目　录

1 课题主要研究内容及进度

1.1 课题主要研究内容

1.1.1 软硬件平台搭建

硬件设计的主要内容是实现矢量推力的机构设计、机体结构设计、电路设计。软件方面包括飞行控制算法的开发、无人机仿真平台的搭建、ROS2 框架下的软件节点开发。

1.1.2 控制器算法设计

PID 控制器相关内容主要为实现基于 PID 的飞行控制和控制分配。在这个过程中利用新框架的优势，并记录开发文档，用高精度模型在仿真中验证效果，然后在实物上进行部署并实验。

自适应 MPC 综合考虑底层执行器动态以及各种可能的约束条件，推导合适的方案并实现。实现 MPC 的重点是在考虑执行器动态、机臂倾转角度限制、自适应等目标下建立合理的模型并用优化的方法实现控制器。采用的框架是 acados。

1.1.3 控制算法性能对比

此部分内容包括设计参考轨迹、进行仿真和实物实验、收集和分析数据、对比 PID 控制器和 MPC 在不同轨迹下的性能等。

1.2 进度介绍

截至报告日期，进度完成情况如下表 1-1 所示，未列写的目标进度均为 0%。

表 1-1 进度情况表

目标	完成情况
完成 ROS2 下开发框架的搭建	100%
在仿真中复现基于 PID 的控制器	100%
在实物上完成 PID 控制器的部署	100%
自适应模型预测控制器架构和设计	100%
用 acados 构建 MPC 求解器	100%
用 Python 测试求解器效果	5%
实现 ROS2 的 MPC 节点	5%

2　已完成的研究工作及结果

2.1　基于 PID 控制的系统概述

2.1.1　软件架构

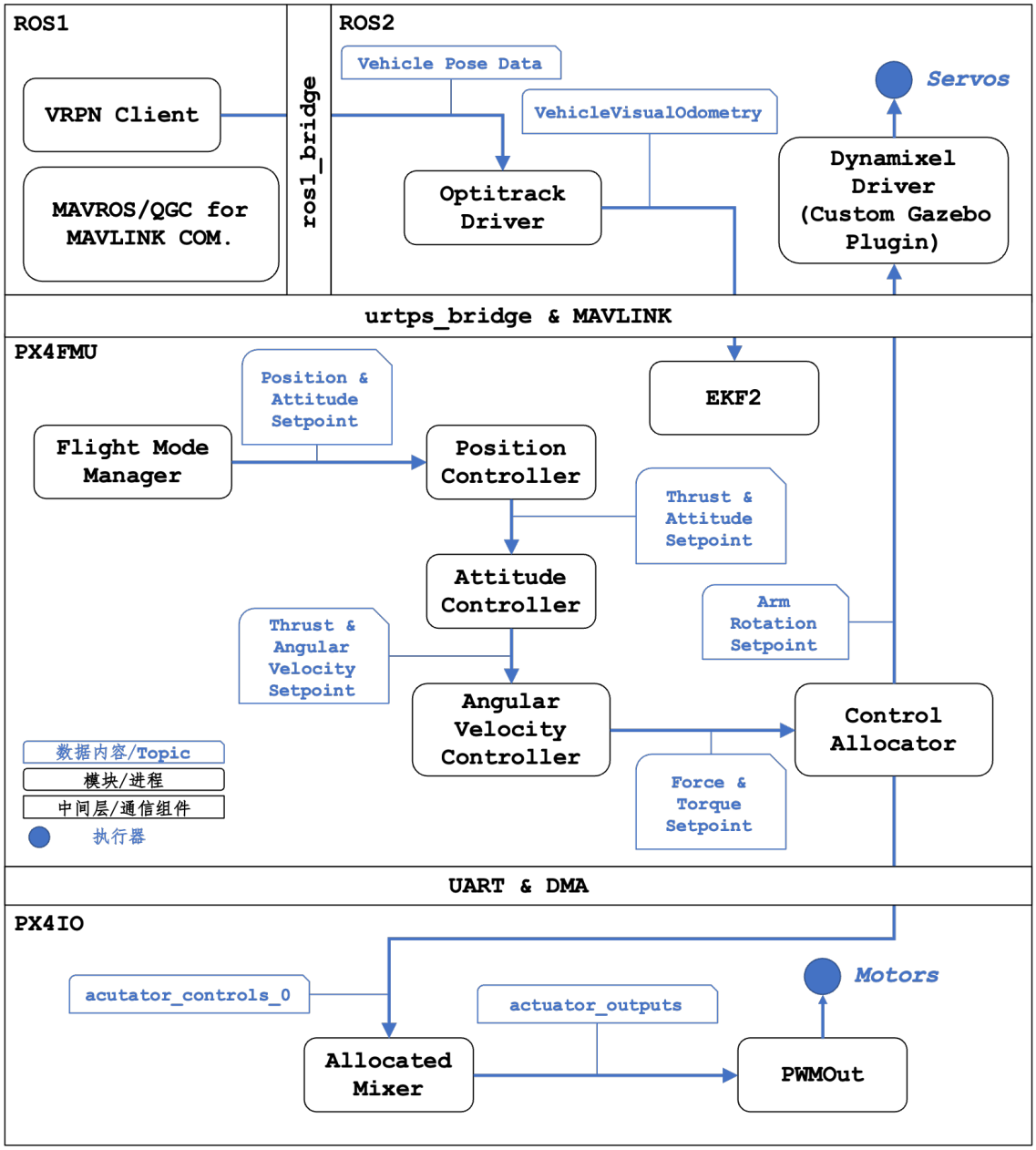

图 2-1　基于 PID 控制的系统概览

图 2-1 从整体上展示了使用 PID 控制时系统的组成和数据流向。其中 ROS1 和 ROS2 节点运行在 NUC11i7PAH 机载电脑上；PX4FMU 和 PX4IO 的实体为 PX4 飞控的两片 MCU，集成于 CUBEORANGE 飞控 PCB 上。

ROS1 环境下主要用到 VRPN Client 和 MAVROS。VRPN Client 用于接受 UDP 协议下的动作捕捉数据并将其发布为 ROS1 的话题的数据。MAVROS 主要用于在 ROS 下可视化一些飞行数据，它和 QGC 均通过 MAVLINK 协议与 PX4 进行通信。

ROS1 与 ROS2 的数据互通依靠 ROS2 官方提供的包 ROS1 Bridge，这样 ROS2 下的 Optitrack Driver 就能获取机器人的动作捕捉位姿数据，然后把数据转换为 PX4 系统中定义的格式然后发布，这些数据随后通过 MicroRTPS Bridge 传输到 PX4 中成为 uORB 消息中的数据并被飞控系统使用。uORB 动捕位姿数据由 EKF2 模块接收，与其他机载传感器数据融合，然后才有最终的状态估计。

