0
```

# 实验结果





深蓝学院
shenlanxueyuan.com

# 感谢各位聆听！

Thanks for Listening