MINCO 轨迹类航点硬约束处理

郑一品

2024 年 10 月 9 日

1　2024 年 10 月 9 日

任务

1、 GCOPTER 源代码

2、 Geometrically Constrained Trajectory Optimization for Multicopters 论文研究

3、 Polynomial-based Online Planning for Autonomous Drone Racing in Dynamic Environments 对比 **2** 的提升

1.1　Geometrically Constrained Trajectory Optimization for Multicopters(T-RO2022)

1.1.1　凸多面体几何约束的处理

对于凸多面体，其表示形式有以下两种；

（1）**$\mathcal{V}$-多面体**: $\mathbb{R}^d$ 中的有限点集 $X = \{x^1, \ldots, x^n\}$ 所构成的凸包，

$$P^{\mathcal{V}} = \mathrm{conv}(X) := \{\sum_{i=1}^{n} \lambda_i x^i \Big| \lambda_1, \ldots, \lambda_n \geq 0, \sum_{i=1}^{n} \lambda_i = 1\}. \tag{1}$$

（2）**$\mathcal{H}$-多面体**: 有限线性不等式方程组的解集，

$$P^{\mathcal{H}} = P(A, b) := \{x \in \mathbb{R}^d \mid a_i^T x \leq b_i \text{ for } 1 \leq i \leq m\}. \tag{2}$$

其中，$A \in \mathbb{R}^{m \times b}$ 是一个以 a_i^T 为行的实矩阵，$b \in \mathbb{R}^b$ 是一个以 b_i 为项的实向量，即

$$P^{\mathcal{H}} = \{x \in \mathbb{R}^n | \mathbf{A}x \preceq b\}. \tag{3}$$

欧式空间到球体的映射: 考虑 m 维空间里以 o 为中心、r 为半径的低维球约束，有如下形式**:**

$$\mathcal{P}^{\mathcal{B}} = \{x \in \mathbb{R}^n \mid \|x - o\|_2 \leq r\}. \tag{4}$$

利用光滑的满射将 $\mathbb{R}^n$ 映射到 $\mathcal{P}^{\mathcal{B}}$，便可以在 $\mathbb{R}^n$ 的代理变量上进行优化隐式地满足 $\mathcal{P}^{\mathcal{B}}$ 的约束。

映射 1：欧氏空间 $\mathbb{R}^n$ → 球体

$$x = o + \frac{2r\xi}{\xi^{\mathrm{T}}\xi + 1} \in \mathcal{P}^{\mathcal{B}}, \forall \xi \in \mathbb{R}^n. \tag{5}$$

其中，引入一个代理变量 ξ，以 ξ 指代 $\mathbf{q}$ 的无约束代理变量。通过计算(5)的 Jacobian 即可得到梯度

$$\frac{\partial J}{\partial \xi_i} = \frac{2r_i g_i}{\xi_i^{\mathrm{T}}\xi_i + 1} - \frac{4r_i(\xi_i^{\mathrm{T}} g_i)\xi_i}{(\xi_i^{\mathrm{T}}\xi_i + 1)^2}. \tag{6}$$

其中，g_i 代表$\partial J/\partial \mathbf{q}$ 中的第i 个元素$\partial J/\partial q_i$。

球体到凸多面体的映射： 考虑凸多面体 $\mathcal{P}^{\mathcal{H}}$, 采用多面体的 $\mathcal{V}$-表示，其顶点数目为 $\hat{n}+1$, 分别为 $(v_0, v_1, \cdots, v_{\hat{n}})$，其在重心坐标下等价于一个标准的 $\hat{n}$-单纯形 $\mathcal{P}_{\omega}^{\mathcal{H}}$，在平方变量代换下，$\mathcal{P}_{\omega}^{\mathcal{H}}$ 等价于一个 $\hat{n}$ 维球。

映射 2：单纯形 → 凸多面体

$$q = v_0 + \widehat{\mathbf{V}}\omega \tag{7}$$

其中，$q \in \mathcal{P}^{\mathcal{H}}$，$\widehat{\mathbf{V}} = (\hat{v_1}, \hat{v_2}, \cdots, \hat{v_{\hat{n}}})$，$\hat{v_i} = v_i - v_0$，$\omega = (\omega_1, \cdots, \omega_{\hat{n}})^T$。由如下单纯形映射到凸多面体(3)：

$$\mathcal{P}_w^{\mathcal{H}} = \left\{ w \in \mathbb{R}^{\hat{n}} \middle| w \succeq 0, \|w\|_1 \leq 1 \right\}. \tag{8}$$

映射 3：球体 → 单纯形

$$\omega = [x]^2 \tag{9}$$

其中，$[\cdot]^2$ $\mathbb{R}^{\hat{n}} \mapsto \mathbb{R}^{\hat{n}}$ 表示逐元素平方的操作。由如下球体映射到单纯形(8)：

$$\mathcal{B}^{\hat{n}} = \left\{ x \in \mathbb{R}^{\hat{n}} \middle| \|x\|_2 \leq 1 \right\}. \tag{10}$$

总映射：欧氏空间 $\mathbb{R}^n$ → 凸多面体： 对(5)、(7)、(9)复合操作，将欧式空间映射到 $\mathcal{P}^{\mathcal{H}}$ 上，定义为

$$q = v_0 + \frac{4\hat{\mathbf{V}}[\xi]^2}{(\xi^{\mathrm{T}}\xi + 1)^2} \in \mathcal{P}^{\mathcal{H}}, \forall \xi \in \mathbb{R}^{\hat{n}}. \tag{11}$$

同样引入一个无约束代理变量 ξ，使得任意的 $\xi \in \mathbb{R}^{\hat{n}}$ 均有对应的 $q \in \mathcal{P}^{\mathcal{H}}$，梯度

$$\frac{\partial J}{\partial \xi_i} = \frac{8\xi_i \circ \hat{\mathrm{V}}^{\mathrm{T}} g_i}{(\xi_i^{\mathrm{T}}\xi_i + 1)^2} - \frac{16 g_i^{\mathrm{T}}\hat{\mathrm{V}}[\xi_i]^2}{(\xi_i^{\mathrm{T}}\xi_i + 1)^3}\xi_i. \tag{12}$$

其中，$\circ$ 为 Hadamard 乘积。

疑问：映射 1、2 和 3 不是一一对应的会对优化产生影响吗？

目前解释：对于任意优化变量 ξ 均有对应的 $q \in \mathcal{P}^{\mathcal{H}}$

1.1.2 时间约束的处理

MINCO 轨迹的时间向量 $\mathbf{T} \in \mathbb{R}_{>0}^{M}$ 是带约束的，为了便于求解将有约束时间变量映射至无约束变量 τ 中。对于不严格要求轨迹时间的情况，可采取如下微分同胚变换

$$\mathbf{T} = e^{[\tau]} \tag{13}$$

1.1.3 连续时间约束优化的惩罚泛函

对于轨迹 $p : [0, T] \mapsto \mathbb{R}^m$，定义如下惩罚泛函

$$I_{\mathcal{G}}^{k}[p] = \int_0^T \max\left[\mathcal{G}(p(t), \ldots, p^{(s)}(t)), \mathbf{0}\right]^k \mathrm{d}t, \tag{14}$$

