深蓝学院
shenlanxueyuan.com

Lecture 6

Soft and Hard Constrained Trajectory Optimization 作业讲解

主讲人 朱江超

homework

In matlab, write a corridor-constrained piecewise Bezier curve generation.

The conversion between Bezier to monomial polynomial is given.

The corridor is pre-defined.

Only position needs to be constrained.

hw5 vs hw6

Define higher order (l^{th}) control points:

$$a_{\mu j}^{0,i} = c_{\mu j}^{i}, a_{\mu j}^{l,i} = \frac{n!}{(n-l)!} \cdot (a_{\mu j}^{l-1,i+1} - a_{\mu j}^{l-1,i}), \quad l \geq 1$$

- Boundary Constraints:

$$\boxed{a_{\mu j}^{l,0} \cdot s_j^{(1-l)} = d_{\mu j}^{(l)}}$$

- Continuity Constraints:

$$\boxed{a_{\mu j}^{\phi,n} \cdot s_j^{(1-\phi)} = a_{\mu,j+1}^{\phi,0} \cdot s_{j+1}^{(1-\phi)}, \quad a_{\mu j}^{0,i} = c_{\mu j}^{i}.}$$

- Safety Constraints:

$$\boxed{\beta_{\mu j}^{-} \leq c_{\mu j}^{i} \leq \beta_{\mu j}^{+}, \ \mu \in \{x, y, z\}, \ i = 0, 1, 2, ..., n,}$$

- Dynamical Feasibility Constraints:

$$v_m^{-} \leq n \cdot (c_{\mu j}^{i} - c_{\mu j}^{i-1}) \leq v_m^{+},$$

$$a_m^{-} \leq n \cdot (n-1) \cdot (c_{\mu j}^{i} - 2c_{\mu j}^{i-1} + c_{\mu j}^{i-2}) / s_j \leq a_m^{+}$$

Stack all of these

$$\begin{aligned} \min \quad & \mathbf{c}^T \mathbf{Q}_o \mathbf{c} \\ \text{s.t.} \quad & \boxed{\mathbf{A}_{eq}\mathbf{c} = \mathbf{b}_{eq},} \\ & \boxed{\mathbf{A}_{ie}\mathbf{c} \leq \mathbf{b}_{ie},} \\ & \boxed{\mathbf{c}_j \in \Omega_j, \quad j = 1, 2, ..., m,} \end{aligned}$$

A typical convex QP formulation.

We only solve this program once to determine whether there is a qualified trajectory exists.

Minimum Snap Trajectory Generation

- Constrained quadratic programming (QP) formulation:

$$\min \quad \begin{bmatrix} \mathbf{p}_1 \\ \vdots \\ \mathbf{p}_M \end{bmatrix}^T \begin{bmatrix} \mathbf{Q}_1 & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \ddots & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{Q}_M \end{bmatrix} \begin{bmatrix} \mathbf{p}_1 \\ \vdots \\ \mathbf{p}_M \end{bmatrix}$$

$$\text{s.t.} \quad \mathbf{A}_{\mathrm{eq}} \begin{bmatrix} \mathbf{p}_1 \\ \vdots \\ \mathbf{p}_M \end{bmatrix} = \mathbf{d}_{eq}$$

It's a typical **convex optimization** program.

about si

$$f_\mu(t) = \begin{cases} s_1 \cdot \sum_{i=0}^{n} c_{\mu 1}^i b_n^i(\frac{t-T_0}{s_1}), & t \in [T_0, T_1] \\ s_2 \cdot \sum_{i=0}^{n} c_{\mu 2}^i b_n^i(\frac{t-T_1}{s_2}), & t \in [T_1, T_2] \\ \vdots & \vdots \\ s_m \cdot \sum_{i=0}^{n} c_{\mu m}^i b_n^i(\frac{t-T_{m-1}}{s_m}), & t \in [T_{m-1}, T_m], \end{cases} \tag{5}$$

$$a_{\mu j}^{l,0} \cdot s_j^{(1-l)} = d_{\mu j}^{(l)}, \begin{cases} position\ constraint: l = 0, a_{\mu j}^{0,0} \cdot s_j = p_{\mu j} \\ velocity\ constriant: l = 1, a_{\mu j}^{1,0} = v_{\mu j} \\ accelerate\ constriant: l = 2, \frac{a_{\mu j}^{2,0}}{s_j} = a_{\mu j} \end{cases}$$

about (n-1)-order control points

$$v_m^- \le n \cdot \left(c_{\mu j}^{i} - c_{\mu j}^{i-1}\right) \le v_m^+$$

$$a_m^- \le n \cdot (n-1) \cdot \frac{c_{\mu j}^{i} - 2c_{\mu j}^{i-1} + c_{\mu j}^{i-2}}{s_j} \le a_m^+$$

$$j_m^- \le n \cdot (n-1) \cdot (n-2) \cdot \frac{c_{\mu j}^{i} - 3c_{\mu j}^{i-1} + 3c_{\mu j}^{i-2} - c_{\mu j}^{i-3}}{s_j^2} \le j_m^+$$

