

دانشگاه صنعتی خواجه نصیرالدین طوسی
دانشکده مهندسی برق - گروه مهندسی مکاترونیک

مکاترونیک ۱

استاد درس: دکتر سعید خان کلانتری
پروژه نهایی: شبیه‌سازی و پیاده سازی کنترل کننده
 مقاوم Fuzzy-PID برای کوادروتور ۶ درجه آزادی

نام و نام خانوادگی	کوثر امین جعفری
شماره دانشجویی	۴۰۳۰۲۳۷۴
نام و نام خانوادگی	محمد رضا شریفی
شماره دانشجویی	۴۰۳۱۵۸۰۴
تاریخ	شهریور ۱۴۰۴

بِسْمِ اللّٰهِ الرَّحْمٰنِ الرَّحِيْمِ

فهرست مطالب

۴	فهرست تصاویر
۶	فهرست جداول
۷	فهرست علائم و نشانه‌ها
۱۰	۱ مقدمه
۱۲	۲ مرور ادبیات
۱۲	۱.۲ کوادروتور به عنوان یک بستر پژوهشی چالش‌برانگیز در حوزه کنترل
۱۲	۱.۱.۲ اهمیت و کاربردهای روزافزون کوادروتورها
۱۳	۲.۱.۲ مدل‌سازی ریاضی و دینامیک پرواز شش درجه آزادی
۱۴	۳.۱.۲ چالش‌های کلیدی در کنترل مقاوم
۱۵	۲.۲ سیر تکامل راهبردهای کنترل: از خطی‌سازی تا یادگیری
۱۵	۱.۲.۲ تکنیک‌های کنترل خطی بنیادین
۱۶	۲.۲.۲ ظهور راهبردهای کنترل غیرخطی و مقاوم
۱۷	۳.۲.۲ رویکردهای پیشرفته برای مقابله با عدم قطعیت
۱۹	۳.۲ تحلیل تطبیقی، مباحثات جاری، و راهبردهای ترکیبی
۱۹	۱.۳.۲ ارزیابی عملکرد و مصالحه
۱۹	۲.۳.۲ مباحثات جاری: کنترل مبتنی بر مدل در مقابل کنترل مبتنی بر داده
۲۰	۳.۳.۲ روند غالب: معماری‌های کنترل ترکیبی
۲۲	۴.۲ شکاف‌های پژوهشی و چشم‌اندازهای آینده
۲۲	۱.۴.۲ شکاف‌های شناسایی شده در پژوهش‌های پیشین
۲۳	۲.۴.۲ مسیرهای تحقیقاتی آینده
۲۴	۳ انتخاب کننده و شرح ویژگی‌ها
۲۴	۱.۳ مدل دینامیکی غیرخطی
۲۶	۲.۳ تحلیل ویژگی‌های ذاتی سیستم
۲۷	۳.۳ چالش‌های عملیاتی: اغتشاشات و عدم قطعیت‌ها
۲۸	۴.۳ کنترل کننده Fuzzy-PID
۲۸	۱.۴.۳ ارزیابی انتقادی کنترل کننده PID
۲۹	۲.۴.۳ مقدمه‌ای بر منطق فازی

۳۲	۴	شبيه‌سازی کنترل‌کننده در نرم افزار متلب/سيمولينك
۳۳	۱.۴	مراحل ابتدائي شروع شبيه‌سازی
۳۳	۱.۱.۴	اضافه کردن فايل‌ها به محيط متلب
۳۳	۲.۱.۴	بارگذاري مسیر برای کنترل موقعیت
۳۹	۳.۱.۴	بارگذاري يا ساخت مدل کوادراتور
۴۴	۴.۱.۴	بارگذاري شرایط اوليه کوادراتور
۴۶	۲.۴	طراحی کنترل کننده PID برای موقعیت
۵۲	۱.۲.۴	نتایج طراحی کنترل کننده موقعیت برای مسیر دایره‌ای
۵۴	۲.۲.۴	نتایج طراحی کنترل کننده موقعیت برای مسیر هشتی
۵۶	۳.۲.۴	نتایج طراحی کنترل کننده موقعیت برای مسیر لوزی
۵۹	۴.۲.۴	تیون کردن ضرایب کنترل کننده
۶۸	۳.۴	طراحی کنترل کننده PID برای وضعیت
۷۰	۴.۴	طراحی کنترل کننده Fuzzy-PID برای وضعیت
۷۸	۵.۴	مقایسه بین PID و Fuzzy-PID
۸۰	۵	آزمون روي سخت‌افزار واقعي
۸۰	۱.۵	معرفی کلي سنسور MPU-9250
۸۵	۲.۵	شرح کد MPU-9250
۸۵	۱.۲.۵	فايل سرآيند (mpu9250.h)
۸۵	۲.۲.۵	فايل منبع (mpu9250.c)
۸۷	۳.۵	شرح کد basic
۸۷	۱.۳.۵	فايل سرآيند (basic.h)
۸۷	۲.۳.۵	فايل منبع (basic.c)
۸۸	۴.۵	شرح کد controller
۸۸	۱.۴.۵	فايل سرآيند (controller.h)
۸۹	۲.۴.۵	فايل منبع (controller.c)
۹۰	۵.۵	شرح کد main
۹۰	۱.۵.۵	فاز راهاندازی (Initialization)
۹۱	۲.۵.۵	حلقه کنترل اصلی (The Main Control Loop)
۹۲	۳.۵.۵	مدیریت وقفه‌ها (Interrupt Handling)
۹۵	۶	نتيجه‌گيري
۹۵		مراجع

فهرست تصاویر

۳۳	نحوه اضافه کردن فایل ها به مسیر	۱.۴
۳۴	کد استفاده شده برای ایجاد مسیر دایره ای	۲.۴
۳۵	کد استفاده شده برای ایجاد مسیر هشتی	۳.۴
۳۶	کد استفاده شده برای ایجاد مسیر لوزی	۴.۴
۳۸	شماتیک مسیرهای ایجاد شده برای آزمون عملکرد کوادراتور	۵.۴
۳۹	رابط گرافیک کاربری برای وارد کردن ابعاد کوادراتور	۶.۴
۴۱	مشخصات فیزیکی کوادراتور	۷.۴
۴۲	کوادراتور استفاده شده برای تست	۸.۴
۴۵	شرایط اولیه کوادراتور	۹.۴
۴۶	شماتیک کلی فایل شبیه سازی سیمولینک کنترل کننده موقعیت کوادراتور	۱۰.۴
۴۷	بلوک بارگذاری مسیر، مدل کوادراتور و شرایط اولیه	۱۱.۴
۴۸	بلوک انیمیشن، نمودارها و ساخت مدل جدید کوادراتور	۱۲.۴
۴۹	رابط گرافیک کاربری برای نمایش عملکرد کوادراتور	۱۳.۴
۵۰	بلوک کنترل موقعیت در نرم افزار سیمولینک	۱۴.۴
۵۱	بلوک کنترل وضعیت در نرم افزار سیمولینک	۱۵.۴
۵۲	عملکرد کوادراتور در مسیر دایره ای	۱۶.۴
۵۳	عملکرد کوادراتور در مسیر دایره ای	۱۷.۴
۵۴	عملکرد کوادراتور در مسیر هشتی	۱۸.۴
۵۵	عملکرد کوادراتور در مسیر هشتی	۱۹.۴
۵۶	عملکرد کوادراتور در مسیر لوزی	۲۰.۴
۵۷	عملکرد کوادراتور در مسیر لوزی	۲۱.۴
۵۹	عملکرد کوادراتور تیون شده در مسیر دایره ای	۲۲.۴
۶۰	عملکرد کوادراتور تیون شده در مسیر دایره ای	۲۳.۴
۶۰	عملکرد کوادراتور تیون شده در مسیر دایره ای	۲۴.۴
۶۱	عملکرد کوادراتور تیون شده در مسیر هشتی	۲۵.۴
۶۱	عملکرد کوادراتور تیون شده در مسیر لوزی	۲۶.۴
۶۲	عملکرد کوادراتور تیون شده در مسیر لوزی	۲۷.۴
۶۴	ضرایب نهایی کنترل کننده	۲۸.۴
۶۶	انیمیشن کوادراتور در مسیر دایره ای	۲۹.۴
۶۶	انیمیشن کوادراتور در مسیر هشتی	۳۰.۴
۶۷	انیمیشن کوادراتور در مسیر لوزی	۳۱.۴

۶۸	شماتیک کلی فایل شبیه‌سازی سیمولینک کنترل کننده وضعیت کوادراتور	۳۲.۴
۶۹	نتیجه تست کنترل کننده وضعیت	۳۳.۴
۷۱	توابع عضویت برای خطأ و مشتق خطأ	۳۴.۴
۷۲	توابع عضویت برای α	۳۶.۴
۷۲	توابع عضویت برای K'_d و K'_p	۳۵.۴
۷۳	قوانين تنظیم فازی برای K'_p	۳۷.۴
۷۴	قوانين تنظیم فازی برای K'_d	۳۸.۴
۷۴	قوانين تنظیم فازی برای α	۳۹.۴
۷۶	فرآیند استنباط یک قانون فازی	۴۰.۴
۷۷	نتیجه تست کنترل کننده وضعیت برای کنترل کننده PID-Fuzzy	۴۱.۴
۹۳	برد استفاده شده برای راهاندازی	۱.۵
۹۴	کواد به همراه برد استفاده شده	۲.۵

فهرست جداول

۱	فهرست جامع علائم و نمادها برای مدل‌سازی و کنترل کوادراتور ۶ درجه آزادی	۷
۱.۲	مقایسه راهبردهای کنترل کلیدی برای کوادراتور	۲۱
۱.۴	مقایسه معیارهای عملکرد برای کنترل‌کننده‌های <i>Fuzzy – PID</i> و <i>PID</i>	۷۸

فهرست علائم و نشانه‌ها

جدول ۱: فهرست جامع علائم و نمادها برای مدل‌سازی و کنترل کوادراتور ۶ درجه آزادی

توضیحات	نماد
بخش ۱: متغیرهای حالت و سینماتیک	
چارچوب مرجع اینرسی یا جهانی (Inertial Frame)	I
چارچوب مرجع متصل به بدن (Body Frame)	B
بردار موقعیت خطی مرکز جرم کوادراتور در چارچوب اینرسی.	$\xi = [x, y, z]^T$
بردار وضعیت زاویه‌ای (زوایای اویلر) در چارچوب اینرسی. زاویه غلتش (Roll) حول محور x بدن. زاویه پیچش (Pitch) حول محور y بدن. زاویه انحراف (Yaw) حول محور z بدن.	$\eta = [\phi, \theta, \psi]^T$ ϕ θ ψ
بردار سرعت خطی در چارچوب بدن.	$v = [u, v, w]^T$
بردار سرعت خطی در چارچوب اینرسی.	$\dot{\xi} = [\dot{x}, \dot{y}, \dot{z}]^T$
بردار سرعت زاویه‌ای در چارچوب بدن. سرعت زاویه‌ای حول محور x بدن (سرعت غلتش). سرعت زاویه‌ای حول محور y بدن (سرعت پیچش). سرعت زاویه‌ای حول محور z بدن (سرعت انحراف).	$\Omega = [p, q, r]^T$ p q r
ماتریس دوران برای تبدیل از چارچوب بدن به چارچوب اینرسی.	R
بخش ۲: پارامترهای فیزیکی، نیروها و گشتاورها	
ادامه در صفحه بعد	

جدول ۱: ادامه از صفحه قبل

توضیحات	نماد
جرم کل کوادراتور (واحد: kg).	m
شتاب گرانش (واحد: m/s^2).	g
تансور اینرسی کوادراتور حول مرکز جرم. ممان‌های اینرسی حول محورهای اصلی x, y, z بدن.	$I = diag(I_{xx}, I_{yy}, I_{zz})$ I_{xx}, I_{yy}, I_{zz}
فاصله مرکز هر موتور تا مرکز جرم کوادراتور (طول بازو).	l
سرعت زاویه‌ای موتور i -ام (واحد: rad/s) ($i=1, 2, 3, 4$).	ω_i
نیروی تراست (پیشران) تولیدی توسط موتور i -ام.	$T_i = k_T \omega_i^2$
مجموع نیروی تراست کل در راستای محور z بدن.	$T = \sum_{i=1}^4 T_i$
گشتاور پسار (Drag Torque) تولیدی توسط موتور i -ام.	$D_i = k_D \omega_i^2$
ضریب تراست موتورها.	k_T or b
ضریب پسار (درگ) موتورها.	k_D or d
بردار گشتاورهای کنترلی اعمالی به بدن. گشتاور کنترلی حول محور x (گشتاور غلتش). گشتاور کنترلی حول محور y (گشتاور پیچش). گشتاور کنترلی حول محور z (گشتاور انحراف).	$\tau = [\tau_\phi, \tau_\theta, \tau_\psi]^T$ τ_ϕ τ_θ τ_ψ

بخش ۳: ورودی‌های کنترلی

ورودی کنترلی متناظر با مجموع نیروی تراست کل (T).	U_1
ورودی کنترلی متناظر با گشتاور غلتش (τ_ϕ).	U_2
ورودی کنترلی متناظر با گشتاور پیچش (τ_θ).	U_3
ورودی کنترلی متناظر با گشتاور انحراف (τ_ψ).	U_4

ادامه در صفحه بعد

جدول ۱: ادامه از صفحه قبل

توضیحات	نماد			
رابطه ماتریسی بین ورودی‌های کنترلی و مربع سرعت زاویه‌ای موتورها.	$\begin{bmatrix} U_1 \\ U_2 \\ U_3 \\ U_4 \end{bmatrix} = \begin{bmatrix} \dots \end{bmatrix}$		ω_1^2	ω_2^2

بخش ۴: کنترل کننده PID استاندارد

.(Setpoint/Reference) مقدار مطلوب یا نقطه تنظیم	$r(t)$ or $x_d(t)$
.(Measured Value) مقدار اندازه‌گیری شده یا واقعی فرآیند	$y(t)$ or $x(t)$
سیگنال خطأ.	$e(t) = r(t) - y(t)$
.(Proportional Gain) بهره تناسبی	K_p
.(Integral Gain) بهره انتگرالی	K_i
.(Derivative Gain) بهره مشتقی	K_d
خروجی کنترل کننده PID	$u(t)$
معادله استاندارد کنترل کننده PID	$u(t) = K_p e(t) + K_i \int_0^t e(\tau) d\tau + K_d \frac{de(t)}{dt}$

فصل ۱

مقدمه

در گذار به عصر فناوری‌های هوشمند و سیستم‌های خودران، وسایل نقلیه هوایی بدون سرنشین، و در میان آن‌ها کوادراتورها، نقشی کلیدی و انکارناپذیر یافته‌اند. این پرندگان که زمانی عمدتاً در حوزه سرگرمی و پژوهش‌های آکادمیک محدود بودند، اکنون به سرعت در حال تبدیل شدن به ستون فقرات صنایع آینده‌نگر، از جمله لجستیک هوشمند، نظارت بر زیرساخت‌های حیاتی، کشاورزی دقیق و عملیات‌های امداد و نجات هستند. این جهش کاربردی، ریشه در یک ویژگی دوگانه و جذاب دارد: ساختار مکانیکی به ظاهر ساده که کنترل آن تنها از طریق تغییر سرعت روتورها صورت می‌پذیرد، در حالی که در پس این سادگی، یک دینامیک پرواز بسیار پیچیده، غیرخطی و چالش‌برانگیز نهفته است. همین ماهیت دوگانه، کوادراتور را به بستری ایده‌آل برای توسعه و اعتبارسنجی پیشرفته‌ترین نظریه‌های مهندسی کنترل تبدیل کرده است.

با این حال، انتقال موقتی آمیز کوادراتورها از محیط‌های کنترل شده آزمایشگاهی به دنیای واقعی، با یک مانع اساسی روبروست: عدم قطعیت. مدل ریاضی که برای توصیف حرکت شش درجه آزادی این پرندگان به کار می‌رود، هرچقدر هم که دقیق باشد، همواره تقریبی از واقعیت است. در عمل، کوادراتور با مجموعه‌ای از عدم قطعیت‌های داخلی و اغتشاشات خارجی مواجه است. تغییرات پیش‌بینی‌نشده در پارامترهای کلیدی مانند جرم و اینرسی به دلیل حمل بارهای متغیر، و همچنین دینامیک‌های مدل‌سازی‌نشده مانند اثرات آیرودینامیکی پیچیده، عملکرد سیستم را تحت تأثیر قرار می‌دهند. مهم‌تر از آن، اغتشاشات خارجی و غیرقابل پیش‌بینی، به ویژه تنببداهای و تلاطم‌های جوی، به دلیل نسبت نیروی پیشران به وزن پایین این پرندگان، می‌توانند به سادگی پایداری آن‌ها را مختل کرده و منجر به انحراف شدید از مسیر مطلوب شوند. بنابراین، مسئله اصلی دیگر صرفاً پایدارسازی یک مدل ایده‌آل نیست، بلکه تضمین عملکرد مقاوم، ایمن و قابل اعتماد در یک محیط عملیاتی پویا و نامشخص است.

ضرورت انجام این پژوهش دقیقاً از همین نقطه برمی‌خیزد. علی‌رغم پیشرفتهای گسترده در حوزه کنترل کوادراتور، شکاف‌های قابل توجهی میان نظریه و عمل باقی مانده است. اولاً، فقدان یک چارچوب استاندارد برای محکزنی و مقایسه منصفانه راهبردهای کنترلی مختلف، پیشرفت این حوزه را با چالش مواجه کرده است. بسیاری از نتایج منتشرشده به دلیل تفاوت در شرایط آزمایشی، سخت‌افزار و معیارهای ارزیابی، قابل تکرار یا مقایسه مستقیم نیستند. ثانیاً، بسیاری از کنترل‌کننده‌ها برای مانورهای تهاجمی و پرواز در محیط‌های ناشناخته و پیچیده، که برای کاربردهای نسل آینده حیاتی است، طراحی نشده‌اند. و در نهایت، با حرکت به سمت الگوریتم‌های کنترلی پیچیده‌تر و مبتنی بر داده، مسئله ایمنی، تأیید‌پذیری و صدور گواهینامه به یک مانع بزرگ، به ویژه برای کاربردهای ایمنی-بحراتی، تبدیل شده است. این پروژه با تمرکز بر طراحی و تحلیل یک کنترل‌کننده مقاوم، تلاشی است در جهت پاسخ به این نیازها و کاهش شکاف میان عملکرد تئوری و قابلیت اطمینان عملی.

برای مقابله با این چالش‌ها، راهبردهای کنترلی در طول زمان تکامل یافته‌اند. رویکردهای اولیه که بر پایه مدل‌های خطی‌شده سیستم بنا شده بودند، هرچند ابزارهای تحلیلی قدرتمندی را فراهم می‌کردند، اما در شرایط پروازی دور از نقطه

تعادل، کارایی خود را از دست می‌دادند. این محدودیت، محققان را به سمت توسعه روش‌های غیرخطی سوق داد که مستقیماً با ذات پیچیده دینامیک سیستم روبرو می‌شوند و مقاومت بیشتری در برابر عدم قطعیت‌ها ارائه می‌دهند. نسل جدیدتر کنترل‌کننده‌ها حتی یک گام فراتر رفته و به جای سرکوب، به طور فعال به تخمین، یادگیری و تطبیق خود با تغییرات سیستم و محیط می‌پردازند. امروزه، اجماع فرایندهای در جامعه علمی وجود دارد که آینده کنترل با عملکرد بالا، در معماری‌های کنترلی ترکیبی نهفته است؛ سیستم‌هایی که به طور هوشمندانه نقاط قوت فلسفه‌های مختلف کنترلی را برای غلبه بر ضعف‌های فردی آن‌ها با یکدیگر ادغام می‌کنند.

این گزارش به صورت نظاممند برای تحلیل این مسئله و ارائه یک راهکار ساختار یافته است. فصل دوم به مرور عمیق ادبیات تحقیق اختصاص دارد، جایی که سیر تکامل رویکردهای کنترلی از مدل‌های کلاسیک تا پارادایم‌های مدرن بررسی شده و با نگاهی انتقادی، چالش‌ها و شکاف‌های پژوهشی کلیدی که محرک اصلی این پژوهه بوده‌اند، شناسایی می‌شوند. در فصل سوم، بر اساس تحلیل‌های صورت گرفته، یک معماری کنترل مقاوم مشخص انتخاب شده و مبانی نظری، ویژگی‌ها و مزایای آن به تفصیل تشریح می‌گردد. فصل چهارم به بخش عملی پژوهه می‌پردازد و فرآیند شبیه‌سازی کنترل‌کننده منتخب را در محیط نرم‌افزاری متلب/اسیمولینک به نمایش می‌گذارد تا عملکرد آن تحت سناریوهای مختلف ارزیابی شود. در فصل پنجم، ملاحظات و چالش‌های پیاده‌سازی این الگوریتم بر روی سخت‌افزار واقعی مورد بحث قرار می‌گیرد که گامی حیاتی برای اعتبارسنجی نهایی است. در نهایت، فصل ششم با ارائه یک نتیجه‌گیری جامع از یافته‌های پژوهه و ترسیم چشم‌اندازهای تحقیقاتی آینده، این گزارش را به پایان می‌رساند.

فصل ۲

مرور ادبیات

۱.۲ کوادروتور به عنوان یک بستر پژوهشی چالش برانگیز در حوزه کنترل

۱.۱.۲ اهمیت و کاربردهای روزافزون کوادروتورها

در دهه‌های اخیر، وسایل نقلیه هوایی بدون سرنشین^۱ از یک فناوری عمدتاً نظامی و سرگرمی به ابزاری حیاتی در طیف گسترده‌ای از کاربردهای غیرنظامی و صنعتی تبدیل شده‌اند [۱، ۲]. در میان انواع مختلف UAV‌ها، کوادروتورها^۲ به دلیل ویژگی‌های منحصر به فرد خود، توجه ویژه‌ای را در جوامع علمی و صنعتی به خود جلب کرده‌اند. این وسایل نقلیه، که نوع خاصی از هواگردهای بالگردان^۳ هستند، به دلیل سادگی مکانیکی نسبی، قابلیت برخاست و فرود عمودی^۴، و توانایی شناوری^۵ در یک نقطه ثابت، انعطاف‌پذیری عملیاتی بالایی از خود نشان می‌دهند [۱، ۴، ۳، ۵]. این ویژگی‌ها، کوادروتورها را برای کاربردهایی نظیر فیلمبرداری هوایی، بازرسی زیرساخت‌های حیاتی (مانند خطوط لوله و پل‌ها)، کشاورزی دقیق، عملیات جستجو و نجات، نظارت و پایش محیطی، و حتی لجستیک و تحویل کالا به گزینه‌ای ایده‌آل بدل کرده است [۷، ۶، ۱، ۲]. رشد تصاعدی مقالات پژوهشی و افزایش علاقه عمومی به این فناوری، همانطور که در روندهای جستجوی جهانی نیز مشهود است، گواهی بر اهمیت روزافزون کوادروتورهاست [۸]. این محبوبیت، خود یک شمشیر دولبه برای پژوهشگران حوزه کنترل است. از یک سو، سادگی مکانیکی کوادروتور (کنترل آن تنها از طریق تغییر سرعت دورانی روتورها انجام می‌شود). آن را به یک پلتفرم در دسترس و کم‌هزینه برای آزمایش تئوری‌های کنترل تبدیل کرده است [۱، ۵]. از سوی دیگر، این سادگی ظاهری، یک پیچیدگی دینامیکی عمیق را پنهان می‌کند که چالش‌های کنترلی قابل توجهی را به همراه دارد. همین ماهیت دوگانه، یعنی دسترسی فیزیکی آسان و چالش‌های کنترلی عمیق، کوادروتور را از یک ابزار کاربردی صرف به یک موضوع پژوهشی بنیادی در علم کنترل تبدیل کرده است که بستر مناسبی برای توسعه و اعتبارسنجی الگوریتم‌های کنترلی پیشرفت‌ههای فراهم می‌آورد [۹، ۳، ۱].

¹Unmanned Aerial Vehicles (UAVs)

²Quadrotors

³Rotary-wing aircraft

⁴Vertical Take-Off and Landing (VTOL)

⁵Hovering

۲.۱.۲ مدل‌سازی ریاضی و دینامیک پرواز شش درجه آزادی

برای طراحی هر کنترل‌کننده مبتنی بر مدلی^۱، در ک دقيق و مدل‌سازی ریاضی رفتار دینامیکی سیستم، یک پیش‌نیاز اساسی است. دینامیک یک کوادراتور معمولاً با یک مدل صلب شش درجه آزادی^۲ توصیف می‌شود که حرکت آن در فضا را به‌طور کامل بیان می‌کند [۱، ۴]. این مدل ریاضی عموماً با استفاده از فرمالیسم نیوتون-اویلر^۳ یا لاغرانژ-اویلر^۴ استخراج می‌شود [۴، ۱۰، ۱۱]. حالت^۵ یک کوادراتور در هر لحظه با دوازده متغیر تعريف می‌شود: سه متغیر برای موقعیت خطی مرکز جرم در یک چارچوب مرجع اینرسی (X, Y, Z)، سه متغیر برای سرعت خطی (Ẋ, Ẏ, Ż)، سه متغیر برای توصیف وضعیت زاویه‌ای (رول^۶، پیچ^۷ و یاو^۸) در یک چارچوب متصل به بدنه، و سه متغیر برای سرعت زاویه‌ای (p, q, r) [۱۱، ۱۲]. دینامیک این سیستم دارای ویژگی‌های ذاتی است که آن را به یک مسئله کنترلی چالش‌برانگیز تبدیل می‌کند:

۱. سیستم کمتر-عملگر^۹: کوادراتور دارای شش درجه آزادی خروجی است، اما تنها چهار ورودی کنترلی مستقل (سرعت دورانی چهار روتور) برای کنترل این شش خروجی در اختیار دارد [۳، ۴، ۶]. این بدان معناست که حرکات انتقالی در راستای محورهای X و Y به‌طور مستقیم قابل کنترل نیستند و باید به صورت غیرمستقیم و از طریق تغییر زوایای رول و پیچ کنترل شوند.

۲. دینامیک شدیداً غیرخطی و کوپل شده^{۱۰}: معادلات حرکت کوادراتور شامل جملات غیرخطی قابل توجهی مانند توابع مثلثاتی زوایای اویلر، جملات درجه دوم سرعت‌های زاویه‌ای (ناشی از اثرات ژیروسکوپی و گشتاورهای آبرودینامیکی)، و کوپل^{۱۱} قوی بین دینامیک انتقالی و دورانی است [۴، ۶، ۹]. این کوپل به این معناست که تغییر در وضعیت زاویه‌ای مستقیماً بر حرکت خطی تأثیر می‌گذارد و بالعکس. به دلیل این ویژگی‌ها، مدل دینامیکی کوادراتور اغلب به دو زیرسیستم تقسیم می‌شود: یک زیرسیستم دورانی (شامل زوایای رول، پیچ و یاو) که کاملاً عملگر^{۱۲} است و یک زیرسیستم انتقالی (شامل موقعیت X, Y, Z) که کمتر-عملگر است [۱۳، ۴]. انتخاب نحوه نمایش وضعیت زاویه‌ای نیز یک تصمیم طراحی بنیادی با پیامدهای مستقیم برای عملکرد کنترل کننده است. در حالی که زوایای اویلر به دلیل ماهیت شهودی خود در بسیاری از مطالعات اولیه استفاده شده‌اند [۴، ۱۱]، این نمایش در زوایای خاصی (مانند پیچ ۹۰ درجه) دچار پدیده قفل گیمبال^{۱۳} می‌شود که منجر به یک تکینگی^{۱۴} ریاضی و از دست رفتن یک درجه آزادی می‌گردد. برای غلبه بر این محدودیت و امکان پذیر ساختن مانورهای تهاجمی^{۱۵}، پژوهش‌های پیشرفته‌تر به استفاده از کواترنیون‌ها^{۱۶} روی آورده‌اند که نمایشی چهاربعدی و بدون تکینگی از وضعیت زاویه‌ای ارائه می‌دهند و برای کنترل در کل پوش پروازی^{۱۷} ضروری هستند [۱۴، ۱۵، ۱۶]. این انتخاب، مستقیماً بر مقاومت و قابلیت اطمینان کنترل کننده در شرایط پروازی دینامیک تأثیر می‌گذارد.

¹Model-based controller

²6-Degrees of Freedom (6-DOF)

³Newton-Euler formalism

⁴Euler-Lagrange formalism

⁵State

⁶Roll (ϕ)

⁷Pitch (θ)

⁸Yaw (ψ)

⁹Under-actuated System

¹⁰Highly Nonlinear and Coupled Dynamics

¹¹Coupling

¹²Fully-actuated

¹³Gimbal Lock

¹⁴Singularity

¹⁵Aggressive maneuvers

¹⁶Quaternions

¹⁷Flight envelope

۳.۱.۲ چالش‌های کلیدی در کنترل مقاوم

هدف اصلی در طراحی یک سیستم کنترل پرواز برای کوادروپرتو، دستیابی به پایداری و ردیابی دقیق مسیرهای مطلوب است. با این حال، در دنیای واقعی، سیستم همواره با عدم قطعیت‌ها و اغتشاشاتی مواجه است که عملکرد آن را تهدید می‌کنند. بنابراین، نیاز به یک کنترل کننده مقاوم^۱ پدید می‌آید؛ کنترل کننده‌ای که بتواند عملکرد مطلوب خود را در حضور این پدیدهای پیش‌بینی‌نشده حفظ کند [۱۷، ۵]. چالش‌های اصلی که یک کنترل کننده مقاوم باید بر آن‌ها غلبه کند، به دو دسته اصلی تقسیم می‌شوند:

۱. عدم قطعیت‌های پارامتری و دینامیک‌های مدل‌سازی‌نشده^۲: مدل ریاضی که برای طراحی کنترل کننده استفاده می‌شود، همواره یک تقریب از سیستم واقعی است. عدم قطعیت‌های پارامتری شامل تغییرات در جرم و ماتریس اینرسی کوادروپرتو به دلیل حمل بارهای ناشناخته یا متغیر است [۷، ۱۸، ۱۹]. دینامیک‌های مدل‌سازی‌نشده نیز شامل اثرات آبرو دینامیکی پیچیده (مانند نیروی درگ^۳ و اثر زمین^۴) است که در مدل‌های ساده‌شده نادیده گرفته می‌شوند اما در مانورهای سریع و تهاجمی تأثیر قابل توجهی دارند [۶، ۲۰].