其中 $k \in \mathbb{R}_{>0}$ 并且 $\max[\cdot, 0]^k$ 代表逐元素最大操作和逐元素幂函数的复合操作。 当参数 $k = 1$ 时，$I_{\mathcal{G}}^{k}[p]$ 是非光滑但精确的，构造如下光滑近似

$$\psi_\mu(x) = \begin{cases} 0 & \text{if} x \le 0, \\ (\mu - x/2)\,(x/\mu)^3 & \text{if} 0 < x < \mu, \\ x - \mu/2 & \text{if} x \ge \mu. \end{cases} \tag{15}$$

该光滑近似或 $I_{\mathcal{G}}^{3}[p]$ 均可用于惩罚泛函。

定义采样函数 $\mathcal{G}_\tau : \mathbb{R}^{2s\times m} \times \mathbb{R}_{>0} \times [0, 1] \mapsto \mathbb{R}^{n_g}$ 如下

$$\mathcal{G}_\tau(\mathbf{c}_i, T_i, \tau) = \mathcal{G}\left(\mathbf{c}_i^{\mathrm{T}}\beta(T_i \cdot \tau), \ldots, \mathbf{c}_i^{\mathrm{T}}\beta^{(s)}(T_i \cdot \tau)\right), \tag{16}$$

其中，τ 代表归一化时间。对 $I_{\mathcal{G}}[p]$ 的数值积分$I : \mathbb{R}^{2Ms\times m} \times \mathbb{R}_{>0}^{M} \mapsto \mathbb{R}_{>0}$ 可以通过采样函数的加权和来计算，即

$$I(\mathbf{c}, \mathbf{T}) = \sum_{i=1}^{M} \frac{T_i}{\kappa_i} \sum_{j=0}^{\kappa_i} \bar{\omega}_j \chi^{\mathrm{T}} \max[\mathcal{G}_\tau(\mathbf{c}_i, T_i, \frac{j}{\kappa_i}), \mathbf{0}]^k. \tag{17}$$

其中 κ_i 直接控制该数值积分的精度,χ 是惩罚向量的权重,并选取权重 $(\bar{\omega}_0, \bar{\omega}_1, \ldots, \bar{\omega}_{\kappa_i-1}, \bar{\omega}_{\kappa_i}) = (1/2, 1, \ldots, 1, 1/2)$。

1.1.4 MINCO 轨迹类（详细参考毕设论文）

假设多段轨迹的时间分配为固定常量，现在先只考虑边界条件约束求解 M 段轨迹的控制量最小化问题，有如下形式：

$$\min_{z(t)} \quad \int_{t_0}^{t_M} v(t)^{\mathrm{T}}\mathbf{W}v(t)\mathrm{d}t \tag{18a}$$

$$\text{s.t.} \quad z^{(s)}(t) = v(t), \forall t \in [t_0, t_M], \tag{18b}$$

$$z^{[s-1]}(t_0) = \bar{z}_o, z^{[s-1]}(t_M) = \bar{z}_f, \tag{18c}$$

$$z^{[d_i-1]}(t_i) = \bar{z}_i, 1 \le i < M, \tag{18d}$$

$$t_{i-1} < t_i, 1 \le i \le M \tag{18e}$$

最优性充要条件：当且仅当轨迹 $z^*(t)$ 满足以下全部条件时，$z^*(t)$ 是问题(18)的最优解，并且 $z^*(t)$ 存在且唯一。
（1）对于所有 $1 \le i \le M$，轨迹 $z*(t):[t_{i-1},t_i] \mapsto \mathbb{R}^m$ 是一个 $2s-1$ 次多项式；
（2）轨迹 $z^*(t)$ 满足轨迹的起终点约束和中间条件，即满足式(18c)和式(18d)；
（3）对于所有 $1 \le i \le M$，轨迹 $z^*(t)$ 在中间点 t_i 处满足 $\bar{d}_i-1$ 阶连续可微，其中 $\bar{d}_i = 2s-d_i$。

借助该最优性充要条件可以直接构造满足约束的唯一最优轨迹，甚至不需要对代价泛函本身进行任何处理，具体构造方法如下。

对于一个 m 维多段轨迹，其每一段都是一个 $2s-1$ 次多项式，表达式如下

$$p_i(t) = \mathbf{c}_i^{\mathrm{T}} \beta(t-t_{i-1}), t \in [t_{i-1}, t_i] \tag{19}$$

其中，$\mathbf{c}_i \in \mathbb{R}^{2s \times m}$ 是多项式的系数矩阵，$\beta(x) = (1, x, \ldots, x^{2s-1})^{\mathrm{T}} \in \mathbb{R}^{2s}$ 是多项式的基。将每一段轨迹中的多项式系数矩阵和时间堆成大矩阵，即

$$\mathbf{c} = \left(\mathbf{c}_1^{\mathrm{T}}, \ldots, \mathbf{c}_M^{\mathrm{T}}\right)^{\mathrm{T}} \in \mathbb{R}^{2Ms \times m}, \mathbf{T} = (T_1, \ldots, T_M)^{\mathrm{T}} \in \mathbb{R}_{>0}^{M} \tag{20}$$

其中，T_i 表示第 i 段多项式轨迹的时间。

对于每一个中间点，给定的 d_i-1 阶中间点状态可以产生 d_i 个给定值的约束方程，并且 $\bar{d}_i - 1 = 2s - d_i - 1$ 阶连续可微性可以产生 $2s-d_i$ 个约束方程。因此，每个中间点可以产生 $2s$ 个约束方程，具体如下

$$\begin{pmatrix} \mathbf{E}_i & \mathbf{F}_i \end{pmatrix} \begin{pmatrix} \mathbf{c}_i \\ \mathbf{c}_{i+1} \end{pmatrix} = \begin{pmatrix} \mathbf{D}_i \\ \mathbf{0}_{\bar{d}_i \times m} \end{pmatrix} \tag{21}$$

$$\mathbf{E}_i = (\beta(T_i), \ldots, \beta^{(d_i-1)}(T_i), \beta(T_i), \ldots, \beta^{(\bar{d}_i-1)}(T_i))^{\mathrm{T}} \in \mathbb{R}^{2s \times 2s} \tag{22}$$

$$\mathbf{F}_i = (\mathbf{0}, -\beta(0), \ldots, -\beta^{(\bar{d}_i-1)}(0))^{\mathrm{T}} \in \mathbb{R}^{2s \times 2s} \tag{23}$$

其中，$\mathbf{D}_i \in \mathbb{R}^{d_i \times 2s}$ 是中间条件(18d)给定的中间点导数值。
对于边界条件(18c)给定的起终点的 $s-1$ 阶导数信息，有

$$\mathbf{F}_0 = \left(\beta(0), \ldots, \beta^{(s-1)}(0)\right)^{\mathrm{T}} \in \mathbb{R}^{s \times 2s}, \tag{24}$$

$$\mathbf{E}_M = \left(\beta(T_M), \ldots, \beta^{(s-1)}(T_M)\right)^{\mathrm{T}} \in \mathbb{R}^{s \times 2s}. \tag{25}$$

