深蓝学院
shenlanxueyuan.com

# 移动机器人运动规划
# ——第五章作业讲解

主讲人 武庆斌

# 作业目录

深蓝学院
shenlanxueyuan.com

- 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

# MATLAB作业

## Minimum Snap Trajectory Generation

- **多项式轨迹描述**

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

$$f_k(t)=\sum_{i=0}^{N} p_{k,i}t^i \qquad 0\le t\le \Delta T_k$$

- **轨迹约束条件**

  Derivative constraints: 规定起始状态和终止状态p,v,a,j、中间点p

  Continuity constraints: 规定前后两端轨迹交点处p,v,a,j连续

- **典型的QP问题**

  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MinimumSnapQP-Solver()和函数MinimumSnapCloseformSolver(),分别表示QP解法和闭式解法

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

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

# MATLAB作业

深蓝学院 shenlanxueyuan.com

## Minimum Snap ---- getQ()分析

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

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

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

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

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

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

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

```
for i=4:n_order
    for j=4:n_order
        Q_k(i+1, j+1)=factorial(i)/factorial(i-4)*factorial(j)/factorial(j-4)/(i+j-7)*ts(k)^(i+j-7);
    end
end
Q = blkdiag(Q, Q_k);
```

$$f(t)=[p_0,p_1,\cdots,p_7]\cdot[t^0,t^1,\cdots,t^7]^T$$

$$\Rightarrow \begin{cases} f(t)=P\cdot[t^0,t^1,t^2,t^3,t^4,t^5,t^6,t^7]^T \\ f^{(1)}(t)=P\cdot[0,1,2t,3t^2,4t^3,5t^4,6t^5,7t^6]^T \\ f^{(2)}(t)=P\cdot[0,0,2,6t,12t^2,20t^3,30t^4,42t^5]^T \\ f^{(3)}(t)=P\cdot[0,0,0,6,24t,60t^2,120t^3,210t^4]^T \\ \boxed{f^{(4)}(t)}=P\cdot[0,0,0,0,24,120t,360t^2,820t^3]^T \end{cases}$$

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

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

# MATLAB作业

## Minimum Snap ---- getQ()分析

$$f^{(4)}(t) = P\cdot[0,0,0,0,24,120t,360t^2,820t^3]^T$$

$$\Rightarrow (f^{(4)}(t))^2 = P\cdot\begin{bmatrix}0\\0\\0\\0\\24\\120t\\360t^2\\820t^3\end{bmatrix}\cdot[0,0,0,0,24,120t,360t^2,820t^3]\cdot P^T$$

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

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

积分

两段轨迹$Q = \begin{bmatrix}Q_1 & 0\\ 0 & Q_2\end{bmatrix}$矩阵示意图：

```
Q =

1.0e+12 *
```

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0.0000 | 0.0000 | 0.0000 | 0.0001 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0.0000 | 0.0000 | 0.0002 | 0.0053 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0.0000 | 0.0002 | 0.0069 | 0.1618 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0.0001 | 0.0053 | 0.1618 | 3.9312 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0.0000 | 0.0000 | 0.0000 | 0.0001 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0.0000 | 0.0000 | 0.0003 | 0.0071 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0.0000 | 0.0003 | 0.0091 | 0.2273 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0.0001 | 0.0071 | 0.2273 | 5.8438 |

# MATLAB作业

深蓝学院 shenlanxueyuan.com

## Minimum Snap ---- getAbeq()分析

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

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

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

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

这里主要根据导数约束与连续性约束构建方程，最终整理成$A_{eq}$与$b_{eq}$

# MATLAB作业

深蓝学院
shenlanxueyuan.com

## Minimum Snap ---- getAbeq()分析

```
%#################################################
% p,v,a,j constraint in start,
Aeq_start = zeros(4, n_all_poly);
beq_start = zeros(4, 1);
% STEP 2.1: write expression of Aeq_start and beq_start
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);
beq_end = zeros(4, 1);
% STEP 2.2: write expression of Aeq_end and beq_end
T=ts(end);
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_end=end_cond';
%#################################################
% position constrain in all middle waypoints
Aeq_wp = zeros(n_seg-1, n_all_poly);
beq_wp = zeros(n_seg-1, 1);
% STEP 2.3: write expression of Aeq_wp and beq_wp
%对每段轨迹的起点进行约束
for i=1:n_seg-1
    Aeq_wp(i,i*8+1)=1;
    beq_wp(i,1)=waypoints(i+1);
end
```

