深蓝学院
shenlanxueyuan.com

Motion Planning for Mobile Robots
第五章作业讲评

主讲人 谢晓佳

HW

Homework 1.1: In matlab, use the quadprog QP solver, write down a minimum snap trajectory generator

Homework 1.2: In matlab, generate minimum snap trajectory based on the closed form solution

Homework 2.1: In C++/ROS, use the OOQP solver, write down a minimum snap trajectory generator

Homework 2.2: In C++/ROS, use Eigen, generate minimum snap trajectory based on the closed form solution

HW 1

Matlab 作业

本次作业采用 `many relative timeline`，分段轨迹表达式如下：

$$f_k(t) = \sum_{i=0}^{N} p_{k,i} t^i, \quad 0 \leq t \leq \Delta \mathrm{T}_k$$

- Derivative constraints : 规定起始状态 p,v,a,j、终止状态起 p,v,a,j、中间点 p
- Continuity constraints : 规定为前后两端轨迹交点处 p,v,a,j 连续

本次作业独立求解 x 和 y 轴的多项式系数，该功能封装为函数 MinimumSnapQPSolver() 和 MinimumSnapCloseformSolver()，分别代表 QP 解法和闭式解法。

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

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

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

HW 1.1

getQ()

$$f(t) = \sum_i p_i t^i$$

$$\Rightarrow f^{(4)}(t) = \sum_{i\geq 4} i(i-1)(i-2)(i-3)t^{i-4}p_i$$

$$\Rightarrow \left(f^{(4)}(t)\right)^2 = \sum_{i\geq 4, l\geq 4} i(i-1)(i-2)(i-3)l(l-1)(l-2)(l-3)t^{i+l-8}p_i p_l$$

$$\Rightarrow J(T) = \int_{T_{j-1}}^{T_j} \left(f^4(t)\right)^2 dt = \sum_{i\geq 4, l\geq 4} \frac{i(i-1)(i-2)(i-3)j(l-1)(l-2)(l-3)}{i+l-7}\left(T_j^{i+l-7} - T_{j-1}^{i+l-7}\right)p_i p_l$$

$$\Rightarrow J(T) = \int_{T_{j-1}}^{T_j} \left(f^4(t)\right)^2 dt$$

$$= \begin{bmatrix} \vdots \\ p_i \\ \vdots \end{bmatrix}^T \begin{bmatrix} & \vdots & \\ \cdots & \frac{i(i-1)(i-2)(i-3)l(l-1)(l-2)(l-3)}{i+l-7} T^{i+l-7} & \cdots \\ & \vdots & \end{bmatrix} \begin{bmatrix} \vdots \\ p_l \\ \vdots \end{bmatrix}$$

$$\Rightarrow \boxed{J_j(T) = \mathbf{p}_j^T \mathbf{Q}_j \mathbf{p}_j}$$ Minimize this!

```
for i = 4:n_order
  for l = 4:n_order
    den = i + l - 7;
    Q_k(i+1,l+1) = i*(i-1)*(i-2)*(i-3)*l*(l-1)*(l-2)*(l-3)/den*(t_k^den);
  end
end
```

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

升幂

$$f(t) = [p_0, p_1, ..., p_7] \cdot [1, t, t^2, ...t^7] = \mathbf{p} \cdot [1, t, t^2, ...t^7]$$

$$f'(t) = \mathbf{p} \cdot [0, 1, 2t, 3t^2, 4t^3, ...7t^6]$$

$$f''(t) = \mathbf{p} \cdot [0, 0, 2, 6t, 12t^2, ...42t^5]$$

$$f'''(t) = \mathbf{p} \cdot [0, 0, 0, 6, 24t, ...210t^4]$$

HW 1.1

getQ()

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	576	1440	2880	5040	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1440	4800	10800	20160	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	2880	10800	25920	50400	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	5040	20160	50400	1E+05	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	576	1440	2880	5040	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	1440	4800	10800	20160	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	2880	10800	25920	50400	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	5040	20160	50400	1E+05	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	576	1440	2880	5040	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1440	4800	10800	20160	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2880	10800	25920	50400	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	5040	20160	50400	1E+05	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	576	1440	2880	5040
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1440	4800	10800	20160
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2880	10800	25920	50400
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	5040	20160	50400	1E+05

```
for i = 4:n_order
  for l = 4:n_order
    den = i + l - 7;
    Q_k(i+1,l+1) = i*(i-1)*(i-2)*(i-3)*l*(l-1)*(l-2)*(l-3)/den*(t_k^den);
  end
end
```