系数组成杨辉三角

1 n=1
1 1 n=2
1 2 1 n=3
1 3 3 1 n=4
1 4 6 4 1 n=5
1 5 10 10 5 1 n=6
1 6 15 20 15 6 1 n=7

```
function coef_list = YangHTriangle(n)
    coef_list = zeros(1, n);
    for i = 1:n
        coef_list(i) = (-1)^(i+n)*nchoosek(n-1, i-1);
    end
end
```

getQM()——Important!!!

不能直接套用使用第五章的结论

$$J = \int_0^T \left(\frac{\mathrm{d}^{\mathrm{k}} \left(s \cdot \sum c_i b^i \left(\frac{t}{s} \right) \right)}{dt^4} \right)^2 \quad dt, t \in [0, T]$$

$$= \int_0^1 s^3 \left(\frac{\mathrm{d}^k \left(\sum c_i b^i(\tau) \right)}{s^k d\tau^4} \right)^2 d\tau, \tau \in [0,1]$$

$$= s^{-(2k-3)} \int_0^1 \left(\frac{\mathrm{d}^k \left(\sum p_i \tau^i \right)}{d\tau^4} \right)^2 d\tau, \tau \in [0,1]$$

$$= s^{-(2k-3)} \int_0^1 (f^4(\tau))^2 d\tau, \tau \in [0,1]$$

$$= \begin{bmatrix} \cdots \\ p_i \\ \cdots \end{bmatrix}^T \left[\cdots \frac{i(i-1)(i-2)(i-3)l(l-1)(l-2)(l-3)s^{-(2k-3)}}{i+l-7} \cdots \right] \begin{bmatrix} \cdots \\ p_i \\ \cdots \end{bmatrix}$$

```
M_j = getM(7);
for j = 0:n_seg-1
    Q_j = zeros(n_seg+1, n_seg+1);
    for i = 4:7
        for l = i:7
            if i <= l
                Q_j(i+1, l+1) = factorial(i)/factorial(i-4)*...
                    factorial(l)/factorial(l-4)*...
                    ts(j+1)^(3-2*4) / (i+l-7);
            else
                Q_j(i+1, l+1) = Q_j(l+1, i+1);
            end
        end
    end
    Q = blkdiag(Q, Q_j);
    M = blkdiag(M, M_j);
end
```

getAbeq()

导数约束：只对起始点和终止点pva(j)施加——和作业5相比少了中间点位置约束
连续性约束：对于中间点施加

```
%#################################################
% Boundary constraints in start point
Aeq_start = zeros(d_order, n_all_poly);
for k = 0:d_order-1
    Aeq_start(k+1, 1:k+1) = ...
        factorial(n_order)/factorial(n_order-k)*...
        YangHTriangle(k+1)*ts(1)^(1-k);
end
beq_start = start_cond';

%#################################################
% Boundary constraints in end point
Aeq_end = zeros(d_order, n_all_poly);
for k = 0:d_order-1
    Aeq_end(k+1, n_all_poly-k:n_all_poly) = ...
        factorial(n_order)/factorial(n_order-k)*...
        YangHTriangle(k+1)*ts(n_seg)^(1-k);
end
beq_end = end_cond';
```

```
%#################################################
% p,v,a,j continuity constrain between each 2 segments
Aeq_con = zeros((n_seg-1)*d_order, n_all_poly);
beq_con = zeros((n_seg-1)*d_order, 1);
for k = 0:d_order-1
    start_idx_1 = k*(n_seg-1);
    for j = 0:n_seg-2
        start_idx_2 = (n_order+1)*(j+1);
        Aeq_con(start_idx_1+j+1, start_idx_2-k:start_idx_2) = ...
                factorial(n_order)/factorial(n_order-k)*...
                YangHTriangle(k+1)*ts(j+1)^(1-k);
        Aeq_con(start_idx_1+j+1, start_idx_2+1:start_idx_2+1+k) = ...
                   -factorial(n_order)/factorial(n_order-k)*...
                   YangHTriangle(k+1)*ts(j+2)^(1-k);

    end
end
```

getAbieq()