综合考虑式(21)、(24)和(25)，可以得到以下线性方程组。

$$\mathbf{M}\mathbf{c}=\mathbf{b} \tag{26}$$

其中，$\mathbf{M}\in\mathbb{R}^{2Ms\times 2Ms}$ 和 $\mathbf{b}\in\mathbb{R}^{2Ms\times m}$ 为

$$\mathbf{M}=\begin{pmatrix}\mathbf{F}_0 & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0}\\ \mathbf{E}_1 & \mathbf{F}_1 & \mathbf{0} & \cdots & \mathbf{0}\\ \mathbf{0} & \mathbf{E}_2 & \mathbf{F}_2 & \cdots & \mathbf{0}\\ \vdots & \vdots & \vdots & \ddots & \vdots\\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{F}_{M-1}\\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{E}_M\end{pmatrix},\mathbf{b}=\begin{pmatrix}\mathbf{D}_0\\ \mathbf{D}_1\\ \mathbf{0}_{\bar{d}_1\times m}\\ \vdots\\ \mathbf{D}_{M-1}\\ \mathbf{0}_{\bar{d}_{M-1}\times m}\\ \mathbf{D}_M\end{pmatrix} \tag{27}$$

最优性充要条件的唯一性使矩阵 $\mathbf{M}$ 对于任意时间向量 $\mathbf{T}\in R^M_{>0}$ 都是非奇异矩阵。

1.1.5 MINCO 轨迹类梯度传导

选定中间点的 0 阶导数信息作为中间条件，规定轨迹的航点向量 $\mathbf{q}=(q_1,\ldots,q_{M-1})$ 与时间向量 $\mathbf{T}=(T_1,\ldots,T_M)^{\mathrm{T}}$。将人为定义的代价函数表示为 $\mathcal{K}(\mathbf{c},\mathbf{T})$，将轨迹参数化后，代价函数为 $\mathcal{W}(\mathbf{q},\mathbf{T})=\mathcal{K}(\mathbf{c}(\mathbf{q},\mathbf{T}),\mathbf{T})$。优化代价函数，需要获得代价函数 $\mathcal{W}$ 关于时间 $\mathbf{T}$ 和空间 $\mathbf{q}$ 的梯度，即 $\partial\mathcal{W}/\partial\mathbf{q}$ 和 $\partial\mathcal{W}/\partial\mathbf{T}$，如下：

$$\frac{\partial\mathcal{W}}{\partial\mathbf{q}}=\left(\mathbf{G}_1^{\mathrm{T}}e_1,\ldots,\mathbf{G}_{M-1}^{\mathrm{T}}e_1\right) \tag{28}$$

$$\frac{\partial\mathcal{W}}{\partial T_i}=\frac{\partial\mathcal{K}}{\partial T_i}-\mathrm{Tr}\left\{\mathbf{G}_i^{\mathrm{T}}\frac{\partial\mathbf{E}_i}{\partial T_i}\mathbf{c}_i\right\} \tag{29}$$

其中，通过式(22)可以直接解析求解 $\partial\mathbf{E}_i/\partial T_i$。因此，通过对所有 $1\le i\le M$ 求解 $\partial\mathbf{E}_i/\partial T_i$ 便可以得到 $\partial\mathbf{E}_i/\partial\mathbf{T}$。$\mathrm{Tr}(\cdot)$ 是矩阵的迹，即对角线元素的乘积。e_1 是单位矩阵 $\mathbf{I_{2s}}$ 的第一个列向量。$\mathbf{G}_i$ 是矩阵 $\mathbf{G}=\left(\mathbf{G}_0^{\mathrm{T}},\mathbf{G}_1^{\mathrm{T}},\ldots,\mathbf{G}_{M-1}^{\mathrm{T}},\mathbf{G}_M^{\mathrm{T}}\right)^{\mathrm{T}}$ 的子矩阵，$\mathbf{G}_0,\mathbf{G}_M\in\mathbb{R}^{s\times m},\mathbf{G}_i\in\mathbb{R}^{2s\times m}$，$\mathbf{G}$ 由下式求解得到

$$\mathbf{M}^{\mathrm{T}}\mathbf{G}=\frac{\partial\mathcal{K}}{\partial\mathbf{c}} \tag{30}$$

1.2 GCOPTER 源代码学习

1.2.1 代码文件目录

功能包：gcopter 、mockmap（地图生成）

gcopter:
——————**config(参数)**
————————global_planning.rviz (rviz 界面参数)
————————global_planning.yaml (优化参数)
——————**include(头文件)**
————————————**gcopter**
————————————(1)firi.hpp H 形凸多面体生成
————————————(2)flatness.hpp 带风阻模型的微分平坦映射
————————————(3)gcopter.hpp 梯度及代价函数计算 + 优化流程
————————————(4)geo_utils.hpp H 形凸多面体->V 形凸多面体
————————————(5)lbfgs.hpp 求解器
————————————(6)minco.hpp MINCO 轨迹定义及求解
————————————(7)quickhull.hpp 快速凸包算法
————————————(8)root_finder.hpp 多项式求根
————————————(9)sdlp.hpp H 形凸多面体生成
————————————(10)sfc_gen.hpp H 形凸多面体生成
————————————(11)trajectory.hpp 多项式轨迹定义
————————————(12)voxel_dilater.hpp 体素地图
————————————(13)voxel_map.hpp 体素地图
————————————misc
————————————visualizer.hpp
——————**launch(启动文件)**
————————global_planning.launch
——————**src(源码)**
————————global_planning.cpp

1.2.2 trajectory.hpp

模板 <D> 类: Piece（定义一段多项式轨迹）
成员:

类型定义:

名称	类型	解释
CoefficientMat	Eigen::Matrix<double, 3, D + 1>	系数矩阵
VelCoefficientMat	Eigen::Matrix<double, 3, D>	速度系数矩阵
AccCoefficientMat	Eigen::Matrix<double, 3, D - 1>	加速度系数矩阵

数据成员:

成员	类型	解释
duration	double	时间
coeffMat	CoefficientMat	系数矩阵

成员函数:

成员函数	返回类型	输入	解释
Piece()	——	时间、系数	构造函数
getDim()	int	——	返回轨迹维度
getDegree()	int	——	返回轨迹多项式阶次
getDuration()	double	——	返回轨迹时间
getCoeffMat()	CoefficientMat	——	返回多项式系数矩阵
getPos()	Eigen::Vector3d	时间 t	返回 t 时刻轨迹位置
getVel()	Eigen::Vector3d	时间 t	返回 t 时刻轨迹速度
getAcc()	Eigen::Vector3d	时间 t	返回 t 时刻轨迹加速度
getJer()	Eigen::Vector3d	时间 t	返回 t 时刻轨迹加加速度
normalizePosCoeffMat()	CoefficientMat	——	时间归一化处理系数矩阵
normalizeVelCoeffMat()	VelCoefficientMat	——	时间归一化处理速度矩阵
normalizeAccCoeffMat()	AccCoefficientMat	——	时间归一化处理加速度矩阵
getMaxVelRate()	double	——	获得轨迹最大速率
getMaxAccRate()	double	——	获得轨迹最大加速度率
checkMaxVelRate()	bool	maxVelRate	速率是否满足限制
checkMaxAccRate()	bool	maxAccRate	加速度率是否满足限制