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

升幂

$$f(t) = [p_0, p_1, ..., p_7] \cdot [1, t, t^2, ...t^7] = \mathbf{p} \cdot [1, t, t^2, ...t^7]$$
$$f'(t) = \mathbf{p} \cdot [0, 1, 2t, 3t^2, 4t^3, ...7t^6]$$
$$f''(t) = \mathbf{p} \cdot [0, 0, 2, 6t, 12t^2, ...42t^5]$$
$$f'''(t) = \mathbf{p} \cdot [0, 0, 0, 6, 24t, ...210t^4]$$

HW 1.1

getAbeq()

- Derivative constraint for one polynomial segment
  - Also models waypoint constraint (0^{th} order derivative)

$$f_j^{(k)}(T_j) = x_j^{(k)}$$
$$\Rightarrow \sum_{i\ge k} \frac{i!}{(i-k)!} T_j^{i-k} p_{j,i} = x_{T,j}^{(k)}$$
$$\Rightarrow \begin{bmatrix} \cdots & \frac{i!}{(i-k)!} T_j^{i-k} & \cdots \end{bmatrix} \begin{bmatrix} \vdots \\ p_{j,i} \\ \vdots \end{bmatrix} = x_{T,j}^{(k)}$$
$$\Rightarrow \begin{bmatrix} \cdots & \frac{i!}{(i-k)!} T_{j-1}^{i-k} & \cdots \\ \cdots & \frac{i!}{(i-k)!} T_j^{i-k} & \cdots \end{bmatrix} \begin{bmatrix} \vdots \\ p_{j,i} \\ \vdots \end{bmatrix} = \begin{bmatrix} x_{0,j}^{(k)} \\ x_{T,j}^{(k)} \end{bmatrix}$$
$$\Rightarrow \mathbf{A}_j \mathbf{p}_j = \mathbf{d}_j$$

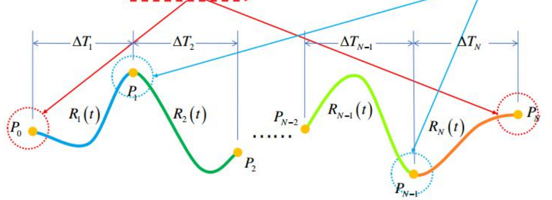
- Continuity constraint between two segments
  - Ensures continuity between trajectory segments when no specific derivatives are given

$$f_j^{(k)}(T_j) = f_{j+1}^{(k)}(T_j)$$
$$\Rightarrow \sum_{i\ge k} \frac{i!}{(i-k)!} T_j^{i-k} p_{j,i} - \sum_{l\ge k} \frac{l!}{(l-k)!} T_j^{l-k} p_{j+1,l} = 0$$
$$\Rightarrow \begin{bmatrix} \cdots & \frac{i!}{(i-k)!} T_j^{i-k} & \cdots & -\frac{l!}{(l-k)!} T_j^{l-k} & \cdots \end{bmatrix} \begin{bmatrix} \vdots \\ p_{j,i} \\ \vdots \\ p_{j+1,l} \\ \vdots \end{bmatrix} = 0$$
$$\Rightarrow \begin{bmatrix} \mathbf{A}_j & -\mathbf{A}_{j+1} \end{bmatrix} \begin{bmatrix} \mathbf{p}_j \\ \mathbf{p}_{j+1} \end{bmatrix} = 0$$

这里主要根据 Derivative constraints 和 Continuity constraints 构建方程，最终整理成 $\mathbf{A}_{eq}$ 和 $\mathbf{b}_{eq}$

Constraints:

- Derivative constraints: $\begin{cases} f_j^{(k)}(T_{j-1}) = x_{0,j}^{(k)} \\ f_j^{(k)}(T_j) = x_{T,j}^{(k)} \end{cases}$
- Continuity constraints: $f_j^{(k)}(T_j) = f_{j+1}^{(k)}(T_j)$