关键数据，比如执行器指令、动作捕捉位姿等，在机载电脑和飞控之间的通讯由 MicroRTPS Bridge 处理，主要是考虑到传输带宽和延迟性能；而 MAVLINK 只作为监控数据或者低频率数据的传输通道。

PX4 采用飞行管理和指令输出分离的设计，主要的逻辑飞控程序运行在 FMU 中，而 IO 处理遥控器的指令输入、控制量的混合、PWM 信号输出等底层任务。FMU 和 IO 之间通过 UART 进行通讯，PX4 实现了 DMA 以降低通信延迟。

Flight Mode Manager 模块根据飞行模式和遥控器输入输出设定位姿的期望值。位姿的期望值经过 Position Controller 位置控制器后得到归一化的力期望值和姿态期望值，进入 Attitude Controller 姿态控制器。在姿态控制器中，力期望值不作任何处理，但姿态设定点与当前姿态相运算后得到角速度期望值。这些数据随后由 Angular Velocity Controller 角速度控制器处理，其中归一化的力期望值被转化为单位为 N 的实际力，并根据机体坐标系下的 Newton-Euler 公式被$_B\boldsymbol{\omega} \times {}_B\boldsymbol{v}$作前馈补偿；姿态方面，角速度控制器根据期望值和当前值获得力矩期望值，并由角速度和质心前馈项作补偿。期望的合力和合力矩由 Control Allocator 控制分配器解算获得 Arm Rotation 机臂倾转角度和 actuator_controls_0 话题中的电机控制量。

电机控制量随着 actuator_controls_0 被送入 IO 中，被 Allocated Mixer 混合后得到实际的 PWM 信号值，随后由 PWMOut 模块输出给电子调速器，控制无刷电机的转速。

机臂倾转角度被 ROS2 中的 Dynamixel Driver 舵机驱动接收，后者驱动机臂按照 Arm Rotation 的数据进行倾转。在仿真中，则是 Gazebo 的插件控制仿真环境中的机器人机臂进行倾转。目前在舵机驱动方面调用的是以位置指令为接口的位置-速度-电流三环控制。

系统中的原创模块为 Optitrack Driver、Dynamixel Driver、Custom Gazebo Plugin，而其他列举模块来源于业界常用的开源项目。但是这些开源项目提供的模块并非直接可用，图 2-1 中列举的模块均被深度修改过以应用于本课题的过驱动六旋翼无人机上。

2.1.2 PX4 代码的修改

对 PX4 代码进行修改的主要目标是：一、按照 Newton-Euler 公式实现更精确的、带有物理意义的刚体运动控制；二、为无人机设计一种新的飞行模式；三、实现控制分配算法并将指令输出管线调试通畅；四、在相关的模块中修复由欠驱动转换为过驱动而产生的自由度缺失问题。关键模块的修改见下表。

表 2-1 PX4 关键模块的修改

模块	修改	备注
Angular Velocity Controller	在扭矩计算中添加了质心的前馈，在合力的计算中添加了角速度和线速度前馈	Newton-Euler 公式
Commander	添加了 ACRO 飞行模式	在 ACRO 模式中，本来用于前进后退的摇杆用于抬头和低头
Control Allocator	添加了过驱动六旋翼需要的分配算法，修改了控制量的输出话题	actuator_controls_0 才是 FMU 和 IO 通信时被优先考虑的信息
Flight Mode Manager	配合 Commander 实现 ACRO 模式	补上缺失的自由度
MC Attitude Controller	正确处理 ACRO 模式下带来的角速度前馈	同时考虑缺失的自由度
MC Position Controller	添加了世界坐标系下的合力到机体坐标系下合力的转换步骤	因为控制分配要使用机体坐标系下的合力/力矩
PX4IO	修改 Allocated Mixer 的输入话题	ESC 校准功能暂未测试

2.2 PID 控制器设计及飞行实验

2.2.1 控制器设计

控制器根据位置、速度、姿态、角速度误差输出期望的合力和合力矩，作为控制分配器的输入。控制器的描述见下式。

$$\begin{bmatrix} {}_B\boldsymbol{f}_a \\ {}_B\boldsymbol{\tau}_a \end{bmatrix} = \begin{bmatrix} m\big(\mathrm{rotate}\big(\mathrm{PID}\big(\boldsymbol{e}_p, \boldsymbol{e}_v\big) + \boldsymbol{a}_{sp} + \boldsymbol{g}, \boldsymbol{q}\big) + {}_B\boldsymbol{\omega} \times {}_B\boldsymbol{v}\big) \\ \boldsymbol{J}\,\mathrm{PID}\big(\boldsymbol{e}_q, \boldsymbol{e}_\omega\big) + {}_B\boldsymbol{\omega} \times \boldsymbol{J}_B\boldsymbol{\omega} + {}_B\boldsymbol{d}_{com} \times {}_B\boldsymbol{f}_a \end{bmatrix} \tag{1}$$

其中，$\boldsymbol{e}_p, \boldsymbol{e}_v$分别是位置和姿态误差，$\boldsymbol{e}_q, \boldsymbol{e}_\omega$是姿态和角速度误差，$\boldsymbol{a}_{sp}$是由摇杆指令得到的加速度设定值；其他的符号定义参见表 2-2。

2.2.2 轨迹跟踪仿真验证

在 SITL（Software In The Loop）仿真中设计了分段连续的六自由度空间八字形轨迹。轨迹$[x(t), y(t), z(t), R(t), P(t), Y(t)]^{\mathrm{T}}$的解析形式为：

$$x(t) = \begin{cases} 0, & t \in [0, t_1) \\ -r + r\cos\left(\dfrac{3\pi}{2t_2}(t - t_1)\right), & t \in [t_1, t_1 + t_2) \\ -r + \dfrac{r}{t_1}(t - t_1 - t_2), & t \in [t_1 + t_2, 3t_1 + t_2) \\ r + r\sin\left(\dfrac{3\pi}{2t_2}(t - 3t_1 - t_2)\right), & t \in [3t_1 + t_2, T - t_1) \\ 0, & t \in [T - t_1, T) \end{cases} \tag{2}$$