۲. اغتشاشات خارجی^۵: این دسته شامل نیروها و گشتاورهای خارجی است که بر کوادروپرتو اعمال می‌شوند و مهم‌ترین آن‌ها تنبایه‌ها^۶ و تلاطم‌های جوی است. به دلیل نسبت نیروی پیشران به وزن پایین و سطح مقطع نسبتاً بزرگ، کوادروپرتوها به شدت تحت تأثیر باد قرار می‌گیرند که می‌تواند پایداری آن‌ها را به خطر اندازد و منجر به انحراف از مسیر مطلوب شود [۶، ۲۱، ۲۲]. یک کنترل کننده مقاوم باید قادر باشد پایداری حلقه-بسته^۷ را تضمین کرده و خطای ردیابی را علی‌رغم وجود این دو منبع اصلی عدم قطعیت، در سطح قابل قبولی نگه دارد. این وظیفه، به‌ویژه برای کنترل کننده‌های خطی ساده که بر پایه یک مدل اسمی و در غیاب اغتشاشات طراحی شده‌اند، بسیار دشوار است [۶، ۱۸]. این چالش بنیادین، محرک اصلی برای تکامل راهبردهای کنترلی بوده و پژوهشگران را به سمت طراحی الگوریتم‌های پیچیده‌تر و هوشمندتر سوق داده است که در بخش‌های بعدی به تفصیل مورد بررسی قرار خواهند گرفت.

¹Robust Controller

²Parametric Uncertainties and Unmodeled Dynamics

³Drag force

⁴Ground effect

⁵External Disturbances

⁶Wind gusts

⁷Closed-loop stability

۲.۲ سیر تکامل راهبردهای کنترل: از خطی‌سازی تا یادگیری

تاریخچه توسعه کنترل کننده‌ها برای کوادراتور، روایتی شفاف از یک فرآیند "مسئله-واکنش" است. محدودیت‌های هر نسل از کنترل کننده‌ها، مستقیماً انگیزه‌ای برای خلق نسل بعدی بوده است. این سیر تکاملی را می‌توان به سه دوره اصلی تقسیم کرد: دوره کنترل خطی، دوره ظهور کنترل غیرخطی و مقاوم، و دوره مدرن که با رویکردهای مبتنی بر تخمين و یادگیری مشخص می‌شود.

۱.۲.۲ تکنیک‌های کنترل خطی بنیادین

اولین موج از کنترل کننده‌های طراحی شده برای کوادراتورها بر پایه یک فرض ساده‌کننده استوار بودند: خطی‌سازی دینامیک غیرخطی سیستم حول یک نقطه کار مشخص، که معمولاً حالت شناوری (hover) است. این رویکرد، امکان استفاده از ابزارهای قدرتمند و شناخته‌شده تئوری کنترل خطی را فراهم می‌کرد.

کنترل کننده تناوبی-انتگرالی-مشتقی

کنترل کننده PID^۱ به دلیل سادگی ساختاری، هزینه محاسباتی پایین و این واقعیت که تنظیم آن لزوماً به مدل دقیق سیستم نیاز ندارد، به پرکاربردترین کنترل کننده در صنعت و همچنین در پیاده‌سازی‌های اولیه کوادراتورها تبدیل شد [۲۱، ۲۲، ۲۳]. مطالعات بنیادین متعددی، مانند کارهای بررسی شده توسط Li (2011) [۱] و Saleh (2022) [۲۱]، کارایی این کنترل کننده را در شبیه‌سازی و آزمایش‌های ساده برای دستیابی به شناوری پایدار و ردیابی مسیرهای ابتدایی نشان داده‌اند [۴، ۲۳].

با این حال، محدودیت اصلی PID دقیقاً در وابستگی آن به نقطه کار خطی‌شده نهفته است. هنگامی که کوادراتور مانورهای تهاجمی انجام می‌دهد یا تحت تأثیر اغتشاشات بزرگ قرار می‌گیرد، حالت سیستم از نقطه کار دور شده و رفتار غیرخطی آن غالب می‌شود. در این شرایط، عملکرد کنترل کننده PID به شدت افت کرده و ممکن است منجر به ناپایداری شود [۱، ۲۳]. علاوه بر این، فرآیند تنظیم بهره‌های PID (Kp, Ki, Kd) اغلب به صورت تجربی و با روش آزمون و خطا انجام می‌شود که سیستماتیک نبوده و تضمینی برای بهینگی عملکرد ارائه نمی‌دهد [۳].

تنظیم کننده خطی درجه دوم

کنترل کننده LQR^۲ یک گام فراتر از PID است و یک رویکرد کنترل بهینه^۳ را معرفی می‌کند. این روش به جای تنظیم دستی بهره‌ها، یکتابع هزینه را که ترکیبی از خطای حالت و تلاش کنترلی است، کمینه می‌سازد [۲۴، ۳]. این ویژگی به طراح اجزاء می‌دهد تا به صورت سیستماتیک یک مصالحه^۴ بین دقت ردیابی و مصرف انرژی برقرار کند. LQR به خوبی با سیستم‌های چند ورودی-چند خروجی (MIMO)^۵ مانند کوادراتور سازگار است [۲۵]. یکی از کارهای کلیدی در این زمینه، پژوهش Reyes-Valeria et al. (2013) است که در آن یک کنترل کننده LQR بر پایه مدل خطی‌شده کواترنیونی طراحی شده و با استفاده از زمان‌بندی بهره^۶، عملکرد آن در رژیمهای پروازی مختلف بهبود یافته است [۱۴، ۲۶]. همچنین، مطالعه تأثیرگذار Bouabdallah et al. (2004) یک مقایسه اولیه و مهم بین عملکرد PID و LQR ارائه داد و برتری نسبی LQR را در شرایط خاص نشان داد [۲۷]. با این وجود، LQR نیز از محدودیت‌های ذاتی کنترل خطی رنج می‌برد.

¹Proportional-Integral-Derivative

²Linear Quadratic Regulator

³Optimal control

⁴Trade-off

⁵Multi-Input Multi-Output

⁶Gain Scheduling

این روش همچنان به یک مدل خطی شده وابسته است و مقاومت آن در برابر عدم قطعیت‌های پارامتری بزرگ، محدود است [۲۷]. همچنین، در غیاب یک جمله انتگرالی (که در کنترل کننده LQI وجود دارد)، ممکن است در ردیابی مسیر دچار خطای حالت ماندگار^۱ شود [۲۸، ۳]. شکست کنترل کننده‌های خطی در ارائه عملکرد قابل قبول در کل پوش پروازی، زمینه را برای ظهور نسل جدیدی از راهبردهای کنترل غیرخطی فراهم کرد.

۲.۲.۲ ظهور راهبردهای کنترل غیرخطی و مقاوم

در حالی که کنترل کننده‌های خطی مانند PID و LQR در پایدارسازی کوادراتور حول نقطه شناوری موفق بودند، عملکرد آن‌ها در مانورهای تهاجمی و در حضور اغتشاشات شدید به شدت افت می‌کرد. این "شکنندگی" در دنیای واقعی، پژوهشگران را ناچار کرد تا از فرض خطی‌سازی فاصله گرفته و به سراغ روش‌هایی بروند که مستقیماً با ذات غیرخطی سیستم روبرو شوند. این نیاز، سرآغاز دوره کنترل غیرخطی مقاوم بود.

خطی‌سازی فیدبک

روش خطی‌سازی فیدبک^۲ یک رویکرد ظریف برای رام کردن دینامیک غیرخطی است. ایده اصلی این است که با استفاده از یک فیدبک حالت غیرخطی و یک تبدیل مختصات، دینامیک غیرخطی اصلی به یک سیستم خطی معادل و ساده (معمولاً مجموعه‌ای از انتگرال‌گیرهای دوگانه) تبدیل شود [۲۹، ۷، ۳]. پس از این تبدیل، می‌توان از ابزارهای کنترل خطی استاندارد برای کنترل سیستم در کل پوش پروازی استفاده کرد. این روش، برخلاف PID و LQR، به یک نقطه کار محدود نیست. با این حال، نقطه ضعف اصلی FBL در حساسیت شدید آن به دقت مدل ریاضی نهفته است. این روش بر "حذف کامل" جملات غیرخطی استوار است و هرگونه خطای مدل‌سازی یا عدم قطعیت پارامتری می‌تواند این حذف را ناقص کرده و منجر به عملکرد ضعیف یا حتی ناپایداری شود [۳۰، ۲۱، ۲۰].

کنترل پس‌گام

کنترل پس‌گام^۳ یک روش طراحی بازگشتی و سیستماتیک است که بر پایه تئوری پایداری لیاپانوف^۴ بنا شده است [۷، ۳]. این روش برای سیستم‌های غیرخطی که ساختار آبشاری^۵ دارند، بسیار مناسب است. با توجه به اینکه دینامیک کوادراتور را می‌توان به یک زیرسیستم دورانی (حلقه داخلی) و یک زیرسیستم انتقالی (حلقه خارجی) تقسیم کرد، BSC به یک انتخاب طبیعی تبدیل شد. مقاله بنیادین (2006) Madani, Benallegue در عمل با چالش "انفجار پیچیدگی"^۶ کوادراتور را ارائه داد. آن‌ها با تجزیه سیستم به زیرسیستم‌های کمتر-عملگر و کاملاً عملگر، یک کنترل کننده پس‌گام طراحی کردند که پایداری کل سیستم را تضمین می‌کرد [۱۲]. با وجود قدرت تئوری، BSC در هر مرحله از فرآیند بازگشتی، از مشتق قوانین کنترل مجازی^۷ مراحل قبل استفاده می‌شود که این امر مواجه است [۳۱]. در هر مرحله از فرآیند بازگشتی، از مشتق قوانین کنترل مجازی^۷ مراحل قبل استفاده می‌شود که این امر منجر به ایجاد عبارات ریاضی بسیار طولانی و پیچیده برای قانون کنترل نهایی می‌گردد. این پیچیدگی، پیاده‌سازی عملی و تنظیم کنترل کننده را دشوار می‌سازد. همچنین، این روش می‌تواند به مشکل "پارامتردهی بیش از حد"^۸ دچار شود [۳۲].

¹Steady-state error

²Feedback Linearization

³Backstepping Control

⁴Lyapunov stability theory

⁵Cascaded structure

⁶Explosion of complexity

⁷Virtual control laws

⁸Over-parameterization

کنترل مد لغزشی

کنترل مد لغزشی^۱ یک تکنیک کنترل مقاوم بسیار قدرتمند است که فلسفه متفاوتی را دنبال می‌کند. به جای تلاش برای حذف دقیق غیرخطی‌ها، SMC یک "سطح لغزش"^۲ را در فضای حالت تعریف می‌کند که نشان‌دهنده دینامیک خطای مطلوب است. سپس یک قانون کنترل سوئیچینگ (غیرپیوسته) طراحی می‌شود تا حالت‌های سیستم را در زمان محدود به این سطح برساند و آن‌ها را روی سطح نگه دارد [۱۱، ۷]. هنگامی که سیستم روی سطح لغزش قرار دارد، رفتار آن دقیقاً مطابق با دینامیک مطلوب تعریف شده توسط سطح است و مهم‌تر از آن، نسبت به دسته‌ای از اغتشاشات و عدم قطعیت‌های مدل که به آن‌ها "عدم قطعیت‌های تطبیق‌یافته"^۳ گفته می‌شود، کاملاً مقاوم است [۱۶]. مطالعات تطبیقی متعدد، مانند آنچه در [۸] و [۲۱] گزارش شده، برتری SMC را از نظر دقت ردیابی و مقاومت در برابر اغتشاشات نسبت به PID و FBL تأیید کرده‌اند. با این حال، بزرگترین عیب SMC، پدیده چترینگ^۴ است [۲۱، ۳۳، ۳۴]. ماهیت سوئیچینگ با فرکانس بالای قانون کنترل در نزدیکی سطح لغزش، باعث ایجاد ارتعاشات با فرکانس بالا در سیگنال کنترل می‌شود. این پدیده نه تنها باعث اتلاف انرژی می‌شود، بلکه می‌تواند دینامیک‌های مدل‌سازی‌نشده فرکانس بالا را تحریک کرده و به عملگرهای فیزیکی (موتورها) آسیب برساند. این مشکل عملی، انگیزه‌ای قوی برای توسعه نسخه‌های بهبودیافته SMC مانند کنترل مد لغزشی مرتبه بالا^۵ و الگوریتم Super-Twisting شد که هدف آن‌ها کاهش یا حذف چترینگ ضمن حفظ مقاومت سیستم است [۳۳].

۳.۲.۲ رویکردهای پیشرفتی برای مقابله با عدم قطعیت

وابستگی بنیادین کنترل کننده‌های مبتنی بر مدل (حتی انواع غیرخطی آن) به یک مدل ریاضی دقیق، بزرگترین نقطه ضعف آن‌ها در کاربردهای دنیای واقعی است. دوره مدرن طراحی کنترل کننده برای کوادراتور با ظهور تکنیک‌هایی مشخص می‌شود که به جای نادیده گرفتن یا سرکوب کردن عدم قطعیت، به طور فعال آن را تخمین زده و خود را با آن تطبیق می‌دهند. این رویکردها نشان‌دهنده یک تغییر پارادایم از کنترل "صلب" به کنترل "هوشمند" و "آگاه از محیط" است. در این میان، یک انشعاب فلسفی در نحوه دستیابی به مقاومت پدیدار می‌شود. یک مسیر، که توسط کنترل کننده‌هایی مانند SMC نمایندگی می‌شود، تلاش می‌کند تا با استفاده از بهره‌های بالا و قوانین کنترل قدرتمند، بر عدم قطعیت "غلبه" کند [۲۴]. مسیر دیگر، که توسط کنترل تطبیقی و رویکردهای مبتنی بر رؤیتگر نمایندگی می‌شود، به دنبال "تخمین و حذف" عدم قطعیت است [۲۲، ۱۷]. یک مسیر سوم و جدیدتر نیز که توسط روش‌های یادگیری تقویتی ارائه شده، به طور کلی مدل خاص را "نادیده" می‌گیرد و سیاستی را می‌آموزد که ذاتاً در طیفی از مدل‌ها مقاوم باشد [۱۸].

کنترل تطبیقی

کنترل تطبیقی^۶ به طور خاص برای مقابله با عدم قطعیت‌های پارامتری طراحی شده است. این روش، پارامترهای کنترل کننده را به صورت آنلاین و در حین پرواز تنظیم می‌کند تا تغییرات در دینامیک سیستم، مانند تغییر جرم به دلیل حمل بار، را جبران نماید [۱۹، ۱۷]. یکی از محبوب‌ترین ساختارها، کنترل تطبیقی مدل مرجع (MRAC)^۷ است. در این روش، یک مدل مرجع ایده‌آل و پایدار تعریف می‌شود و کنترل کننده تطبیقی وظیفه دارد پارامترهای خود را به گونه‌ای تنظیم کند که رفتار سیستم واقعی، رفتار مدل مرجع را دنبال کند [۱۵، ۳۵]. کارهایی مانند پژوهش Islam et al. (2015) یک طراحی

¹Sliding Mode Control

²Sliding surface

³Matched uncertainties

⁴Chattering

⁵Higher-Order SMC

⁶Adaptive Control

⁷Model Reference Adaptive Control

مقاوم تطبیقی را ارائه می‌دهند که جملات PD کلاسیک را با قوانین تطبیقی مبتنی بر پایداری لیپانوف ترکیب می‌کند [۱۷]. مزیت بزرگ این روش‌ها این است که نیازی به دانستن کران بالای عدم قطعیت‌ها از قبل ندارند [۱۹]. با این حال، طراحی و اثبات پایداری آن‌ها می‌تواند پیچیده باشد و عملکردشان ممکن است به نویز اندازه‌گیری و دینامیک‌های مدل‌سازی نشده حساس باشد [۳۵، ۳۶].

کنترل مبتنی بر رؤیتگر اغتشاش

کنترل مبتنی بر رؤیتگر اغتشاش^۱ یک راهبرد عملی و مؤثر برای افزایش مقاومت یک کنترل‌کننده اسمی (مانند PID یا LQR) است. ایده اصلی این است که یک رؤیتگر^۲ طراحی شود تا مجموع اغتشاشات خارجی و عدم قطعیت‌های مدل را به صورت یک "اغتشاش کلی" در زمان واقعی تخمین بزند. سپس این اغتشاش تخمین‌زده شده از طریق یک حلقه پیشخور^۳ در قانون کنترل حذف می‌شود [۲۲، ۳۷، ۳۸]. مطالعه Benevides et al. (2022) یک نمونه برجسته از این رویکرد است که در آن از یک فیلتر کالمون^۴ برای تخمین تنببدادها استفاده شده و نتایج نشان‌دهنده بهبود چشمگیر در دقت ریدیابی مسیر در حضور باد است [۲۲]. عملکرد DOBC به دقت رؤیتگر و محدودیت‌های پهنه‌ای باند سیستم بستگی دارد، اما این روش یک راه حل مازولار و کارآمد برای مقاوم‌سازی کنترل‌کننده‌های موجود ارائه می‌دهد.

کنترل مبتنی بر یادگیری

یادگیری تقویتی عمیق^۵ یک تغییر پارادایم کامل از طراحی مبتنی بر مدل به طراحی مبتنی بر داده است. در این رویکرد، یک عامل^۶ (کنترل‌کننده) به جای پیروی از معادلات از پیش تعیین‌شده، یاد می‌گیرد که چگونه با تعامل با محیط (که می‌تواند یک شبیه‌ساز فیزیکی دقیق باشد) و دریافت سیگنال پاداش، یک سیاست کنترلی بهینه را کشف کند [۱۸، ۳۹]. پژوهش پیشگامانه Vaidya Keshavan^۷ یک کنترل‌کننده DRL را معرفی می‌کند که با آموزش بر روی طیف وسیعی از دینامیک‌های شبیه‌سازی شده (با تغییر جرم و اینرسی)، به یک سیاست کنترلی "نامتفاوت نسبت به دینامیک"^۸ دست می‌یابد. نتایج آن‌ها بهبود دقت ریدیابی تا ۸۵ درصد را نسبت به روش‌های پایه نشان می‌دهد [۱۸، ۴۰]. با وجود پتانسیل بسیار بالا، روش‌های DRL با چالش‌های جدی روبرو هستند. مهم‌ترین چالش، "شکاف شبیه‌سازی به واقعیت"^۹ است؛ سیاستی که در شبیه‌سازی عملکرد عالی دارد، ممکن است در دنیای واقعی به دلیل تفاوت‌های جزئی در دینامیک و نویز سنسورها، عملکرد ضعیفی داشته باشد یا حتی ناپایدار شود [۱۸، ۴۰]. علاوه بر این، این روش‌ها به حجم عظیمی از داده‌های آموزشی نیاز دارند و قادر تضمین‌های پایداری رسمی هستند، که این امر یک مانع بزرگ برای استفاده از آن‌ها در کاربردهای ایمنی-بحراتی^۹ محسوب می‌شود [۱۸، ۴۱].

¹Disturbance Observer-Based Control

²Observer

³Feedforward loop

⁴Kalman Filter

⁵Deep Reinforcement Learning

⁶Agent

⁷Dynamics-invariant

⁸Sim-to-real gap

⁹Safety-critical applications

۳.۲ تحلیل تطبیقی، مباحثات جاری، و راهبردهای ترکیبی

انتخاب یک استراتژی کنترلی برای کوادروتور، فرآیندی پیچیده است که مستلزم درک عمیق مصالحه‌های موجود بین عملکرد، مقاومت و پیچیدگی است. هیچ کنترل کننده واحدی وجود ندارد که در تمام معیارها برتر باشد. این بخش به تحلیل تطبیقی این راهبردها، بررسی مباحثات کلیدی در این حوزه، و معرفی روند غالب به سمت معماری‌های ترکیبی می‌پردازد.

۱.۳.۲ ارزیابی عملکرد و مصالحه

کنترل کننده‌های خطی مانند PID و LQR از نظر پیاده‌سازی ساده و از نظر محاسباتی کم‌هزینه هستند، اما مقاومت و عملکرد آن‌ها در شرایط پروازی تهاجمی محدود است [۱، ۴۲]. در مقابل، کنترل کننده‌های غیرخطی مانند SMC و Backstepping مقاومت بالایی ارائه می‌دهند اما با هزینه پیچیدگی طراحی و چالش‌هایی مانند چترینگ و انفجار پیچیدگی همراه هستند [۸، ۴۲]. یک مطالعه تطبیقی در [۴۳] نتیجه‌گیری می‌کند که SMC به دلیل دینامیک سریع و مقاومت بالا، امیدوار کننده‌ترین گزینه برای پایدارسازی وضعیت زاویه‌ای است. MPC^۱ به دلیل توانایی در مدیریت بهینه قیود حالت و ورودی، عملکرد بسیار خوبی ارائه می‌دهد، اما هزینه محاسباتی بالای آن به طور سنتی یک مانع بزرگ برای پیاده‌سازی بر روی پردازنده‌های تعبیه‌شده^۲ بوده است [۶، ۴۴]. روش‌های مبتنی بر یادگیری، پیچیدگی طراحی مبتنی بر مدل را با پیچیدگی جمع‌آوری داده و آموزش جایگزین می‌کنند و پتانسیل بالاترین عملکرد را دارند، اما با عدم قطعیت در اینمی همراه هستند [۱۸، ۴۱]. جالب توجه است که مفهوم "پیچیدگی محاسباتی" یک هدف متحرک است. آنچه در گذشته غیرقابل اجرا تلقی می‌شد، مانند MPC، امروزه با پیشرفت پرداختاب کنترل کننده مناسب برای یک کاربرد خاص، بیش از آنکه به دنبال یافتن "بهترین" کنترل کننده باشد، به یافتن "مناسب‌ترین" آن با توجه به محدودیت‌ها و الزامات پروژه مربوط می‌شود. مطالعات تطبیقی متعدد، مانند آنچه در [۱]، [۹] و [۴۲] ارائه شده، به روشن شدن این مصالحه‌ها کمک کرده‌اند.

با پیشرفت پردازنده‌های تعبیه‌شده، الگوریتم‌های کنترلی به صورت بلاذرنگ^۳ قابل پیاده‌سازی شدند [۴۴، ۲۵]. این تحول، کانون مصالحه را از "آیا می‌توانیم آن را اجرا کنیم؟" به "آیا مدل‌سازی و تنظیم آن ارزش تلاش را دارد؟" تغییر داده است. این تغییر به نفع روش‌های مبتنی بر یادگیری است که مدل‌سازی پیچیده را با آموزش فشرده (اما آفلاین) جایگزین می‌کنند.

۲.۳.۲ مباحثات جاری: کنترل مبتنی بر مدل در مقابل کنترل مبتنی بر داده

مبحثی که امروزه بیش از هر موضوع دیگری در حوزه کنترل کوادروتور جریان دارد، تقابل بین رویکرد سنتی مبتنی بر مدل^۴ و پارادایم نوظهور مبتنی بر داده^۵ است. رویکرد مبتنی بر مدل، که شامل تمام کنترل کننده‌های کلاسیک و غیرخطی مورد بحث است، بر یک مدل ریاضی دقیق از سیستم تکیه دارد و عملکرد آن به طور مستقیم به وفاداری^۶ این مدل به واقعیت وابسته است [۴۵]. در مقابل، رویکرد مبتنی بر داده، به ویژه یادگیری تقویتی، می‌تواند بدون نیاز به یک مدل صریح، کنترل سیستم را بیاموزد [۱۸، ۱۵]. برخی مطالعات اولیه ادعا کرده‌اند که RL به طور قابل توجهی از روش‌های سنتی بهتر عمل می‌کند [۴۰]. با این حال، یک مطالعه انتقادی توسط Song et al. [۴۱] استدلال می‌کند که بسیاری از این مقایسه‌ها منصفانه نیستند. زمانی که یک کنترل کننده هندسی^۷ (یک روش پیشرفتی مبتنی بر مدل) به طور بهینه تنظیم شود، شکاف عملکرد بین آن و RL بسیار کمتر از آن چیزی است که قبلاً گزارش شده بود. یافته‌های آن‌ها نشان می‌دهد که RL در عملکرد

¹Model Predictive Control

²Embedded processors

³Real-time

⁴Model-based

⁵Data-driven

⁶Fidelity

⁷Geometric controller

گذرا^۱ برتری دارد، در حالی که کنترل هندسی خطای حالت ماندگار کمتری را به دست می‌آورد [۴۱]. این مبحث صرفاً یک بحث آکادمیک در مورد عملکرد نیست، بلکه پیامدهای عمیقی برای اعتماد و صدور گواهینامه^۲ دارد. کنترل کننده‌های مبتنی بر مدل دارای اثبات‌های پایداری ریاضی (مانند تئوری لیاپانوف) هستند که رفتار آن‌ها راقابل پیش‌بینی و تحلیل‌پذیر می‌سازد [۱۳، ۱۷، ۴۶]. این ویژگی برای صدور گواهینامه در حوزه‌های ایمنی-بحرانی مانند هوانوردی ضروری است. در مقابل، کنترل کننده‌های DRL اغلب به عنوان "جعبه سیاه"^۳ عمل می‌کنند [۴۱]. اگرچه عملکرد آن‌ها می‌تواند چشمگیر باشد، اثبات اینکه در یک حالت دیده‌نشده با شکست مواجه نخواهد شد، یک مسئله پژوهشی باز و دشوار است. این مانع عملی، مسیر آینده را به سمت پژوهش در زمینه هوش مصنوعی قابل تأیید^۴ و یادگیری تقویتی ایمن^۵، یا روش‌های ترکیبی که در آن یک کنترل کننده پشتیبان با پایداری اثبات‌شده بر عملکرد یک کنترل کننده مبتنی بر یادگیری نظارت می‌کند، هدایت می‌کند.

۳.۳.۲ روند غالب: معماری‌های کنترل ترکیبی

با توجه به اینکه هیچ راهبرد کنترلی "حالص" واحدی وجود ندارد که بتواند به تمام چالش‌های کنترل کوادراتور پاسخ دهد، یک روند غالب در این حوزه، حرکت به سمت معماری‌های کنترل ترکیبی^۶ است. این رویکردها با ترکیب هوشمندانه نقاط قوت چندین استراتژی، به دنبال غلبه بر ضعف‌های فردی آن‌ها هستند [۳۲، ۶]. بلوغ این حوزه را می‌توان در کثرت و تنوع طرح‌های ترکیبی مشاهده کرد که نشان‌دهنده یک اجماع در میان پژوهشگران است: آینده کنترل مقاوم و با عملکرد بالا در این ترکیبات هوشمندانه نهفته است.

نمونه‌هایی از این معماری‌های ترکیبی عبارتند از:

ترکیب کنترل خطی و هوشمند: مانند Fuzzy PID که در آن از منطق فازی برای تنظیم آنلاین بهره‌های کنترل کننده PID استفاده می‌شود تا سازگاری آن با شرایط متغیر بهبود یابد. یا استفاده از RL برای پیش‌بینی و تنظیم بهره‌های PID در حین پرواز [۳۹].

ترکیب کنترل کننده‌های غیرخطی: مانند ترکیب SMC و Backstepping که در آن ساختار سیستماتیک Backstepping با مقاومت SMC ادغام می‌شود [۲۱]. اخیراً، کنترل کننده‌های ترکیبی الهام گرفته از طبیعت^۷ که این دو روش را برای حذف همزمان پرش سرعت و چترینگ به کار می‌گیرند، مورد توجه قرار گرفته‌اند [۴۷].

ترکیب کنترل بهینه و مقاوم: مانند ترکیب MPC با H-infinity که در آن کنترل کننده H-infinity برای پایدارسازی اولیه و کاهش بار محاسباتی MPC استفاده می‌شود [۴۸].

ترکیب کنترل تطبیقی و مقاوم: مانند Adaptive Backstepping SMC که در آن قوانین تطبیقی برای تخمین عدم قطعیت‌ها به کار می‌روند و SMC مقاومت در برابر اغتشاشات باقی‌مانده را تضمین می‌کند [۳۲]. این روند به وضوح نشان می‌دهد که میدان تحقیق از جستجو برای یک "گلوله نقره‌ای" عبور کرده و به سمت طراحی سیستم‌های کنترلی یکپارچه و چندلایه حرکت کرده است که هر لایه برای مقابله با جنبه خاصی از چالش کنترل کوادراتور بهینه شده است.