模板 <D> 类: Trajectory（定义多段多项式轨迹）
成员:

类型定义:

名称	类型	解释
Pieces	std::vector<Piece <D> >	多项式容器

数据成员:

成员	类型	解释
pieces	Pieces	多段多项式

成员函数:

成员函数	返回类型	输入	解释
Trajectory()	——	时间、系数	构造函数
getPieceNum()	int	——	返回轨迹个数
getDurations()	Eigen::VectorXd	——	返回每段轨迹时间
getTotalDuration()	double	——	返回轨迹总时间
getPositions()	Eigen::Matrix3Xd	——	得到航点及始末点位置
locatePieceIdx()	int	t	定位轨迹段索引
append()	——	traj	将一个轨迹添加到当前轨迹
getJuncPos()	Eigen::Vector3d	index	获得 idx 航点位置
getJuncVel()	Eigen::Vector3d	index	获得 idx 航点速度
getJuncAcc()	Eigen::Vector3d	index	获得 idx 航点加速度
getPos()	Eigen::Vector3d	时间 t	返回 t 时刻轨迹位置
getVel()	Eigen::Vector3d	时间 t	返回 t 时刻轨迹速度
getAcc()	Eigen::Vector3d	时间 t	返回 t 时刻轨迹加速度
getJer()	Eigen::Vector3d	时间 t	返回 t 时刻轨迹加加速度
getMaxVelRate()	double	——	获得轨迹最大速率
getMaxAccRate()	double	——	获得轨迹最大加速度率
checkMaxVelRate()	bool	maxVelRate	速率是否满足限制
checkMaxAccRate()	bool	maxAccRate	加速度率是否满足限制

1.2.3 minco.hpp

命名空间 minco BandedSystem 类

数据成员:

成员	类型	解释
N	int	带状矩阵大小
lowerBw	int	下带宽
upperBw	int	上带宽

成员函数:

成员函数	返回类型	输入	解释
solve()	——	矩阵 b	求解带状线性方程组 Ax=b
solveAdj()	——	矩阵 b	求解带状线性方程组 ATx=b
factorizeLU()	——	——	带状矩阵 LU 分解

MINCO_S2NU 类 数据成员:

成员	类型	解释
N	int	轨迹段的数量
headPV	Eigen::Matrix<double, 3, 2>	轨迹初始位置、速度矩阵
tailPV	Eigen::Matrix<double, 3, 2>	轨迹最终位置、速度矩阵
A	BandedSystem	带状矩阵 M
b	Eigen::MatrixX3d	存 D 矩阵
T1	Eigen::VectorXd	时间向量的一次方
T2	Eigen::VectorXd	时间向量的二次方
T3	Eigen::VectorXd	时间向量的三次方

成员函数:

成员函数	返回类型	输入	解释
setConditions()	——		初始化
setParameters()	——	航点、时间	求解系数矩阵 c
getTrajectory()	——	traj	存储求得的轨迹
getEnergy()	——	energy	定义轨迹能量
getEnergyPartialGradByCoeffs()	——	gdc	能量关于系数的偏导
getEnergyPartialGradByTimes()	——	gdT	能量关于时间的偏导
propogateGrad()	——		代价函数对航点和时间的梯度

MINCO_S3NU 类

MINCO_S4NU 类

1.2.4 lbfgs.hpp

命名空间 lbfgs

lbfgs_parameter_t 类: 优化参数

函数指针

typedef double
(*lbfgs_evaluate_t)(void instance, const Eigen::VectorXd &x, Eigen::VectorXd &g);
instance：自定义指针 || x：优化变量 || g：代价对 x 的梯度

函数

line_search_lewisoverton() 线搜索
inline int lbfgs_optimize(
Eigen::VectorXd &x,
double &f,
lbfgs_evaluate_t proc_evaluate,
lbfgs_stepbound_t proc_stepbound,
lbfgs_progress_t proc_progress,
void instance,
const lbfgs_parameter_t &param)

lbfgs 测试

函数：$z = (x-1)^2 + y^2$

```
#include <iostream>
#include <Eigen/Eigen>
#include "lbfgs.hpp"

using namespace lbfgs;

double evaluate_fun(void *instance,
                    const Eigen::VectorXd &x,
                    Eigen::VectorXd &g)
{
    g(0) = 2 * (x(0) - 1);
    g(1) = 2 * x(1);
    return pow(x(0) - 1, 2) + pow(x(1), 2);
}
int main()
{
```

```
        Eigen::VectorXd x(2);
        x << 2, 2;
        double f;
        lbfgs_parameter_t param;
        int state = lbfgs_optimize(x, f, evaluate_fun,
                                   nullptr, nullptr, nullptr,
                                   param);
        if(state >= 0)
        {
            std::cout << "min_f: " << f << std::endl
                      << "optimal x :" << x.transpose() << std::endl;
        }
        else
        {
            std::cout << "error" << std::endl;
        }
        return 0;
    }
```

运行结果：

min_f: 6.16298e-32

optimal x : 1 -2.22045e-16

1.2.5 gcopter.hpp

该头文件主要实现了 GCOPTER_PolytopeSFC 类，其中包含：

1、代价函数对多项式系数矩阵和时间向量的梯度传导至航点及时间向量

部分成员函数：

成员函数	解释
backwardT()	带约束的 T 范围 T>0-> 无约束的 τ
forwardT()	$\tau -> T$
backwardGradT	得到代价函数对 τ 的梯度

```
    static inline void forwardP(const Eigen::VectorXd &xi,
                                const Eigen::VectorXi &vIdx,
                                const PolyhedraV &vPolys,
```

```
                                        Eigen::Matrix3Xd &P)
    {
        const int sizeP = vIdx.size();
        P.resize(3, sizeP); //P初始化
        Eigen::VectorXd q;
        for (int i = 0, j = 0, k, l; i < sizeP; i++, j += k)
        {
            l = vIdx(i);
            k = vPolys[l].cols();
            q = xi.segment(j, k).normalized().head(k - 1);
            P.col(i) = vPolys[l].rightCols(k - 1) * q.cwiseProduct(q) +
                       vPolys[l].col(0);
        }
        return;
    }
```

问题：多面体映射的函数与论文中不同

2、连续性约束的惩罚主要包括：

（1）attachPenaltyFunctional 函数：各种连续性惩罚梯度及代价叠加。

（2）smoothedL1：一阶惩罚项的光滑近似。

3、总代价函数及梯度：

（1）costFunctional：主要流程为——逆映射求时间和航点——构造 minco 轨迹——能量代价对时间和系数的偏导——连续性惩罚——MINCO 梯度传导——时间代价——梯度传导至 tau 和 xi——范数惩罚。