位置约束：2*n_seg*(n_order+1)

速度约束：2*n_seg*n_order

加速度约束：2*n_seg*(n_order-1)

```
constraint_range = zeros(n_seg, 2*d_order);
for j = 0:n_seg-1
    constraint_range(j+1,:) = [corridor(j+1, axis)+corridor(j+1, 2+axis), ...
        -(corridor(j+1, axis)-corridor(j+1, 2+axis)),...
        v_max, v_max, a_max, a_max, j_max, j_max];
end
[Aieq, bieq] = getAbieq(n_seg, d_order, constraint_range, ts);
```

```
function [Aieq, bieq] = getAbieq(n_seg, d_order, constraint_range, ts)
    n_order = 2*d_order-1;
    n_all_poly = n_seg*2*d_order;
    Aieq = zeros(2*n_seg*d_order*(n_order+1-(d_order-1)/2), n_all_poly);
    bieq = zeros(2*n_seg*d_order*(n_order+1-(d_order-1)/2), 1);

    %#####################################################
    % p,v,a,j
    for k = 0:d_order-1
        for j = 0:n_seg-1
            start_idx_1 = 2*n_seg*k*(n_order+1-(k-1)/2)+2*j*(1+n_order-k);
            start_idx_2 = (n_order+1-k)*j;
            for i = 0:n_order-k
                Aieq(start_idx_1+2*i+1:start_idx_1+2*i+2, start_idx_2+i+1:start_idx_2+i+1+k) = ...
                    [YangHTriangle(k+1)*factorial(n_order)/factorial(n_order-k)*ts(j+1)^(1-k);
                    -YangHTriangle(k+1)*factorial(n_order)/factorial(n_order-k)*ts(j+1)^(1-k)];
                bieq(start_idx_1+2*i+1:start_idx_1+2*i+2, 1) = constraint_range(j+1,2*k+1:2*k+2)';
            end
        end
    end
```

Thanks for listening!

运动规划第六章作业

1、基于 Bezier 曲线 MinimumSnapTrajetory 问题。

首先 Bezier 曲线由如下公式表示：

$$\boldsymbol{B}_j(t)=c_j^0 b_n^0(t)+c_j^1 b_n^1(t)+\cdots+c_j^n b_n^n(t)=\sum_{i=0}^{i=n} c_j^i b_n^i(t) \tag{1-1}$$

其中

$$b_n^i(t)=\binom{n}{i}\cdot t^i\cdot(1-t)^{n-i} \tag{1-2}$$

观察式（1-1），我们可以发现 Bezier 曲线由控制点c_j^i以及对应的权重函数b_n^i（也称 Bernstein polynomial basis）组成。对于$\mu\in\{x,y,z\}$三个轴，我们将每个对应坐标轴上表示为由多段 Bezier 曲线构成的轨迹，如下所示：

$$f_\mu(t)=\begin{cases} s_1\cdot\sum_{i=0}^{i=n} c_{\mu 1}^i b_n^i\left(\frac{t-T_0}{s_1}\right), & t\in[T_0,T_1]\\ s_2\cdot\sum_{i=0}^{i=n} c_{\mu 2}^i b_n^i\left(\frac{t-T_1}{s_2}\right), & t\in[T_1,T_2]\\ \cdots & \\ s_m\cdot\sum_{i=0}^{i=n} c_{\mu m}^i b_n^i\left(\frac{t-T_{m-1}}{s_m}\right), & t\in[T_{m-1},T_m]\end{cases} \tag{1-3}$$

每段 Bezier 曲线对应不同的时间分配，考虑到 Bezier 曲线的特性，上述式（1-3）将$t=[T_{i-1},T_i]$映射到$\tau=[0,1]$（也称归一化），这个映射关系如下所示：

$$\frac{t-T_j}{s_{j+1}}=\tau \tag{1-4}$$

同时为了保证数值优化的稳定性（**这个目前还不知道怎么解释数值稳定性问题**）在每段 Bezier 曲线上都乘以一个标量系数s_m。接着我们以式(1-3)为基础构建 MinimumSnapTrajetory 问题，其目标函数如下所示：

$$J=\sum_{\mu\in\{x,y,z\}}\int_0^T\left(\frac{\boldsymbol{d}^k\boldsymbol{f}_\mu(t)}{\boldsymbol{dt}^k}\right)^2 dt \tag{1-4}$$