起点状态约束

终点状态约束

中间点位置约束

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

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

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

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

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

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

# MATLAB作业

## Minimum Snap ---- getAbeq()分析

```
%#############################################################
% 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
% STEP 3: get the coefficients of i-th segment of both x-axis
% and y-axis
Pxi = [];
Pyi = [];
Pxi=flipud(poly_coef_x(i*8+1:i*8+8));
Pyi=flipud(poly_coef_y(i*8+1:i*8+8));
for t = 0:tstep:ts(i+1)
    X_n(k)  = polyval(Pxi, t);
    Y_n(k)  = polyval(Pyi, t);
    k = k + 1;
end
```

K=0,1,2,3分别表示位置、速度、加速度、加加速度

连续性约束

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

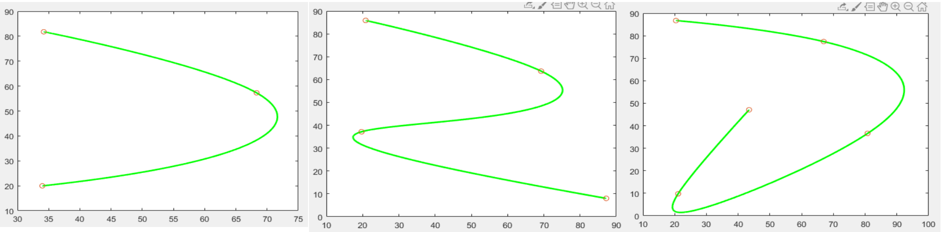
$$f_j^{(k)}(T_j) = f_{j+1}^{(k)}(T_j)$$

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

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

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

可视化



# MATLAB作业

## Minimum Snap ---- getM()分析（闭式解法）

- We have $\boldsymbol{M}_{\mathrm{j}}\mathbf{p}_j = \mathbf{d}_j$, where $\boldsymbol{M}_{\mathrm{j}}$ is a mapping matrix that maps polynomial coefficients to derivatives

$$J = \begin{bmatrix} \mathbf{p}_1 \\ \vdots \\ \mathbf{p}_M \end{bmatrix}^T \begin{bmatrix} \mathbf{Q}_1 & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & \mathbf{Q}_M \end{bmatrix} \begin{bmatrix} \mathbf{p}_1 \\ \vdots \\ \mathbf{p}_M \end{bmatrix} \qquad J = \begin{bmatrix} \mathbf{d}_1 \\ \vdots \\ \mathbf{d}_M \end{bmatrix}^T \boxed{\begin{bmatrix} \boldsymbol{M}_1 & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & \boldsymbol{M}_M \end{bmatrix}}^{-T} \begin{bmatrix} \mathbf{Q}_1 & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & \mathbf{Q}_M \end{bmatrix} \boxed{\begin{bmatrix} \boldsymbol{M}_1 & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & \boldsymbol{M}_M \end{bmatrix}}^{-1} \begin{bmatrix} \mathbf{d}_1 \\ \vdots \\ \mathbf{d}_M \end{bmatrix}$$

这里主要使用映射矩阵M将多项式系数映射为导数，提高求解数值稳定性。

# MATLAB作业

深蓝学院
shenlanxueyuan.com

## Minimum Snap ---- getM()分析（闭式解法）

```
M = [];
for n = 1:n_seg
    M_k = zeros(n_order+1,n_order+1);
    %#################################################
    % STEP 1.1: calculate M_k of the k-th segment
    T=0;
    for k=0:3
        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
        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
```

实现

$$\mathbf{M}_j \mathbf{p}_j = \mathbf{d}_j$$

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

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

凝练

定义$P_{total} = \begin{pmatrix} P_1 \\ P_2 \\ \vdots \\ P_M \end{pmatrix}^T = \begin{bmatrix} p_{0,1} \\ p_{1,1} \\ \cdots \\ p_{7,1} \\ \cdot \\ \cdot \\ \cdot \\ p_{0,K} \\ p_{1,K} \\ \cdots \\ p_{7,K} \end{bmatrix}$，$d_{total} = \begin{bmatrix} d_1 \\ d_2 \\ \cdot \\ \cdot \\ \cdot \\ d_K \end{bmatrix} = \begin{bmatrix} p_0 \\ v_0 \\ a_0 \\ j_0 \\ p_1 \\ v_1 \\ a_1 \\ j_1 \\ \cdots \\ p_k \\ v_k \\ a_k \\ j_K \end{bmatrix}$，由$M_j p_j = d_j$，可得

