Homework 5: Minimum Snap Trajectory Generation

Huan Chen

1. Matlab Part

Polynomial coefficients are organized as $\boldsymbol{p} = [p_7, p_6, p_5, p_4, p_3, p_2, p_1, p_0]^T$. The sparsity patter of matrix Q is showed in Figure 1. The minimum snap trajectory generation results are showed in Figure 2.

Closed-form solution: the shape of mapping matrix is showed in Figure 3, result is presented in Figure 4.

2. Main Takeaways

Previous three lectures are about front-end path finding. Here we start to work on back-end optimization-based trajectory generation, using high-order polynomials. These contents are more than fundamental in path planning, more details see the slides.

1. What are required to generate trajectory.
   - Boundary condition: start, goal positions (orientations)
   - Intermediate condition: waypoint positions (orientations), wit A*, RRT*, etc.
   - Smoothness criteria (this homework is to minimize snap)
2. Quadrotor modelling and differential flatness, $\boldsymbol{\sigma} = [x, y, z, \phi]^T$.
3. Multi-segment minimum snap trajectory generation, with QP solver. For M segments.
   - Polynomial functions for each segment.

$$f(t) = \begin{cases} f_1(t) := \sum_{i=0}^{N} p_{1,i} t^i, & T_0 \le t \le T_1 \\ f_2(t) := \sum_{i=0}^{N} p_{2,i} t^i, & T_1 \le t \le T_2 \\ \vdots & \\ f_M(t) := \sum_{i=0}^{N} p_{M,i} t^i, & T_{M-1} \le t \le T_M \end{cases}$$

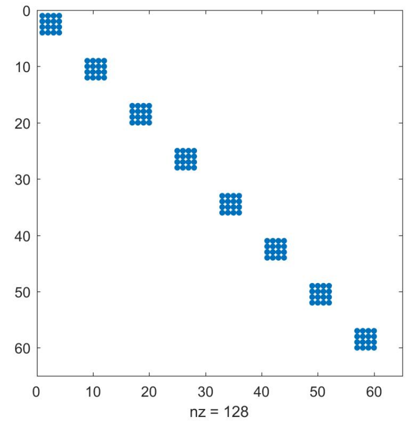

Figure 1: The shape of Q.

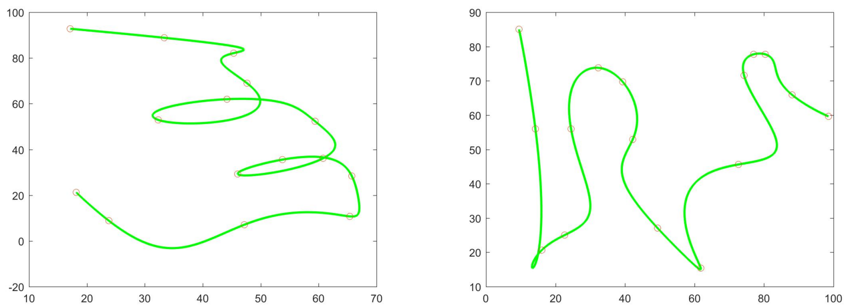

Figure 2: The result of minimum snap trajectory generation.

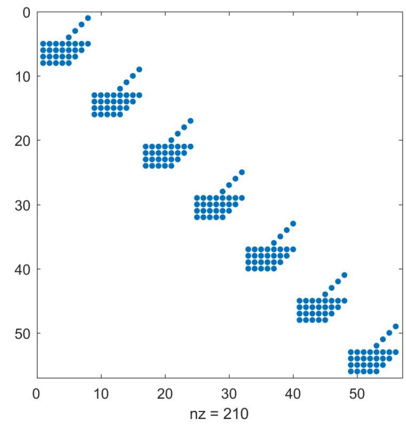

Figure 3: The shape of M.

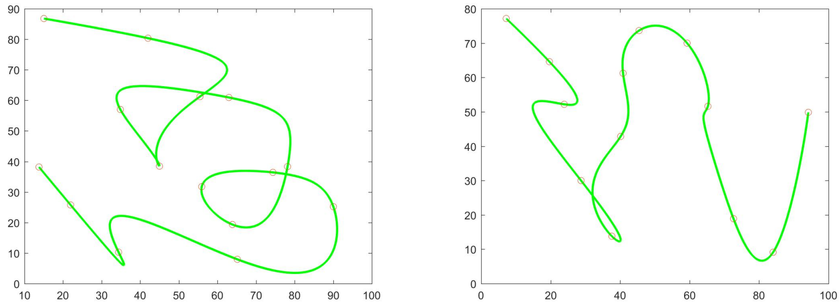

Figure 4: The result of closed-form solution.

- Cost function, e.g. minimize snap.

$$
\begin{aligned}
\rightarrow J(T) &= \int_{T_{j-1}}^{T} {}_j (f^{(4)}(t))^2 dt \\
&= \sum_{i \geq 4, l \geq 4} \frac{i(i-1)(i-2)(i-3)l(l-1)(l-2)(l-3)}{i+l-7} (T_j^{i+l-7} - T_{j-1}^{i+l-7}) p_i p_l \\
&= \begin{bmatrix} \vdots \\ p_i \\ \vdots \end{bmatrix} \begin{bmatrix} & & \vdots & & \\ \dots & & \frac{i(i-1)(i-2)(i-3)l(l-1)(l-2)(l-3)}{i+l-7} T^{i+l-7} & & \dots \\ & & \vdots & & \end{bmatrix} \begin{bmatrix} \vdots \\ p_l \\ \vdots \end{bmatrix}
\end{aligned}
$$

$$\rightarrow J_j(T) = \boldsymbol{p}_j^T \boldsymbol{Q}_j \boldsymbol{p}_j$$

- Constraints: derivative constraint for all the waypoints, continuity constraint between two segments.
- Constrained QP formulation, convex optimization.

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

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

4. Closed-from solution to minimum snap.
   - More numerically stable.
   - Use mapping matrix $\boldsymbol{M}_j$ maps polynomial coefficients to derivatives, $\boldsymbol{M}_j \boldsymbol{p}_j = \boldsymbol{d}_j$.
   - Use selection matrix $\boldsymbol{C}$ to separate free ($\boldsymbol{d}_P$) and constrained ($\boldsymbol{d}_F$)
5. Numerical stability.
   - Use relative timeline.

- Solve 3 axis independently is stable and faster.
- Time allocation significantly affect the final trajectory.