这里以u轴上第j段 Bezier 曲线举例：

$$J_{uj}=\int_0^{s_j}\left(\frac{\boldsymbol{d}^k\boldsymbol{f}_{\mu j}(t)}{\boldsymbol{dt}^k}\right)^2 dt \tag{1-5}$$

从而得到以下式子

$$J_{uj}=\int_0^1 s_j\left(\frac{s_j\cdot\boldsymbol{d}^k\boldsymbol{g}_{\mu j}(\tau)}{\boldsymbol{d}(s_j\cdot\tau)^k}\right)^2 d\tau=\frac{1}{s_j^{2k-3}}\int_0^1\left(\frac{\boldsymbol{d}^k\boldsymbol{g}_{\mu j}(\tau)}{\boldsymbol{d}\tau^k}\right)^2 d\tau \tag{1-6}$$

对比基于传统多项式 MinimumSnapTrajetory 问题，基于 Bezier 曲线 MinimumSnapTrajetory 的目标函数表达式相对复杂，不利于构建形如$J=\boldsymbol{p}^T\boldsymbol{Q}\boldsymbol{p}$二次函数形式。为此我们通过传统多项式系数$\boldsymbol{p}$与 Bezier 曲线系数$\boldsymbol{c}$的线性转换关系式，将基于 Bezier 曲线 MinimumSnapTrajetory 的目标函数转换成$J=\boldsymbol{c}^T\boldsymbol{M}^T\boldsymbol{Q}\boldsymbol{M}\boldsymbol{c}$。以下针对阶数为 3 的 Bezier 曲线如何转换成传统多项式曲线进行推导：

$$\boldsymbol{B}(t)=(1-t)^3c_0+3t(1-t)^2c_1+3t^2(1-t)c_2+t^3c_3 \tag{1-7}$$

由式（1-7）可得：

$$\boldsymbol{B}(t)=\begin{bmatrix}1 & -3 & 3 & -1\\ 0 & 3 & -6 & 3\\ 0 & 0 & 3 & -3\\ 0 & 0 & 0 & 1\end{bmatrix}\begin{bmatrix}1\\ t\\ t^2\\ t^3\end{bmatrix} \tag{1-8}$$

$$\boldsymbol{B}(t)=[c_0 \quad c_1 \quad c_2 \quad c_3]\begin{bmatrix}1 & -3 & 3 & -1\\ 0 & 3 & -6 & 3\\ 0 & 0 & 3 & -3\\ 0 & 0 & 0 & 1\end{bmatrix}\begin{bmatrix}1\\ t\\ t^2\\ t^3\end{bmatrix}=[p_0 \quad p_1 \quad p_2 \quad p_3]\begin{bmatrix}1\\ t\\ t^2\\ t^3\end{bmatrix} \tag{1-9}$$

由式（1-9）可得

$$\boldsymbol{c}^{\boldsymbol{T}}\boldsymbol{M}^{\boldsymbol{T}}=\boldsymbol{p}^{\boldsymbol{T}} \tag{1-10}$$

$$\boldsymbol{M}^{\boldsymbol{T}}=\begin{bmatrix}1 & -3 & 3 & -1\\ 0 & 3 & -6 & 3\\ 0 & 0 & 3 & -3\\ 0 & 0 & 0 & 1\end{bmatrix} \tag{1-11}$$

因此基于 Bezier 曲线 MinimumSnapTrajetory 的归一化目标函数为：

$$J=\boldsymbol{a}^{\boldsymbol{T}}\boldsymbol{s}^{\boldsymbol{T}}\boldsymbol{M}^{\boldsymbol{T}}\boldsymbol{QMsa} \tag{1-12}$$

其中$\boldsymbol{a}$为归一化 Bezier 曲线的控制点。

接下来构建相关约束，基于 Bezier 曲线 MinimumSnapTrajetory 问题主要包含固定等式约束，连续性等式约束、安全性不等式约束以及动力学不等式约束。

固定等式约束主要包含起点 Start、终点 Goal 的$p, v, a, jerk$等约束，通常表示成下列形式：

$$a_{uj}^{l,i}\cdot s_j^{1-l}=d_{uj}^{(l)} \tag{1-13}$$

$$a_{uj}^{l,\mathrm{i}}=\frac{\boldsymbol{n}!}{(\boldsymbol{n}-\boldsymbol{l})!}\left(a_{uj}^{l-1,\mathrm{i}+1}-a_{uj}^{l-1,\mathrm{i}}\right) \tag{1-14}$$