$$y(t) = \begin{cases} \dfrac{rt}{t_1}, & t \in [0, t_1) \\ r + r\sin\left(\dfrac{3\pi}{2t_2}(t - t_1)\right), & t \in [t_1, t_1 + t_2) \\ 0, & t \in [t_1 + t_2, 3t_1 + t_2) \\ -r + r\cos\left(\dfrac{3\pi}{2t_2}(t - 3t_1 - t_2)\right), & t \in [3t_1 + t_2, T - t_1) \\ -r + \dfrac{r(t - T + t_1)}{t_1}, & t \in [T - t_1, T) \end{cases} \tag{3}$$

$$z(t) = -h - d_h + d_h\cos\left(\frac{2\pi t}{2t_1 + t_2}\right), \qquad t \in [0, T) \tag{4}$$

$$R(t)=\begin{cases}0, & t\in[0,t_1)\\ \frac{\pi}{4}-\frac{\pi}{4}\cos\left(\frac{2\pi}{t_2}(t-t_1)\right), & t\in[t_1,t_1+t_2)\\ 0, & t\in[t_1+t_2,3t_1+t_2)\\ -\frac{\pi}{4}+\frac{\pi}{4}\cos\left(\frac{2\pi}{t_2}(t-3t_1-t_2)\right), & t\in[3t_1+t_2,T-t_1)\\ 0, & t\in[T-t_1,T)\end{cases} \tag{5}$$

$$P(t)=\frac{dz}{dt},\qquad t\in[0,T) \tag{6}$$

$$Y(t)=\begin{cases}\frac{\pi}{2}, & t\in[0,t_1)\\ \frac{\pi}{2}+\frac{3\pi}{2t_2}(t-t_1), & t\in[t_1,t_1+t_2)\\ 0, & t\in[t_1+t_2,3t_1+t_2)\\ -\frac{3\pi}{2t_2}(t-3t_1-t_2), & t\in[3t_1+t_2,T-t_1)\\ \frac{\pi}{2}, & t\in[T-t_1,T)\end{cases} \tag{7}$$

当参数设置为$T=40, t_1=\frac{T}{3\pi+4}, t_2=\frac{3\pi T}{6\pi+8}, r=1.5, h=2.5, d_h=0.5$时，仿真结果和数据如下列图所示：