$$p_{0,j} = p_{j-1}$$
$$p_{1,j} = v_{j-1}$$
$$2p_{2,j} = a_{j-1}$$
$$6p_{3,j} = j_{j-1}$$
$$p_{0,j} + p_{1,j}t_j + p_{2,j}t_j^2 + p_{3,j}t_j^3 + p_{4,j}t_j^4 + p_{5,j}t_j^5 + p_{6,j}t_j^6 + p_{7,j}t_j^7 = p_j$$
$$p_{1,j} + 2p_{2,j}t_j + 3p_{3,j}t_j^2 + 4p_{4,j}t_j^3 + 5p_{5,j}t_j^4 + 6p_{6,j}t_j^5 + 7p_{7,j}t_j^6 = v_j$$
$$2p_{2,j} + 6p_{3,j}t_j + 12p_{4,j}t_j^2 + 20p_{5,j}t_j^3 + 30p_{6,j}t_j^4 + 42p_{7,j}t_j^5 = a_j$$
$$6p_{3,j} + 24p_{4,j}t_j + 60p_{5,j}t_j^2 + 120p_{6,j}t_j^3 + 210p_{7,j}t_j^4 = j_j$$

# MATLAB作业

深蓝学院
shenlanxueyuan.com

## Minimum Snap ---- getCt()分析（闭式解法）

- 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

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

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

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

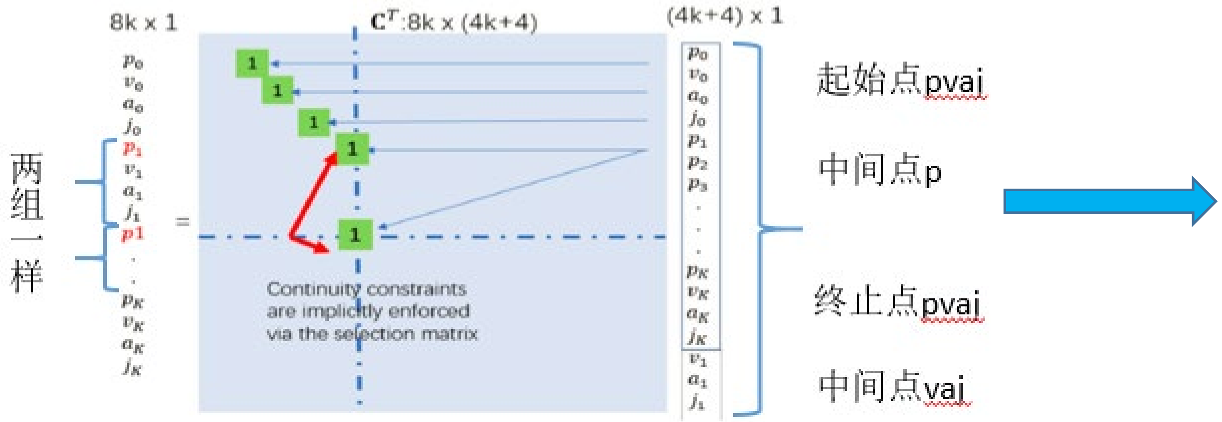
$$\mathbf{d}_P^* = -\mathbf{R}_{PP}^{-1}\mathbf{R}_{FP}^T\mathbf{d}_F$$

这里通过使用选择矩阵$C$分离自由变量$d_P$和固定变量$d_F$，转化为无约束二次规划以闭式求解。

# MATLAB作业

## Minimum Snap ---- getCt()分析（闭式解法）

选择矩阵Ct构造规则：

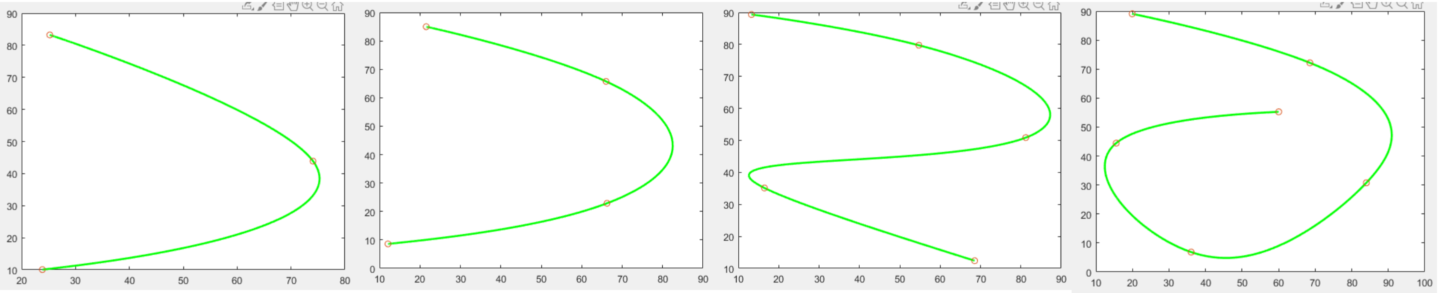



```
%#################################################################
% STEP 2.1: finish the expression of Ct
%fixed derivatives
%起点状态: p0,v0,a0,j0
for i=1:4
    Ct(i,i)=1;
end
%中间点位置: p1,p2,...,p(n-1)
idx_df = size(Ct,2);
for i=1:n_seg-1
    idx=8*i;
    Ct(idx-4+1,idx_df+i)=1;
    Ct(idx+1,idx_df+i)=1;
end
%终点状态: pn,vn,an,jn
idx_df = size(Ct,2);
for i=1:4
    idx=(n_seg-1)*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=8*i;
    idx_df2=idx_df+(i-1)*3;
    Ct(idx-4+2,idx_df2+1)=1;
    Ct(idx-4+3,idx_df2+2)=1;
    Ct(idx-4+4,idx_df2+3)=1;
    Ct(idx+2,idx_df2+1)=1;
    Ct(idx+3,idx_df2+2)=1;
    Ct(idx+4,idx_df2+3)=1;
end
```