连续性等式约束主要包含两端 Bezier 曲线连接处保证$p, v, a, jerk$连续，通常表示成下列形式：

$$a_{uj}^{l,n}\cdot s_j^{1-l}=a_{uj+1}^{l,0}\cdot s_{j+1}^{1-l} \tag{1-15}$$

安全性约束主要通过约束 Bezier 曲线上所有控制点在事先分析周围环境所生成的飞行 Corridor 内，由于 Bezier 曲线的凸包特性，从而使得整段 Bezier 曲线都是出于安全的区域。通常安全性约束表示成下列形式：

$$\beta_{uj}^{-}\le a_{uj}^{0,i}\cdot s_j\le\beta_{uj}^{+} \tag{1-16}$$

动力学约束主要针对无人机的物理运动极限，通常表示成下列形式：

$$v_m^{-}\le a_{uj}^{1,i}\le v_m^{+} \tag{1-17}$$

$$a_m^{-}\le a_{uj}^{2,i}\cdot s_j^{-1}\le a_m^{+} \tag{1-18}$$

以下将简要展示不同时间分配的仿真结果：

1、五段轨迹时间分配为[1s,1s,1s,1s,1s]

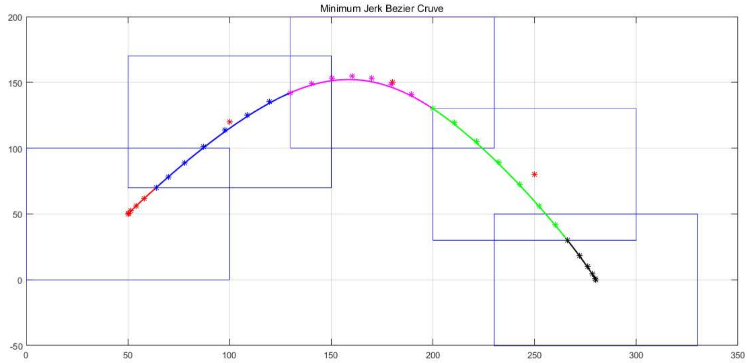

图 1 Minimum Jerk Trajectory

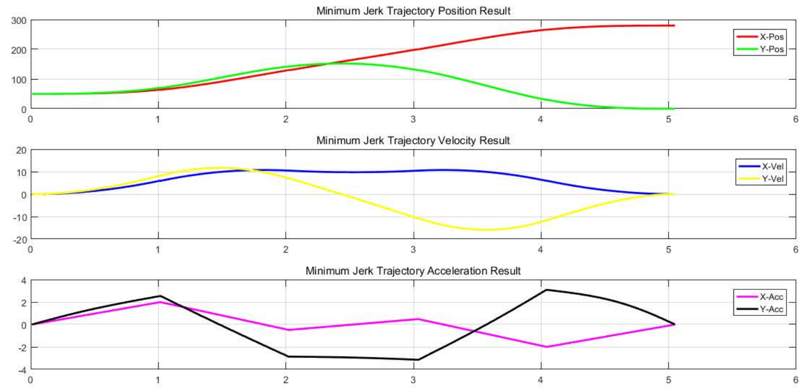

图 2 Minimum Jerk Trajectory 位置、速度、加速度结果

2、五段轨迹时间分配为[1s,3s,2s,4s,1s]