¹Transient performance

²Trust and certification

³Black box

⁴Certifiable AI

⁵Safe RL

⁶Hybrid Control Architectures

⁷Bio-inspired

جدول ۱.۲: مقایسه راهبردهای کنترل کلیدی برای کوادرotor

راهبرد کنترل	اصل بنیادی	نقاط قوت کلیدی	ضعف‌ها و چالش‌های اصلی
PID	فیدبک مبتنی بر خطاب سادگی، پیچیدگی کم، تنظیم بدون نیاز به مدل	عملکرد ضعیف برای سیستم‌های شدیداً غیرخطی، تنظیم دستی	
LQR	کنترل بهینه از طریق کمینه‌سازی تابع هزینه	بهینه برای سیستم‌های خطی، MIMO مدیریت	نیاز به خطی‌سازی، احتمال خطای حالت ماندگار
Backstepping	طراحی بازگشتی مبتنی بر لیاپانوف	سیستماتیک برای سیستم‌های غیرخطی، تضمین پایداری بیش از حد	"انجار پیچیدگی"، پارامتردهی
SMC	وادار کردن حالت به یک سطح لغزش	مقاومت بالا در برابر اغتشاشات و عدم قطعیت‌ها	پدیده چترینگ، نیاز به دانستن کران اغتشاشات
MPC	بینه‌سازی آنلاین روی یک افق پیش‌بینی	مدیریت قیود، قابلیت پیش‌بینی	هزینه محاسباتی بالا، وابسته به مدل
کنترل تطبیقی	تنظیم آنلاین پارامترها	جبران عدم قطعیت‌های پارامتری (مانند بار)	تحلیل پایداری پیچیده، حساسیت به نویز
یادگیری تقویتی	بینه‌سازی سیاست بدون مدل از طریق پاداش بهینه	مقاومت در برابر دینامیک‌های مدل نشده، یادگیری سیاست‌های شکاف شبیه‌سازی به واقعیت، نیاز به داده زیاد، نگرانی‌های ایمنی	

جدول ۱.۲ یک نمای کلی و مقایسه‌ای از راهبردهای کنترل کلیدی برای کوادرotorها ارائه می‌دهد که در بخش مرور ادبیات به تفصیل مورد بحث قرار گرفتند. این جدول هر روش کنترلی از PID و LQR کلاسیک گرفته تا روش‌های پیشرفته‌تر مانند SMC، کنترل تطبیقی و یادگیری تقویتی را بر اساس چهار معیار اساسی ارزیابی می‌کند: اصل بنیادی عملکرد، نقاط قوت کلیدی، و ضعف‌ها و چالش‌های اصلی آن. هدف از این مقایسه، نمایش تصویری مصالحه‌ها بین عملکرد، پیچیدگی و مقاومت است؛ برای مثال، تقابل بین سادگی PID و مقاومت بالای SMC در برابر اغتشاشات، یا پتانسیل بالای یادگیری تقویتی در مقابل نگرانی‌های مربوط به ایمنی آن. این جمع‌بندی به خواننده کمک می‌کند تا به سرعت درک کند که چرا هیچ کنترل‌کننده واحدی برای تمام کاربردها ایده‌آل نیست و انتخاب نهایی به الزامات خاص پروژه بستگی دارد.

۴.۲ شکاف‌های پژوهشی و چشم‌اندازهای آینده

علی‌رغم پیشرفت‌های چشمگیر در زمینه کنترل کوادروتور، هنوز شکاف‌های قابل توجهی در ادبیات علمی وجود دارد و مسیرهای پژوهشی متعددی برای آینده باز است. شناسایی این شکاف‌ها نه تنها برای توجیه پژوهش‌های جدید ضروری است، بلکه نقشه راهی برای پیشرفت این حوزه در سال‌های آینده فراهم می‌کند.

۱.۴.۲ شکاف‌های شناسایی‌شده در پژوهش‌های پیشین

ارزیابی انتقادی مجموعه کارهای انجام‌شده، چندین محدودیت و شکاف کلیدی را آشکار می‌سازد:

شکاف ۱: فقدان محکزی استاندارد و تکرارپذیری: یکی از بزرگ‌ترین موانع در پیشرفت این حوزه، دشواری در مقایسه منصفانه نتایج مطالعات مختلف است. بسیاری از مقالات، نتایج خود را با استفاده از سخت‌افزارهای شرایط آزمایشی، و معیارهای ارزیابی متفاوتی گزارش می‌دهند [۸، ۲۰]. پارامترهای کلیدی مانند جرم کوادروتور، بهره‌های کنترل کننده، سرعت باد، و مشخصات مسیر اغلب حذف می‌شوند که این امر تکرارپذیری^۱ نتایج را تقریباً غیرممکن می‌سازد. همانطور که در [۸] و [۲۰] به درستی اشاره شده، نیاز مبرمی به استانداردهایی برای گزارش‌دهی (مانند ارائه همزمان خطای جذر میانگین مربعات (RMSE)^۲ و حداقل خطای ردیابی (MTE)^۳ و همچنین برای کمی‌سازی اختشاشات (مثالاً بیان نیروی باد به صورت نسبتی از وزن وسیله نقلیه) وجود دارد. مطالعه [۴۱] نیز نشان می‌دهد که چگونه مقایسه‌های غیراستاندارد می‌تواند به نتایج گمراه‌کننده در مورد برتری یک کلاس از کنترل‌کننده‌ها بر دیگری منجر شود.

شکاف ۲: مقاومت در سناریوهای تهاجمی و بدون ساختار: در حالی که بسیاری از کنترل‌کننده‌ها مقاومت خوبی در برابر اختشاشات متوسط از خود نشان می‌دهند، پژوهش‌های کمی در مورد کنترل برای مانورهای بسیار تهاجمی در محیط‌های شلoug و ناشناخته، مانند پرواز از میان شکاف‌های باریک، انجام شده است [۴۹، ۵۰]. این سناریوها چالش "بینایی فعال"^۴ را مطرح می‌کنند که در آن کنترل و ادراک به شدت به یکدیگر وابسته هستند؛ کوادروتور باید وضعیت خود را به گونه‌ای کنترل کند که شکاف همیشه در میدان دید سنسورهایش باقی بماند [۴۹]. به طور مشابه، کنترل کوادروتورها هنگام حمل بارهای معلق بزرگ و دینامیک (مانند بارهای آویزان) یک چالش بزرگ باقی مانده است و حوزه‌هایی مانند برخاست و فرود در این شرایط به طور خاص کمتر مورد مطالعه قرار گرفته‌اند [۲۰].

شکاف ۳: شکاف شبیه‌سازی به واقعیت برای کنترل مبتنی بر یادگیری: یک مانع اساسی برای استقرار عملی روش‌های مبتنی بر یادگیری، به ویژه DRL، انتقال سیاست‌های آموخته شده در شبیه‌سازی به دنیای واقعی بدون افت قابل توجه عملکرد است [۱۸، ۴۰]. اگرچه تلاش‌هایی برای کاهش این شکاف از طریق تکنیک‌هایی مانند تصادفی‌سازی دینامیک^۵ و استفاده از داده‌های واقعی در فرآیند آموزش صورت گرفته است [۱۸]، این مسئله همچنان یک چالش بنیادی برای این حوزه باقی مانده است.

شکاف ۴: ایمنی، تأیید و صدور گواهینامه: برای کنترل کننده‌های مبتنی بر یادگیری و الگوریتم‌های تطبیقی بسیار پیچیده، یک شکاف عمیق در زمینه روش‌های صوری^۶ برای تأیید پایداری و تضمین عملکرد ایمن وجود دارد. این شکاف، بزرگ‌ترین مانع برای استقرار این فناوری‌های قدرتمند در کاربردهای دنیای واقعی و ایمنی-بحرانی است [۴۱]. این شکاف‌های پژوهشی به طور عمیقی به یکدیگر مرتبط هستند. به عنوان مثال، حل مسئله پرواز تهاجمی در محیط‌های شلoug (شکاف ۲) نیازمند ادراک بهتر است. ادراک بهتر به سنسورها متکی است که نویز را به سیستم وارد می‌کنند. مدیریت نویز سنسور و سایر اثرات دنیای واقعی، مشکل شکاف شبیه‌سازی به واقعیت (شکاف ۳) را برای کنترل کننده‌های مبتنی بر یادگیری تشدید

¹Reproducibility

²Root Mean Square Error

³Maximum Tracking Error

⁴Active vision

⁵Dynamics Randomization

⁶Formal methods

می‌کند. استفاده از یک کنترل‌کننده مبتنی بر یادگیری برای چنین وظیفه ایمنی-بحرانی، بلافاصله مسئله تأیید و صدور گواهینامه (شکاف^(۴)) را مطرح می‌کند. این درهم‌تنیدگی نشان می‌دهد که تأثیرگذارترین پژوهش‌های آینده، پژوهش‌هایی خواهند بود که به صورت کل‌نگر و یکپارچه به مسائل کنترل، ادراک و ایمنی بپردازند.

۲.۴.۲ مسیرهای تحقیقاتی آینده

مسیرهای پژوهشی آینده مستقیماً از شکاف‌های شناسایی‌شده نشأت می‌گیرند و به دنبال ارائه راه حل‌هایی برای چالش‌های باقی‌مانده هستند:

مسیر ۱: توسعه کنترل‌کننده‌های تطبیقی و مبتنی بر یادگیری قابل تأیید: پژوهش‌های آینده باید بر روی معماری‌های ترکیبی تمرکز کنند که یک کنترل‌کننده مبتنی بر یادگیری با عملکرد بالا را با یک لایه ناظارتی ایمنی-بحرانی مبتنی بر روش‌های با پایداری اثبات‌شده (مانند SMC یا LQR) ترکیب می‌کنند. این رویکرد مستقیماً به شکاف ایمنی و تأیید (شکاف^(۴)) پاسخ می‌دهد.

مسیر ۲: کنترل و ادراک یکپارچه برای ناوبری چابک: کارهای آینده باید بر روی ادغام تنگاتنگ الگوریتم‌های ادراک پیشرفت‌ه (به ویژه مبتنی بر بینایی ماشین) با کنترل‌کننده‌های مقاوم تمرکز کنند. این امر، ناوبری خودران در محیط‌های ناشناخته و پویا را امکان‌پذیر می‌سازد و به چالش مقاومت در سناریوهای بدون ساختار (شکاف ۲) و مسئله "بینایی فعال" مطرح‌شده در [۴۹]^(۵) می‌پردازد.

مسیر ۳: یادگیری ماشین آگاه از فیزیک: برای پر کردن شکاف شبیه‌سازی به واقعیت (شکاف ۳)، پژوهش‌ها باید به سمت روش‌هایی حرکت کنند که اصول فیزیکی شناخته‌شده (مانند معادلات دینامیکی $DOF-6$) را در فرآیند یادگیری ادغام می‌کنند. این رویکرد که به آن یادگیری ماشین آگاه از فیزیک^(۱) گفته می‌شود، می‌تواند کارایی نمونه‌برداری^(۲) را بهبود بخشد و به سیاست‌های تعیین‌پذیرتر^(۳) منجر شود.

مسیر ۴: کنترل مقاوم برای تعامل فیزیکی و دستکاری: گسترش دامنه کاربرد کوادراتورها از پرواز صرف به وظایفی که شامل تماس فیزیکی هستند، مانند دستکاری هوایی با بازووهای رباتیک [۵۱]^(۴)، نیازمند نسل جدیدی از کنترل‌کننده‌ها است. این کنترل‌کننده‌ها باید بتوانند نیروهای تماسی و تغییرات شدید دینامیک سیستم ناشی از تعامل با محیط را مدیریت کنند. در نهایت، آینده کنترل کوادراتور تنها در طراحی یک الگوریتم بهتر خلاصه نمی‌شود، بلکه در طراحی یک اکوسیستم توسعه و اعتبارسنجی بهتر نهفته است. درخواست برای محکزنی استاندارد و تکرارپذیری (شکاف ۱)^(۸)، در واقع فراخوانی برای یک متداول‌وزی پژوهشی جدید است. نیاز به انتقال بهتر از شبیه‌سازی به واقعیت (شکاف ۳)، به نیاز برای شبیه‌سازهای با وفاداری بالاتر و تکنیک‌های تصادفی‌سازی دامنه بهتر اشاره دارد. چالش ایمنی و صدور گواهینامه (شکاف^(۴)) نیز به نیاز برای ایزارها و چارچوب‌های تأیید جدید اشاره می‌کند. بنابراین، یک سهم بزرگ در آینده این حوزه ممکن است نه یک کنترل‌کننده جدید، بلکه یک مجموعه محکزنی عمومی، با وفاداری بالا و استاندارد باشد که امکان مقایسه عادلانه و تکرارپذیر همه راهبردهای کنترلی را فراهم آورد و پیشرفت در کل این حوزه را تسريع بخشد.

¹Physics-Informed Machine Learning

²Sample efficiency

³Generalizable policies

فصل ۳

انتخاب کننده و شرح ویژگی‌ها

۱.۳ مدل دینامیکی غیرخطی

اساس هر تحلیل کنترلی، درک صحیح از مدل ریاضی سیستم است. دینامیک یک کوادراتور با شش درجه آزادی، که شامل سه حرکت انتقالی در امتداد محورهای z, y, x و سه حرکت دورانی حول این محورها (ψ, θ, ϕ) است، با استفاده از فرمالیسم نیوتون-اویلر توصیف می‌شود. این مدل، رابطه بین نیروهای ورودی کنترلی و شتاب‌های انتقالی و زاویه‌ای حاصل را بیان می‌کند. ورودی‌های کنترلی سیستم چهار عدد هستند:

نیروی تراست کلی (U_1) که از مجموع نیروی تولیدی هر چهار موتور حاصل می‌شود و سه گشتاور دورانی (U_2, U_3, U_4) که از اختلاف سرعت موتورها ایجاد می‌شوند.

معادلات حاکم بر دینامیک سیستم، ذاتاً غیرخطی هستند. این غیرخطی بودن به دلیل وجود جملات مثلثاتی زوایای اویلر (مانند $\cos(\theta)$ و $\sin(\phi)$) و حاصل ضرب سرعت‌های زاویه‌ای در معادلات گشتاور است. این معادلات پیچیده، رفتار دینامیکی کوادراتور را توصیف می‌کنند و مبنای طراحی هر کنترل‌کننده‌ای قرار می‌گیرند.

$$\ddot{x} = (\cos \phi \sin \theta \cos \psi + \sin \phi \sin \psi) \frac{U_1}{m} \quad (1.3)$$

$$\ddot{y} = (\cos \phi \sin \theta \sin \psi - \sin \phi \cos \psi) \frac{U_1}{m} \quad (2.3)$$

$$\ddot{z} = -g + (\cos \phi \cos \theta) \frac{U_1}{m} \quad (3.3)$$

$$\ddot{\phi} = \dot{\theta} \dot{\psi} \frac{I_{yy} - I_{zz}}{I_{xx}} + \frac{U_2}{I_{xx}} \quad (4.3)$$

$$\ddot{\theta} = \phi \dot{\psi} \frac{I_{zz} - I_{xx}}{I_{yy}} + \frac{U_3}{I_{yy}} \quad (5.3)$$

$$\ddot{\psi} = \dot{\phi} \dot{\theta} \frac{I_{xx} - I_{yy}}{I_{zz}} + \frac{U_4}{I_{zz}} \quad (6.3)$$

در این معادلات، m جرم کوادراتور، g شتاب گرانش و I_{yy}, I_{xx}, I_{zz} ممان‌های اینرسی حول محورهای بدن هستند. ارائه این مدل ریاضی پیچیده، اولین گام در درک این موضوع است که چرا یک کنترل‌کننده خطی ساده برای مدیریت چنین

سیستمی کافی نیست و نیاز به رویکردهای پیشرفته‌تر احساس می‌شود.

۲.۳ تحلیل ویژگی‌های ذاتی سیستم

دینامیک پیچیده کوادراتور دارای سه ویژگی ذاتی است که چالش‌های کنترلی اساسی را به وجود می‌آورند:

کم تحریریکی (Underactuation): کوادراتور یک نمونه کلاسیک از سیستم‌های کم تحریریک است. این سیستم دارای شش درجه آزادی (سه موقعیت و سه زاویه) است که باید کنترل شوند، اما تنها چهار ورودی کنترلی مستقل (سرعت چهار موتور) در اختیار دارد. این عدم تطابق بین تعداد درجات آزادی و ورودی‌های کنترلی به این معناست که نمی‌توان تمام حالت‌های سیستم را به طور مستقیم و مستقل کنترل کرد.

کوپل شدید (Strong Coupling): ماهیت کم تحریریک سیستم مستقیماً منجر به کوپل شدید بین دینامیک دورانی و انتقالی می‌شود. به عنوان مثال، برای ایجاد شتاب در جهت افقی (مثلاً در امتداد محور x)، کوادراتور باید با ایجاد زاویه رول یا پیچ، بردار تراست کلی خود را کج کند تا مؤلفه‌ای از نیرو در آن جهت ایجاد شود. این بدان معناست که کنترل موقعیت و کنترل وضعیت (attitude) به طور جداگانه ناپذیری به هم مرتبط هستند. این ارتباط یک محدودیت بنیادی است که معماري کنترل را دیکته می‌کند و مهندسان را مجبور به استفاده از استراتژی‌های کنترل سلسله‌مراتبی یا آبشاری (cascaded) می‌کند. در این معماری، یک حلقه کنترل بیرونی کنترل برای موقعیت، مقادیر مطلوب (setpoints) را برای یک حلقه کنترل داخلی سریع‌تر برای وضعیت تولید می‌کند.

ناپایداری ذاتی (Inherent Instability): کوادراتور در حلقه باز یک سیستم ذاتاً ناپایدار است. این سیستم، مشابه یک آونگ معکوس، برای حفظ پایداری خود نیازمند کنترل فیدبک فعال و مداوم است. حتی عمل ساده‌ای مانند پرواز ایستا (hovering) نیز یک وظیفه کنترلی غیربدیهی است که نیازمند تنظیم مداوم سرعت موتورها برای مقابله با کوچکترین انحرافات است.

وابستگی حیاتی دقت موقعیت به پایداری وضعیت، یک پیامد مهم دیگر دارد: عملکرد کل سیستم به طور نامتناسبی به مقاومت و دقت حلقه کنترل وضعیت داخلی وابسته است. یک خطای کوچک و گذرا در کنترل وضعیت، در طول زمان انتگرال‌گیری شده و به یک خطای بزرگ و بالقوه غیرقابل جبران در موقعیت تبدیل می‌شود. این موضوع اهمیت کنترل کننده‌هایی را که پایداری وضعیت سریع، مقاوم و دقیقی را فراهم می‌کنند، دوچندان می‌کند و توجیه کننده استفاده از کنترل کننده‌های بسیار پاسخگو مانند Fuzzy-PID یا کنترل مدل‌لغزشی (SMC) در حلقه داخلی است.

۳.۲ چالش‌های عملیاتی: اغتشاشات و عدم قطعیت‌ها

علاوه بر پیچیدگی‌های ذاتی مدل، کوادراتورها در محیط‌های واقعی با مجموعه‌ای از چالش‌های عملیاتی مواجه هستند که کنترل کننده باید بر آن‌ها غلبه کند:

اغتشاشات خارجی: سیستم به شدت به اغتشاشات خارجی مانند تندبادها و تلاطم‌های جوی حساس است. این عوامل نیروها و گشتاورهای مدل‌سازی نشده‌ای را بر بدن پرنده اعمال می‌کنند که می‌توانند به سرعت آن را از مسیر مطلوب منحرف کنند.

عدم قطعیت‌های داخلی و پارامتری: کنترل کننده با عدم قطعیت‌های متعددی از منابع داخلی نیز روبرو است که شامل موارد زیر می‌شود:

عدم دقیق مدل: همواره تفاوت‌هایی بین مدل ریاضی و سیستم فیزیکی وجود دارد.

تغییرات باز: در کاربردهایی مانند پهپادهای سه‌پاشهای کشاورزی یا حمل محموله، تغییرات در جرم و مرکز ثقل سیستم، خواص اینرسی آن را تغییر می‌دهد و دینامیک پرواز را تحت تأثیر قرار می‌دهد.

خطاهای عملگر/حسگر: احتمال آسیب به موتورها، نوسانات ولتاژ یا نقص در عملکرد حسگرهای می‌تواند رفتارهای غیرقابل پیش‌بینی را در سیستم ایجاد کرده و پایداری را به خطر اندازد.

در مجموع، دینامیک غیرخطی، کوپل شده، کم‌تحریک و ناپایدار کوادراتور، همراه با حساسیت آن به اغتشاشات و عدم قطعیت‌ها، یک مسئله کنترلی چالش‌برانگیز را ایجاد می‌کند که راه حل‌های کنترل خطی سنتی را به مرزهای کارایی خود می‌رساند و نیاز به یک رویکرد کنترل مقاوم و تطبیقی را به وضوح نشان می‌دهد.

۴.۳ کنترل کننده Fuzzy-PID

کوادراتور، یک وسیله نقلیه هوایی بدون سرنشین (*UAV*), به عنوان یک پلتفرم رباتیک شش درجه آزادی ($6 - DOF$) تعریف می‌شود. این شش درجه آزادی شامل سه حرکت انتقالی در امتداد محورهای مختصات دکارتی (x, y, z) و سه حرکت دورانی حول این محورها است که به ترتیب با نام‌های رول (*Roll*), پیچ (*Pitch*) و یاوه (*Yaw*) شناخته می‌شوند. در سال‌های اخیر، به دلیل سادگی ساختار مکانیکی، قابلیت برخاست و فرود عمودی (*VTOL*) و قدرت مانور بالا، کوادراتورها به یکی از محبوب‌ترین پلتفرم‌ها در تحقیقات دانشگاهی و کاربردهای صنعتی تبدیل شده‌اند. با این حال، این سادگی مکانیکی، یک چالش کنترلی پیچیده را پنهان می‌کند. کوادراتور ذاتاً یک سیستم ناپایدار، غیرخطی و کم-عملکر (*Underactuated*) است. این ویژگی‌ها به این معناست که بدون یک سیستم کنترل حلقه-بسته فعال و مقاوم، پرنده قادر به حفظ تعادل و پرواز پایدار نخواهد بود. بنابراین، طراحی یک کنترل کننده کارآمد که بتواند بر این ناپایداری‌ها غلبه کرده و پروازی دقیق و قابل اعتماد را تضمین کند، هسته اصلی مهندسی کنترل در این حوزه را تشکیل می‌دهد.

۱.۴.۳ ارزیابی انتقادی کنترل کننده PID

کنترل کننده تناسبی-انتگرالی-مشتقی (*PID* یا *Proportional – Integral – Derivative*) به دلیل سادگی ساختاری و عملکرد قابل قبول در طیف وسیعی از سیستم‌ها، پرکاربردترین الگوریتم کنترل بازخوردی در صنعت است. قانون کنترل *PID* به صورت ریاضی با معادله زیر تعریف می‌شود:

$$u(t) = K_p \left(e(t) + \frac{1}{T_i} \int_0^t e(\tau) d\tau + T_d \frac{d}{dt} e(t) \right) \quad (7.3)$$

در این معادله، $u(t)$ خروجی کنترل کننده، $e(t)$ سیگنال خطای تفاوت بین مقدار مطلوب و مقدار واقعی)، و Kd, Ki, Kp به ترتیب بهره‌های تناسبی، انتگرالی و مشتقی هستند. نقش هر یک از این جملات در کنترل کوادراتور به شرح زیر است: جمله تناسبی (*P*): این جمله متناسب با خطای فعلی عمل می‌کند و نیروی محرکه اصلی برای کاهش خطای فراهم می‌آورد. بهره Kp بالا، سرعت پاسخ سیستم را افزایش می‌دهد اما می‌تواند منجر به نوسانات و ناپایداری شود. جمله انتگرالی (*I*): این جمله با جمع‌آوری خطاهای گذشته، خطای حالت ماندگار (*steady-state error*) را از بین می‌برد. این ویژگی تضمین می‌کند که کوادراتور در نهایت دقیقاً به زاویه یا موقعیت مطلوب خود می‌رسد. با این حال، افزایش بهره Ki می‌تواند پاسخ گذرا را کند کرده و فراجهش (*overshoot*) را افزایش دهد. جمله مشتقی (*D*): این جمله با پاسخ به نرخ تغییرات خطای رفتار آینده سیستم را پیش‌بینی می‌کند. عملکرد آن شبیه یک ترمز است که نوسانات را میرا کرده و از فراجهش جلوگیری می‌کند و به پایداری سیستم کمک می‌کند. بهره Kd بالا، پایداری را بهبود می‌بخشد اما سیستم را به نویز فرکانس بالا بسیار حساس می‌کند.

در کنترل کوادراتور، معمولاً از یک ساختار کنترلی آشیاری (*cascaded*) استفاده می‌شود که در آن دو حلقة خارجی به کار گرفته می‌شود: یک حلقة داخلی برای کنترل وضعیت (زوايا) و یک حلقة خارجی برای کنترل موقعیت. حلقة خارجی خطای موقعیت را به عنوان ورودی دریافت کرده و زوایای رول و پیچ مطلوب را به عنوان خروجی تولید می‌کند. سیس حلقة داخلی این زوایای مطلوب را به عنوان نقطه تنظیم (*setpoint*) خود دریافت کرده و با کنترل گشتاورها، پرنده را به آن وضعیت هدایت می‌کند. کنترل کننده‌های *PID* به دلیل پیاده‌سازی آسان و عملکرد رضایت‌بخش در شرایط پروازی محدود و مشخص، به طور گسترده در کوادراتورهای تجاری و تحقیقاتی اولیه مورد استفاده قرار گرفته‌اند.

با وجود کاربرد گسترده، کنترل کننده *PID* مرسوم دارای محدودیت‌های ذاتی است که عملکرد آن را در کاربردهای پیچیده‌ای مانند کنترل کوادراتور به شدت به چالش می‌کشد.

۱. چالش تنظیم بهینه بهره‌ها: فرآیند یافتن مقادیر بهینه برای سه بهره Kp, Ki, Kd که به آن "تنظیم" (*Tuning*)

گفته می‌شود، یک فرآیند پیچیده و زمانبر است. روش‌های کلاسیک مانند زیگلر-نیکولز یا آزمون و خطای دستی، اغلب به نتایج زیربهینه منجر می‌شوند و تضمینی برای عملکرد مطلوب در تمام شرایط ندارند. پیچیدگی این فرآیند تا حدی است که اغلب از آن به عنوان یک "هنر" یاد می‌شود تا یک علم دقیق، که خود نشان‌دهنده نبود یک رویکرد سیستماتیک برای سیستم‌های پیچیده است.

۲. افت عملکرد با تغییر پارامترهای سیستم: یک کنترل کننده PID برای یک مجموعه شرایط خاص (مثلًاً یک کوادراتور با باتری پر و بدون بار اضافی) تنظیم می‌شود. هرگونه تغییر در پارامترهای سیستم، مانند کاهش ولتاژ باتری در طول پرواز یا تغییر در جرم کل به دلیل حمل بار (مانند یک کوادراتور سمپاچ)، باعث تغییر در دینامیک پرنده می‌شود. کنترل کننده PID با بهره‌های ثابت قادر به تطبیق خود با این تغییرات نیست و در نتیجه عملکرد آن به شدت افت کرده و حتی ممکن است ناپایدار شود.

۳. ناتوانی در مدیریت غیرخطی بودن شدید: همانطور که در بخش اول نشان داده شد، دینامیک کوادراتور به شدت غیرخطی است. یک کنترل کننده خطی با بهره‌های ثابت نمی‌تواند عملکردی یکنواخت و بهینه را در سراسر پوشش پروازی (برای مثال، در مانورهای سریع با زوایای زیاد در مقابل پرواز شناور آرام) ارائه دهد.

۴. حساسیت به نویز: جمله مشتقی (D) به نویز فرکانس بالا که به طور معمول در خروجی سنسورهای $MEMS$ (مانند ژیروسکوپ و شتابسنج) وجود دارد، بسیار حساس است. این نویز می‌تواند توسط جمله مشتقی تقویت شده و منجر به تولید سیگنال‌های کنترلی نامنظم و پرنوسان شود که به عملکردها (موتورها) آسیب می‌رساند و باعث اشتعاع آن‌ها می‌شود. این مشکل اغلب طراحان را مجبور می‌کند تا از فیلترهای پایین‌گذر استفاده کنند یا به طور کلی جمله مشتقی را حذف کرده و به یک کنترل کننده PI با عملکرد ضعیف‌تر بسته کنند.

۵. توازن ناکارآمد بین سرعت پاسخ و پایداری: در کنترل PID کلاسیک، یک توازن ($trade-off$) ذاتی بین سرعت پاسخ و پایداری وجود دارد. افزایش بهره تنبایی (K_p) برای دستیابی به پاسخ سریع‌تر، معمولاً منجر به افزایش فراجهش و کاهش حاشیه پایداری می‌شود. دستیابی همزمان به زمان صعود کوتاه و فراجهش کم، با استفاده از روش‌های تنظیم مرسوم تقریباً غیرممکن است.

محدودیت اصلی کنترل کننده PID صرفاً دشواری تنظیم آن نیست، بلکه "شکنندگی بهینگی" آن است. به عبارت دیگر، هر مجموعه "بهینه" از بهره‌های PID ، تنها برای یک لحظه خاص و یک مجموعه شرایط کاملاً مشخص، بهینه است. از آنجایی که کوادراتور یک سیستم پویا در حال تغییر مداوم است (جرم متغیر، ولتاژ باتری در حال کاهش، شرایط آبودینامیکی متغیر)، یک کنترل کننده با پارامترهای ثابت، یک پارادایم اساساً ناقص برای این کاربرد است. مشکل، ساختار PID نیست، بلکه ماهیت ایستا و غیرتطبیقی آن در یک دنیای دینامیک است. این تحلیل نشان می‌دهد که سیستم کنترل باید دارای قابلیت تطبیق‌پذیری برای حفظ عملکرد باشد، ویژگی‌ای که کنترل کننده PID مرسوم فاقد آن است. این نتیجه‌گیری، بحث را از "PID دشوار است" به "PID اساساً برای این کاربرد نامناسب است" تغییر می‌دهد و راه را برای جستجوی راه حل‌های هوشمند و تطبیقی هموار می‌سازد.

۲.۴.۳ مقدمه‌ای بر منطق فازی

منطق فازی، که توسط پروفسور لطفی‌زاده معرفی شد، شکلی از منطق چند-ارزشی است که برخلاف منطق کلاسیک بولین که تنها با دو ارزش "صحیح" (۱) یا "غلط" (۰) سروکار دارد، به متغیرها اجازه می‌دهد تا درجات عضویتی بین ۰ و ۱ داشته باشند. این ویژگی، منطق فازی را قادر می‌سازد تا با عدم قطعیت، ابهام و اطلاعات ناقص که مشخصه دنیای واقعی هستند، به شیوه‌ای مؤثر برخورد کند. منطق فازی در هسته خود، روشی برای فرموله کردن دانش و تجربه انسانی و استدلال‌های مبتنی

بر زبان طبیعی در یک چارچوب ریاضی قابل محاسبه است. این پارادایم به جای تکیه بر مدل‌های ریاضی دقیق، از دانش طراح برای تصمیم‌گیری استفاده می‌کند.