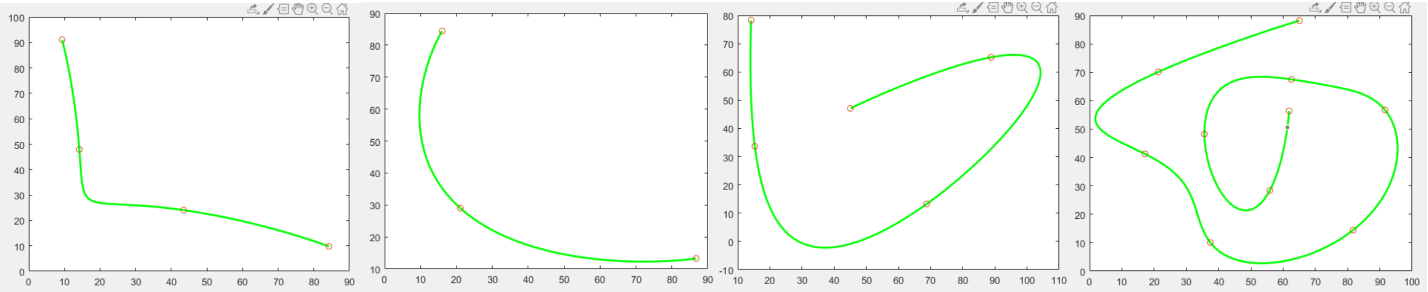
# MATLAB作业

## Minimum Snap ---- 效果展示

- MinimumSnapQPSolver()求解结果：



- MinimumSnapCloseformSolver()求解结果：

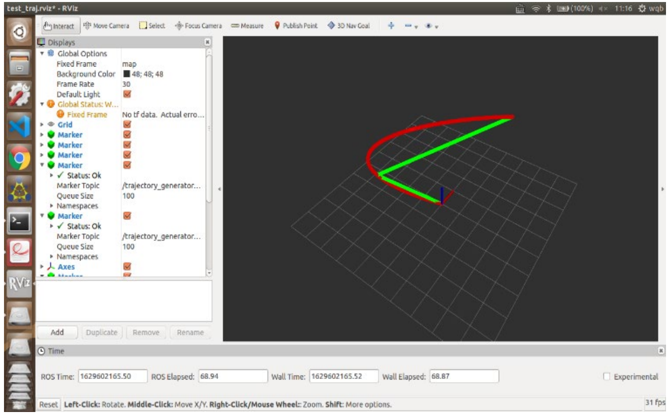


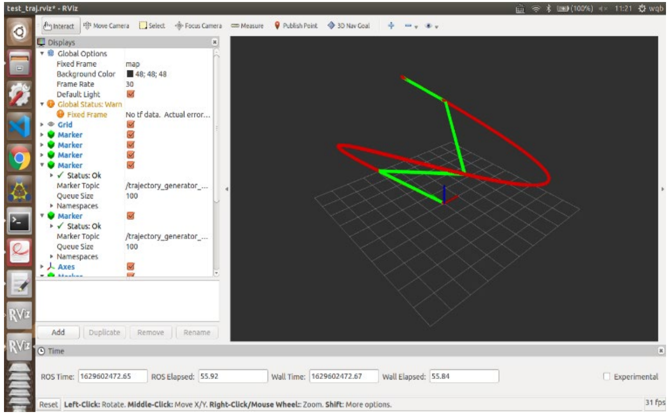
# ROS作业



## Minimum Snap ---- 效果展示

ROS环境下仍旧可以按照求解器与闭式求解两种方法进行，大家可以按照所述Minimum Snap 方法进行实现，下面不在赘述具体过程，直接给出运行效果。





深蓝学院
shenlanxueyuan.com

# 感谢各位聆听！

Thanks for Listening