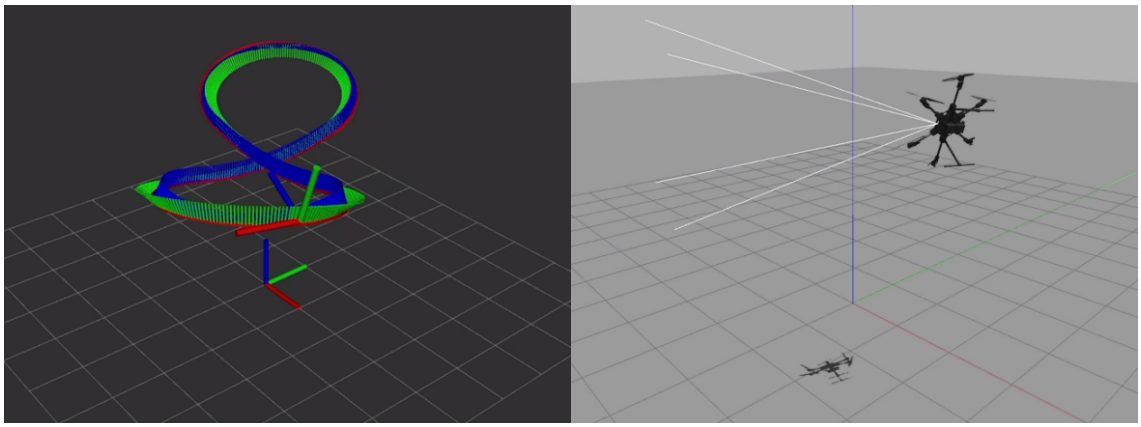

图 2-2　RViz 中轨迹和位姿可视化（左）和 Gazebo 仿真机器人位姿（右）

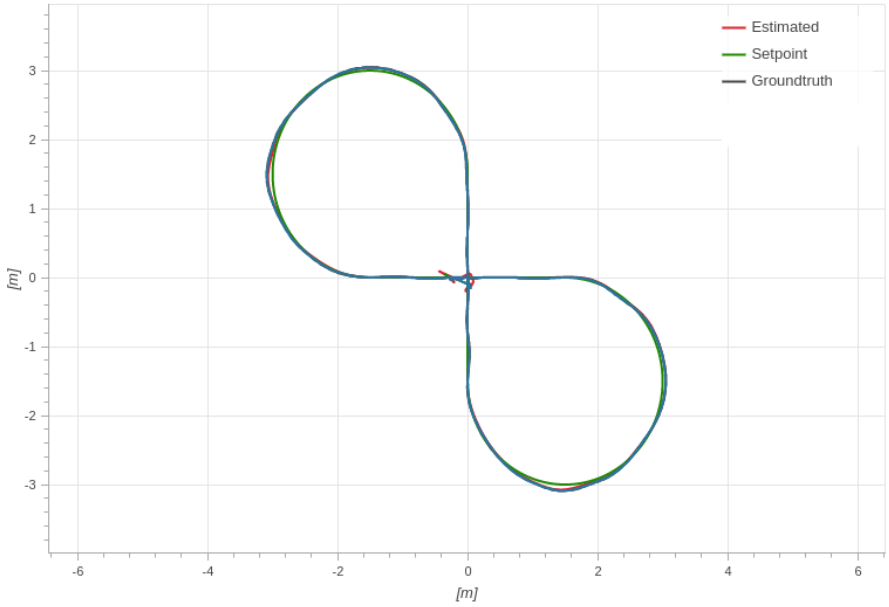

图 2-3　轨迹期望值和实际值在 XY 平面上的投影（Flight Review）

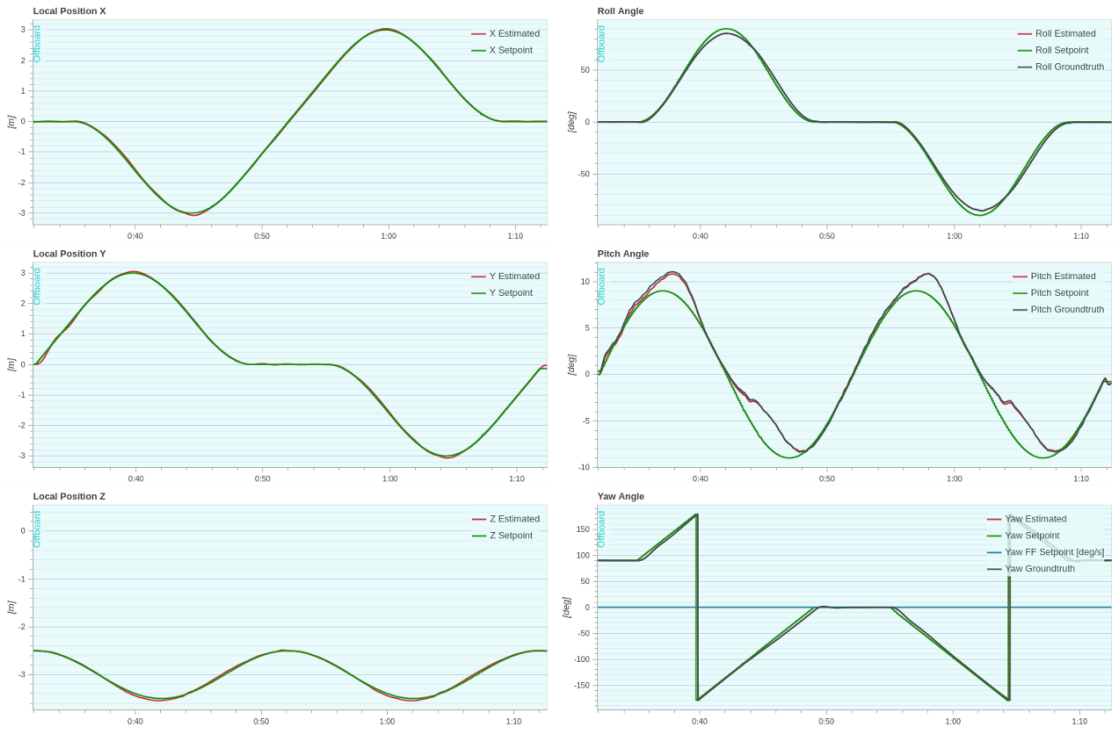

图 2-4　六自由度的轨迹追踪情况（Flight Review）

注意到图 2-4 的“Pitch Angle”子图，无人机对$P(t)$轨迹的追踪效果不理想，是由于$R(t)$的变化造成的。令轨迹$R(t) = 0,\ t \in [0, T]$，其余不变，则对$P(t)$的追

踪效果如图 2-5 所示。图中红线代表 Estimated 数据，为 EKF2 模块对各个传感器数据融合后得出的估计值。

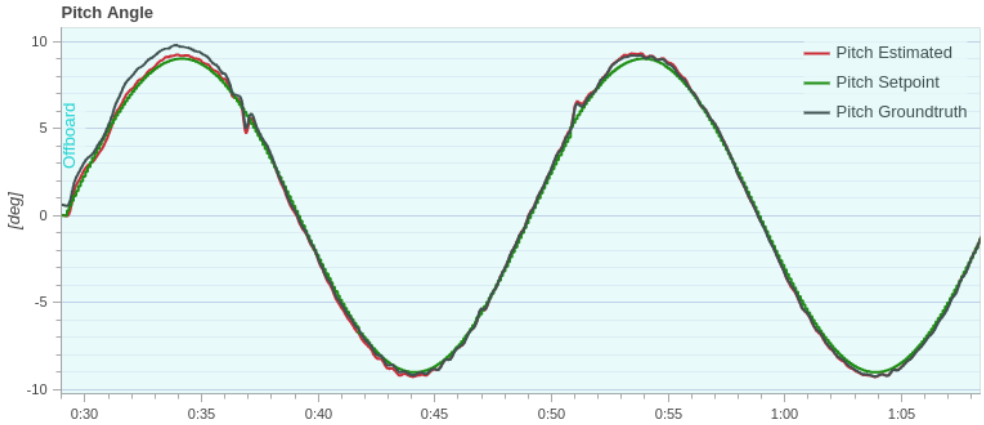

图 2-5　轨迹$P(t)$追踪情况（Flight Review）

图 2-5 所展示的追踪效果有显著改善。对比图 2-4 和图 2-5，可以归纳 Pitch 姿态的控制效果在 Roll 姿态变化时会降低，可能是在不同状态下对 Pitch 的控制需要的增益不同，可以适当调节 PID 参数，或者使用 Gain Scheduling 方法解决；也有可能是两轴之间的控制器相对独立，Pitch 轴无法预料 Roll 轴的行为，使用模型预测控制可能可以解决此问题，需要后续进一步研究。

2.2.3　无人机的遥控飞行

目前已成功在实际飞行器上完成 PID 控制算法的验证。在室内动捕提供位姿估计的环境下，飞行器可以稳定悬停，位置和姿态控制可以解除耦合。图 2-6 展示了飞行器在不同姿态下的悬停情况。

图 2-6　不同姿态下的悬停飞行

左图展示的姿态为$R=0, P=30°, Y=90°$，中图为$R=0, P=-20°, Y=135°$，右图为$R=0, P=28°, Y=180°$。可以明显看到机臂倾转的情况，符合效率最优的分配逻辑。一次典型的多模式的遥控飞行测试记录的位姿数据如下图所示。

图 2-7　不同飞行模式下的位姿数据（Flight Review）

为遥控操作设计了三种飞行模式，分别是 Altitude、Position、ACRO（Acrobatic 模式），对应图 2-7 中的浅绿、绿、黄三种底色。

Position 模式下，Roll 和 Pitch 锁定为刚进入此模式的值，Yaw 的期望值可以由摇杆输入改变；位置控制则是：在摇杆不输入信号时，锁定最近一次停止输入时的位置为期望值，若摇杆有输入则不设定期望值，而是将摇杆输入换算成加速度直接产生期望合力。

ACRO 模式下，原本用于前进和后退的摇杆变为改变 Pitch，其他期望值的产生逻辑与位置控制相同；图 2-7 展示的 ACRO 模式下，没有改变位置和 Roll 的期望值，但是通过摇杆改变了 Yaw 和 Pitch 的期望值。

Altitude 模式和 Position 模式的摇杆输入逻辑相同，但是不设定位置期望值，即位置控制不起作用；进入 Altitude 模式会使 Roll 和 Pitch 期望值阶跃为 0。

图 2-7 测试使用的 PID 参数为 PX4 提供的默认值。实际位置均保持在期望位置±0.2m 的范围内，实际姿态欧拉角基本保持在期望值的±2 度范围内。注意 Yaw 的数据范围是$[-\pi,\pi]$，所以“Yaw Angle”图中数据在π附近发生突变，但姿态并未发生突变。若仔细调节控制器参数，应该可以获得更好的控制效果。