یک سیستم استنتاج فازی (Fuzzy Inference System) از چهار جز اصلی تشکیل شده است که با همکاری یکدیگر، یک ورودی عددی (*crisp*) را به یک خروجی عددی تبدیل می‌کنند.

۱. فازی‌سازی (Fuzzification): این مرحله، اولین گام در یک سیستم فازی است و وظیفه آن تبدیل مقادیر ورودی عددی و دقیق (مانند خطای زاویه برابر با 5° -درجه) به مجموعه‌های فازی است. این کار از طریق "تابع عضویت" (Membership Functions) انجام می‌شود. یک تابع عضویت، درجه تعلق (بین 0 و 1) یک مقدار ورودی به یک "متغیر زبانی" (Linguistic Variable) را مشخص می‌کند. متغیرهای زبانی، مفاهیمی کیفی مانند "منفی بزرگ"، "صفر"، "ثبت کوچک" هستند. تابع عضویت می‌تواند اشکال مختلفی مانند مثلثی، ذوزنقه‌ای یا گوسی داشته باشد. برای مثال، خطای 5° -درجه ممکن است 80% به مجموعه "منفی کوچک" و 20% به مجموعه "صفر" تعلق داشته باشد.

۲. پایگاه دانش (Knowledge Base): این بخش قلب هوشمند یک سیستم فازی است و از دو جز تشکیل شده است: پایگاه داده (تعریف تابع عضویت) و "پایگاه قوانین" (Rule Base). پایگاه قوانین مجموعه‌ای از قوانین شرطی "اگر-آنگاه" ($IF - THEN$) است که استراتژی کنترل را بر اساس دانش خبره و به صورت زبانی تعریف می‌کند. این قوانین، روابط بین متغیرهای ورودی و خروجی فازی را بیان می‌کنند. یک قانون نمونه می‌تواند به این صورت باشد: "اگر (IF) خطا منفی بزرگ است و (AND) نرخ تغییر خطا صفر است، آنگاه ($THEN$) خروجی کنترل منفی بزرگ است".

۳. موتور استنتاج (Inference Engine): این مؤلفه، مغز عملیاتی سیستم است که قوانین فازی را بر روی ورودی‌های فازی‌شده اعمال می‌کند تا یک خروجی فازی تولید کند. این فرآیند شامل دو مرحله است:

ارزیابی مقدم (Antecedent): با استفاده از عملگرهای فازی (مانند AND که معمولاً با عملگر \min پیاده‌سازی می‌شود، و OR که با \max پیاده‌سازی می‌شود)، درجه صدق بخش "اگر" (IF) هر قانون محاسبه می‌شود. استلزم (Implication): نتیجه به دست آمده از مرحله قبل برای "شکل‌دهی" به بخش "آنگاه" ($THEN$) یا نتیجه (Consequent) هر قانون استفاده می‌شود. این کار معمولاً با "برش دادن" (*clipping*) یا "مقیاس‌دهی" (*scaling*) تابع عضویت خروجی انجام می‌شود.

در نهایت، نتایج فازی حاصل از تمام قوانینی که فعال شده‌اند، با یکدیگر "تجمیع" (Aggregation) می‌شوند تا یک مجموعه فازی واحد برای خروجی نهایی ایجاد کنند.

۴. فازی‌زدایی (Defuzzification): از آنجایی که سیستم‌های فیزیکی (مانند موتورهای کوادراتور) به یک سیگنال کنترلی عددی و دقیق نیاز دارند، خروجی فازی تجمیع شده باید به یک مقدار عددی تبدیل شود. این فرآیند، فازی‌زدایی نام دارد. روش‌های مختلفی برای این کار وجود دارد که متد اداول‌ترین آن‌ها "روش مرکز ثقل" (Centroid of Area) است. در این روش، مرکز ثقل سطح زیر نمودار تابع عضویت خروجی تجمیع شده محاسبه می‌شود و به عنوان خروجی نهایی سیستم در نظر گرفته می‌شود.

منطق فازی یک چارچوب قدرتمند برای فرموله کردن دانش شهودی و مبتنی بر هیوریستیک یک اپراتور انسانی خبره فراهم می‌کند. یک خلبان ماهر برای کنترل پرنده، معادلات دیفرانسیل را در ذهن خود حل نمی‌کند؛ بلکه بر اساس تجربه و اکنش نشان می‌دهد ("پهپاد به سرعت در حال کله کردن به جلو است، پس باید یک گشتاور مخالف قوی اعمال کنم"). پایگاه قوانین فازی، یک پیاده‌سازی مستقیم و قابل محاسبه از همین دانش خبره است. منطق فازی این شکاف را با اجازه

دادن به ما برای نوشتن هیوریستیک‌های انسانی در یک ساختار منظم (پایگاه قوانین) و استفاده از نظریه مجموعه‌های فازی برای درون‌یابی بین این قوانین، پر می‌کند و یک سطح کنترل غیرخطی و هموار ایجاد می‌نماید. این ویژگی آن را به ابزاری ایده‌آل برای سیستم‌هایی تبدیل می‌کند که مدل‌سازی دقیق آن‌ها دشوار است یا به پاسخ‌های تطبیقی و شبکه‌ای انسانی نیاز دارند.

فصل ۴

شبیه‌سازی کنترل کننده در نرم افزار متلب / سیمولینک

پس از تبیین مبانی نظری، مرور ادبیات و انتخاب معماری کنترل کننده مقاوم Fuzzy-PID در فصل‌های پیشین، در این فصل چارچوب پیاده‌سازی و ارزیابی عددی سامانه ارائه می‌شود. هدف اصلی، اعتبارسنجی عملکرد حلقه‌های کنترلی بر روی مدل دینامیکی شش‌درجه‌آزادی کوادراتور و سنجش میزان پایداری، دقت رهگیری و تاب آوری در برابر عدم قطعیت‌ها و اغتشاشات محیطی است. محیط پیاده‌سازی MATLAB/Simulink انتخاب شده تا همخوان با نمادگذاری و متغیرهای حالت/ورودی معرفی شده در گزارش باشد (وضعیت

$$[x, y, z, \phi, \theta, \psi]$$

$$[u, v, w, p, q, r]$$
 سرعت‌ها

$$U_1 \text{ تا } U_4$$
 و ورودی‌های کنترلی

).

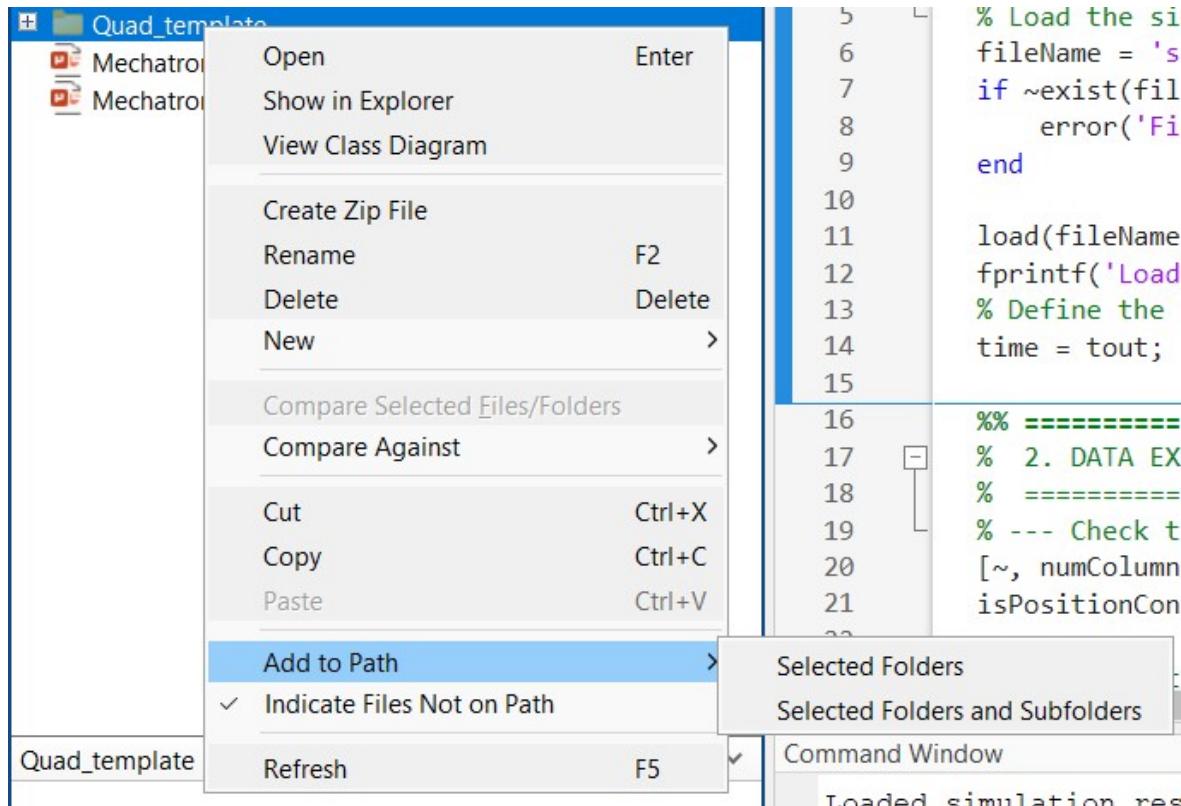
ساختار شبیه‌سازی به صورت آبشاری سازمان‌دهی شده است: حلقه بیرونی «کنترل موقعیت» فرمان‌های مسیر را به ارجاع‌های وضعیت تبدیل می‌کند و حلقه درونی «کنترل وضعیت اارتفاع» این ارجاع‌ها را به گشتاورها و تراست متناظر رسانده و از طریق بخش «اختلاط فرامین» به اامر چهار موتور نگاشت می‌کند؛ سپس مدل دینامیکی پرنده (Plant) پاسخ سامانه را تولید می‌کند. نمودار کلی فایل شبیه‌سازی و جریان سیگنال‌ها در ابتدای فصل نمایش داده می‌شود تا اجزای فرمان، کنترل کننده‌ها، اختلاط و پرنده (Plant) به صورت یکپارچه قبله ریدایی باشند.

سازمان فصل به این صورت است: ابتدا پیکربندی مدل و داده‌های ورودی (بارامترهای فیزیکی، شرایط اولیه، فرمان مسیر و ثبت داده‌ها) شرح داده می‌شود؛ سپس پیاده‌سازی کنترل کننده و جزئیات تنظیم آن ارائه می‌گردد؛ در ادامه سناریوهای آزمایشی و نتایج عددی/نموداری به همراه بحث و تحلیل مقایسه‌ای آورده می‌شود؛ و نهایتاً جمع‌بندی فصل، نکات پیاده‌سازی و توصیه‌های کاربردی برای انتقال به سخت‌افزار بیان می‌گردد. این فصل پل میان طراحی نظری و ارزیابی عملی است و بستر قضاوت درباره کارایی و دوام راهبرد کنترلی پیشنهادی را فراهم می‌سازد.

۱.۴ مراحل ابتدایی شروع شبیه‌سازی

۱.۱.۴ اضافه کردن فایل‌ها به محیط متلب

ابتدا پوشه مربوط به پروژه و تمامی زیرپوشه‌های مربوط به آن باید به مسیر اضافه شوند، روی پوشه مربوط به پروژه کلیک کرده و مانند تصویر زیر تمام فایل‌های مورد نیاز را به پروژه اضافه می‌کنیم:



شکل ۱.۴: نحوه اضافه کردن فایل‌ها به مسیر

بعد از انتخاب گزینه Selected Folders and Subfolders تمامی فایل‌های موجود در شبیه‌سازی به مسیر متلب اضافه شده و قابل استفاده می‌باشد.

۲.۱.۴ بارگذاری مسیر برای کنترل موقعیت

فرآیند بارگذاری مسیر یک مرحله مقدماتی و ضروری است که توسط بلوک خاکستری رنگ LOAD در مدل سیمولینک مدیریت می‌شود. این بلوک وظیفه دارد تا پیش از شروع شبیه‌سازی، تمام داده‌های لازم را به محیط متلب و سیمولینک تزریق کند تا شبیه‌سازی با یک مجموعه اطلاعات منسجم و صحیح آغاز شود.

برای هر یک از سه مسیر تعریف شده، یک اسکریپت مجازی MATLAB توسعه داده شده است تا مسیرهای مرجع مورد نیاز برای شبیه‌سازی را تولید کند. این اسکریپتها با هدف ایجاد داده‌هایی با ساختار یکسان و سازگار با مدل سیمولینک طراحی شده‌اند. فرآیند کلی در هر سه اسکریپت شامل تعریف پارامترهای اصلی مسیر (مانند ابعاد، زمان و نرخ نمونه‌برداری)، تولید بردارهای زمانی و مختصات برای هر بخش از مسیر، ترکیب این بخش‌ها برای ساخت مسیر نهایی، و در نهایت بسته‌بندی داده‌ها در قالب یک ساختار به نام path است که هر یک از مؤلفه‌های آن (x, y, z, psi) یک شیء timeseries متلب می‌باشد. این ساختار استاندارد، بارگذاری و استفاده از مسیرها را در محیط سیمولینک تسهیل می‌کند.

```

% --- Script to Generate a Path: Takeoff, Circle, Return, and Land at Origin ---
%% 1. Define Path Parameters
height = 2;           % meters
radius = 1.5;          % meters
sample_rate = 50;       % points per second (Hz)

takeoff_time = 4;      % seconds to reach altitude
circle_time = 10;       % seconds to complete the circle
return_time = 3;        % --- NEW: seconds to return to the center ---
land_time = 4;          % seconds to land

%% 2. Generate Time and Angle Vectors
% Create time vectors for each of the four phases
t1 = (0:1/sample_rate:takeoff_time)';
t2 = (t1(end) + 1/sample_rate : 1/sample_rate : t1(end) + circle_time)';
t3 = (t2(end) + 1/sample_rate : 1/sample_rate : t2(end) + return_time)'; % --- NEW: Return time ---
t4 = (t3(end) + 1/sample_rate : 1/sample_rate : t3(end) + land_time)'; % --- NEW: Landing time ---

% Full time vector
time_vector = [t1; t2; t3; t4];

% Angle vector for the circle
angle = linspace(0, 2*pi, length(t2))';

%% 3. Generate Coordinates for Each Phase
% --- Phase 1: Takeoff ---
x_takeoff = zeros(size(t1));
y_takeoff = zeros(size(t1));
z_takeoff = linspace(0, height, length(t1))';

% --- Phase 2: Circle ---
x_circle = radius * cos(angle);
y_circle = radius * sin(angle);
z_circle = ones(size(t2)) * height;

% --- Phase 3: Return to Center (NEW) ---
x_return = linspace(x_circle(end), 0, length(t3)); % Fly from circle's end X to 0
y_return = linspace(y_circle(end), 0, length(t3)); % Fly from circle's end Y to 0
z_return = ones(size(t3)) * height;                 % Maintain altitude

% --- Phase 4: Land at Origin (Modified) ---
x_land = zeros(size(t4)); % Land at X=0
y_land = zeros(size(t4)); % Land at Y=0
z_land = linspace(height, 0, length(t4)); % Descend to Z=0

% --- Combine All Four Phases ---
x_path = [x_takeoff; x_circle; x_return; x_land];
y_path = [y_takeoff; y_circle; y_return; y_land];
z_path = [z_takeoff; z_circle; z_return; z_land];
psi_path = zeros(size(time_vector));

%% 4. Create the Path Structure with Timeseries Objects
path.x = timeseries(x_path, time_vector);
path.y = timeseries(y_path, time_vector);
path.z = timeseries(z_path, time_vector);
path.psi = timeseries(psi_path, time_vector);

%% 5. Save the Path to a new .mat File
save('Path_Takeoff_Circle.mat', 'path');
disp('File "Path_Takeoff_Circle.mat" created successfully.');

%% 6. Plot the Generated Path
try
    figure('Name', 'Generated 3D Path: Takeoff, Circle, Return, Land');
    plot3(x_path, y_path, z_path, 'b-', 'LineWidth', 2);

    hold on;
    plot3(x_path(1), y_path(1), z_path(1), 'go', 'MarkerFaceColor', 'g', 'MarkerSize', 10); % Start
    plot3(x_path(end), y_path(end), z_path(end), 'ro', 'MarkerFaceColor', 'r', 'MarkerSize', 10); % End
    hold off;

    grid on;
    xlabel('X-axis (m)');
    ylabel('Y-axis (m)');
    zlabel('Z-axis (m)');
    title('Flight Trajectory: Takeoff, Circle, Return, Land');
    legend('Path', 'Start', 'End', 'Location', 'best');
    axis equal;
    view(30, 20);

    disp('3D plot generated successfully.');
catch ME
    warning(['Could not generate 3D plot. Error: %s', ME.message]);
end

```

```
% --- Script to Generate Path: Takeoff, Figure-Eight, Land ---

clear;
clc;

%% 1. Parameters
height = 5; % meters
x_radius = 4; % meters (width of the eight)
y_radius = 2; % meters (height of the eight lobes)
sample_rate = 50; % Hz
takeoff_time = 5;
figure_eight_time = 20;
land_time = 5;

%% 2. Time Vectors
t1 = (0:1/sample_rate:takeoff_time)';
t2 = (takeoff_time + 1/sample_rate : 1/sample_rate : takeoff_time + figure_eight_time)';
t3 = (t2(end) + 1/sample_rate : 1/sample_rate : t2(end) + land_time)';
time_vector = [t1; t2; t3];
angle = linspace(0, 2*pi, length(t2))';

%% 3. Coordinates
% Takeoff
x_takeoff = zeros(size(t1));
y_takeoff = zeros(size(t1));
z_takeoff = linspace(0, height, length(t1))';
% Figure-Eight
x_figure_eight = x_radius * sin(angle);
y_figure_eight = y_radius * sin(2*angle);
z_figure_eight = ones(size(t2)) * height;
% Land
x_land = ones(size(t3)) * x_figure_eight(end);
y_land = ones(size(t3)) * y_figure_eight(end);
z_land = linspace(height, 0, length(t3))';
% Combine
x_path = [x_takeoff; x_figure_eight; x_land];
y_path = [y_takeoff; y_figure_eight; y_land];
z_path = [z_takeoff; z_figure_eight; z_land];
psi_path = zeros(size(time_vector));

%% 4. Create Structure
path.x = timeseries(x_path, time_vector);
path.y = timeseries(y_path, time_vector);
path.z = timeseries(z_path, time_vector);
path.psi = timeseries(psi_path, time_vector);

%% 5. Save File
save('Path_Takeoff_FigureEight_Land.mat', 'path');
disp('File "Path_Takeoff_FigureEight_Land.mat" created successfully.');

%% 6. Plot
figure('Name', 'Generated Path: Figure-Eight');
plot3(x_path, y_path, z_path, 'b-', 'LineWidth', 2);
hold on;
plot3(x_path(1), y_path(1), z_path(1), 'go', 'MarkerFaceColor', 'g', 'MarkerSize', 10);
plot3(x_path(end), y_path(end), z_path(end), 'ro', 'MarkerFaceColor', 'r', 'MarkerSize', 10);
grid on; xlabel('X (m)'); ylabel('Y (m)'); zlabel('Z (m)');
title('Generated Path: Takeoff, Figure-Eight, Land'); axis equal; view(45, 25);
```

شکل ۴.۳: کد استفاده شده برای ایجاد مسیر هشتی

```
% --- Script to Generate Path: Diamond with Return and Landing at Origin --
%% 1. Parameters
height = 3; % meters
points_per_leg = 100; % Number of points for each segment
takeoff_time = 4;
return_time = 3; % --- NEW: Time to return to center ---
land_time = 4; % Time for landing phase
sample_rate = 50;

% Diamond vertices [x, y, z]
v1 = [0, 1.5, height];
v2 = [-1.5, 0, height];
v3 = [0, -1.5, height];
v4 = [1.5, 0, height];
vertices = [v1; v2; v3; v4; v1]; % Path ends back at v1

%% 2. Coordinates & Time
% --- Phase 1: Takeoff ---
t_takeoff = (0:1/sample_rate:takeoff_time)';
x_takeoff = zeros(size(t_takeoff));
y_takeoff = zeros(size(t_takeoff));
z_takeoff = linspace(0, height, length(t_takeoff)');

% --- Phase 2: Diamond Legs ---
x_diamond = []; y_diamond = []; z_diamond = [];
for i = 1:size(vertices, 1)-1
    x_leg = linspace(vertices(i,1), vertices(i+1,1), points_per_leg);
    y_leg = linspace(vertices(i,2), vertices(i+1,2), points_per_leg);
    z_leg = ones(points_per_leg, 1) * height;

    x_diamond = [x_diamond; x_leg(2:end)];
    y_diamond = [y_diamond; y_leg(2:end)];
    z_diamond = [z_diamond; z_leg(2:end)];
end
total_diamond_points = length(x_diamond);
diamond_duration = (total_diamond_points / sample_rate);
t_diamond = linspace(t_takeoff(end) + 1/sample_rate, t_takeoff(end) + diamond_duration,
total_diamond_points);

% --- Phase 3: Return to Center (NEW) ---
t_return = (t_diamond(end) + 1/sample_rate : 1/sample_rate : t_diamond(end) + return_time)';
x_return = linspace(x_diamond(end), 0, length(t_return)); % Fly from diamond's end X to 0
y_return = linspace(y_diamond(end), 0, length(t_return)); % Fly from diamond's end Y to 0
z_return = ones(size(t_return)) * height; % Maintain altitude

% --- Phase 4: Land at Origin (Modified) ---
t_land = (t_return(end) + 1/sample_rate : 1/sample_rate : t_return(end) + land_time)';
x_land = zeros(size(t_land)); % Land at X=0
y_land = zeros(size(t_land)); % Land at Y=0
z_land = linspace(height, 0, length(t_land)); % Descend to Z=0

% --- Combine all phases ---
x_path = [x_takeoff; x_diamond; x_return; x_land];
y_path = [y_takeoff; y_diamond; y_return; y_land];
z_path = [z_takeoff; z_diamond; z_return; z_land];
time_vector = [t_takeoff; t_diamond; t_return; t_land];
psi_path = zeros(size(time_vector));

%% 3. Create Structure
path.x = timeseries(x_path, time_vector);
path.y = timeseries(y_path, time_vector);
path.z = timeseries(z_path, time_vector);
path.psi = timeseries(psi_path, time_vector);

%% 4. Save File
save('Path_Diamond_Return_Land.mat', 'path');
disp('File "Path_Diamond_Return_Land.mat" created successfully.');

%% 5. Plot
figure('Name', 'Generated Path: Diamond with Return and Landing');
plot3(x_path, y_path, z_path, 'b-', 'LineWidth', 2);
hold on;
plot3(x_path(1), y_path(1), z_path(1), 'go', 'MarkerFaceColor', 'g', 'MarkerSize', 10);
plot3(x_path(end), y_path(end), z_path(end), 'ro', 'MarkerFaceColor', 'r', 'MarkerSize', 10);
hold off;
grid on; xlabel('X (m)'); ylabel('Y (m)'); zlabel('Z (m)');
title('Generated Path: Diamond with Return and Landing'); axis equal; view(30, 20);
```

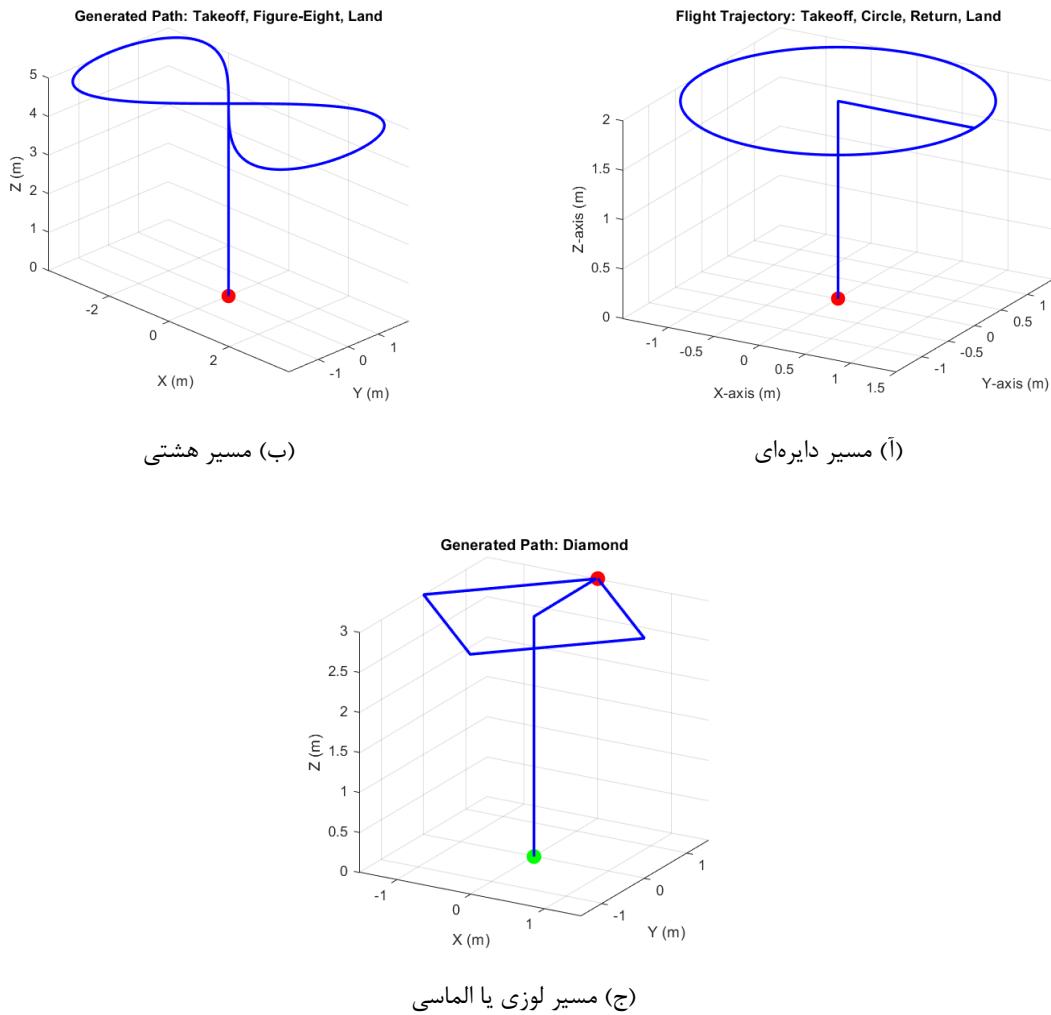
شکل ۴.۴: کد استفاده شده برای ایجاد مسیر لوزی

۱. مسیر برخاست و حرکت دایره‌ای (**Takeoff and Circle**): این اسکریپت یک مسیر پایه و استاندارد برای ارزیابی پایداری و قابلیت رهگیری منحنی‌های هموار تولید می‌کند. کد ابتدا مسیر را به دو فاز مجزا تقسیم می‌کند: برخاست (Takeoff) و حرکت دایره‌ای (Circle). برای فاز برخاست، مختصات X و Y صفر نگه داشته می‌شوند و مختصات Z با استفاده از دستور `linspace` به صورت خطی از صفر تا ارتفاع مطلوب (۲ متر) در بازه زمانی مشخص افزایش می‌یابد. برای فاز حرکت دایره‌ای، ارتفاع ثابت باقی می‌ماند و مختصات X و Y با استفاده از معادلات پارامتری دایره ($x=rcos(\theta), y=rsin(\theta)$) محاسبه می‌شوند، که در آن زاویه theta به صورت خطی از 0 تا 2π تغییر می‌کند تا یک دایره کامل با شعاع ۱.۵ متر ایجاد شود. در نهایت، داده‌های دو فاز به یکدیگر متصل شده و به همراه بردار زمانی کامل، در ساختار `path` ذخیره می‌شوند.

۲. مسیر برخاست، حرکت هشتی و فرود (**Takeoff, Figure-Eight, Land**): برای تولید این مسیر پیچیده و تهاجمی، اسکریپت مسیر را به سه فاز برخاست، حرکت هشتی و فرود تقسیم می‌کند. پس از فاز برخاست که مشابه مسیر قبلی است، فاز حرکت هشتی با استفاده از منحنی لیساژو (Lissajous curve) پیاده‌سازی می‌شود. در این روش، مختصات X و Y با توابع سینوسی با فرکانس‌های متفاوت تعریف می‌شوند (عموماً با نسبت فرکانس ۱ به ۲، مانند $x=A\sin(t), y=B\sin(2t)$) تا شکل عدد هشت انجلیسی ایجاد شود. در این فاز، ارتفاع ثابت باقی می‌ماند. برای فاز فرود، موقعیت X و Y در نقطه پایانی مسیر هشتی ثابت نگه داشته شده و ارتفاع با دستور `linspace` به صورت خطی از مقدار بیشینه به صفر کاهش می‌یابد. این سه بخش در نهایت با یکدیگر ترکیب شده و یک مسیر کامل و پیوسته را تشکیل می‌دهند که قابلیت مانورپذیری کنترل کننده را به چالش می‌کشد.