（2）：normRetrictionLayer：范数惩罚。

4、最小距离路径：

（1）costDistance：代价函数及梯度。

（2）getShortestPath：获得最短路径。

5、获得飞行走廊：processCorridor

6、初始化：setup：最短路径——优化维度设置——多面体索引设置——MINCO 轨迹计算——各种对象初始化。

7、优化：optimize

GCOPTER 代码基本阅读完，除 quickhull 等飞行走廊生成有关算法、带风阻微分平坦推导等以外，主要代码流程以及 MINCO 轨迹求解优化已经基本看完。

2 2024 年 10 月 21 日

任务

1、 上次未完成的内容：Polynomial-based Online Planning for Autonomous Drone Racing in Dynamic Environments 对比 **2** 的提升

2、 思考 MINCO 技术还存在什么缺陷

3、 在最初的硕士方向和现在所研究的有什么关系，如果不足以提供支撑，要继续阅读文献

2.1 Polynomial-based Online Planning for Autonomous Drone Racing in Dynamic Environments(IROS2023)

2.1.1 论文主要研究内容

采用时间均匀 MINCO 作为轨迹表示形式，对航点作硬约束处理，并结合动态障碍物和大姿态飞行构建了一个重规划框架。

（1）**核心方法**

对于 N 个目标航点，采用 $(N+1)$ 段 MICNO 轨迹，下述简称为 MINCOS 轨迹。其中，N 个目标航点的位置信息即为 MICNOS 轨迹中每一段 MINCO 轨迹的起始或终止点的位置信息。因此，优化变量即可由单段 MINCO 轨迹的时间向量和航点和目标航点的速度和加速度向量组成。经论文 Geometrically Constrained Trajectory Optimization for Multicopters 中公式（56、57、63）推导得，代价函数 $\mathcal{W}$ 对第 n 段轨迹的起始点 $\mathbf{z_{n-1}}$ 和终止点 $\mathbf{z_n}$ 的 i 阶导数的梯度如下

$$\frac{\partial \mathcal{W}}{\partial \mathbf{z}_{\mathbf{n-1}}^{(\mathbf{i})}} = \mathbf{G}_{\mathbf{n0}}^{\mathbf{T}} e_i \tag{31}$$

$$\frac{\partial \mathcal{W}}{\partial \mathbf{z}_{\mathbf{n}}^{(\mathbf{i})}} = \mathbf{G}_{\mathbf{nM}}^{\mathbf{T}} e_i \tag{32}$$

其中，e_i 是 $\mathbf{I_2 s}$ 的第 i 列。

（2）**实现功能**

目前实现了对于给定 N 个航点的的全局路径规划。相比于单段 MINCO 轨迹优化，对固定航点的速度也进行了优化。效果如图1所示。

（3）**还未实现功能**

钻运动的圈的功能、及多拓扑规划的功能暂未实现。一方面仿真环境有限，该仿真环境是生成的静态体素地图。第二，是该仿真是基于全局路径规划的框架，未涉及局部路径规划。

（4）**后续实现方向**

在 EGO-PLANNER-V2 既存在全局路径规划，又含有局部的路径规划。不包含动态障碍物但仿真环境更贴近实机，其相比于 EGO-PLANNER 采用 MINCO 轨迹，方便代码修改。

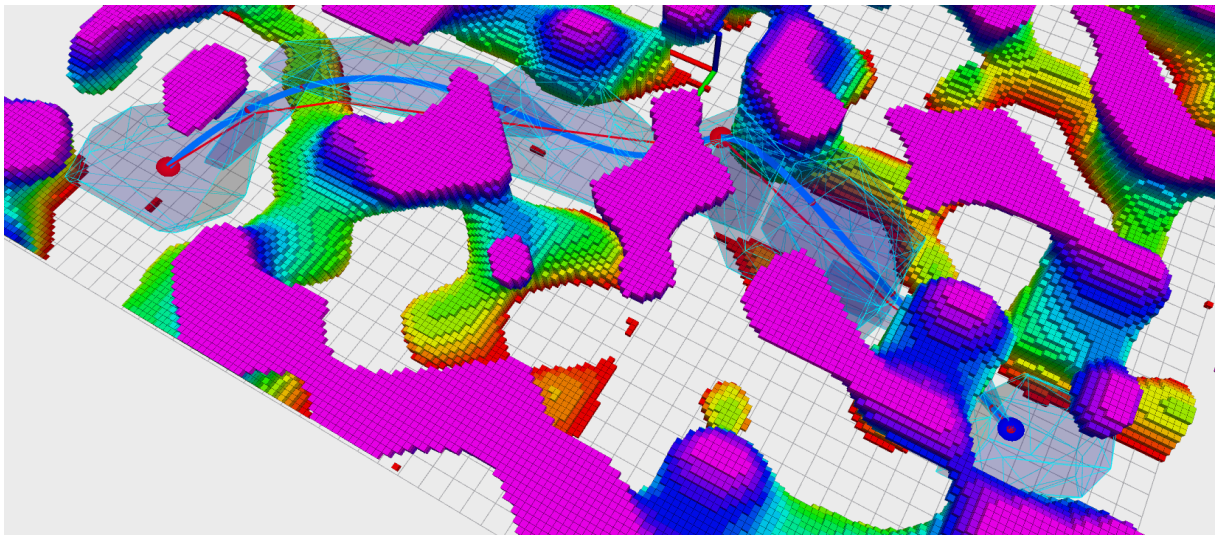

图 1: 航点硬约束的全局路径轨迹效果图

尝试对 EGO-PLANNER-V2 进行修改，并给出一些固定航点接口等。但 EGO-PLANNER-V2 代码体量大、耦合程度高，存在一定难度。

3 2024 年 10 月 30 日

任务

1、 将 GCOPTER-MINCOS 代码移植至 EGO-PLANNER-V2

2、 继续复现 Polynomial-based 其余部分（动态环、拓扑路径规划）

3.1 将 GCOPTER-MINCOS 代码移植至 EGO-PLANNER-V2

（1）**设计状态机 FSM**

在原有状态机的基础上增加 **环的距离检测**和**固定航点重规划**。

具体逻辑：在检测到到达环（航点）附近时，规划到下一个航点的全局轨迹；获得下一段全局轨迹之后，获得该段轨迹的局部路径规划目标点，最后进行固定航点重规划。在飞行至环中心时，切换至原有重规划。

Algorithm 1 固定航点重规划

Input: 无人机位置 $odom\_pos\_$, 圆环中心位置 $circle\_pos\_$, 固定航点重规划标志 $touch\_circle\_$, 当前时间 $time\_now\_$, 重规划界限 $planning\_horizen\_$, 全局规划界限 $no\_replan\_thresh\_$, 当前目标点 $goal$

case $EXEC\_TRAJ$**:**
if $touch\_circle\_$ **and** $(time\_now\_ > circle\_cross\_time\_ - 0.1$ **or** $(odom\_pos\_ - circle\_pos\_).norm() < 0.09)$ **then**
    $cir\_ps\_.erase(cir\_ps\_.begin())$
    $touch\_circle\_ = false$
    $already\_cross = true$
    $have\_planed\_next\_waypoint = false$
end if
if $(goal - odom\_pos\_).norm() < no\_replan\_thresh\_$ **and** $planNextWaypoint()$ **then**
    $have\_planed\_next\_waypoint = true$
    if $!cir\_ps\_.empty()$ **and** $(odom\_pos\_ - circle\_pos\_).norm() < planning\_horizen\_$ **then**
        $touch\_circle\_ = true$
        $planFromLocalTraj(1)$
    end if
end if