HW 1.1

```matlab
%#####################################################
% p,v,a,j constraint in start,
Aeq_start = zeros(4, n_all_poly);
T = 0;
for k = 0 : 3 % p,v,a,j
    for i = k : n_order % i >= k
        Aeq_start(k+1,i+1) = factorial(i)/factorial(i-k)*(T^(i-k));
    end
end
beq_start = start_cond';

%#####################################################
% p,v,a,j constraint in end
Aeq_end = zeros(4, n_all_poly);
T = ts(end);
idx = (n_seg-1)*(n_order+1);
for k = 0 : 3 % p,v,a,j
    for i = k : n_order % i >= k
        Aeq_end(k+1,idx+i+1) = factorial(i)/factorial(i-k)*(T^(i-k));
    end
end
beq_end = end_cond';

%#####################################################
% position constrain in all middle waypoints
Aeq_wp = zeros(n_seg-1, n_all_poly);
for n = 0 : n_seg-1-1
    T = ts(n+1);
    idx = (n)*(n_order+1);
    for i = 0 : n_order
        Aeq_wp(n+1, idx+i+1) = T^i;
    end
end
beq_wp = waypoints(2:end-1);
```

- Derivative constraint for one polynomial segment
  - Also models waypoint constraint (0^{th} order derivative)

$$f_j^{(k)}(T_j) = x_j^{(k)}$$

$$\Rightarrow \sum_{i \geq k} \frac{i!}{(i-k)!} T_j^{i-k} p_{j,i} = x_{T,j}^{(k)}$$

$$\Rightarrow \begin{bmatrix} \cdots & \frac{i!}{(i-k)!} T_j^{i-k} & \cdots \end{bmatrix} \begin{bmatrix} \vdots \\ p_{j,i} \\ \vdots \end{bmatrix} = x_{T,j}^{(k)}$$

$$\Rightarrow \begin{bmatrix} \cdots & \frac{i!}{(i-k)!} T_{j-1}^{i-k} & \cdots \\ \cdots & \frac{i!}{(i-k)!} T_j^{i-k} & \cdots \end{bmatrix} \begin{bmatrix} \vdots \\ p_{j,i} \\ \vdots \end{bmatrix} = \begin{bmatrix} x_{0,j}^{(k)} \\ x_{T,j}^{(k)} \end{bmatrix}$$

$$\Rightarrow \mathbf{A}_j \mathbf{p}_j = \mathbf{d}_j$$

HW 1.1

深蓝学院
shenlanxueyuan.com

```matlab
%#####################################################
% position continuity constrain between each 2 segments
Aeq_con_p = zeros(n_seg-1, n_all_poly);
beq_con_p = zeros(n_seg-1, 1);
% STEP 2.4: write expression of Aeq_con_p and beq_con_p
k = 0; % k = 0,1,2,3
for n = 0 : n_seg-1-1
    T = ts(n+1);
    idx = (n)*(n_order+1);
    for i = k : n_order
        Aeq_con_p(n+1, idx+i+1) = factorial(i)/factorial(i-k)*(T^(i-k));
    end
    T = 0;
    idx = (n+1)*(n_order+1);
    for i = k : n_order
        Aeq_con_p(n+1, idx+i+1) = -factorial(i)/factorial(i-k)*(T^(i-k));
    end
end
```

```matlab
for i=0:n_seg-1
    %#####################################################
    % STEP 3: get the coefficients of i-th segment of both x-axis and
    idx = (i*n_poly_perseg+1) : ((i+1)*n_poly_perseg);
    Pxi = flipud(poly_coef_x(idx));
    Pyi = flipud(poly_coef_y(idx));
    for t = 0:tstep:ts(i+1)
        X_n(k)  = polyval(Pxi, t);
        Y_n(k)  = polyval(Pyi, t);
        k = k + 1;
    end
end
```