۳. مسیر الماسی (**Diamond Path**): رویکرد ساخت این مسیر با دو مسیر قبلی متفاوت است و بر اساس تعریف نقاط کلیدی (Waypoints) انجام می‌شود. این اسکریپت ابتدا رئوس یک شکل لوزی را در صفحه افقی در ارتفاع مطلوب (۳ متر) تعریف می‌کند. مسیر از ترکیب یک فاز برخاست اولیه و چهار فاز حرکت خطی بین این رئوس تشکیل می‌شود. پس از فاز برخاست، کد با استفاده از دستور `linspace`، به صورت متوالی بین هر دو رأس یک خط مستقیم تولید می‌کند. با اتصال این چهار پاره خط، یک مسیر الماسی شکل با گوشه‌های تیز ایجاد می‌شود. این روش تولید مسیر، به طور خاص برای ارزیابی پاسخ گذرای کنترل کننده به تغییرات ناگهانی در مسیر مرجع (وروودی پله در سرعت) بسیار مناسب است. در تمام مراحل، زاویه یا و (psi) صفر در نظر گرفته شده تا تمرکز آزمون بر روی کنترل موقعیت باشد.

شکل ۵.۴: شبیه‌سازی مسیرهای ایجاد شده برای آزمون عملکرد کوادراتور



هر یک از سه مسیر پرواز تعریف شده، هدفمندانه برای ارزیابی جنبه‌ای خاص از عملکرد و مقاومت کنترل کننده کوادراتور انتخاب شده‌اند. این مسیرها صرفاً مسیرهای تصادفی نیستند، بلکه سناریوهای آزمون استانداردی هستند که هر کدام یک چالش کنترلی مشخص را مطرح می‌کند.

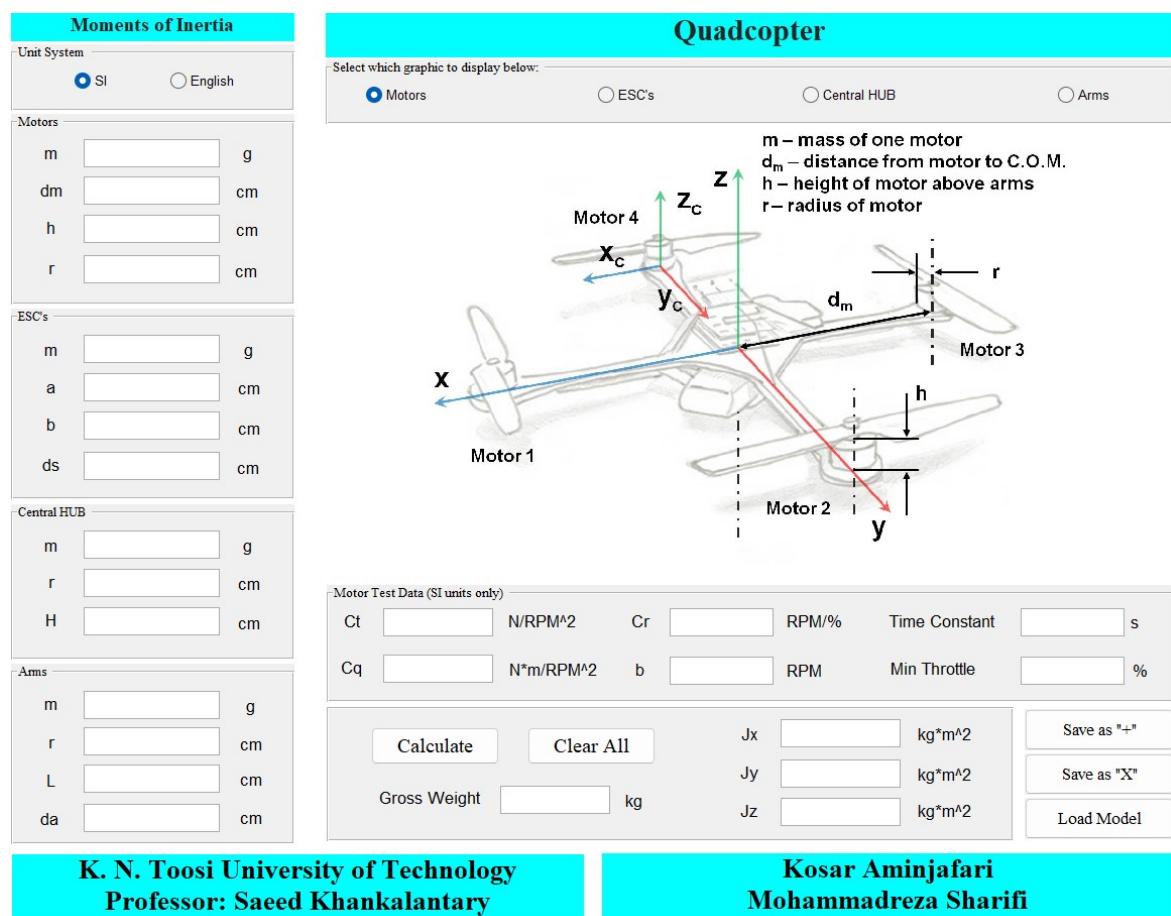
مسیر اول، «برخاست و حرکت دایره‌ای»، به عنوان یک آزمون بنیادین عمل می‌کند. چالش اصلی در اینجا، ارزیابی قابلیت کنترل کننده در پایدارسازی اولیه و رهگیری یک مسیر هموار و پیوسته است. برخاست عمودی، توانایی حلقه کنترل ارتفاع در رسیدن به نقطه تنظیم مطلوب و غلبه بر نیروی گرانش را می‌سنجد. پس از آن، حرکت روی یک دایره با شعاع ثابت، عملکرد حلقه کنترل موقعیت را در تبدیل یک منحنی ساده به فرمان‌های رول و پیچ مناسب و پایدار ارزیابی می‌کند. این مسیر، محک اولیه برای سنجش کارایی پایه‌ی کنترل کننده در شرایط عادی و قبل پیش‌بینی است.

مسیر دوم، «حرکت هشتی»، سطح چالش را به شکل قابل توجهی افزایش می‌دهد. این مسیر برای آزمون مانور پذیری و عملکرد دینامیکی کنترل کننده در شرایط تهاجمی تر طراحی شده است. برخلاف دایره، مسیر هشتی دارای انحنایان متغیر و نقاط عطفی است که کنترل کننده را مجبور به اجرای سریع و مداوم تغییرات در زوایای رول و پیچ می‌کند. این سناریو به طور خاص، توانایی کنترل کننده در مدیریت کوپلینگ شدید بین دینامیک دورانی و انتقالی را به چالش می‌کشد؛ جایی که تغییرات سریع در وضعیت (attitude) برای دنبال کردن مسیر، مستقیماً بر پایداری موقعیت تأثیر می‌گذارد. این آزمون نشان می‌دهد که آیا کنترل کننده می‌تواند در سرعت‌های بالاتر و در مانورهای پیچیده، پایداری و دقیقت خود را حفظ کند یا

خیر.

در نهایت، مسیر سوم، «مسیر الماسی»، برای ارزیابی پاسخ گذرا (transient response) و دقت کنترل کننده در برابر تغییرات ناگهانی طراحی شده است. هر گوشه از مسیر لوزی، معادل یک ورودی پله در سرعت مطلوب است. این تغییرات ناگهانی، رفتار کنترل کننده را در مواجهه با خطاها بزرگ و آنی آشکار می‌سازد. هدف از این آزمون، سنجش معیارهایی مانند فراجهش (overshoot)، زمان نشت (Settling time) و توانایی سیستم در خنثی‌سازی نوسانات پس از یک تغییر سریع در مسیر مرجع است. این سناریو به خوبی نشان می‌دهد که کنترل کننده چقدر در مدیریت تغییرات ناگهانی فرمان، مقاوم و کارآمد است.

۳.۱.۴ بارگذاری یا ساخت مدل کوادروتور



شکل ۴.۶: رابط گرافیک کاربری برای وارد کردن ابعاد کوادروتور

مدلسازی دینامیکی یک سیستم، اولین و یکی از اساسی‌ترین گام‌ها در فرآیند طراحی و تحلیل سیستم‌های کنترل است. نرم‌افزار ارائه شده، یک واسط گرافیکی به منظور تجمعی و پردازش پارامترهای فیزیکی یک کوادکوپتر جهت استخراج مدل ریاضی آن، مشخصاً جرم کل و ماتریس اینرسی، طراحی شده است. این مدل ریاضی، پایه‌ای برای شبیه‌سازی رفتار پروازی سیستم، طراحی کنترل کننده‌های خطی و غیرخطی، و پیش‌بینی پاسخ پهپاد به فرامین کنترلی پیش از آزمون‌های پروازی واقعی است. در ادامه، تحلیل جامعی از پارامترهای وارد شده برای یک کوادکوپتر دست‌ساز ارائه می‌گردد. این چهار نما اجزای مختلف رابط GUI _Modeling را برای پارامترسازی هندسی جرمی کوادروتور نشان می‌دهند؛ کاربر با جایه‌جایی میان گزینه‌های Arms، Central HUB، ESC's، Motors، همان اسکلت بدنه را می‌بیند اما برچسب‌گذاری

هندسی و فیلدهای ورودی متناسب با جزء منتخب تغییر می‌کند تا ورود داده‌ها دقیق و منطبق با فیزیک سازه انجام شود. پس از تکمیل هر نما و فشردن دکمه Calculate، ممان‌های اینرسی کل وسیله بر اساس قضیه محورهای موازی محاسبه و در خروجی‌های J_x , J_y و J_z نمایش داده می‌شود و وزن کل نیز در Gross Weight گزارش می‌گردد؛ در صورت نیاز پیکربندی نهایی برای آرایش‌های "+" و "X" ذخیره می‌شود تا در ماتریس اختلاط و مدل دینامیکی به کار رود.

در نمای Motors هندسه و جرم هر موتور ملخ نسبت به مرکز جرم بدن معرفی می‌شود و نمادگذاری‌های m , d_m , r و h به ترتیب جرم یک موتور، فاصله افقی از مرکز جرم، اختلاف ارتفاع موتور نسبت به بازوها و شعاع مؤثر مجموعه ملخ را مشخص می‌کنند. این اطلاعات تعیین‌کننده سهم موتور در اینرسی حول محورهای است؛ اگر $J_{m,c}$ ممان ذاتی مجموعه موتور ملخ حول مرکز خودش باشد، سهم آن در حول مبدأ بدن طبق قضیه محورهای موازی برابر است با

$$J_m = J_{m,c} + m(d_m^2 + h^2),$$

و برای چهار موتور، جمع این مقادیر به ممان کل افزوده می‌شود. به علاوه، تغییر علامت h یا جابه‌جایی d_m می‌تواند کوپل‌های ناخواسته بین محورها ایجاد کند و بهمین دلیل این نما با محورهای بدن (x, y, z) و بردارهای (x_c, y_c, z_c) کاربر را در تشخیص جهت‌گیری‌های صحیح یاری می‌دهد.

در نمای ESC's فرض می‌شود هر ESC در امتداد یکی از بازوها نصب شده و با پارامترهای a , b , m و d_s توصیف می‌گردد که به ترتیب جرم، عرض، طول و فاصله افقی ESC از مرکز جرم بدن هستند. این جزء معمولاً یک منشور مستطیلی نازک مدل می‌شود و ممان ذاتی آن حول مرکز خودش با تقریب منشور نازک محاسبه شده و سپس با جابه‌جایی d_s به مبدأ بدن منتقل می‌شود؛ بنابراین اگر $J_{e,c}$ ممان ذاتی باشد، سهم هر ESC برابر است با

$$J_e = J_{e,c} + m d_s^2,$$

که نشان می‌دهد حتی جرم‌های کوچک الکترونیک، وقتی دور از مرکز جرم نصب شوند، می‌توانند اثر معنی‌داری بر J داشته باشند. نمایش گرافیکی این نما محل تقریبی نصب ESC و جهت طول عرض آن را نسبت به محورها روشن می‌کند تا ورود a و b با چیدمان واقعی سازگار باشد.

در نمای Central HUB بخش میانی بدن به صورت استوانه‌ای با جرم m ، شعاع r و ارتفاع کل H مدل می‌شود و چون این جزء دقیقاً پیرامون مرکز جرم قرار دارد، ممان‌های ذاتی استوانه نقش پررنگی دارند. برای یک استوانه توپر، ممان‌ها حول مرکز جرم طبق روابط استاندارد برابرند با

$$J_z = \frac{1}{2} m r^2, \quad J_x = J_y = \frac{1}{12} m (3r^2 + H^2),$$

و چون مرکز این قطعه بر مرکز جرم منطبق است، افزودن جملات انتقالی قضیه محورهای موازی لازم نیست. این نما به صورت شماتیک محل توپی مرکزی، اندازه شعاع و ارتفاع آن را روی بدن نمایش می‌دهد تا انتخاب r و H با هندسه واقعی هم خوان بماند و اثر تغییر ابعاد مرکزی بر توزیع جرم و سختی دورانی به خوبی دیده شود.

در نمای Arms هر بازو به صورت میله‌ای با جرم m ، شعاع مؤثر r ، طول کل L و آفست d_a نسبت به مرکز جرم معرفی می‌شود؛ d_a جابه‌جایی نقطه میانی بازو از مرکز جرم را بیان می‌کند. برای مدل‌سازی بازو به عنوان میله نازک، ممان حول محوری عمود بر طول و گذرنده از مرکز خود بازو تقریباً

$$J_{\perp} \approx \frac{1}{12} m L^2$$

و حول محور طولی با تقریب پوسته نازک

$$J_{\parallel} \approx m r^2$$

است. با انتقال به مبدأ بدنه، سهم هر بازو حول محورهای لخت به صورت

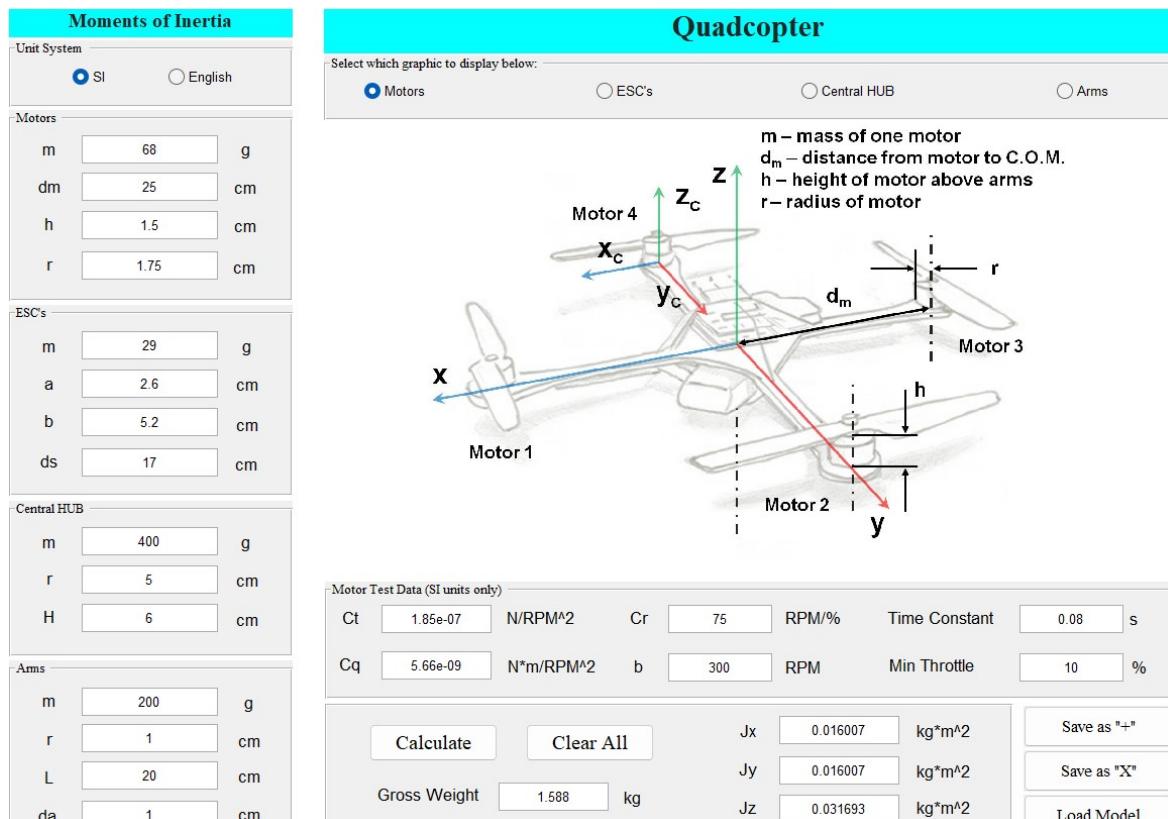
$$J_{\text{arm}} = J_{\perp} + m d_a^2$$

به ممان کل اضافه می‌شود و انتخاب آرایش "+" یا "X" روی همنهی این سهم‌ها و تقارن J_x و J_y اثر مستقیم دارد. نمایش گرافیکی این نما، طول L ، شعاع مؤثر r و جایگاه d_a را نسبت به محورها مشخص می‌کند تا سازگاری پارامترهای واردشده با هندسه واقعی شاسی به صورت شهودی کنترل شود.

در مجموع، این چهار نما زنجیرهای منسجم برای توصیف اجزای اصلی کوادروتور فراهم می‌کنند تا پس از تبدیل یک‌ها و اعمال قضیه محورهای موازی، ممان‌های اینرسی کل به صورت

$$\mathbf{J} = \text{diag}(J_x, J_y, J_z) = \sum_{i \in \{\text{motors, ESCs, hub, arms}\}} (J_{i,c} + m_i d_i^2)$$

دقیقاً محاسبه و به مدل Simulink تزریق شود؛ به این ترتیب تغییر هر پارامتر در هر نما اثر خود را بلاfacله در \mathbf{J} و وزن کل نشان می‌دهد و مسیر طراحی مکانیکی و کنترل به طور یکپارچه پیش می‌رود.





شکل ۸.۴: کوادروتور استفاده شده برای تست

بخش اول: خصوصیات جرمی و هندسی

این بخش به تعریف مشخصات فیزیکی اجزای اصلی سازه پهپاد اختصاص دارد که در نهایت، مرکز جرم و ماتریس اینرسی کل سیستم را تعیین می‌کنند.

برای **موتورها** (*Motors*) و **اسپید کنترل کنندها** (*SCs*), مقادیر جرمی به ترتیب ۶۸ و ۲۹ گرم به همراه ابعاد فیزیکی و فواصل نصب آن‌ها از مرکز جرم (ds و dm) وارد شده است. این مقادیر با مشخصات فنی قطعات منتخب (موتور $AX - 2810Q$ و اسپید کنترل کننده $Emax30A$) مطابقت دارند. جرم این قطعات، به دلیل قرارگیری در انتهای بازوها و با فاصله قابل توجه از مرکز، تأثیر بسزایی بر ممان‌های اینرسی سیستم، به ویژه حول محور عمودی (J_z)، خواهد داشت.

بدنه مرکزی (*Central HUB*) به عنوان هسته اصلی سازه، با جرمی معادل ۴۰۰ گرم و شعاع ۵ سانتی‌متر مدل‌سازی شده است. این جرم، علاوه بر صفات اصلی بدنه، می‌تواند وزن مجموعه اویونیک شامل کنترل کننده پرواز، واحد *GPS* گیرنده رادیویی و سایر حسگرهای را نیز نمایندگی کند که از نظر مهندسی، یک تقریب قابل قبول است.

بخش **بازوهای اهرم** (*Arms*) که نقش اهرمهای انتقال نیرو و گشتاور را ایفا می‌کنند، با مقادیر اصلاح شده و دقیقی مدل‌سازی شده‌اند. جرم ۲۰۰ گرم برای هر بازو با توجه به جنس آلومینیومی و پایه‌های اتصال موتور، مقداری واقع‌بینانه است. همچنین، طول بازو (L) به مقدار ۲۰ سانتی‌متر با توجه به فاصله موتور تا مرکز (dm) برابر ۲۵ سانتی‌متر) و شعاع بدنه مرکزی (۵ سانتی‌متر)، اکنون از نظر هندسی کاملاً صحیح و سازگار است. صحت این پارامترها برای محاسبه دقیق ممان اینرسی ضروری است.

بخش دوم: خصوصیات آیرودینامیکی و دینامیکی موتور

این بخش رفتار هر واحد موتور-ملخ را به عنوان یک عملگر (*Actuator*) توصیف می‌کند و برای شبیه‌سازی دقیق نیروهای تولیدی، حیاتی است.

پارامترهای C_t (ضریب تراست) و C_q (ضریب گشتاور)، اصلی‌ترین مشخصه‌های آیرودینامیکی سیستم پیش‌ران هستند. این ضرایب، رابطه بین سرعت زاویه‌ای موتور (ω) و نیروی تراست (T) و گشتاور آیرودینامیکی (Q) را از طریق روابط غیرخطی زیر مدل‌سازی می‌کنند:

$$T = C_t \cdot \omega^2$$

$$Q = C_q \cdot \omega^2$$

مقادیر وارد شده، که معمولاً از آزمون‌های عملی روی تراست استند به دست می‌آیند، امکان ترجمه دقیق سرعت موتور به نیروهای وارد بر بدنه پهپاد را فراهم می‌سازند.

پارامترهای Cr و b یک مدل خطی مرتبه اول را برای رابطه بین سیگنال ورودی از کنترل کننده (درصد تراطل) و سرعت خروجی موتور (*RPM*) ارائه می‌دهند. این تقریب خطی، پیچیدگی‌های دینامیکی درایور *ESC* و موتور را ساده‌سازی کرده و طراحی کنترل کننده را تسهیل می‌بخشد. **Time Constant** (ثابت زمانی)، پارامتری دینامیکی است که سرعت پاسخ موتور به تغییرات در فرمان ورودی را مشخص می‌کند. این مقدار برابر با زمانی است که پاسخ پله موتور به حدود ۶۳.۲ درصد مقدار نهایی خود می‌رسد و مقادیر کوچکتر، نشان‌دهنده پاسخ سریع‌تر و قابلیت مانورپذیری بالاتر پهپاد است. در نهایت، **Min Throttle** (حداقل تراطل) یک مقدار آستانه عملی برای جلوگیری از خاموش شدن موتورها در حین پرواز است.

بخش سوم: خروجی‌های محاسباتی و نتیجه‌گیری نهایی

خروجی نهایی این مدل‌سازی، **Gross Weight** (جرم کل) و ممان‌های اینرسی (J_x , J_y , J_z) است. ممان اینرسی، مقاومت سیستم در برابر شتاب‌گیری زاویه‌ای را توصیف می‌کند. همانطور که در نتایج دیده می‌شود، به دلیل تقارن ساختاری پهپاد، ممان اینرسی حول محورهای *Pitch* و *Roll* کاملاً برابر است ($J_x = J_y$). مقدار بزرگتر برای J_z نیز صحیح و قابل انتظار است، زیرا بخش قابل توجهی از جرم سیستم (موتورها و بازوها) در فاصله شعاعی زیادی از محور عمودی Z توزیع شده است.

در مجموع، مدل ارائه شده با پارامترهای اصلاح شده، یک تقریب دقیق و معتبر از پهپاد واقعی است. این مدل، پایه‌ای قابل اعتماد برای مراحل بعدی پروژه، یعنی طراحی و تنظیم دقیق کنترل‌کننده‌های پرواز در محیط شبیه‌سازی، فراهم می‌آورد. توصیه نهایی برای تکمیل مدل، افزودن جرم باتری به جرم بدنه مرکزی است تا وزن کل مدل با وزن پروازی واقعی (All-Up Weight) پهپاد تطبیق کامل یابد.

۴.۱.۴ بارگذاری شرایط اولیه کوادراتور

در شبیه‌سازی این پروژه، مرحله مقداردهی اولیه یک گام حیاتی برای تعریف نقطه شروع محاسبات و تضمین تکرارپذیری آزمون‌ها است که توسط بلوک *LOAD* در مدل سیمولینک مدیریت می‌شود. در این فرآیند، شرایط اولیه کوادراتور از یک فایل داده با نام *IC-OnGround-MotorsOff.mat* بارگذاری می‌گردد.

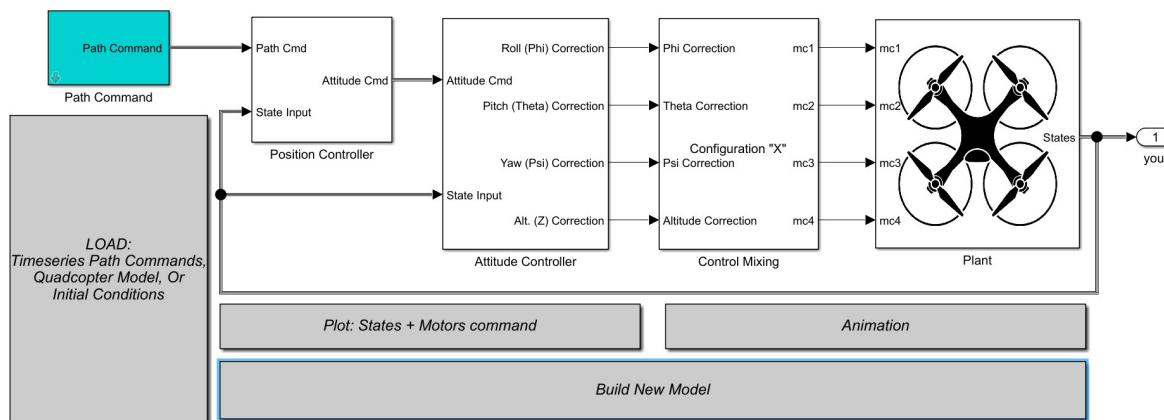
این فایل حاوی یک ساختار (*struct*) در متلب است که وضعیت کامل پرنده را در لحظه صفر از طریق ۱۶ فیلد مجزا تعریف می‌کند. این فیلدها شامل موقعیت در سه محور (X , Y , Z), وضعیت زاویه‌ای (*Phi*, *Theta*, *Psi*), سرعت‌های خطی و زاویه‌ای در چارچوب بدنه (*U*, *V*, *W*, *P*, *Q*, *R*) و سرعت اولیه چهار موتور (*w1* تا *w4*) می‌باشند. نام‌گذاری این فایل به وضوح سناریوی پیش‌فرض را توصیف می‌کند: شروع شبیه‌سازی از یک حالت سکون کامل در مبدأ مختصات، با وضعیتی کاملاً افقی و موتورهای خاموش، که در آن تمامی ۱۶ مقدار اولیه صفر تنظیم شده‌اند. این داده‌ها به صورت مستقیم برای مقداردهی اولیه به انتگرال‌گیرهای مدل دینامیکی (*Plant*) به کار رفته و نقطه شروع دقیق و یکسانی را برای تمامی آزمون‌ها تضمین می‌کنند.

Field ▲	Value
P	0
Q	0
R	0
Phi	0
The	0
Psi	0
U	0
V	0
W	0
X	0
Y	0
Z	0
w1	0
w2	0
w3	0
w4	0

شکل ۹.۴: شرایط اولیه کوادراتور

برای تغییر دادن هر کدام از شرایط اولیه به سادگی می‌توان روی پارامتر مورد نظر double-click کرد و مقدار موردنظر را بازگذاری کرد.