（2）**定义 MINCOS 类**

（3）**增加固定航点重规划算法**

（4）运行效果

正常运行效果，如图2。

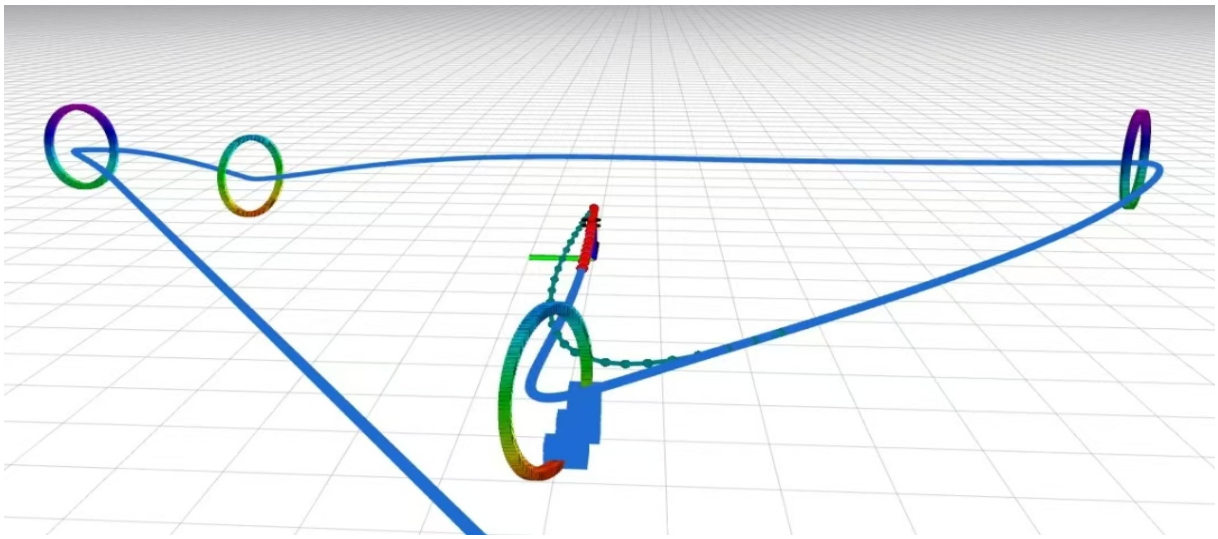

图 2: 航点硬约束的局部路径规划全局效果图

有小概率优化之后的轨迹会打圈，并且很难从环的另一端钻出，如图3，初步分析可能原因：

一方面是逻辑还需修改，因为固定经过环的中心即可，在需要大幅转弯时，可能存在掉头飞行的情况。

另一方面可能是优化的代价函数需修改，所采用的优化代价函数的每一项代价是采用 EGO-V2 的代价函数进行计算的，可能并不适用于固定航点优化。

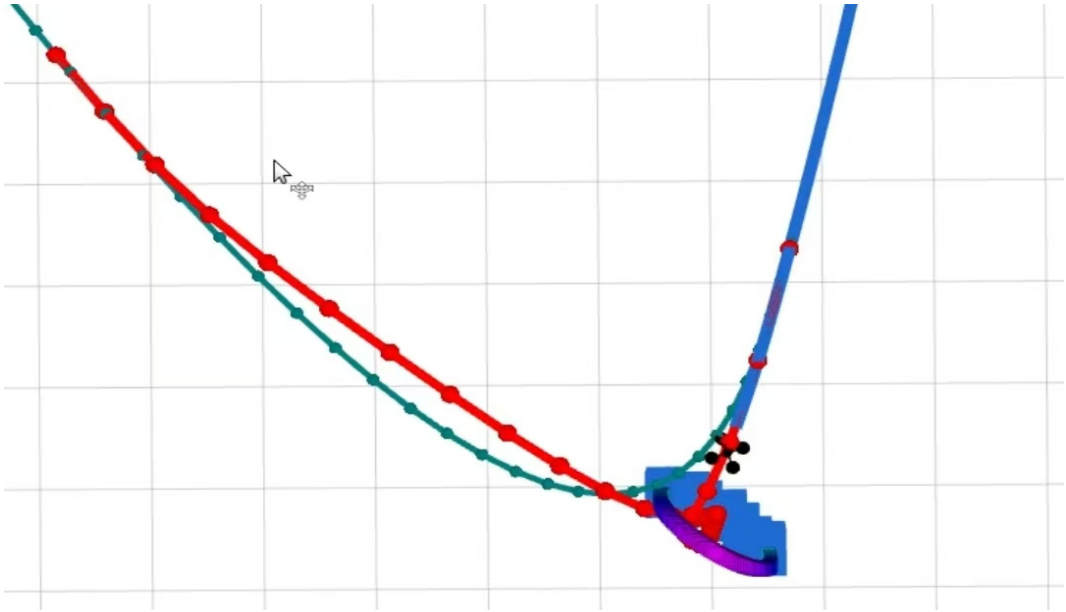

图 3: 航点硬约束的局部路径规划失败图

3.2 Time-Optimal Gate-Traversing Planner for Autonomous Drone Racing(ICRA2024)

Code: https://github.com/FSC-Lab/TOGT-Planner

3.2.1 论文所解决的问题

根据所提供的门的信息，不考虑环境中的其他环境信息，计算出一条穿过所有门的时间最优且符合无人机动力学的全局轨迹。

3.2.2 论文算法

（1）四旋翼无人机模型

状态变量：$\mathbf{x}=[\mathbf{p}^W,\mathbf{q}_{WB},\mathbf{v}^W,\boldsymbol{\omega}^B]^T\in\mathbb{R}^n,n=13$，分别为世界坐标系下的位置、由机体坐标系到世界坐标系的单位四元数旋转矩阵、世界坐标系下的速度、机体坐标系下的角速度

输入变量：$\mathbf{u}=[f_1,f_2,f_3,f_4]^T\in\mathbb{R}^m,m=4$，依次为四个电机的推力

$$\dot{\mathbf{p}}=\mathbf{v},\quad \dot{\mathbf{q}}=\frac{1}{2}\mathbf{\Lambda}(\mathbf{q})\begin{bmatrix}0\\ \boldsymbol{\omega}\end{bmatrix}, \tag{33}$$

$$\dot{\mathbf{v}}=\mathbf{g}+\frac{1}{m}\mathbf{R}(\mathbf{q})\mathbf{F}_T,\quad \dot{\boldsymbol{\omega}}=\mathbf{J}^{-1}(\boldsymbol{\tau}-\boldsymbol{\omega}\times\mathbf{J}\boldsymbol{\omega}) \tag{34}$$

其中

$$\mathbf{F}_T=\begin{bmatrix}0\\0\\\sum f_i\end{bmatrix},\boldsymbol{\tau}=\begin{bmatrix}l(f_1+f_2-f_3-f_4)\\ l(-f_1+f_2+f_3-f_4)\\ c_\tau(f_1-f_2+f_3-f_4)\end{bmatrix}. \tag{35}$$