2.3 自适应 MPC 架构和设计

2.3.1 总体架构设计

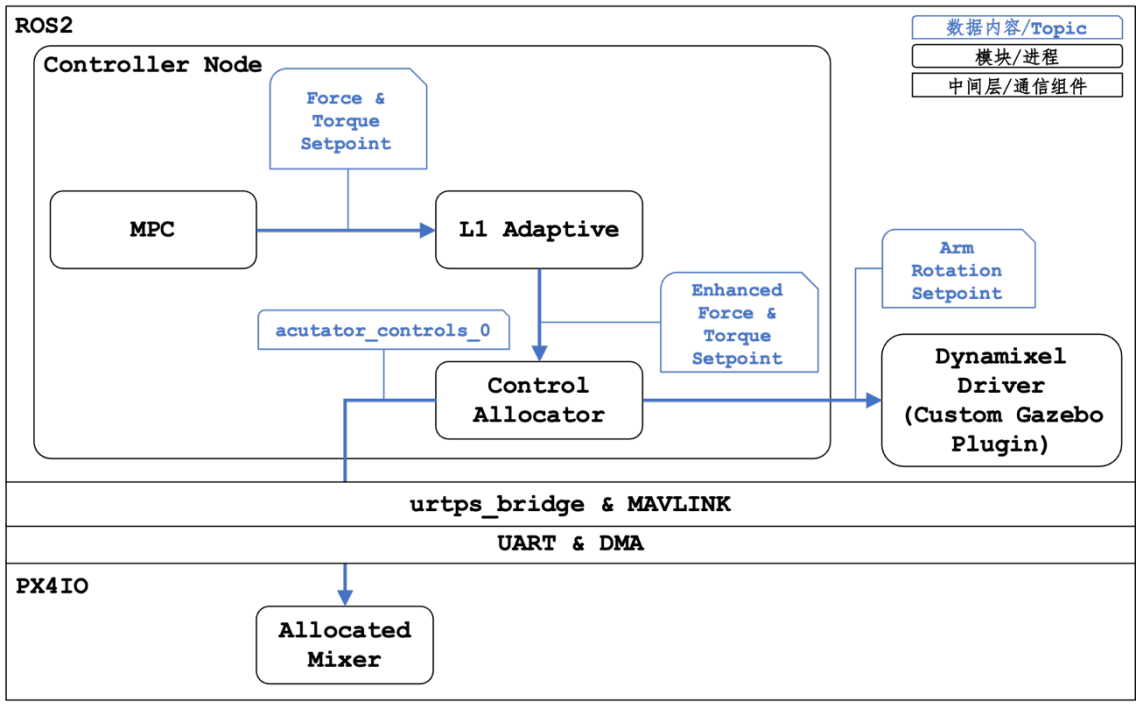

图 2-8　自适应 MPC 的总体架构设计

控制器节点包括三个小部分：MPC，L1 自适应控制器，以及控制分配器。MPC 在不考虑模型误差情况下输出执行器合力/力矩，使用它的原因是 MPC 能够对更远未来的控制目标作出反应，相比于 PID 控制器，MPC 的滞后小，对轨迹追踪的效果强。L1 自适应控制器将模型参数误差和未建模动态一起当作合力/力矩意义上的不确定性，通过自适应控制律将其估计出来，并把估计值与 MPC 的输出值相加作为最后的输出。

2.3.2 名义模型 MPC 设计

名义模型 MPC 设计中不考虑模型误差以及未建模动态，同时也不会考虑外界环境带来的干扰。这样设计出来的 MPC 是非常理想化的，在实际应用场景下可能性能并不好。本项目中采用 MPC 的动机是它能够对近未来的期望值作出响应，对比以 PID 为代表的基于当前误差的控制器来说，对期望值的变化追踪得更好，同时它能够系统性地考虑约束，在实际应用中非常有用。而模型误差带来的性能损失将在 2.3.4 节中进行讨论。

定义表 2-2 所示的记号对应的意义。

表 2-2 符号和意义

符号	维度	意义
$\boldsymbol{x}$	19×1	状态向量
$\boldsymbol{u}$	6×1	输入向量
$\boldsymbol{p}$	3×1	NED 世界坐标系下位置（m）
$\boldsymbol{q}=[\omega,x,y,z]^{\intercal}$	4×1	表示姿态的单位四元数
${}_B\boldsymbol{v}$	3×1	FRD 机体坐标系下线速度（m/s）
${}_B\boldsymbol{\omega}$	3×1	FRD 机体坐标系下角速度（rad/s）
${}_B\boldsymbol{f}_a$	3×1	FRD 机体坐标系下执行器合力（N）
${}_B\boldsymbol{\tau}_a$	3×1	FRD 机体坐标系下执行器合力矩（Nm）
m	1×1	机体质量（kg）
${}_B\boldsymbol{d}_{com}$	3×1	FRD 坐标系下质心位置（m）
$\boldsymbol{J}$	3×3	转动惯量
$\boldsymbol{e}$	18×1	代价向量
$\boldsymbol{q}_{err}$	3×1	基于四元数的姿态误差
$\boldsymbol{Q}$	18×18	中间阶段对$\boldsymbol{e}$的代价权重矩阵
$\boldsymbol{Q}_N$	18×18	终止阶段对$\boldsymbol{e}$的代价权重矩阵
$\boldsymbol{R}$	6×6	对$\boldsymbol{u}$的代价权重矩阵
$\boldsymbol{g}=[0,0,9.8]^{\intercal}$	3×1	NED 世界坐标系下重力加速度
$\boldsymbol{h}$	12×1	约束向量
$\boldsymbol{F}=[f_1,f_2\cdots f_6]^{\intercal}$	6×1	电机推力向量（N）
$\boldsymbol{B}$	6×12	合力/力矩分配矩阵
α_i	1×1	第i个机臂倾转角度（rad）