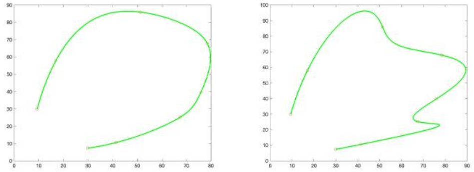
- Continuity constraint between two segments
  - Ensures continuity between trajectory segments when no specific derivatives are given

$$f_j^{(k)}(T_j) = f_{j+1}^{(k)}(T_j)$$

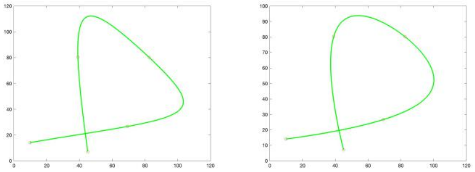
$$\Rightarrow \sum_{i\geq k} \frac{i!}{(i-k)!} T_j^{i-k} p_{j,i} - \sum_{l\geq k} \frac{l!}{(l-k)!} T_j^{l-k} p_{j+1,l} = 0$$

$$\Rightarrow \begin{bmatrix} \cdots & \frac{i!}{(i-k)!} T_j^{i-k} & \cdots & -\frac{l!}{(l-k)!} T_j^{l-k} & \cdots \end{bmatrix} \begin{bmatrix} \vdots \\ p_{j,i} \\ \vdots \\ p_{j+1,l} \\ \vdots \end{bmatrix} = 0$$

$$\Rightarrow \begin{bmatrix} \mathbf{A}_j & -\mathbf{A}_{j+1} \end{bmatrix} \begin{bmatrix} \mathbf{p}_j \\ \mathbf{p}_{j+1} \end{bmatrix} = 0$$



运行截图(左：时间间隔固定，右：时间间隔按距离划分)



运行截图(左：时间间隔固定，右：时间间隔按距离划分)

HW 1.2

- We have $\boldsymbol{M}\mathbf{p}=\mathbf{d}$, where $\boldsymbol{M}$ is a mapping matrix that maps polynomial coefficients to derivatives
- Use a selection matrix $\mathbf{C}$ to separate free $(\mathbf{d}_P)$ and constrained $(\mathbf{d}_F)$ variables
  - Free variables : derivatives unspecified, only enforced by continuity constraints

$$\begin{bmatrix} \mathbf{d}_1 \\ \vdots \\ \mathbf{d}_M \end{bmatrix} = \mathbf{C}^T \begin{bmatrix} \mathbf{d}_F \\ \mathbf{d}_P \end{bmatrix} \qquad J = \begin{bmatrix} \mathbf{d}_F \\ \mathbf{d}_P \end{bmatrix}^T \underbrace{\mathbf{C}\boldsymbol{M}^{-T}\mathbf{Q}\boldsymbol{M}^{-1}\mathbf{C}^T}_{\mathbf{R}} \begin{bmatrix} \mathbf{d}_F \\ \mathbf{d}_P \end{bmatrix} = \begin{bmatrix} \mathbf{d}_F \\ \mathbf{d}_P \end{bmatrix}^T \begin{bmatrix} \mathbf{R}_{FF} & \mathbf{R}_{FP} \\ \mathbf{R}_{PF} & \mathbf{R}_{PP} \end{bmatrix} \begin{bmatrix} \mathbf{d}_F \\ \mathbf{d}_P \end{bmatrix}$$

- Turned into an unconstrained quadratic programming that can be solved in closed form:

$$J = \mathbf{d}_F^T \mathbf{R}_{FF} \mathbf{d}_F + \mathbf{d}_F^T \mathbf{R}_{FP} \mathbf{d}_P + \mathbf{d}_P^T \mathbf{R}_{PF} \mathbf{d}_F + \mathbf{d}_P^T \mathbf{R}_{PP} \mathbf{d}_P$$

$$\mathbf{d}_P^* = -\mathbf{R}_{PP}^{-1} \mathbf{R}_{FP}^T \mathbf{d}_F$$

HW 1.2

getM()

```
M = [];
for n = 1:n_seg
    M_k = [];
    %#####################################################
    % STEP 1.1: calculate M_k of the k-th segment
    T = 0;
    for k = 0 : 3 % p,v,a,j at t0
        for i = k : n_order
            M_k(k+1,i+1) = factorial(i)/factorial(i-k)*(T^(i-k));
        end
    end
    T = ts(n);
    for k = 0 : 3 % p,v,a,j at T
        for i = k : n_order
            M_k(4+k+1,i+1) = factorial(i)/factorial(i-k)*(T^(i-k));
        end
    end
    M = blkdiag(M, M_k);
end
```