（2）门约束的定义

球形门：指在飞行过程中必须要经过的航点。

$$\mathcal{G}_{\mathcal{B}}=\{\mathbf{p}\in\mathbb{R}^3|\|\mathbf{p}-\mathbf{p}_w\|_2\le\delta\}, \tag{36}$$

其中，$\mathbf{p}_w$ 是球形门的中心，δ 是其半径。

凸多边形门：指在飞行过程中要钻过的门或隧道。

$$\mathcal{G}_{\mathcal{P}}=\{\mathbf{p}\in\mathbb{R}^3|\mathbf{A}\mathbf{p}\le\mathbf{b}\}, \tag{37}$$

（3）时间最优门遍历问题

$$\min_{\mathbf{x},\mathbf{u},t_f} t_f \tag{38}$$

$$\text{s.t.}\mathbf{x}(0)=\bar{\mathbf{x}}_0,\mathbf{x}(t_f)=\bar{\mathbf{x}}_f, \tag{39}$$

$$\dot{\mathbf{x}}=\mathbf{f}(\mathbf{x},\mathbf{u}),\mathbf{h}(\mathbf{x},\mathbf{u})\le 0, \tag{40}$$

$$\exists 0<t_1<t_2<...<t_L<t_f, \tag{41}$$

$$\mathbf{h}_{\mathcal{G}^i}(\mathbf{p}_{\mathbf{x}(t_i)})\le 0,1\le i\le L, \tag{42}$$

（4）问题转化与求解

门约束与时间解耦

$$\begin{aligned}&\min_{\mathbf{P},\mathbf{T}} T_\Sigma + I_{\mathcal{T}(\mathbf{P})}(\mathbf{T})\\&\text{s.t.}\mathbf{h}_{\mathcal{G}^i}(\mathbf{p}_i) \le \mathbf{0}, 1 \le i \le L,\end{aligned} \tag{43}$$

其中，

$$I_{\mathcal{T}(\mathbf{P})}(\mathbf{T}) = \begin{cases} 0 & ,\text{if } \mathbf{T} \in \mathcal{T}(\mathbf{P}) \\ \infty & ,\text{if } \mathbf{T} \notin \mathcal{T}(\mathbf{P}) \end{cases} \tag{44}$$

$$\begin{aligned}\mathcal{T}(\mathbf{P}) = \{\mathbf{T} \in \mathbb{R}^{L+1}_{>} \mid \exists \mathbf{x}, \mathbf{u} : [0, T_\Sigma] \to \mathbb{R}^n, \mathbb{R}^m\\ \text{s.t.}\mathbf{x}(0) = \bar{\mathbf{x}}_0, \mathbf{x}(T_\Sigma) = \bar{\mathbf{x}}_f\\ \dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}), \mathbf{h}(\mathbf{x}, \mathbf{u}) \le 0\\ \mathbf{p}_{\mathbf{x}(t_{\Sigma_i})} = \mathbf{p}_i, 1 \le i \le L\}.\end{aligned} \tag{45}$$

给定门的位置向量后，可行时间分配向量的集合如式45所示。采用 MINCO 轨迹能够消除尽可能多的约束，并且具有最少数量的系数，大大降低了问题的复杂性。通过结合微分平坦特性消除对系统动力学、初始状态和终端状态的约束，而且可以确保精确的航路点遍历，从45中删除大量约束，只留下状态输入约束。连续状态输入约束通过采样来定义惩罚泛函。

$$\begin{aligned}I_{\hat{\mathcal{T}}(\mathbf{P})}(\mathbf{T}) &= \int_0^{T_\Sigma} \max[\mathbf{h}_\Psi(\mathbf{y}^{[s]}(t)), \mathbf{0}]^3 dt,\\ &\approx \sum_{i=1}^{L+1} \sum_{j=0}^{\kappa_i} \max[\mathbf{h}_\Psi(\mathbf{y}^{[s]}(t_{i-1} + j\Delta t_i)), \mathbf{0}]^3 \Delta t_i,\end{aligned} \tag{46}$$

（5）门约束和时间约束

通过将门约束和时间约束的可行解空间映射至 n 维欧式空间后，进行无约束优化。

（6）无约束优化

$$\min_{\mathbf{D},\mathbf{K}} \quad T_\Sigma(\mathbf{K}) + I_{\hat{\mathcal{T}}(\mathbf{P}(\mathbf{D}))}(\mathbf{T}(\mathbf{K})). \tag{47}$$

推导上述函数梯度，并采用 L-BFGS 算法进行优化，优化完成后可由 **D K** 获得航点向量 **P** 和时间向量 **T**，然后构建 MINCO 轨迹，之后通过微分平坦便可得到全状态变量和输入变量。

3.2.3 论文启发

本篇论文通过对穿过所有门的全局路径规划问题进行问题建模，并且采用 MINCO 轨迹来进行轨迹表示，一方面能够减少优化问题中的约束，另一方面可以减少优化参数；对连续性状态输入约束采用离散化的惩罚泛函；对凸的门约束和时间约束映射至 n 维欧氏空间进行优化。从本篇论文中能够了解 MINCO 轨迹以及几何优化方法的应用。

4 2024 年 11 月 12 日

任务

1、 读文献学习代价函数构建方法

2、 解决优化轨迹“打圈”的问题

3、 文献进展报告为四部分：（1）论文所解决的问题（2）算法（3）结果（4）论文启发

4.1 Minimum Snap Trajectory Generation and Control for Quadrotors(ICRA2011)

4.1.1 论文所解决的问题

论文提出了四旋翼无人机在严格约束的三维空间中的控制器设计和轨迹生成方法。另外，论文对无人机的微分平坦特性进行了详细推导，并提出飞行走廊约束方法。

4.1.2 论文算法

（1）无人机微分平坦特性

无人机动力学模型如下：

$$\begin{cases} m\ddot{\mathbf{r}} = -mg\mathbf{z}_W + u_1\mathbf{z}_B & (48a) \\ \dot{\boldsymbol{\omega}}_{BW} = \mathcal{I}^{-1}\left[-\boldsymbol{\omega}_{BW}\times\mathcal{I}\boldsymbol{\omega}_{BW} + \begin{bmatrix} u_2 \\ u_3 \\ u_4 \end{bmatrix}\right] & (48b) \end{cases}$$

平坦输出：$\sigma(t) = [x, y, z, \psi]^T$。由式48a可知，机体坐标系的 Z_B 轴方向 $\mathbf{z_B}$ 满足下式

$$\mathbf{z_B} = \frac{\mathbf{t}}{\|\mathbf{t}\|}, \mathbf{t} = [\ddot{\sigma}_1, \ddot{\sigma}_2, \ddot{\sigma}_3 + g]^T \tag{49}$$