۲.۴ طراحی کنترل کننده PID برای موقعیت



شکل ۱۰.۴: شماتیک کلی فایل شبیه‌سازی سیمولینک کنترل کننده موقعیت کوادروتور

این شکل نمای کلی فایل شبیه‌سازی کنترل کوادروتور در Simulink را نشان می‌دهد که در آن زنجیره تولید فرمان از «مرجع مسیر» تا «فرمان‌های چهار موتور» و نیز حلقه بازخورد حالت‌ها به صورت یکپارچه دیده می‌شود. در سمت چپ، ناحیه LOAD مسئول بارگذاری داده‌های اولیه است؛ این داده‌ها شامل فرمان‌های مسیر به صورت timeseries، پارامترهای مدل کوادروتور و شرایط اولیه حالت‌هاست تا شبیه‌سازی با مجموعه اطلاعات سازگار شروع شود. بلوک Path Command همان منبع مرجع مسیر است و بردار «فرمان مسیر» را تولید می‌کند که در ادامه به بلوک Position Controller وارد می‌شود. علاوه بر فرمان مسیر، «بازخورد حالت» را نیز از مسیر State Input دریافت می‌کند و با محاسبه خطاهای موقعیتی، آن را به «فرمان وضعیت» تبدیل می‌نماید؛ به عبارت دیگر، مرجع‌های مناسب برای رول، پیچ، یا و ارتفاع را می‌سازد تا حلقه درونی بتواند پایداری بدنه را تضمین کند. خروجی این بلوک با عنوان Attitude Cmd به Attitude Controller ارسال می‌شود و همزمان همان State Input نیز برای محاسبه خطاهای زاویه‌ای و ارتفاعی به این کنترل کننده داده می‌شود. این از مقایسه وضعیت مطلوب و وضعیت واقعی، چهار اصلاحه کنترلی تولید می‌کند که در شکل با عنوانین Alt. (Z) Correction، Yaw (Psi) Correction، Pitch (Theta) Correction و Roll (Phi) Correction مشخص شده‌اند؛ این اصلاحه‌ها معادل نیازهای نیروی تراست کل و گشتاورهای سه‌محوره بدنه هستند و ورودی‌های مرحله بعدی را تشکیل می‌دهند. در بلوک Control Mixing این چهار کمیت پیوسته به فرمان‌های مجزای هر موتور نگاشت می‌شود و خروجی‌های mc1 تا mc4 شکل می‌گیرند؛ این نگاشت معمولاً با یک ماتریس اختلاط انجام می‌شود به صورت

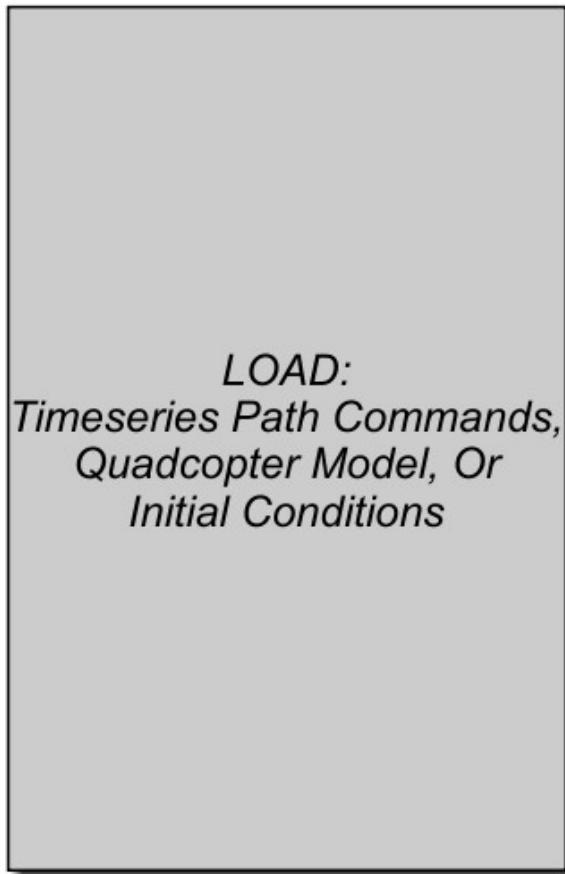
$$\begin{bmatrix} mc_1 \\ mc_2 \\ mc_3 \\ mc_4 \end{bmatrix} = M \begin{bmatrix} T \\ \tau_\phi \\ \tau_\theta \\ \tau_\psi \end{bmatrix},$$

که در آن T تراست کل و $\tau_\phi, \tau_\theta, \tau_\psi$ به ترتیب گشتاورهای رول، پیچ و یا و هستند و M با توجه به آرایش پره‌ها، بازوی موثر و ضرایب تراست/درگ تعیین می‌شود. فرمان‌های موتوری mc1 تا mc4 سپس به بلوک Plant اعمال می‌شوند که مدل دینامیکی شش درجه‌آزادی کوادروتور را پیاده‌سازی می‌کند؛ این مدل با انتگرال گیری از معادلات حرکت، بردار حالت را به روزرسانی کرده و در خروجی سیگنال States را تولید می‌کند که هم برای ثبت نتایج (yout) و هم برای بستن حلقه

بازخورد (State Input) استفاده می‌شود. برای تحلیل و ارائه نتایج، ناحیه Plot: States + Motors command به ترسیم پاسخ‌های زمانی حالتها و فرمان‌های موتوری اختصاص دارد و بخش Animation نیز یک نمایش سه‌بعدی از مسیر و وضعیت پرنده فراهم می‌کند تا درک شهودی از رفتار سیستم به دست آید؛ این تجسم‌ها بر دینامیک اثر ندارند و صرفاً مصرف کننده داده‌اند. در پایین شکل ناحیه Build New Model قرار دارد که به منزله فضای کمکی برای ساخت یا پیکربندی سفاربوهای جدید یا جایگزینی مدل‌ها به کار می‌رود. به طور خلاصه، جریان اطلاعات از چپ به راست ابتدا مرجع مسیر را به مرجع‌های وضعیت تبدیل می‌کند، سپس در حلقه درونی به گشتاورها و تراست بدلت می‌شود و پس از اختلاط، به فرمان‌های چهار موتور می‌رسد؛ همزمان، بردار حالت

$$\mathbf{x} = [x, y, z, \phi, \theta, \psi, u, v, w, p, q, r]^\top$$

در Plant محاسبه شده و از طریق مسیر بازخورد به کنترل کننده‌های بیرونی و درونی تغذیه می‌گردد تا یک سامانه حلقه‌بسته آبشاری با امکان پایش عددی و تجسم گرافیکی در بستر Simulink شکل گیرد.



شکل ۱۱.۴: بلوک بارگذاری مسیر، مدل کوادروتور و شرایط اولیه

این تصویر نمایانگر بلوک LOAD در فایل شبیه‌سازی است؛ ناحیه‌ای مقدماتی که پیش از آغاز حل عددی، تمامی داده‌های ورودی لازم را به محیط Simulink تزریق می‌کند و تضمین می‌نماید که زنجیره کنترلی با یک مجموعه منسجم از اطلاعات آغاز به کار کند. مأموریت اصلی این بخش سه گروه داده را در بر می‌گیرد: نخست «فرمان‌های مسیر» به صورت

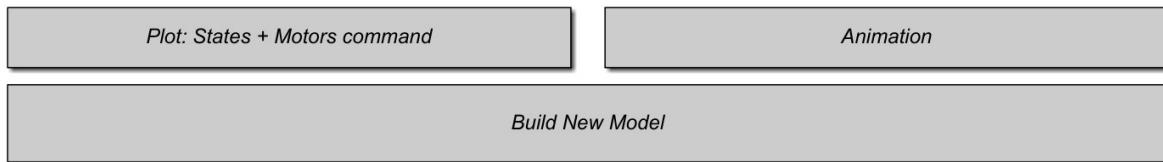
timeseries که معمولاً برداری از مراجع موقعیت و سرجهت است، مانند

$$\mathbf{r}_d(t) = [x_d(t), y_d(t), z_d(t), \psi_d(t)]^\top,$$

و از طریق بلوک‌های From Workspace به حلقه «کنترل موقعیت» اعمال می‌شود؛ برای سازگاری با حل‌گر لازم است بردار زمان یکنواخت و صعودی بوده و نوع داده یا به صورت شیء timeseries و یا ساختار استاندارد struct با فیلدهای time و signals تعریف شود تا بعد و نمونه‌زمان‌ها به درستی تفسیر گردد. دوم «پارامترهای مدل کوادراتور» که به‌طور معمول شامل جرم m ، ممان‌های اینرسی (I_{xx}, I_{yy}, I_{zz})، فاصله بازو L ، ضرایب تراست و درگ ملخ‌ها و ثابت گرانش g است و اغلب در قالب یک ساختار یکپارچه (model) در Base Workspace قرار می‌گیرد تا هم در بلوک Plant و هم در ماتریس اختلاط فرمان‌ها مورد استفاده واقع شود. سوم «شرایط اولیه» که وضعیت آغازین دستگاه را مشخص می‌کند، برای نمونه

$$\mathbf{x}(0) = [x_0, y_0, z_0, \phi_0, \theta_0, \psi_0, u_0, v_0, w_0, p_0, q_0, r_0]^\top,$$

و به صورت مستقیم در انتگرال‌گیرهای Plant و سازوکارهای ثبت خروجی به کار می‌رود. از منظر پیاده‌سازی، LOAD معمولاً به اسکریپت‌های m. mat. یا فایل‌های InitFcn متصل است و از طریق رویداد آغازین قرارداد چارچوب مختصات و نام‌گذاری دقیق متغیرها در این گام حیاتی است، زیرا بلوک‌های پایین‌دست مانند Position Controller و Attitude Controller می‌توانند مقدیر اتکا دارند. در نتیجه، بلوک LOAD نقش «درگاه داده» را ایفا کرده و امکان تعریف سناریوهای گوناگون شبیه‌سازی را تنها با تعویض فایل مسیر، جدول پارامترها یا بردار شرایط اولیه فراهم می‌سازد؛ به این ترتیب بدون تغییر در شماتیک اصلی، تکرار پذیری آزمایش‌ها، کنترل نسخه و جابه‌جایی سریع میان سناریوهای مختلف ممکن می‌شود.



شکل ۱۲.۴: بلوک اینیمیشن، نمودارها و ساخت مدل جدید کوادراتور

در این تصویر سه ناحیه کمکی از فایل شبیه‌سازی نمایش داده شده است که کار دیده‌بانی، تجسم و توسعه سناریوهای جدید را بر عهده دارند. ناحیه چپ-بالا با عنوان Plot: States + Motors command مژول ترسیم است که روی داده‌های ثبت‌شده شبیه‌سازی عمل می‌کند؛ در اینجا سیگنال‌های حالت و فرمان‌های موتوری که به کمک بلوک‌های To Workspace یا ثبت خودکار Simulink به صورت timeseries در فضای کاری ذخیره می‌شوند، خوانده شده و نمودارهای زمانی تولید می‌گردد. معمولاً بردار حالت به صورت

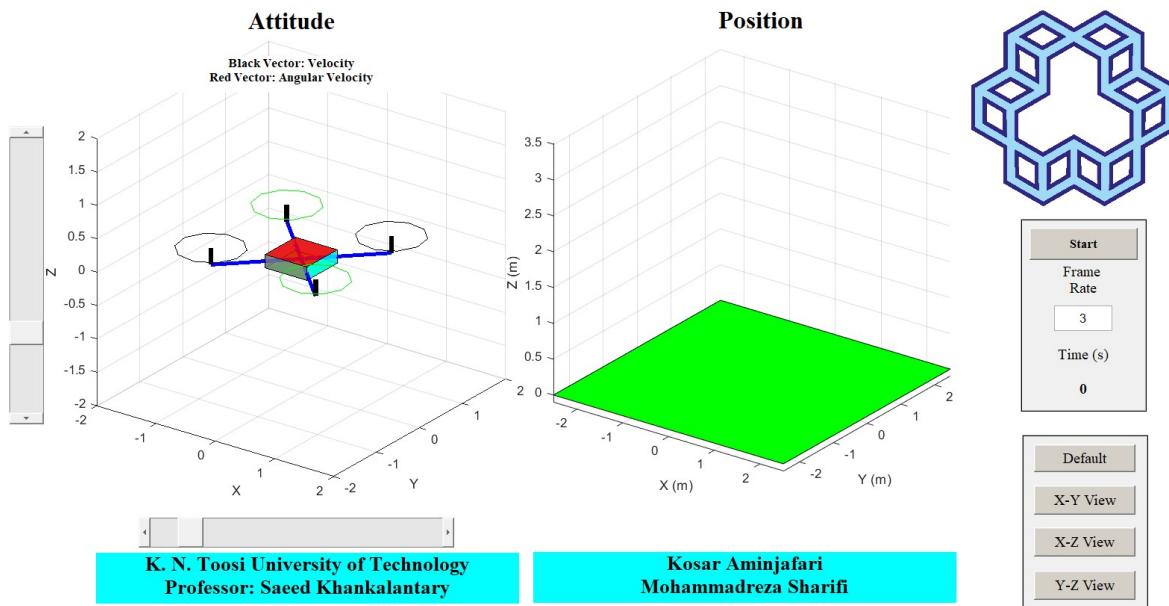
$$\mathbf{x}(t) = [x(t), y(t), z(t), \phi(t), \theta(t), \psi(t), u(t), v(t), w(t), p(t), q(t), r(t)]^\top$$

$$\mathbf{u}(t) = [mc_1(t), mc_2(t), mc_3(t), mc_4(t)]^\top$$

ثبت می‌شود تا بتوان کیفیت پاسخ گذرا، اشباع محرک‌ها و توازن بار بین موتورها را ارزیابی کرد. ناحیه راست-بالا با عنوان

Animation وظیفه تجسم سه بعدی حرکت کوادراتور را دارد و با دریافت موقعیت $\mathbf{p}(t) = [x(t), y(t), z(t)]^\top$ و زوایا (ψ, ϕ, θ) پیکربندی بدن را در دستگاه لخت به روزرسانی می‌کند. مبنای این نمایش، تبدیل چارچوب بدن به لخت از طریق ماتریس چرخش $\mathbf{R}_{IB}(\phi, \theta, \psi)$ است تا جهت‌گیری لحظه‌ای پرنده درست اعمال شود؛ مسیر طی شده، بردارهای سرعت و حتی فرمان‌های موتوری می‌توانند به صورت overlay نمایش داده شوند. این بخش برای دینامیک تأثیری ندارد و صرفاً مصرف کننده داده است، اما برای راستی آزمایی شهودی و کشف خطاهای علامتی (مانند جابه‌جایی محورها یا ترتیب زوایا) بسیار مؤثر است. در نسخه‌های جدید MATLAB ممکن است برای روانی نمایش، کاهش نرخ رسم یا استفاده از video logging توصیه شود.

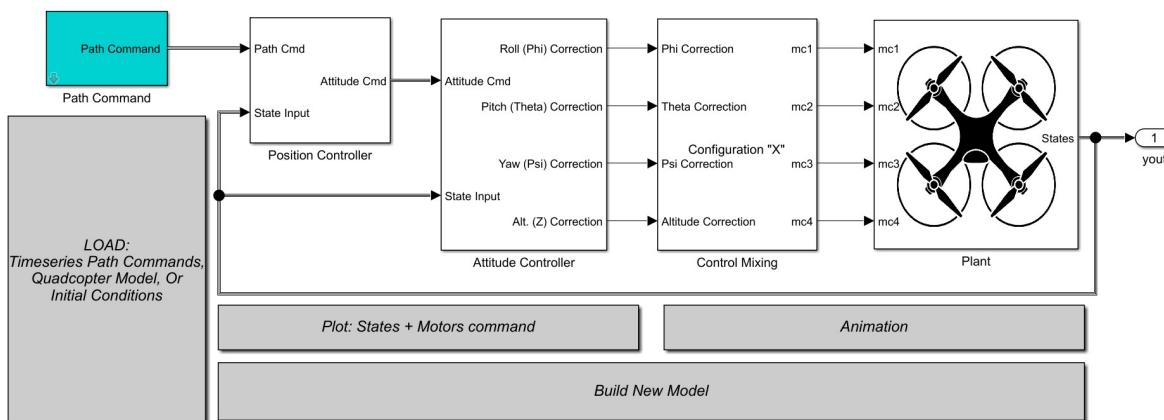
نوار پایینی با عنوان Build New Model فضای کار برای توسعه سناریوهای جدید مدل است؛ در این بخش کاربر می‌تواند با اجرای اسکریپت‌های پارامتری، ایجاد Variant Subsystem برای کنترل کننده‌های مختلف، یا کپی برداری از اسکلت مدل موجود، یک پیکربندی تازه بسازد و آن را به زنجیره اصلی متصل کند. معمولاً پارامترهای فیزیکی، ماتریس اختلاط، تنظیمات حل‌گر و نرخ نمونه‌برداری در همین ناحیه تنظیم شده و با callbacks راهاندازی (مانند InitFcn) به مدل تزریق می‌شوند تا تکرار پذیری تضمین شود. به این ترتیب سه بخش Build New Model، Animation Plot و حلقه کامل «داده‌برداری → تجسم → توسعه» را کنار هسته دینامیک و کنترل فراهم می‌کنند و امکان مستندسازی دقیق نتایج و بازآفرینی سریع آزمایش‌ها را مهیا می‌سازند.



شکل ۱۳.۴: رابط گرافیک کاربری برای نمایش عملکرد کوادراتور

این نما خروجی ماثول Animation را در یک محیط تعاملی نشان می‌دهد که در آن رفتار کوادراتور همزمان از دو زاویه مکمل «وضعیت» و «مکان» قابل مشاهده است. پنل سمت چپ با عنوان Attitude پیکربندی لحظه‌ای بدن را در دستگاه مختصات سه بعدی نمایش می‌دهد؛ حلقه‌های سبز نمایانگر صفحه ملخ‌ها و بدن رنگی، شاسی پرنده را صورت‌بندی می‌کند. بر فراز بدن دو بردار مرجع ترسیم شده که مطابق راهنمای بالای شکل، بردار سیاه سرعت خطی و بردار قرمز سرعت زاویه‌ای را نمایش می‌دهند؛ بنابراین در هر گام زمانی می‌توان بزرگی و جهت $v(t)$ و $\omega(t)$ را نسبت به محورهای بدن ارزیابی کرد. محورهای X, Y و Z با برچسب‌گذاری شفاف در صحنه حضور دارند تا زاویه‌های چرخش و تغییر راستا به صورت بصری و بی‌ابهام قابل پیگیری باشد؛ اسکرول‌بارهای کنار و پایین پنجره نیز برای بزرگ‌نمایی، جابه‌جایی و تنظیم سطح دید در حین پخش به کار می‌روند تا کاربر بدون توقف شبیه‌سازی بتواند روی جزئیات مورد نظر تمرکز کند.

پنل سمت راست با عنوان Position مسیر سه بعدی حرکت را بر فراز صفحه زمین ترسیم می‌کند؛ نشانگرهای دایرهای آبی، نمونه‌های زمانی مسیر را مشخص می‌سازند و پیوستگی آن‌ها تغییرات لحظه‌ای موقعیت $(x(t), y(t), z(t))$ را آشکار می‌کند. صفحه سبز رنگ کف صحنه مرز ground plane را می‌نمایاند و درک فاصله عمودی را ساده می‌کند، بهویژه در مانورهایی که فراز altitude تغییرات چشمگیر دارد. در ستون کنترلی سمت راست، دکمه Stop اجرای انیمیشن را متوقف/ازسر می‌گیرد، باکس Frame Rate نرخ فریم را بر حسب فریم بر ثانیه تنظیم می‌کند تا تعادل میان روانی نمایش و بار محاسباتی برقرار شود، و ساختار Time (s) زمان جاری شبیه‌سازی را به طور زنده گزارش می‌کند؛ این سه مولفه امکان X-Y View، Default همگام‌سازی دقیق با نمودارهای زمانی و تحلیل‌های کمی را فراهم می‌آورند. مجموعه دکمه‌های X-Z View و Y-Z View نیز نماهای از پیش‌تعریف شده دوربین را فراهم می‌کنند تا با یک کلیک بتوان مسیر را در صفحه افقی، نماهای جانبی عمودی یا دید پیش‌فرض مرور کرد؛ تغییر دیدگاه بهویژه برای ارزیابی فراجهش ارتفاعی، انحراف جانبی و کیفیت رهگیری مسیر در هر صفحه بسیار سودمند است.



شکل ۱۴.۴: بلوک کنترل موقعیت در نرم افزار سیمولینک

این نمودار، ساختار حلقه بیرونی «کنترل موقعیت» را نشان می‌دهد که چگونه فرمان‌های X_cmd Path Cmd شامل Alt_Cmd، Y_Cmd، Psi_Cmd و Alt_Cmd به همراه بازخورد State Input از جمله X، Y و Psi و سرعت‌های بدنه U و V وارد زنجیره تولید «فرمان وضعیت» می‌شوند. هسته این زنجیره، بلوک سبزرنگ Error Rotations است که به جای کار کردن مستقیم با خطای مکان در چارچوب لخت، ابتدا خطای افقی را به چارچوب بدنه می‌چرخاند تا کنترل‌پذیر توسط وضعیت (رول/پیچ) شود. بنابراین با استفاده از زاویه یا ψ داریم:

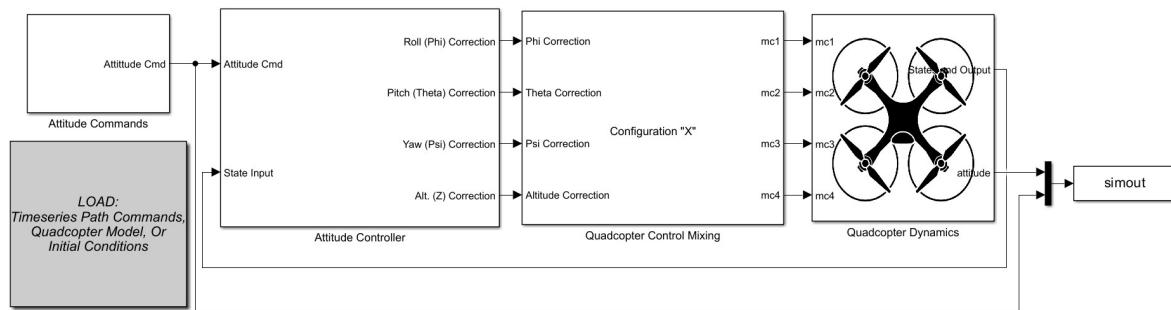
$$\begin{bmatrix} e_{x_b} \\ e_{y_b} \end{bmatrix} = \begin{bmatrix} \cos \psi & \sin \psi \\ -\sin \psi & \cos \psi \end{bmatrix} \begin{bmatrix} x_{cmd} - x \\ y_{cmd} - y \end{bmatrix}.$$

بر این مبنای یک «سرعت مطلوب در چارچوب بدنه» ساخته می‌شود و آنگاه کنترل کننده‌های Theta Command Control و Phi Command Control PD روی خطای سرعت بدنه عمل می‌کنند و به ترتیب θ_{cmd} و ϕ_{cmd} را می‌سازند. فرم معمول این دو نگاشت (با اشعاع $\pm 8^\circ$) به شکل زیر است:

$$\theta_{cmd} = \text{sat}_{\pm 8^\circ} \left(K_{p,\theta} (v_{x_b}^d - v_{x_b}) + K_{d,\theta} \frac{d}{dt} (v_{x_b}^d - v_{x_b}) \right),$$

$$\phi_{cmd} = \text{sat}_{\pm 8^\circ} \left(K_{p,\phi} (v_{yb}^d - v_{yb}) + K_{d,\phi} \frac{d}{dt} (v_{yb}^d - v_{yb}) \right).$$

در این جا v_{xb} و v_{yb} همان مولفه‌های سرعت بدن‌اند که در مدل با U و V اندازه‌گیری می‌شوند و v_{yb}^d خروجی‌های Alt_Cmd و Psi_Cmd به همراه Error Rotations مسقیم θ و ϕ بدهمراه عبور مستقیم Altitude Cmd و Goto Quadcopter Dynamics اشاره‌گر Attitude Cmd در گاه جمع می‌شوند و از طریق Block به حلقه درونی و مدل دینامیکی ارسال می‌گردند. ایده کلیدی این معماری آن است که چون متغیر واقعاً کنترل پذیر با وضعیت، «سرعت در چارچوب لخت» است، ابتدا خطای مکان به سرعت مطلوب در چارچوب بدن نگاشته می‌شود و سپس همین سرعت مطلوب به فرمان وضعیت تبدیل می‌گردد؛ بدین ترتیب مسیر attitude → body-velocity → position به صورت منسجم و قابل تنظیم پیاده می‌شود.



شکل ۱۵.۴: بلوک کنترل وضعیت در نرم افزار سیمولینک

این دیاگرام معماری حلقه درونی «کنترل وضعیت» را نشان می‌دهد که در آن چهار تنظیم‌گر مستقل برای رول، پیچ، یاو و ارتفاع بر مبنای قانون‌های PID/PD پیاده‌سازی شده‌اند و ورودی‌های خود را از دو گذرگاه State و Attitude Cmd می‌گیرند. گذرگاه نخست حامل مراجع ψ_{cmd} , θ_{cmd} , ϕ_{cmd} و z_{cmd} است که از حلقه بیرونی یا اپراتور می‌آیند؛ گذرگاه دوم نیز حالت‌های اندازه‌گیری شده را در اختیار می‌گذارد، شامل زوایا ϕ , θ , ψ و نرخ‌های بدن p , q , r (که در مدل با برچسب‌های P, Q, R نشان داده شده‌اند) و همچنین ارتفاع z . در مسیر رول (بلوک آبی) خطای زاویه به صورت $K_i = 0.5$, $K_p = 2$ تشکیل می‌شود و با یک قانون PID با ضرایب نمونه درج شده روی بلوک (1) به «تصحیح رول» نگاشت می‌شود؛ ترم مشتق عملاً با فیدبک نرخ p نقش میراکنندگی را ایفا می‌کند تا پاسخ سریع بدون فراجهش زیاد حاصل شود. صورت‌بندی متداول فرمان رول چنین است:

$$u_\phi = K_{p,\phi} (\phi_{cmd} - \phi) - K_{d,\phi} p + K_{i,\phi} \int (\phi_{cmd} - \phi) dt,$$

که پس از اشباع انتگرال (آن‌تی‌ویندایپ) با کران‌های ± 1 به خروجی «Phi Correction» فرستاده می‌شود. مسیر پیچ (بلوک قرمز) به طور مشابه با $e_\theta = \theta_{cmd} - \theta$ و همان ضرایب نمونه (2) عمل می‌کند و با بهره‌گیری از نرخ q ترم مشتق مؤثر را می‌سازد؛ خروجی این کanal «Theta Correction» است:

$$u_\theta = K_{p,\theta} (\theta_{cmd} - \theta) - K_{d,\theta} q + K_{i,\theta} \int (\theta_{cmd} - \theta) dt.$$

در مسیر یاو (بلوک سبز) به سبب کندی دینامیک و حساسیت جهت‌گیری، بهره مشتق بزرگ‌تری انتخاب شده است ($K_p = 2$) تا تغییرات نرخ r به خوبی میرا شود و انحراف سرفرامن به سرعت خاموش گردد؛ خروجی این کanal $K_d = 5$, $K_i = 1$

«Psi Correction» خواهد بود:

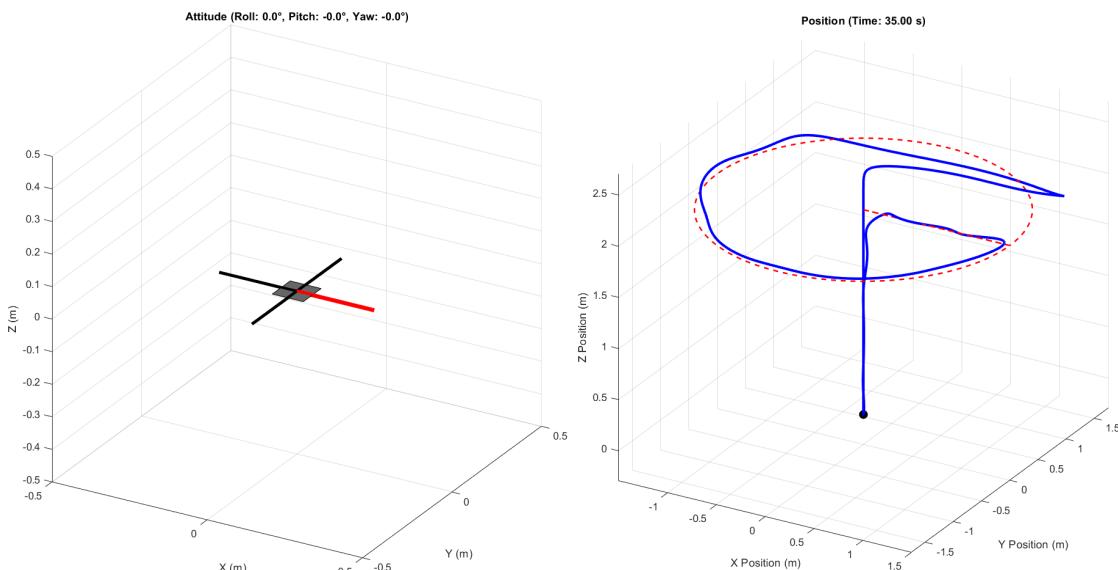
$$u_\psi = K_{p,\psi} (\psi_{cmd} - \psi) - K_{d,\psi} r + K_{i,\psi} \int (\psi_{cmd} - \psi) dt.$$

کanal ارتفاع (بلوک بنفس) با مرجع z_{cmd} و حالت \dot{z} کار می‌کند و علاوه بر تنظیم‌گر PID با ضرایب نمونه $K_p = 2, K_d = 3.3, K_i = 1.1$ ، یک جزء پیشخور ثابت نیز دارد که در بلوک با عنوان Gravity Offset = 57.1762 نمایش داده شده است؛ این بایاس معادل نیروی ادریچه شناوری نامی است که وزن را خنثی می‌کند و فرمان تنظیم‌گر صرفاً «تصحیح حول شناوری» را تولید می‌نماید. فرم رایج فرمان ارتفاع به شکل زیر نوشته می‌شود:

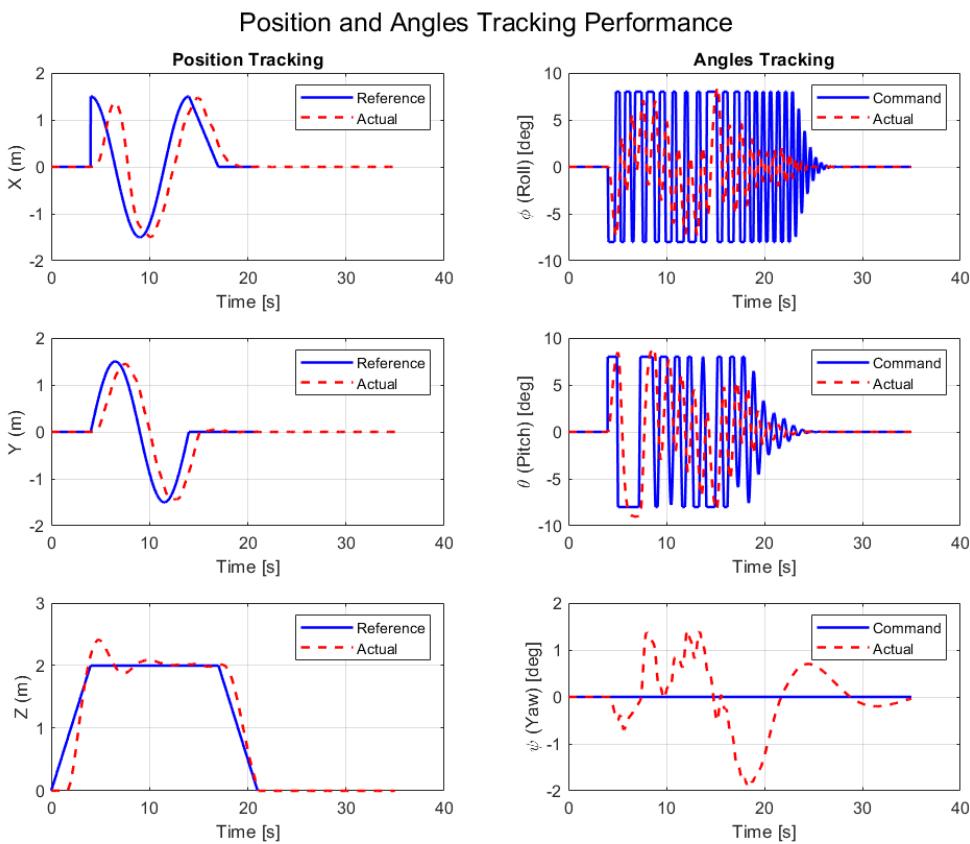
$$u_z = u_{hover} + K_{p,z} (z_{cmd} - z) + K_{i,z} \int (z_{cmd} - z) dt - K_{d,z} \dot{z},$$

که در آن u_{hover} همان بایاس گرانشی و \dot{z} می‌تواند از فیلتر مشتق یا حسگر ارتفاع-سرعت محاسبه شود. در هر چهار مسیر، «Integral Saturation: on» با کران‌های ± 1 فعال است تا از پدیده بادشگی انتگرال در اشباع‌ها جلوگیری شود و در «Roll (Phi) Correction» نتیجه بازیابی نرم پس از اشباع تضمین گردد. خروجی‌های چهار تنظیم‌گر به ترتیب با برچسب‌های «Alt. (Z) Correction»، «Yaw (Psi) Correction»، «Pitch (Theta) Correction» و «Roll (Phi) Correction» از بلوک‌ها خارج شده و در مرحله بعدی اختلاط فرامین به گشتاورهای بدنه و تراست کل نگاشت می‌شوند؛ این معماری با جداسازی محورها، تنظیم و تیونینگ را ساده می‌کند، امکان زمان‌بندی نمونه‌برداری ناهم‌نرخ را فراهم می‌سازد و به دلیل استفاده از نرخ‌های بدنه r, p, q به عنوان فیدبک مشتق، پاسخ گذرا را پایدار و قابل‌پیش‌بینی نگه می‌دارد.