其中关键向量的定义和动态关系用下列等式表示：

$$\boldsymbol{x}=\begin{bmatrix}\boldsymbol{p}\\ \boldsymbol{q}\\ {}_B\boldsymbol{v}\\ {}_B\boldsymbol{\omega}\\ {}_B\boldsymbol{f}_a\\ {}_B\boldsymbol{\tau}_a\end{bmatrix},\qquad \boldsymbol{u}=\begin{bmatrix}{}_B\dot{\boldsymbol{f}}_a\\ {}_B\dot{\boldsymbol{\tau}}_a\end{bmatrix},\qquad \boldsymbol{e}=\begin{bmatrix}\boldsymbol{p}_{err}\\ \boldsymbol{q}_{err}\\ {}_B\boldsymbol{v}_{err}\\ {}_B\boldsymbol{\omega}_{err}\\ {}_B\boldsymbol{f}_a\\ {}_B\boldsymbol{\tau}_a\end{bmatrix} \tag{8}$$

$$\dot{\boldsymbol{x}}=\begin{bmatrix}\text{rotate}({}_B\boldsymbol{v},\boldsymbol{q})\\ \frac{1}{2}\boldsymbol{q}\otimes\begin{bmatrix}0\\ {}_B\boldsymbol{\omega}\end{bmatrix}\\ \frac{{}_B\boldsymbol{f}_a}{m}+\text{rotate}(\boldsymbol{g},\boldsymbol{q}^{-1})-{}_B\boldsymbol{\omega}\times{}_B\boldsymbol{v}\\ \boldsymbol{J}^{-1}\left({}_B\boldsymbol{\tau}_a+{}_B\boldsymbol{d}_{com}\times{}_B\boldsymbol{f}_a-{}_B\boldsymbol{\omega}\times(\boldsymbol{J}_B\boldsymbol{\omega})\right)\\ {}_B\dot{\boldsymbol{f}}_a\\ {}_B\dot{\boldsymbol{\tau}}_a\end{bmatrix}\triangleq\text{dyn}(\boldsymbol{x},\boldsymbol{u}) \tag{9}$$

$$\boldsymbol{h}=\begin{bmatrix}{}_B\boldsymbol{v}\\ {}_B\boldsymbol{\omega}\\ \boldsymbol{F}\end{bmatrix}=\begin{bmatrix}{}_B\boldsymbol{v}\\ {}_B\boldsymbol{\omega}\\ \text{allocate}({}_B\boldsymbol{f}_a,{}_B\boldsymbol{\tau}_a,\boldsymbol{B})\end{bmatrix} \tag{10}$$

$$\begin{bmatrix}{}_B\boldsymbol{f}_a\\ {}_B\boldsymbol{\tau}_a\end{bmatrix}=\boldsymbol{B}[\sin(\alpha_1)f_1,\cos(\alpha_1)f_1\cdots\sin(\alpha_6)f_6,\cos(\alpha_6)f_6]^{\intercal}\triangleq\boldsymbol{B}\boldsymbol{y} \tag{11}$$

式(7)中的位置误差$\boldsymbol{p}_{err}$、线速度误差${}_B\boldsymbol{v}_{err}$和角速度误差${}_B\boldsymbol{\omega}_{err}$均为当前值与参考值相减。姿态误差则满足下式：

$$\boldsymbol{q}\otimes\boldsymbol{q}_{ref}^{-1}=\begin{bmatrix}w\\ \boldsymbol{q}_{err}\end{bmatrix} \tag{12}$$

式(8)中的$\text{rotate}(\boldsymbol{v},\boldsymbol{q})$表示基于单位四元数$\boldsymbol{q}$对三维向量$\boldsymbol{v}$进行旋转，具体算法是将$\boldsymbol{v}$扩展成$\boldsymbol{v}_q=[0,\boldsymbol{v}]^{\intercal}$，进行运算得到$\boldsymbol{v}'_q=\boldsymbol{q}\otimes\boldsymbol{v}_q\otimes\boldsymbol{q}^{-1}$，取$\boldsymbol{v}'_q$的向量部分作为输出。

式(9)中的$\text{allocate}({}_B\boldsymbol{f}_a,{}_B\boldsymbol{\tau}_a,\boldsymbol{B})$的算法是利用$\boldsymbol{B}$的伪逆$\boldsymbol{B}^{\dagger}$从式(10)中获取$\boldsymbol{y}$，然后由$\boldsymbol{y}$求出$\boldsymbol{F}$：

$$\boldsymbol{y}=\begin{bmatrix}y_1\\ \vdots\\ y_{12}\end{bmatrix}=\boldsymbol{B}^{\dagger}\begin{bmatrix}{}_B\boldsymbol{f}_a\\ {}_B\boldsymbol{\tau}_a\end{bmatrix} \tag{13}$$

$$\boldsymbol{F}=\begin{bmatrix}\sqrt{y_1^2+y_2^2} & \cdots & \sqrt{y_{11}^2+y_{12}^2}\end{bmatrix}^{\intercal} \tag{14}$$

基于上述定义和关系，可以构造 MPC 中的 OCP（Optimal Control Problem）求解预测区间内的系统输入$[\boldsymbol{u}_0 \cdots \boldsymbol{u}_{N-1}]$，在满足约束的情况下使代价函数最小，完整列写如下：

$$\min_{[\boldsymbol{u}_0 \cdots \boldsymbol{u}_{N-1}]} \sum_{k=0}^{N-1} (\boldsymbol{e}_k^{\mathrm{T}} \boldsymbol{Q} \boldsymbol{e}_k + \boldsymbol{u}_k^{\mathrm{T}} \boldsymbol{R} \boldsymbol{u}_k) + \boldsymbol{e}_N^{\mathrm{T}} \boldsymbol{Q}_N \boldsymbol{e}_N \tag{15}$$

s.t.

$$\begin{gathered} \boldsymbol{h}_k \in [\boldsymbol{h}_{lb}, \boldsymbol{h}_{ub}] \\ \boldsymbol{u}_k \in [\boldsymbol{u}_{lb}, \boldsymbol{u}_{ub}] \\ \dot{\boldsymbol{x}} = \mathrm{dyn}(\boldsymbol{x}, \boldsymbol{u}) \\ \boldsymbol{x}_0 = \boldsymbol{x}(t) \end{gathered}$$

MPC 的基本思想是滚动优化，取 OCP 的最初解$\boldsymbol{u}_0$作为系统的输入。注意到这里的系统输入的意义是合力/力矩的变化率，这里假定它在控制周期内值恒定，于是 MPC 的实际输出的合力/力矩可以由上一控制周期的值加上本周期的$\boldsymbol{u}$乘以控制周期而得到。至此，理想条件下的 MPC 已经构建完毕，其输出的合力/力矩随后由控制分配得到执行器的指令。

2.3.3 MPC 的实现

acados 是为非线性最优控制设计的求解框架。用 acados 能够快速、模块化地生成 OCP 求解器。主要工作流程是：一，用 CasADi 定义模型的变量、动态和各种函数；二，在 Python 或者 MATLAB 中使用 acados 提供的接口定义 OCP；三，生成求解器的 C 代码；四，在应用程序中调用求解器，构造 MPC。