给定 ψ，可以计算出 $\mathbf{x}_C = [\cos\sigma_4, \sin\sigma_4, 0]^T$，由于采用 $Z-X-Y$ 的旋转顺序，坐标轴 $\mathbf{x}_C$ 和 $\mathbf{x}_B$、$\mathbf{z}_B$ 在一个平面，$\mathbf{y}_B$ 与 $\mathbf{z}_B$ 和 $\mathbf{x}_C$ 正交。因此，有

$$\mathbf{y}_B = \frac{\mathbf{z}_B\times\mathbf{x}_C}{\|\mathbf{z}_B\times\mathbf{x}_C\|},\ \mathbf{x}_B = \mathbf{y}_B\times\mathbf{z}_B \tag{50}$$

之后可以唯一确定从机体坐标系到世界坐标系的旋转矩阵 ${}^WR_B = [\mathbf{x}_B\ \ \mathbf{y}_B\ \ \mathbf{z}_B]$，通过旋转矩阵可以确定无人机的滚动角 ϕ 和俯仰角 θ。对式48a求导得

$$m\dot{\mathbf{a}} = \dot{u}_1\mathbf{z}_B + \boldsymbol{\omega}_{BW}\times u_1\mathbf{z}_B \tag{51}$$

沿 $\mathbf{z}_B$ 投影可得 $\dot{u}_1 = \mathbf{z}_B\cdot m\dot{\mathbf{a}}$，并定义向量 $\mathbf{h}_\omega = \boldsymbol{\omega}_{BW}\times\mathbf{z}_B = \frac{m}{u_1}(\dot{\mathbf{a}} - (\mathbf{z}_B\cdot\dot{\mathbf{a}})\mathbf{z}_B)$ 为 $\frac{m}{u_1}\dot{\mathbf{a}}$ 在 $X_B - Y_B$ 平面上的投影。因此

$$p = -\mathbf{h}_\omega\cdot\mathbf{y}_B, q = \mathbf{h}_\omega\cdot\mathbf{x}_B \tag{52}$$

另外，r 是 $\boldsymbol{\omega}_{BW}$ 在 $\mathbf{z}_B$ 方向的分量，并将 $\boldsymbol{\omega}_{BW}$ 分解为 $\boldsymbol{\omega}_{BC}+\boldsymbol{\omega}_{CW}$ 能够得出

$$r=(\omega_{BC}+\omega_{CW})\cdot\mathbf{z}_B=\omega_{CW}\cdot\mathbf{z}_B=\dot{\psi}\mathbf{z}_W\cdot\mathbf{z}_B \tag{53}$$

同样，对式48a求二阶导可以得出，角加速度也可以被平坦输出及其有限阶导数的函数表示。

对于控制输入 u，由式48a、49可知，净推力 $u_1=m\|\mathbf{t}\|$；另外，由于角速度和角加速度是 z 及其导数的函数，因此可以通过式48b来计算输入 u_2，u_3 和 u_4。综上，四旋翼无人机系统的 12 个状态和 4 个控制输入都可以写成平坦输出 σ 及其导数的形式。

（2）**轨迹生成**

轨迹表示为 m 段 n 阶多项式轨迹，如下

$$\sigma_T(t)=\begin{cases}\sum_{i=0}^{n}\sigma_{Ti1}t^i & t_0\le t<t_1\\ \sum_{i=0}^{n}\sigma_{Ti2}t^i & t_1\le t<t_2\\ \vdots & \\ \sum_{i=0}^{n}\sigma_{Tim}t^i & t_{m-1}\le t\le t_m\end{cases} \tag{54}$$

优化问题的构建：代价函数包含位置 $\mathbf{r_T}$ 的 k_r 阶导的平方和偏航角 ψ 的 k_ψ 阶导的平方，如下

$$\begin{aligned}\min\quad & \int_{t_0}^{t_m}\mu_r\left\|\frac{d^{k_r}\mathbf{r_T}}{dt^{k_r}}\right\|^2+\mu_\psi\frac{d^{k_\psi}\cdot\psi_\mathbf{T}}{dt^{k_\psi}}^2dt\\ \text{s.t.}\quad & \sigma_T(t_i)=\sigma_i,i=0,...,m\\ & \frac{d^px_T}{dt^p}|_{t=t_j}=0\text{ or free},j=0,m;p=1,...,k_r\\ & \frac{d^py_T}{dt^p}|_{t=t_j}=0\text{ or free},j=0,m;p=1,...,k_r\\ & \frac{d^pz_T}{dt^p}|_{t=t_j}=0\text{ or free},j=0,m;p=1,...,k_r\\ & \frac{d^p\psi_T}{dt^p}|_{t=t_j}=0\text{ or free},j=0,m;p=1,...,k_\psi\end{aligned} \tag{55}$$

其中，μ_r、μ_ψ 的作用是使积分量无量纲。由于 u_2,u_3 由位置的四阶导表示，并且 u_4 由偏航角的二阶导表示，因此取 $k_r=4,k_\psi=2$ 将优化变量 $\sigma_{Tij}=[x_{Tij},y_{Tij},z_{Tij},\psi_{Tij}]^T$ 写成 $4mn\times1$ 的向量 $\mathbf{c}$，则问题转换为一个 QP 问题，如下

$$\begin{aligned}\min\quad & c^THc+f^Tc\\ \text{s.t.}\quad & Ac\le b\end{aligned} \tag{56}$$

（3）**无量纲优化**

式55中的变量 x_T,y_T,z_T,ψ_T 是解耦的，因此可以将问题分离为四个优化问题。考虑单一无量纲变量 $\tilde{\omega}(\tau)$ 的优化问题，如式57所示。其中 τ 代表无量纲时间。

$$\begin{aligned}\min\quad & \int_0^1\frac{d^k\tilde{w}(\tau)}{d\tau^k}^2d\tau\\ \text{s.t.}\quad & \tilde{w}(\tau_i)=\tilde{w}_i,\qquad i=0,...,m\\ & \frac{d^p\tilde{w}(\tau)}{d\tau^p}|_{\tau=\tau_j}=0\text{ or free},\quad \tau_j=0,1;p=1,...,k\end{aligned} \tag{57}$$

引入时间维度 $t=\alpha\tau$ 和变量 ω，定义为 $w(t)=w(\alpha\tau)=\beta_1+\beta_2\tilde{w}(\tau)$。则问题57可变换为如下问题

$$\begin{aligned}\min \quad & \frac{\alpha^{2k-1}}{\beta_2}\int_0^{\alpha}\frac{d^k w(t)}{dt^k}dt \\ \text{s.t.} \quad & w(t_i)=\beta_1+\beta_2\tilde{w}_i, \quad i=1,...,m \\ & \frac{d^p w(t)}{dt^p}|_{t=t_j}=0 \text{ or free}, \quad t_j=0,\alpha; p=1,...,k\end{aligned} \tag{58}$$

在问题58中边界条件在空间维度平移了 β_1 缩放了 β_2，在时间维度缩放了 α。因此，如果无量纲优化问题57的最优解是 $\tilde{\omega}^*$，则问题58的最优解为

$$w^*(t)=\beta_1+\beta_2\tilde{w}^*\left(t/\alpha\right) \tag{59}$$

整体优化思路：对于问题55，在解耦后分别与问题58对应，得到时间放缩尺度 α 与和每个变量在空间中的变换 β_1,β_2 的比例关系。每个变量的时间放缩尺度相同，空间放缩尺度可以不同。