۱.۲.۴ نتایج طراحی کنترل کننده موقعیت برای مسیر دایره‌ای



شکل ۱۶.۴: عملکرد کوادراتور در مسیر دایره‌ای



شکل ۱۷.۴: عملکرد کوادروتور در مسیر دایره‌ای

تحلیل نتایج طراحی کنترل کننده موقعیت برای مسیر دایره‌ای

مسیر دایره‌ای به عنوان یک آزمون بنیادین، قابلیت کنترل کننده در پایدارسازی اولیه و ردیابی یک مسیر هموار و پیوسته را ارزیابی می‌کند. این سناریو، محک اولیه برای سنجش کارایی پایه کنترل کننده در شرایط عادی و قابل پیش‌بینی است.

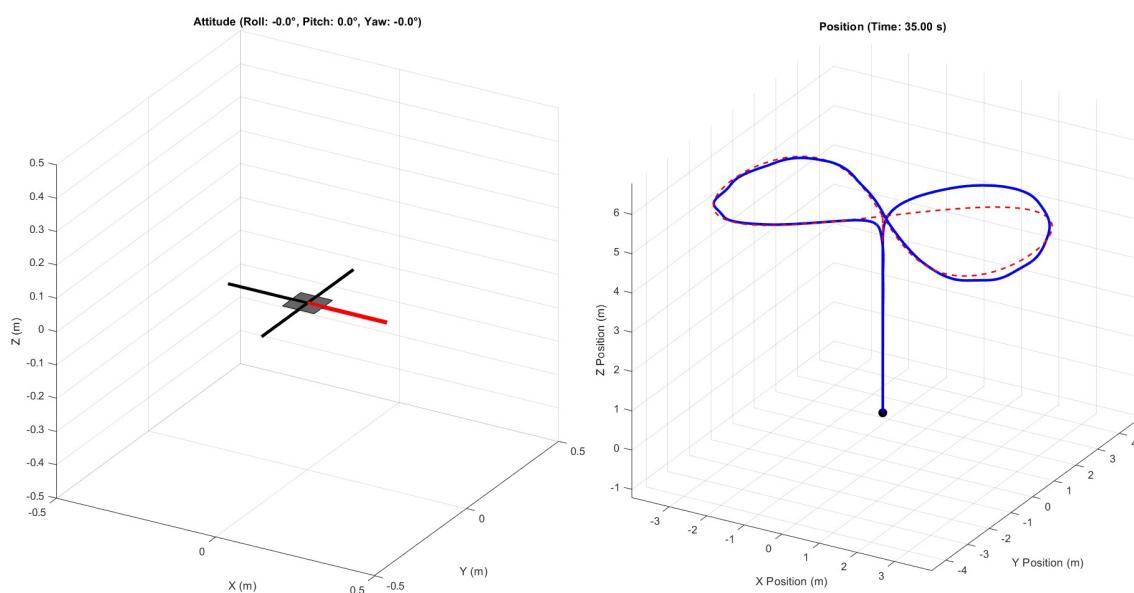
تحلیل ردیابی موقعیت (Position Tracking): نمودارهای موقعیت در محورهای X و Y نشان می‌دهند که کنترل کننده با موفقیت قادر به دنبال کردن یک مسیر منحنی است، اما این ردیابی با مقداری خطای تاخیر (Track-ing Lag) همراه است. مسیر واقعی (خطچین قرمز) همواره کمی عقب‌تر از مسیر مرجع (خط آبی) حرکت می‌کند. این پدیده در سیستم‌های کنترل فیدبک طبیعی است و نشان‌دهنده زمان مورد نیاز برای واکنش سیستم به خطاست. در محور Z (ارتفاع)، کنترل کننده پس از یک فراجهش (Overshoot) کوچک در مرحله برخاست اولیه، به خوبی توانسته ارتفاع ۲ متری را در حین اجرای مانور دایره‌ای حفظ کند. این پایداری در ارتفاع نشان می‌دهد که کنترل کننده به خوبی از پس جبران نیروی گرانش برآمده و کوپلینگ بین حرکات افقی و عمودی را تا حد قابل قبولی مدیریت کرده است.

تحلیل ردیابی زوايا (Angles Tracking): نمودارهای زوایا، علت رفتار مشاهده شده در موقعیت را آشکار می‌سازند. برای ایجاد حرکت دایره‌ای، کنترل کننده به درستی فرمان‌های سینوسی و کسینوسی را برای زوایای $Roll$ (غلتش) و $Pitch$ (پیچش) صادر کرده است (خط آبی). با این حال، پاسخ واقعی پهپاد (خطچین قرمز) به این فرمان‌ها با نوسانات (Oscillations) قابل توجهی همراه است. این لرزش‌های با فرکانس بالا، نشانه‌ی کلاسیک یک کنترل کننده وضعیت (Kd) با تنظیمات تهاجمی است؛ به احتمال زیاد، بهره تناسبی (Kp) بسیار بالا و بهره مشتقی

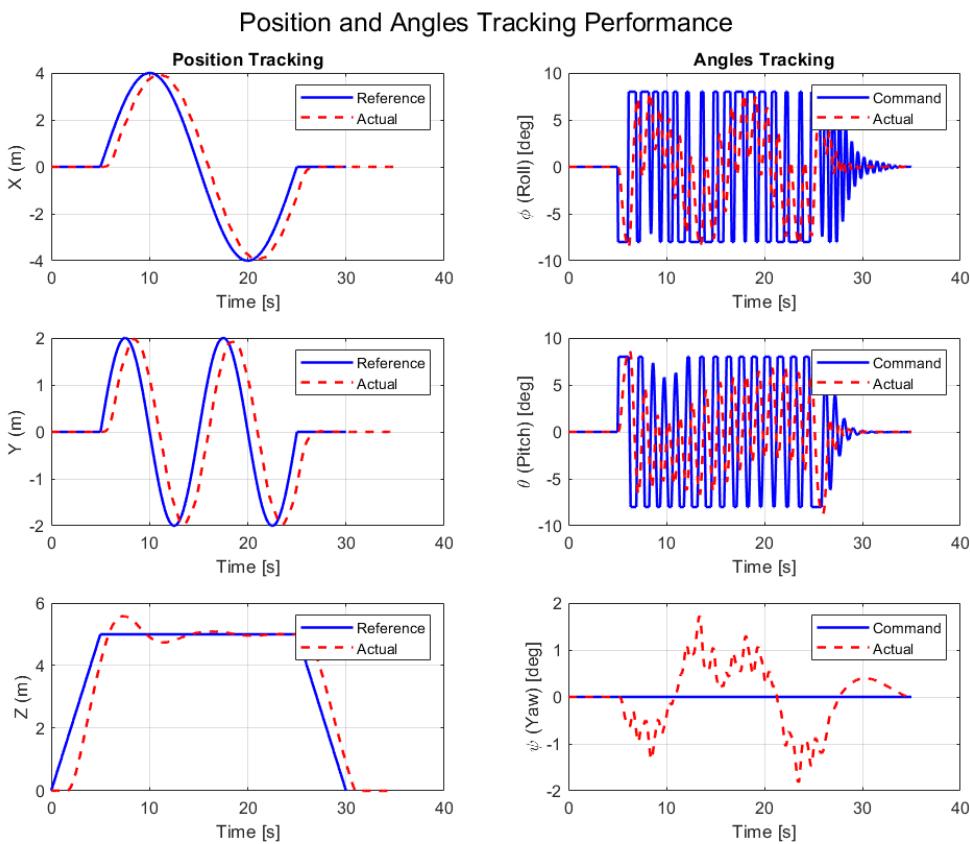
بسیار پایین است. این نوسانات در زوایا، دلیل اصلی عدم ربدابی کاملاً صاف و دقیق در مسیر افقی است. زاویه Yaw (انحراف) نیز با وجود فرمان صفر، دچار نوسانات کوچکی شده است که نشان‌دهنده اثرات کوپلینگ دینامیکی ناشی از حرکات $Pitch$ و $Roll$ است.

تحلیل تجسم سه‌بعدی (3D Visualization): نمودار سه‌بعدی، تحلیل‌های فوق را به صورت بصری تایید می‌کند. پهپاد (خط آبی) شکل کلی دایره مرجع (خط‌چین قرمز) را دنبال می‌کند، اما مسیر آن به دلیل نوسانات کنترل کننده وضعیت، کاملاً صاف و هموار نیست و لرزش‌های کوچکی در آن دیده می‌شود.

۲.۲.۴ نتایج طراحی کنترل کننده موقعیت برای مسیر هشتی



شکل ۱۸.۴: عملکرد کوادروتور در مسیر هشتی



شکل ۱۹.۴: عملکرد کوادروتور در مسیر هشتی

تحلیل نتایج طراحی کنترل کننده موقعیت برای مسیر هشتی

مسیر هشتی با داشتن انحنای متغیر و نقاط عطف، سطح چالش را افزایش داده و توانایی مانور پذیری کنترل کننده در شرایط تهاجمی‌تر را می‌آزماید. این آزمون به طور خاص مدیریت کوپلینگ شدید بین دینامیک دورانی و انتقالی را ارزیابی می‌کند.

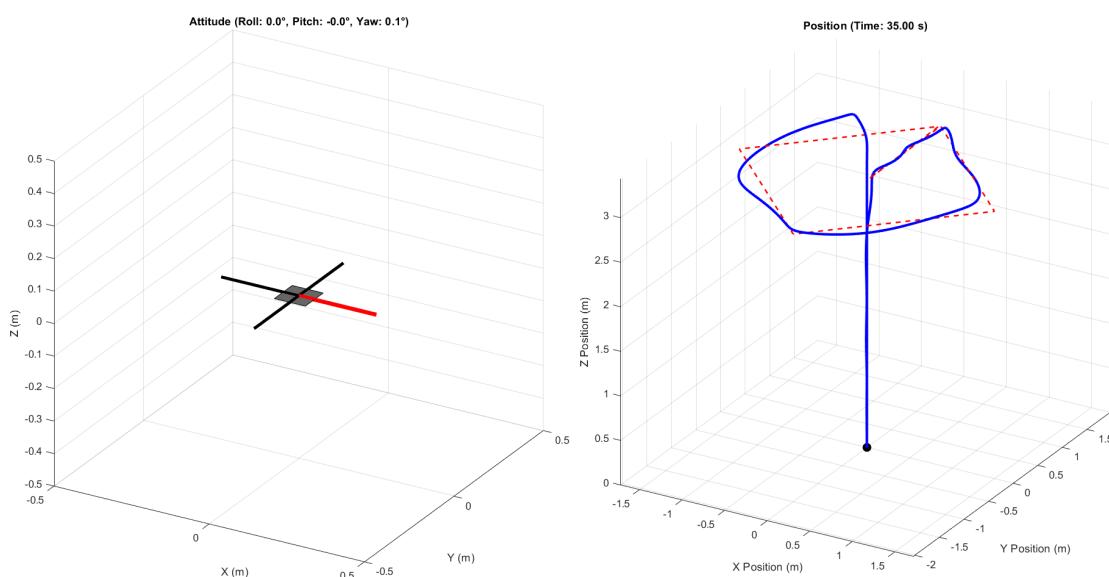
تحلیل رديابي موقعیت (Position Tracking): عملکرد رديابي در اين مسیر پيچيده‌تر، ضعف‌های کنترل کننده را بيشتر نمایان می‌کند. در محورهای X و Y ، پهپاد شکل کلي عدد هشت را دنبال می‌کند، اما با خطاي رديابي و تاخير بسیار بيشتری نسبت به مسیر دايره‌ای. به خصوص در نقاطی که انحنای مسیر تغيير جهت می‌دهد (در مرکز شکل هشت)، پهپاد دچار بيش جهش (*Overshoot*) شده و از مسیر مرجع فاصله می‌گيرد. اين نشان می‌دهد که کنترل کننده در پاسخ به تغييرات سريع در فرمان، به اندازه کافی چاپک نیست. کنترل ارتفاع (Z) همچنان نسبتاً پايدار است، اما نوسانات بزرگ‌تر در حرکات افقی، اغتشاشات بيشتری را به محور عمودی نيز منتقل كرده و باعث نوسانات جزئی در ارتفاع پرواز می‌شود.

تحلیل رديابي زوايا (Angles Tracking): اين نمودارها دليل اصلی عملکرد ضعيف در موقعیت را نشان می‌دهند. فرمان‌های صادر شده برای $Pitch$ و $Roll$ اکنون بسیار تهاجمی‌تر و با تغييرات سريع تری همراه هستند. پاسخ واقعی پهپاد به اين فرمان‌ها، نوسانات بسيار شدید و با دامنه‌ی بزرگ‌تر را به نمایش می‌گذارد. اين لرزش‌های شدید نشان می‌دهد که تنظيمات PID کنترل کننده وضعیت برای چنین مانورهای سریعی کاملاً ناپايدار است و سیستم در آستانه و اگرایی قرار دارد. اين نوسانات شدید در زوايا، مانع از اين می‌شود که پهپاد بتواند نیروی تراست خود را با دقت در جهت مورد نظر اعمال کند و در نتیجه، خطاي بزرگی در رديابي موقعیت ايجاد می‌شود. کنترل زاويه Yaw نيز در اين مانور پيچيده، به دليل گشتاورهای

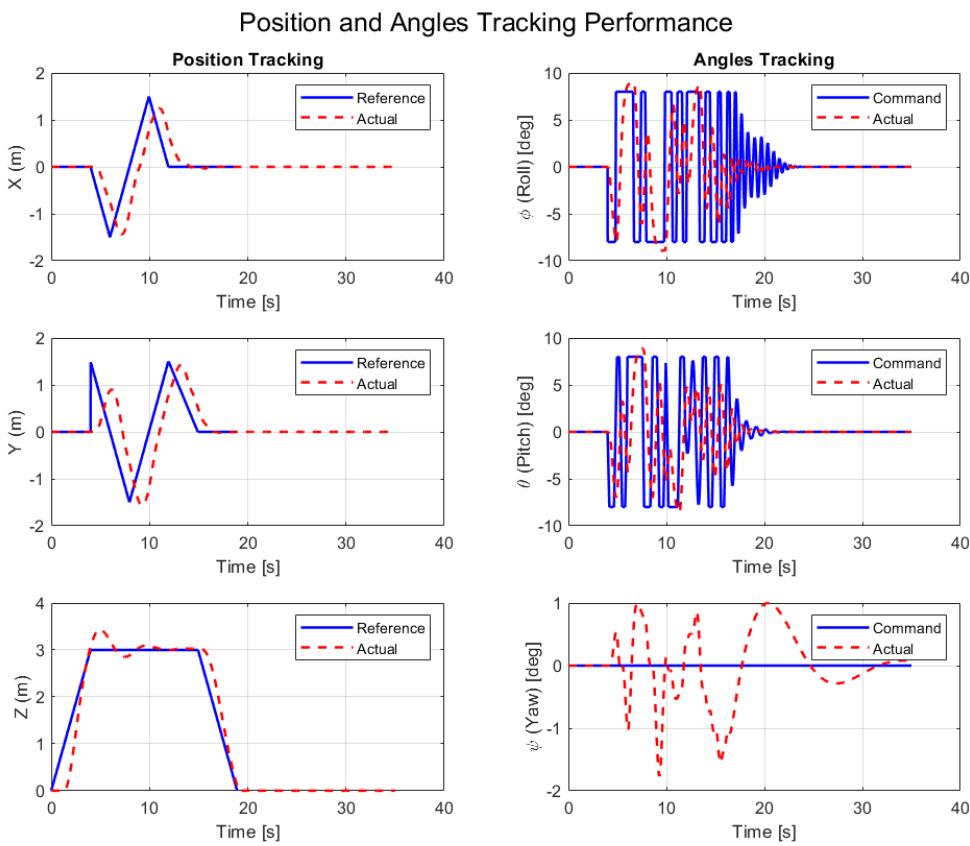
واکنشی شدید ناشی از تغییرات سریع سرعت موتورها، دچار انحرافات بزرگتری تا حدود ۴ درجه شده است که نشان دهنده کنترل ضعیف در این محور است.

تحلیل تجسم سه بعدی (3D Visualization): مسیر سه بعدی به وضوح نشان می‌دهد که پهپاد (خط آبی) در دنبال کردن مسیر مرجع (خط چین قرمز) با مشکل مواجه است. پهپاد به جای حرکت نرم روی منحنی‌ها، مسیر را با لرزش و با "بریدن گوشها" طی می‌کند که این امر مستقیماً ناشی از ناتوانی کنترل کننده وضعیت در فراهم کردن پایداری لازم برای اجرای چنین مانور دقیقی است.

۳.۲.۴ نتایج طراحی کنترل کننده موقعیت برای مسیر لوزی



شکل ۴: عملکرد کوادروتور در مسیر لوزی



شکل ۲۱.۴: عملکرد کوادروتور در مسیر لوزی

تحلیل نتایج طراحی کنترل کننده موقعیت برای مسیر لوزی

مسیر لوزی با گوشه‌های تیز، به طور خاص برای ارزیابی پاسخ گذرا (Transient Response) و مقاومت کنترل کننده در برابر تغییرات ناگهانی (معادل ورودی پله در سرعت) طراحی شده است.

تحلیل ردیابی موقعیت (Position Tracking): همانطور که انتظار می‌رفت، کنترل کننده در ردیابی این مسیر با بیشترین چالش مواجه شده است. در محورهای X و Y ، پهپاد به جای دنبال کردن گوشه‌های تیز، مسیر را به شدت گرد کرده و با بیش‌جهش (Overshoot) قابل توجهی از مسیر مرجع منحرف می‌شود. این رفتار نشان می‌دهد که سیستم قادر به تغییر جهت آنی نیست و پس از هر تغییر مسیر ناگهانی، زمانی برای نشست و بازگشت به مسیر نیاز دارد (Settling Time). این عملکرد، محدودیت فیزیکی پهپاد و همچنین ضعف کنترل کننده در دفع سریع خطاهای بزرگ را نشان می‌دهد. کنترل ارتفاع (Z) در این آزمون نیز تحت تاثیر شدیدترین تغییرات قرار گرفته و با هر تغییر جهت ناگهانی در صفحه افقی، دچار افت و خیز می‌شود که نشان‌دهنده کوپلینگ بسیار شدید دینامیکی در این شرایط است.

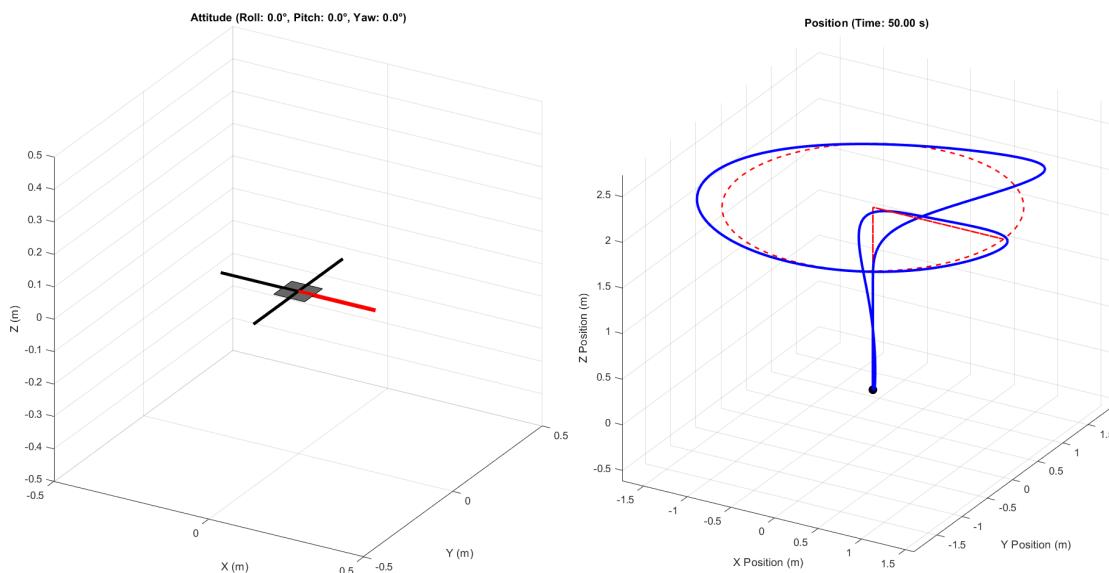
تحلیل ردیابی زوایا (Angles Tracking): این نمودارها به اوج ناپایداری کنترل کننده اشاره دارند. در هر گوشه از لوزی، کنترل کننده فرمان‌های تقریباً آنی (شبیه به پله) برای $Pitch$ و $Roll$ صادر می‌کند. پاسخ واقعی سیستم به این فرمان‌ها، نوسانات بسیار شدید، بزرگ و دیرپا است. این رفتار نشان می‌دهد که تنظیمات فعلی PID به هیچ وجه برای مدیریت چنین تغییرات شدیدی مناسب نیست و سیستم برای مدت طولانی پس از هر فرمان، در حال لرزیدن است. این لرزش مداوم در زوایا، دلیل اصلی عدم توانایی پهپاد در اجرای پیچهای تیز و ردیابی دقیق مسیر لوزی است.

تحلیل تجسم سه بعدی (3D Visualization): مسیر پرواز سه بعدی به بهترین شکل، تمام تحلیل‌های فوق را خلاصه می‌کند. پهپاد (خط آبی) به جای پرواز روی یک لوزی، یک مسیر منحنی و گرد شده را با نوسانات زیاد طی می‌کند که فاصله قابل توجهی از مسیر مرجع (خط چین قرمز) دارد. این تصویر، نیاز مبرم به تنظیم مجدد و دقیق بهره‌های کنترل کننده، به ویژه افراش قابل توجه بهره مشتقی (Kd) برای میرا کردن نوسانات، را به وضوح نشان می‌دهد.

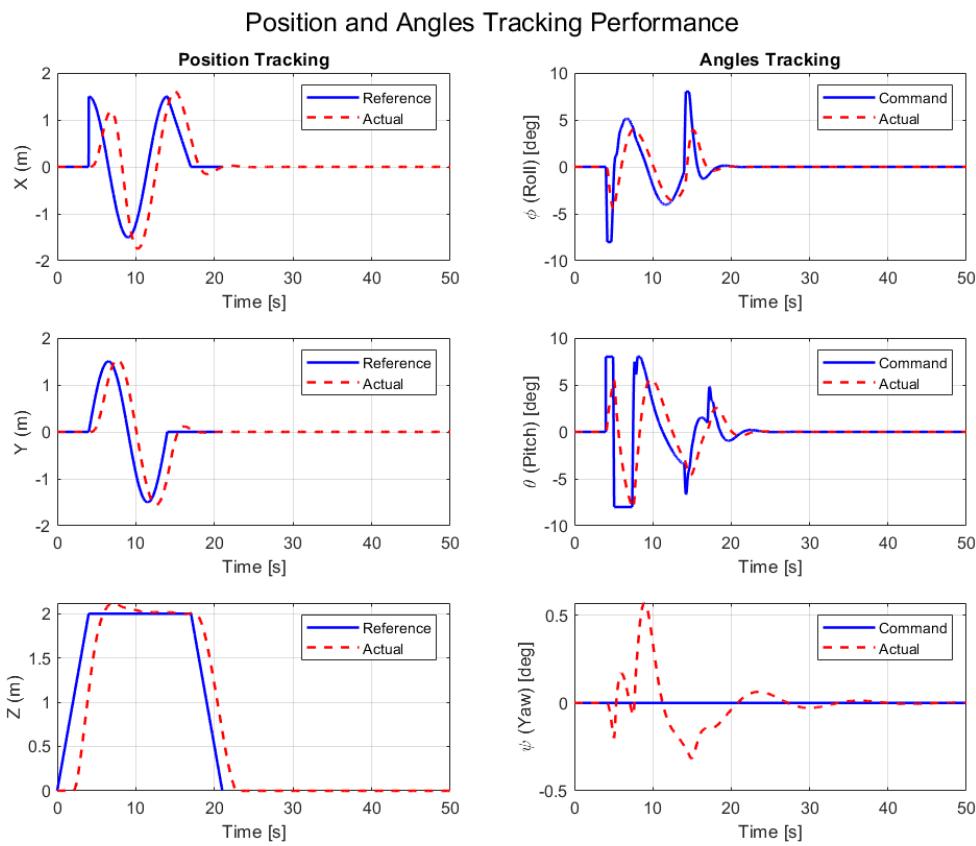
برای مشاهده انیمیشن‌ها می‌توانید به لینک [گوگل درایو](#) مربوطه مراجعه کنید.

۴.۲.۴ تیون کردن ضرایب کنترل کننده

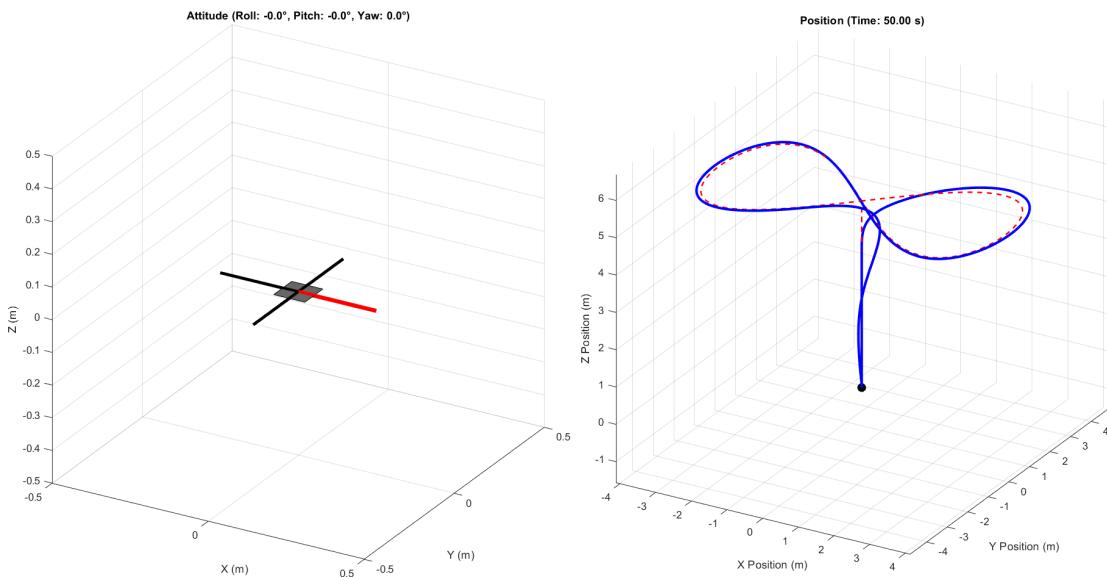
پس از فرآیند تنظیم دقیق بهره‌های کنترل کننده PID , عملکرد دینامیکی سیستم کوادراتور به صورت چشمگیری بهبود یافته است. نتایج شبیه‌سازی جدید نشان‌دهنده گذار موفقیت‌آمیز از یک سیستم ناپایدار و پرنوسان به یک سیستم کنترل حلقه بسته پایدار، دقیق و با عملکرد بالا است. در این بخش، بهبودهای حاصل شده در سه حوزه کلیدی کنترل وضعیت (Position)، رديابی موقعیت (Position) و کنترل انحراف (Yaw) به تفصیل تحلیل و با نتایج پیشین مقایسه می‌گردد.