本课题的 OCP 属于 NLP（Non-Linear Programming），在 acados 框架中是使用 SQP-RTI（Sequential Quadratic Programming with Real Time Iteration）的方法将 NLP 转化为一系列 QP（Quadratic Programming）问题，然后用高性能的 QP 求解器进行求解。用 CasADi 定义模型的变量、动态以及各种函数的关键是正确使用其提供的符号系统。CasADi 未提供四元数相关的符号运算，需自行实现。

生成求解器 C 代码时，需要设置求解器有关的选项。使用的一些关键设置是：一，使用 Implicit Runge-Kutta Method（RK4）对动态方程进行离散化；二，使用对实时迭代优化过的 SQP 求解 NLP；三，使用 Full Condensing HPIPM 作为 QP 求解器；四，不对寻找全局最优解作严格要求，因为实时优化强调的在时间限定内得到足够好的解即可。

在应用程序中调用求解器具体来说就是在 MPC 节点中调用上一步生成的代码中的函数，得到需要的$\boldsymbol{u}_0$以及合力/力矩。

2.3.4 L1 自适应控制

获得名义 MPC 之后，为了增加 MPC 的鲁棒性和准确性，需要考虑 MPC 模型中的误差/不确定性（Uncertainty）。参数误差、未建模动态、底层传感器对指令的跟踪误差可以统一成合力/力矩上的不确定性考虑，现定义如下表所示的额外符号。

表 2-3 补充符号和意义

符号	维度	意义
${}_B\boldsymbol{f}_{L1}$	3×1	L1 控制器的输出力（N）
${}_B\boldsymbol{f}_{\Delta}$	3×1	力不确定性（N）
${}_B\boldsymbol{\tau}_{L1}$	3×1	L1 控制器的输出力矩（Nm）
${}_B\boldsymbol{\tau}_{\Delta}$	3×1	力矩不确定性（Nm）
$\boldsymbol{u}_{L1}$	6×1	L1 控制器的总输出向量
$\boldsymbol{\sigma}$，$\hat{\boldsymbol{\sigma}}$	6×1	不确定性向量与其估计值
$\boldsymbol{z}$，$\hat{\boldsymbol{z}}$	6×1	简化的状态向量与其估计值
$\mathbb{I}_i$	$i\times i$	单位矩阵
$\mathbb{O}_i$	$i\times i$	零矩阵
$\boldsymbol{\mathcal{A}}$	6×1	理想系统动态
$\boldsymbol{\mathcal{B}}$	6×6	输入增益矩阵
T_s	1×1	离散化周期

式(8)为系统动态的表达式。不确定性只出现在速度对时间的导数等式上；而其他的等式关系都是严格的数学求导，不存在不确定性（假设状态估计算法足够好），况且此处假设不确定性的表达形式为力和力矩，所以在设计 L1 自适应控制器的时候可以不考虑它们。按照图 2-8 所表述的架构将不确定性和 L1 控制器的补偿值放到系统动态中，有：

$$\dot{\boldsymbol{z}}=\begin{bmatrix}{}_B\dot{\boldsymbol{v}}\\ {}_B\dot{\boldsymbol{\omega}}\end{bmatrix}=\begin{bmatrix}\dfrac{{}_B\boldsymbol{f}_a+{}_B\boldsymbol{f}_{L1}+{}_B\boldsymbol{f}_{\Delta}}{m}+\mathrm{rotate}(\boldsymbol{g},\boldsymbol{q}^{-1})-{}_B\boldsymbol{\omega}\times{}_B\boldsymbol{v}\\ \boldsymbol{J}^{-1}\left({}_B\boldsymbol{\tau}_a+{}_B\boldsymbol{\tau}_{L1}+{}_B\boldsymbol{\tau}_{\Delta}+{}_B\boldsymbol{d}_{com}\times{}_B\boldsymbol{f}_a-{}_B\boldsymbol{\omega}\times(\boldsymbol{J}_B\boldsymbol{\omega})\right)\end{bmatrix} \tag{16}$$

$$\dot{\boldsymbol{z}}=\underbrace{\begin{bmatrix}\dfrac{{}_B\boldsymbol{f}_a}{m}+\mathrm{rotate}(\boldsymbol{g},\boldsymbol{q}^{-1})-{}_B\boldsymbol{\omega}\times{}_B\boldsymbol{v}\\ \boldsymbol{J}^{-1}\left({}_B\boldsymbol{\tau}_a+{}_B\boldsymbol{d}_{com}\times{}_B\boldsymbol{f}_a-{}_B\boldsymbol{\omega}\times(\boldsymbol{J}_B\boldsymbol{\omega})\right)\end{bmatrix}}_{\boldsymbol{\mathcal{A}}}+\underbrace{\begin{bmatrix}\dfrac{\mathbb{I}_3}{m} & \mathbb{O}_3\\ \mathbb{O}_3 & \boldsymbol{J}^{-1}\end{bmatrix}}_{\boldsymbol{\mathcal{B}}}\left(\underbrace{\begin{bmatrix}{}_B\boldsymbol{f}_{L1}\\ {}_B\boldsymbol{\tau}_{L1}\end{bmatrix}}_{\boldsymbol{u}_{L1}}+\underbrace{\begin{bmatrix}{}_B\boldsymbol{f}_{\Delta}\\ {}_B\boldsymbol{\tau}_{\Delta}\end{bmatrix}}_{\boldsymbol{\sigma}}\right) \tag{17}$$

为了表示方便，考虑了不确定性之后，系统动态可以更简洁地写成：

$$\dot{\boldsymbol{z}} = \boldsymbol{\mathcal{A}} + \boldsymbol{\mathcal{B}}(\boldsymbol{u}_{L1} + \boldsymbol{\sigma}) \tag{18}$$

其中$\boldsymbol{\mathcal{A}}$是时变的，数据与 MPC 共享。$\boldsymbol{\mathcal{B}}$则是一个常数输入增益矩阵。

在欠驱动多旋翼上，不确定性一般被分为匹配的不确定性和不匹配的不确定性，因为欠驱动多旋翼无法产生机体 XY 平面（FRD 配置）上的力，于是出现在 XY 平面上的不确定性无法直接被抵消，这一部分的不确定性被称作为不匹配的不确定性。而过驱动六旋翼可以通过执行器力矩直接抵消 XY 平面上的不确定性，所以此方面的设计实际上比欠驱动型简单。

L1 自适应的基本思想是对$\boldsymbol{\sigma}$进行估计，即获取$\hat{\boldsymbol{\sigma}}$，然后用某种方法通过$\boldsymbol{u}_{L1}$将其抵消。将$\hat{\boldsymbol{\sigma}}$带入式(17)：

$$\dot{\hat{\boldsymbol{z}}} = \widehat{\boldsymbol{\mathcal{A}}} + \boldsymbol{\mathcal{B}}(\boldsymbol{u}_{L1} + \hat{\boldsymbol{\sigma}}) \tag{19}$$