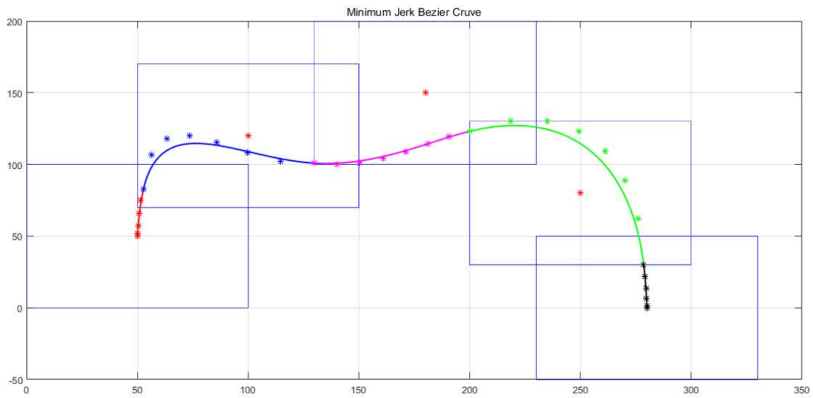

图 3 Minimum Jerk Trajectory

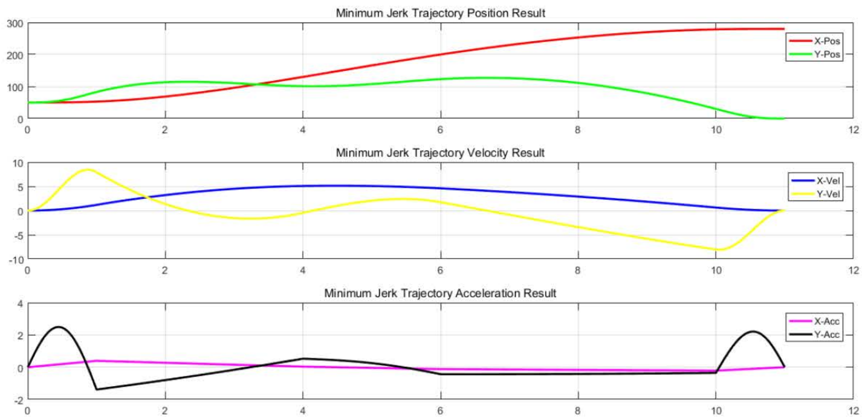

图 4 Minimum Jerk Trajectory 位置、速度、加速度结果

3、五段轨迹时间分配为[3s,1s,1s,1s,3s]

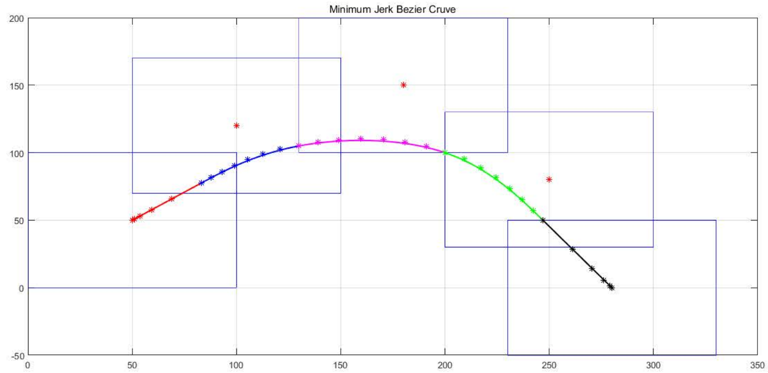

图 5 Minimum Jerk Trajectory

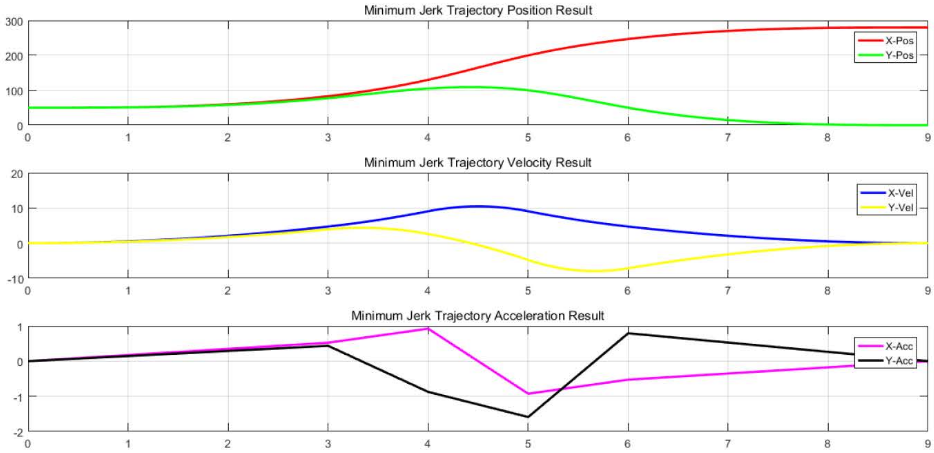

图 6 Minimum Jerk Trajectory 位置、速度、加速度结果

总结：对比三个实验结果，我们可以发现时间分配的问题更像是权重分配问题，分配的时间越多，等价于在归一化的 Minimum Jerk Trajectory 目标函数$J = \boldsymbol{a}^T \boldsymbol{s}^T \boldsymbol{M}^T \boldsymbol{QMsa}$中所占的比重越大，意味着在这段时间内轨迹所产生的 Cost 必须要越小。我们也能够从图 4、图 6 我们能够发现，时间分配多的轨迹，加速度变化率（Jerk）小，而时间分配少的轨迹，加速度变化率大，符合上述理论推导的结果。