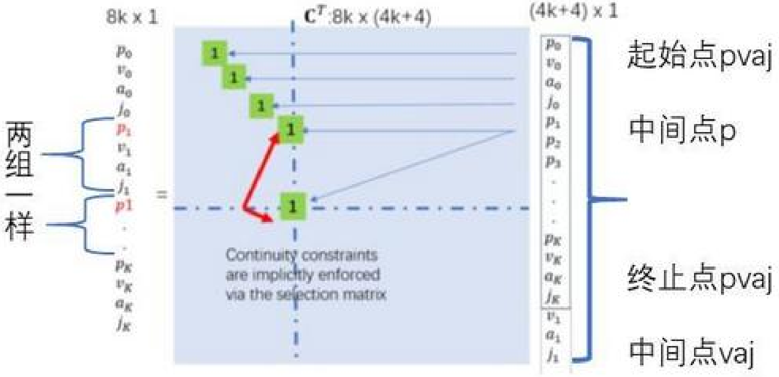
$$\boldsymbol{M}_{\mathrm{j}}\mathbf{p}_j = \mathbf{d}_j$$

$$f_j^{(k)}(T_j) = x_j^{(k)}$$
$$\Rightarrow \sum_{i \geq k} \frac{i!}{(i-k)!} T_j^{i-k} p_{j,i} = x_{T,j}^{(k)}$$

HW 1.2

getCt()

```
%% Check Ct
syms p0 v0 a0 j0 p1 p2 p3 v3 a3 j3 a1 v1 j1 a2 v2 j2 real
dfdp = [p0 v0 a0 j0 p1 p2 p3 v3 a3 j3, v1 a1 j1, v2 a2 j2]';
d = [p0 v0 a0 j0 p1 v1 a1 j1 p1 v1 a1 j1 p2 v2 a2 j2 p2 v2 a2 j2 p3 v3 a3 j3]';
Ct = getCt(3, 7);
dd = Ct*dfdp;
assert(isequaln(d,dd))
```




```
%#####################################################
% STEP 2.1: finish the expression of Ct
% fixed derivatives
% p0,v0,a0,j0
for i = 1 : 4
    Ct(i,i) = 1;
end
% p1,p2,...,p(n-1)
idx_df = size(Ct,2);
for i = 1 : n_seg - 1
    idx = i*8;
    Ct(idx-4+1, idx_df+i) = 1;
    Ct(idx+1, idx_df+i) = 1;
end
% pn,vn,a0,jn
idx_df = size(Ct,2);
for i = 1 : 4
    idx = n_seg*8-4;
    Ct(idx+i, idx_df+i) = 1;
end

% free derivatives
% v1,a1,j1,...,v(n-1),a(n-1),j(n-1)
idx_df = size(Ct,2);
for i = 1 : n_seg-1
    idx = i*8;
    idxf2 = idx_df + (i-1)*3;
    Ct(idx-4+2, idxf2+1) = 1;
    Ct(idx-4+3, idxf2+2) = 1;
    Ct(idx-4+4, idxf2+3) = 1;
    Ct(idx+2, idxf2+1) = 1;
    Ct(idx+3, idxf2+2) = 1;
    Ct(idx+4, idxf2+3) = 1;
end
```

HW 2

```
double accInv = 1.0 / _Acc;
double velInv = 1.0 / _Vel;

// 加速时间与距离，减速需要耗费同样资源
double accTime = _Vel * accInv;          // v = v0 + at, v0 = 0
double accDist = 0.5 * _Vel * accTime;   // d = 0.5 * (v0 + v) * t, v0 = 0
double accTime2 = 2.0 * accTime;
double accDist2 = 2.0 * accDist;     // 包含加速和减速的总距离

for(int i = 0; i < Path.rows() - 1; i++)
{
  double t = 0.0;
  double dist = (Path.row(i + 1) - Path.row(i)).norm();

  if(dist <= accDist2)
    t = 2.0 * std::sqrt(dist * accInv);      // 0.5 * d = v0 * t + 0.5 * a * t * t, v0 = 0
  else
    t = accTime2 + (dist - accDist2) * velInv;

  time(i) = t;
}
```

- OOPQ
- OSQP
- qpOASES

感谢各位聆听！

Thanks for Listening