希望$\hat{\boldsymbol{z}} \to \boldsymbol{z}$，$\hat{\boldsymbol{\sigma}} \to \boldsymbol{\sigma}$，即希望估计值向真实值收敛，这样状态的估计值也向真实值收敛，伴随而来同理有$\widehat{\boldsymbol{\mathcal{A}}} \to \boldsymbol{\mathcal{A}}$。引入记号$\tilde{\boldsymbol{z}} = \hat{\boldsymbol{z}} - \boldsymbol{z}$，$\tilde{\boldsymbol{\sigma}} = \hat{\boldsymbol{\sigma}} - \boldsymbol{\sigma}$构造如下线性系统：

$$\dot{\tilde{\boldsymbol{z}}} = \boldsymbol{A}\tilde{\boldsymbol{z}} + \boldsymbol{\mathcal{B}}\tilde{\boldsymbol{\sigma}} \tag{20}$$

其中$\boldsymbol{A}$为 Hurwitz 矩阵，这样能保证这个系统的平衡点$(\tilde{\boldsymbol{z}}, \tilde{\boldsymbol{\sigma}}) = (0,0)$为渐进稳定的，以此作为能估计出不确定性的一个先决条件。注意此处仅有$\boldsymbol{A}$为 Hurwitz 矩阵是不能保证$(\tilde{\boldsymbol{z}}, \tilde{\boldsymbol{\sigma}})$的有界性的，也不能对系统的状态转移情况作出判定。

现假设能得到可靠的$\hat{\boldsymbol{\sigma}}$，即$\hat{\boldsymbol{\sigma}} = \boldsymbol{\sigma}$，并以此为基础讨论消去不确定性影响的方法。朴素的思想是令$\boldsymbol{u}_{L1} + \hat{\boldsymbol{\sigma}} = 0$，这样式(17)变为$\dot{\boldsymbol{z}} = \boldsymbol{\mathcal{A}}$，即完全成为理想的样子了。但是考虑到现实的系统，执行器是没办法对超出自身能力范围的不确定性补偿的：比如有些$\boldsymbol{\sigma}$频率太高，系统从物理上没办法补偿它们，算法却还将它们一起纳入计算$\boldsymbol{u}_{L1}$，这样往往引起系统震荡，产生消极影响，限制了算法对$\boldsymbol{\sigma}$感知的增益，也就间接地使对不确定性的估计速度下降。L1 自适应理论给出的解决办法是令：

$$\boldsymbol{u}_{L1} = \mathrm{LPF}(\hat{\boldsymbol{\sigma}}) \tag{21}$$

即对$\hat{\boldsymbol{\sigma}}$低通滤波之后作为$\boldsymbol{u}_{L1}$，通过适当调节截止频率，能够剔除上述消极影响，并增大不确定性感知增益（自适应增益），以此提高自适应性能。

接下来讨论获取$\hat{\boldsymbol{\sigma}}$的方法，式(19)本身无法完成对$\hat{\boldsymbol{z}}$和$\hat{\boldsymbol{\sigma}}$的更新，因为输入$\tilde{\boldsymbol{\sigma}}$无从获取。而利用式(18)形式的所谓“状态预测器”，则可在式(20)，和后面要设计的自适应控制率$\hat{\boldsymbol{\sigma}} = \mathrm{adapt}(\tilde{\boldsymbol{z}})$的帮助下更新$\hat{\boldsymbol{z}}$和$\hat{\boldsymbol{\sigma}}$。但式(18)中存在的$\widehat{\boldsymbol{\mathcal{A}}}$会使后续分析

较为复杂，所以需要利用式(19)进行一次代换，将式(17)和式(19)相加得到另一种状态预测器的形式：

$$\dot{\hat{\boldsymbol{z}}} = \boldsymbol{\mathcal{A}} + \boldsymbol{\mathcal{B}}(\boldsymbol{u}_{L1} + \hat{\boldsymbol{\sigma}}) + \boldsymbol{A}\tilde{\boldsymbol{z}} \tag{22}$$

于是剩下的目标为合理地设计自适应控制率$\hat{\boldsymbol{\sigma}} = \text{adapt}(\tilde{\boldsymbol{z}})$。L1 自适应理论给出了在离散情况下的结果：

$$\hat{\boldsymbol{\sigma}}_k = -\boldsymbol{\mathcal{B}}^{-1}(e^{\boldsymbol{A}T_s} - \mathbb{I}_6)^{-1}\boldsymbol{A}e^{\boldsymbol{A}T_s}\tilde{\boldsymbol{z}}_k \tag{23}$$

至此 L1 自适应 MPC 设计完毕。至于式(22)的详细推导，是后续工作之一，此处是为了文档的完整性而不加证明地列举结论。

3 后期拟完成的研究工作及进度安排

3.1 后期拟完成的研究工作

后续需要完成的工作主要是继续自适应 MPC 的开发和部署。拟分为下列几步走：测试求解器、在 ROS2 中以 C++节点的形式实现 MPC、实现 L1 自适应控制律对 MPC 进行补偿、将控制器与实际 PX4 子系统进行对接。

实现上述目标后即进行试验，依靠轨迹规划课题的相关成果，测试轨迹追踪效果，对比不同控制器的性能。随后总结成果，并写论文。

3.2 进度安排

具体进度安排如下：至 4 月 1 日完成求解器的测试和 MPC 的 C++节点，至 5 月 15 日实现 L1 自适应控制器，至 6 月 1 日完成控制器节点与 PX4 子系统的对接。在最终截止日期前完成实际测试和论文编写。

4　存在的困难及解决方案

4.1　存在的困难

一，工作量比较大，没有直接可用的软件包。

二，暂未弄清楚如何选取非线性 MPC 中的$\boldsymbol{Q}_N$，以及 OCP 中的$\boldsymbol{u}$上下限。

三，L1 自适应控制理论比较复杂，相关资料较少，无开源代码参考，复现较为困难。

4.2　解决方案

针对目前存在的问题和困难，解决办法可以是：修改原定计划/目标，或增加人手/效率/时数。

5　论文按时完成的可能性

本课题目前已经完成了基于 PX4 的 PID 控制器的仿真和实际飞行测试。

目前正在实现未进行自适应补偿的 MPC，近期未来将要进行 MPC 求解器的测试以及 L1 自适应控制器的实现。论文按期完成有一定的困难，但会尽量克服，尽力实现按期完成的目标。