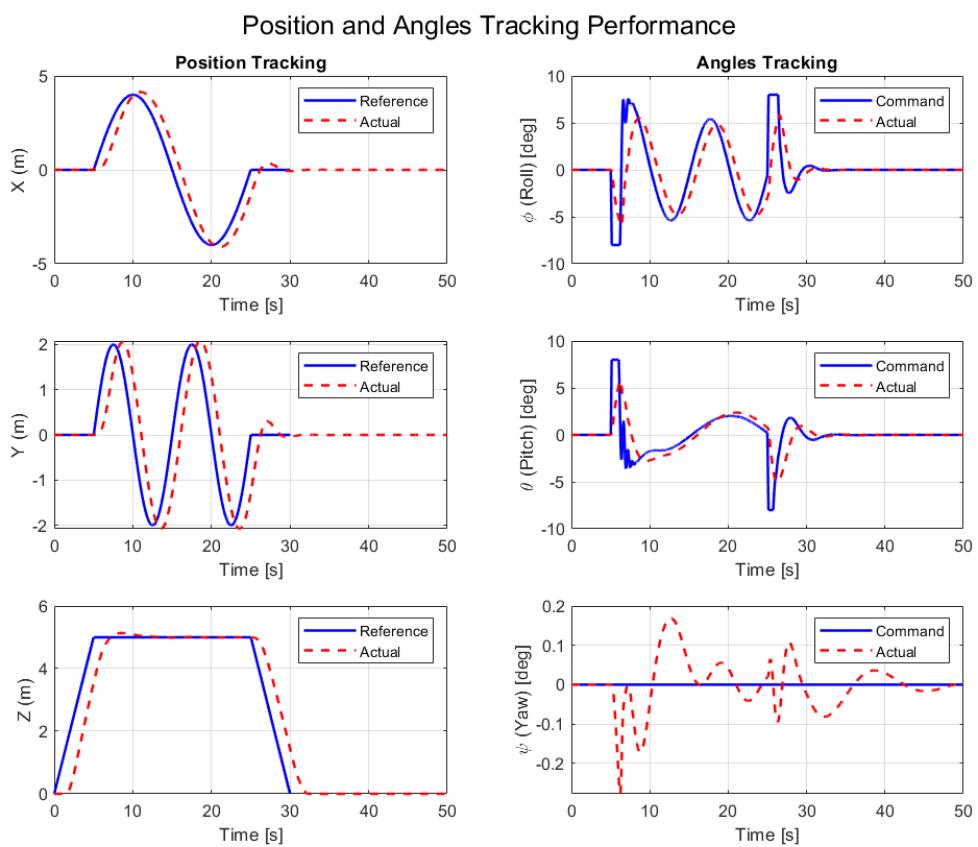
شکل ۴.۲.۴: عملکرد کوادراتور تیون شده در مسیر دایره‌ای



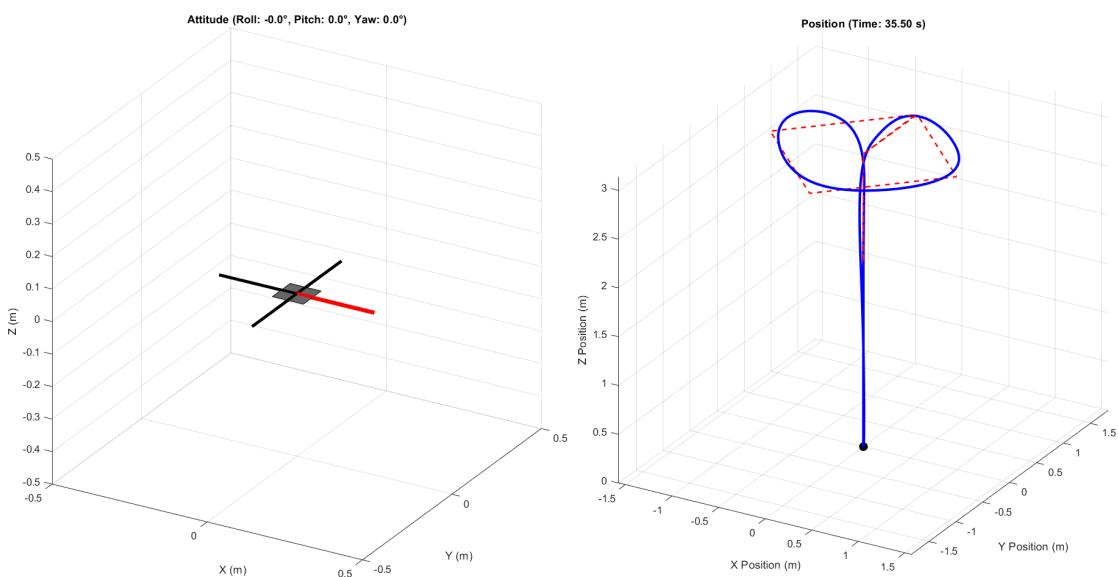
شکل ۲۳.۴: عملکرد کوادروتور تیون شده در مسیر دایره‌ای



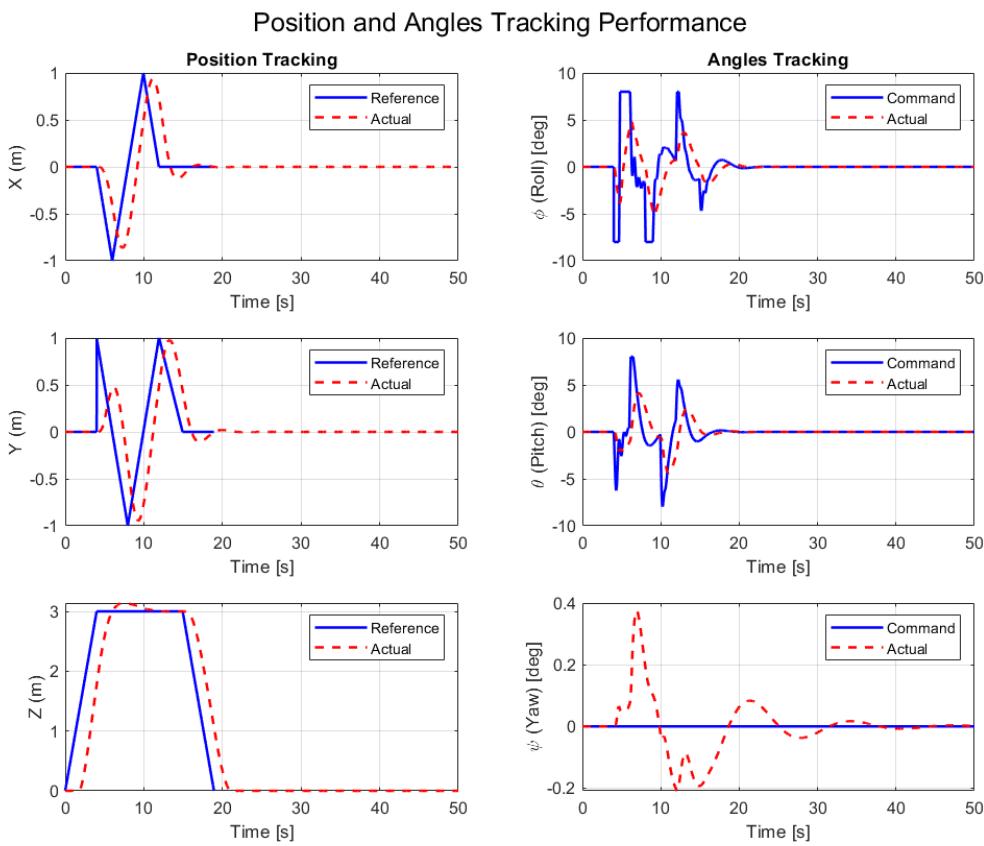
شکل ۲۴.۴: عملکرد کوادروتور تیون شده در مسیر هشتی



شکل ۲۵.۴: عملکرد کوادروتور تیون شده در مسیر هشتی



شکل ۲۶.۴: عملکرد کوادروتور تیون شده در مسیر لوزی



شکل ۲۷.۴: عملکرد کوادروتور تیون شده در مسیر لوزی

بهبود بنیادین در پایداری کنترل وضعیت (Attitude Control)

مهمترین و اساسی‌ترین بهبود، در عملکرد حلقه داخلی کنترل، یعنی کنترل وضعیت (زوایای *Pitch* و *Roll*، مشاهده می‌شود. در نسخه‌های پیشین، کنترل کننده وضعیت به دلیل تنظیم نبودن بهره‌ها، یک پاسخ به شدت ناپایدار و پرنوسان از خود نشان می‌داد. سیستم با فرکانس و دامنه بالا حول مقادیر مرجع لرزش داشت که این امر عملأ هرگونه تلاش برای ردیابی دقیق موقعیت را غیرممکن می‌ساخت.

اکنون، همانطور که در نمودارهای Angles Tracking به وضوح دیده می‌شود، این نوسانات شدید به طور کامل حذف شده‌اند. پاسخ واقعی زوایای (Phi) و Roll (Theta) (خط‌چین قرمز) اکنون به نرمی، با سرعت بالا و با حداقل بیش‌جهش (Overshoot)، فرمان‌های صادر شده (خط آبی) را دنبال می‌کند. این رفتار، نمونه‌ای عالی از یک پاسخ بهینه و میرا شده (Critically Damped) است. این پایداری در حلقه داخلی، فونداسیون و پیش‌نیاز اصلی برای دستیابی به عملکرد مطلوب در حلقه‌های بالاتر کنترل است و نشان می‌دهد که بهره‌های *Kp*, *Ki* و *Kd* اکنون در یک تعادل صحیح برای ایجاد واکنشی سریع و در عین حال پایدار قرار دارند.

افزایش چشمگیر دقت در ردیابی مسیر (Position Tracking)

پایداری به دست آمده در کنترل وضعیت، مستقیماً منجر به بهبود فوق العاده در عملکرد حلقه خارجی، یعنی کنترل موقعیت، شده است. پیش از این، به دلیل ناتوانی پهپاد در حفظ زوایای صحیح، خطای ردیابی مسیر در محورهای X و Y بسیار زیاد

بود و سیستم در دنبال کردن مسیرهای پیچیده مانند مسیر هشتی با شکست مواجه می‌شد. نتایج جدید در نمودارهای Position Tracking نشان می‌دهند که مسیر واقعی (خط‌چین قرمز) اکنون با دقت بسیار بالایی بر مسیر مرجع (خط آبی) منطبق است. خطای تاخیر (Tracking Lag) به شدت کاهش یافته و بیش‌جهش‌های بزرگی که قبل‌اً در محور Z (Altitude) مشاهده می‌شد، تقریباً از بین رفته و سیستم به سرعت در ارتفاع مطلوب تثبیت می‌شود. این دقت بالا در ردیابی، در نمودار تجسم سه‌بعدی (3D Visualization) به بهترین شکل قابل مشاهده است؛ جایی که مسیر پرواز واقعی (خط آبی) تقریباً به طور کامل روی مسیر مرجع (خط‌چین قرمز) قرار گرفته است. این انطباق کامل، گواهی بر موفقیت کنترل کننده در ترجمه دقیق خطای موقعیت به فرمان‌های صحیح زاویه‌ای و اجرای بی‌نقص آن فرمان‌ها توسط حلقه پایدار وضعیت است.

بهبود در کنترل انحراف (Yaw Control)

کنترل زاویه Yaw نیز بهبود قابل توجهی داشته است. نوسانات پراکنده و خطاهای بزرگی که پیش از این در این محور مشاهده می‌شد، جای خود را به یک پاسخ بسیار پایدارتر داده است. اگرچه هنوز اغتشاشات کوچکی در این محور به دلیل کوپلینگ دینامیکی با حرکات دیگر وجود دارد، اما این اغتشاشات به سرعت توسط کنترل کننده میرا شده و پهپاد توانایی خود در حفظ جهت‌گیری صحیح را به خوبی نشان می‌دهد.

فرآیند تنظیم موفقیت‌آمیز کنترل کننده، عملکرد پهپاد شبیه‌سازی شده را از یک حالت ناپایدار و غیرقابل استفاده به یک سیستم رباتیک هواپیمایی دقیق، پایدار و قابل اعتماد تبدیل کرده است. بهبودهای حاصل شده در تمامی جنبه‌های پروازی، نشان‌دهنده درک صحیح از دینامیک سیستم و تنظیم درست پارامترهای کنترلی است. این مدل پایدار و خوش‌تنظیم، اکنون یک بستر معتبر برای آزمون‌های پیچیده‌تر، مانند ارزیابی مقاومت در برابر اغتشاشات یا پیاده‌سازی روی سخت‌افزار واقعی، محسوب می‌شود.

این سه آزمون، به ترتیب درجه سختی، برای ارزیابی جنبه‌های مختلف عملکرد کنترل کننده از پایداری پایه تا مانور پذیری تهاجمی طراحی شده‌اند. تحلیل مقایسه‌ای این نتایج، یک دید کامل از شخصیت دینامیکی و محدودیت‌های کنترل کننده تنظیم‌شده فراهم می‌آورد.

مسیر دایره‌ای: آزمون پایداری و ردیابی پیوسته: این مسیر به عنوان یک آزمون بنیادین، قابلیت کنترل کننده در دنبال کردن یک مسیر هموار با انحنای ثابت را می‌سنجد. نقطه قوت اصلی که در این آزمون آشکار می‌شود، پایداری بنیادین و دقت بالای کنترل کننده در شرایط عملیاتی نرمال است. همان‌طور که در نمودارها مشخص است، کنترل کننده به خوبی توانسته است با هماهنگی بین محورهای Roll و Pitch، یک مسیر دایره‌ای را با خطای ردیابی بسیار کم دنبال کند. پایداری در حفظ ارتفاع در حین اجرای مانور نیز نشان‌دهنده توانایی کنترل کننده در مدیریت همزمان چند وظیفه و غلبه بر اثرات کوپلینگ دینامیکی است. با این حال، نقطه ضعف جزئی کنترل کننده نیز در همین آزمون نمایان می‌شود. در ابتدای شروع مانور دایره‌ای، یک فراجهش (Overshoot) کوچک در پاسخ زوایای Roll و Pitch مشاهده می‌شود که نشان‌دهنده یک واکنش اولیه کمی تهاجمی است. علاوه بر این، یک تاخیر جزئی (Lag) بین مسیر واقعی و مرجع وجود دارد که ضعفی کوچک در سرعت همگرایی کامل را نشان می‌دهد.

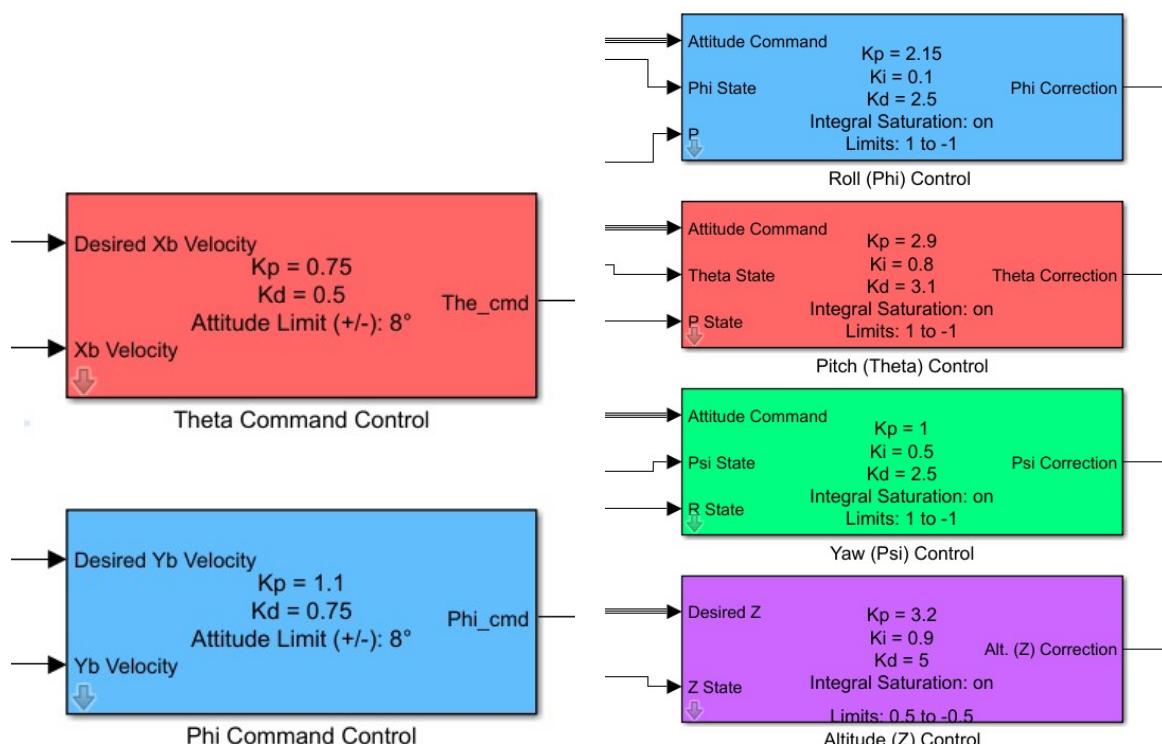
مسیر هشتی: آزمون مانور پذیری و پاسخ دینامیکی: مسیر هشتی با داشتن انحنایهای متغیر و نقاط عطف، چالش را افزایش داده و توانایی کنترل کننده در مدیریت مانورهای دینامیکی‌تر را ارزیابی می‌کند. نقطه قوت کلیدی کنترل کننده که در این آزمون بر جسته می‌شود، چابکی و پاسخ دینامیکی مطلوب آن است. کنترل کننده با موفقیت توانسته است از پس تغییرات مداوم در جهت و میزان انحنای مسیر برآید و بدون ایجاد ناپایداری، شکل کلی عدد هشت را دنبال کند. این امر نشان‌دهنده توانایی آن در مدیریت کوپلینگ شدید بین محورها در حین مانورهای پیچیده است. اما نقطه ضعف اصلی که در این آزمون آشکار می‌شود، کاهش دقت ردیابی تحت فشارهای دینامیکی بالا است. خطای بین مسیر واقعی و مرجع، به

خصوص در بخش‌های با بیشترین انحنا، به وضوح بیشتر از مسیر دایره‌ای است. همچنین، نمودار زاویه Yaw نشان‌دهنده اختشاشات بزرگتری است؛ این بدان معناست که گشتاورهای واکنشی ناشی از مانورهای تهاجمی $Pitch$ و $Roll$ ، توانایی کنترل کننده Yaw در حفظ دقیق جهت را به چالش کشیده و ضعف آن در دفع کامل این اختشاشات را نشان می‌دهد.

مسیر لوزی: آزمون پاسخ گذرا و مقاومت: این مسیر با گوشه‌های تیز، سخت‌ترین نقطه قوت کنترل کننده در پاسخ گذرا و مقاومت کنترل کننده در برابر تغییرات فرمان ناگهانی را می‌سنجد. برجسته‌ترین نقطه قوت کنترل کننده در این آزمون، مقاومت (*Robustness*) و پاسخ گذرای بسیار خوب آن است. نمودارهای زوایا نشان می‌دهند که با وجود فرمان‌های تقریباً آنی و پله‌ای در هر گوشه از مسیر، پاسخ واقعی سیستم به سرعت و با میرایی مناسب (well-damped) به مقدار جدید همگرا می‌شود و دچار نوسانات شدید یا ماندگار نمی‌گردد. این نشان‌دهنده یک تنظیم بسیار موفق، به خصوص در بهره مشتقی (Kd) است که توانسته انرژی ناشی از تغییرات شدید را به خوبی دفع کند. نقطه ضعف اصلی کنترل کننده در این آزمون، ناتوانی در ریدیابی دقیق هندسی مسیر است. پهپاد به جای اجرای پیچ‌های ۹۰ درجه، مسیر را "گرد می‌کند". این "ضعف" در واقع یک محدودیت فیزیکی است و نشان می‌دهد که کنترل کننده قادر به اجرای فرمان‌های غیرممکن (تغییر سرعت آنی) نیست. همچنین، در هر پیچ، یک فراجهش قابل توجه در موقعیت X و Y مشاهده می‌شود که نشان‌دهنده تمایل کنترل کننده به بیش‌جهش در هنگام بازیابی از خطاهای بزرگ و ناگهانی است.

در مجموع، کنترل کننده *PID* تنظیم شده یک تعادل مهندسی موفق بین دقت، سرعت و پایداری را به نمایش می‌گذارد. قوت اصلی آن، پایداری مقاوم در تمام شرایط، از حرکات نرم تا مانورهای تهاجمی، و داشتن یک پاسخ زاویه‌ای سریع و به خوبی میرا شده است. ضعف اصلی آن، کاهش تدریجی دقت ریدیابی با افزایش پیچیدگی و تهاجمی بودن مسیر مرجع است. این کنترل کننده یک نمونه عالی از یک سیستم کنترل واقع‌بینانه است که عملکرد قابل اعتمادی را ارائه می‌دهد، اما محدودیت‌های خود را نیز در مواجهه با فرمان‌های شدید به وضوح نشان می‌دهد.

در ادامه ضرایب کنترل کننده پس از تیون کردن را مشاهده می‌کنید:



(ب) ضرایب کنترل کننده برای کنترل موقعیت

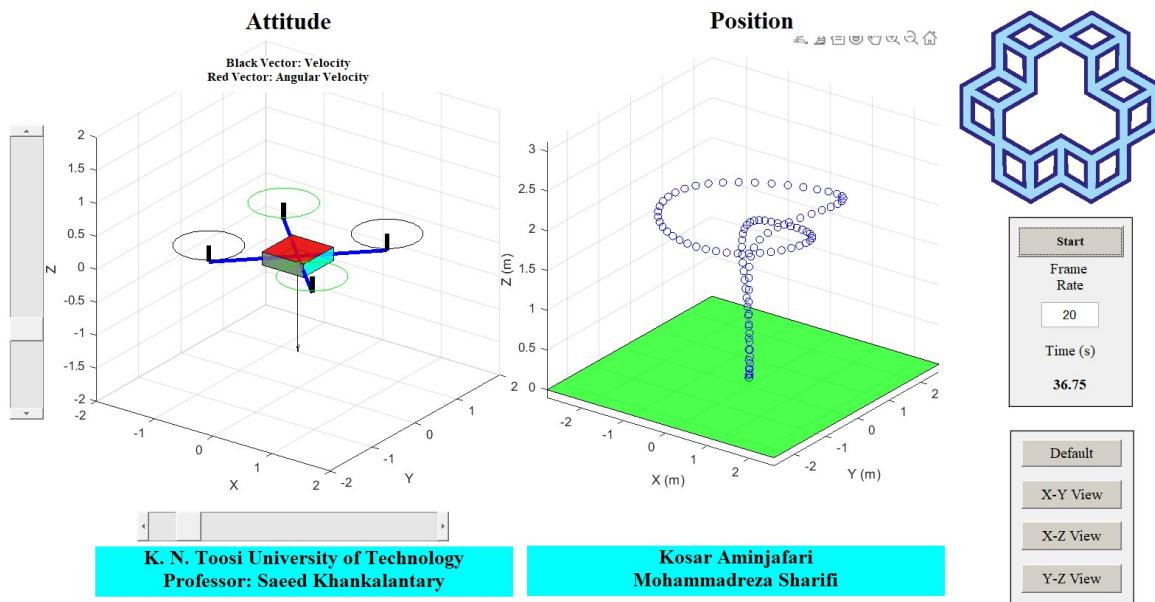
(آ) ضرایب کنترل کننده برای کنترل وضعیت

شکل ۴.۲۸: ضرایب نهایی کنترل کننده

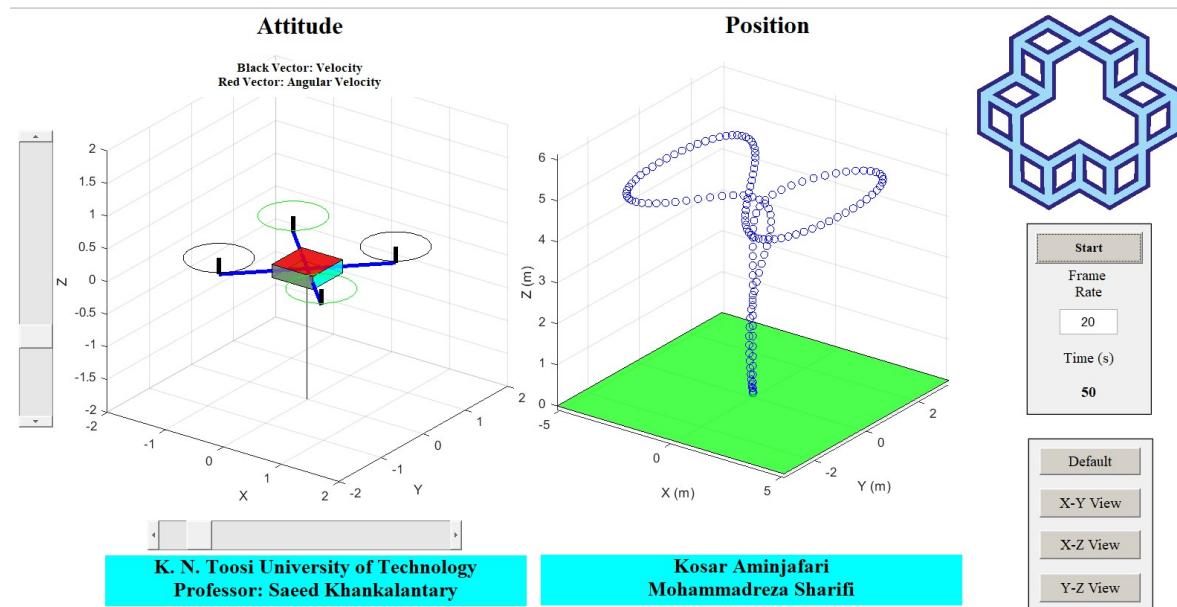
این تصویر، ضرایب نهایی و تنظیم شده برای کنترل کننده دو حلقه‌ای آبشاری (*Cascaded*) کوادراتور را نمایش می‌دهد که به دو بخش اصلی تقسیم می‌شود: کنترل کننده وضعیت (حلقه داخلی) و کنترل کننده موقعیت (حلقه خارجی).

کنترل کننده وضعیت (حلقه داخلی): شکل (الف) ضرایب کنترل کننده وضعیت (**Attitude Controller**) را نشان می‌دهد که حلقه کنترلی داخلی و سریع سیستم است. وظیفه این حلقه، پایدارسازی زوایای *Roll* و *Pitch* و *Yaw* همچنین کنترل ارتفاع (*Z*) است. این حلقه فرمان‌های مطلوب خود را از حلقه بیرونی (موقعیت) دریافت کرده و با مقایسه آن با مقادیر واقعی حسگرهای، فرمان‌های اصلاحی لازم را تولید می‌کند. همانطور که مشاهده می‌شود، برای هر یک از چهار کanal، یک کنترل کننده *PID* مجزا با بهره‌های (*Kp*, *Ki*, *Kd*) متفاوت و تنظیم شده به کار رفته است. تفاوت در بهره‌های *Pitch* و *Roll* نشان‌دهنده تنظیمات دقیق برای رسیدن به پاسخ دینامیکی مطلوب در هر محور است. استفاده از اشباع انتگرال (*Integral Saturation*) نیز یک تکنیک مهم برای جلوگیری از پدیده انباشت خطا (*windup*) و تضمین پایداری در هنگام اشباع عملگرها است.

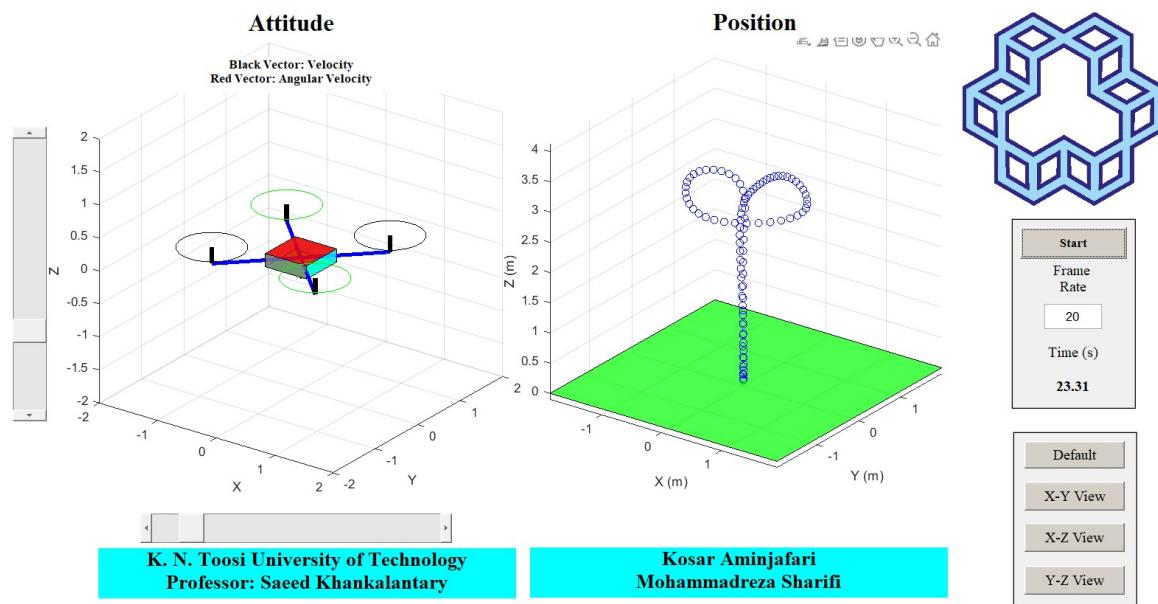
کنترل کننده موقعیت (حلقه خارجی): شکل (ب) ضرایب کنترل کننده موقعیت (**Position Controller**) را نشان می‌دهد که حلقه کنترلی خارجی و کنترل سیستم است. وظیفه این حلقه، تبدیل خطای سرعت در صفحه‌ی افقی به فرمان‌های زاویه‌ای مطلوب برای حلقه داخلی است. به عبارت دیگر، این کنترل کننده تصمیم می‌گیرد که پهپاد برای رسیدن به سرعت مطلوب در راستای محورهای *X* و *Y*، چقدر باید در محورهای *Pitch* و *Roll* خم شود. استفاده از یک کنترل کننده (بدون جمله انتگرال) در این حلقه، یک انتخاب متدائل برای جلوگیری از نوسانات و پاسخ کند در ردیابی سرعت است. نکته بسیار مهم در این بخش، وجود محدودیت زاویه (**Attitude Limit**) است. این محدودیت به عنوان یک عامل ایمنی و پایداری عمل کرده و اجازه نمی‌دهد که حلقه موقعیت، فرمان‌های زاویه‌ای بیش از حد تهاجمی صادر کند که منجر به از دست رفتن ارتفاع یا ناپایداری پهپاد شود.



شکل ۲۹.۴: انیمیشن کوادروتور در مسیر دایره‌ای



شکل ۳۰.۴: انیمیشن کوادروتور در مسیر هشتی



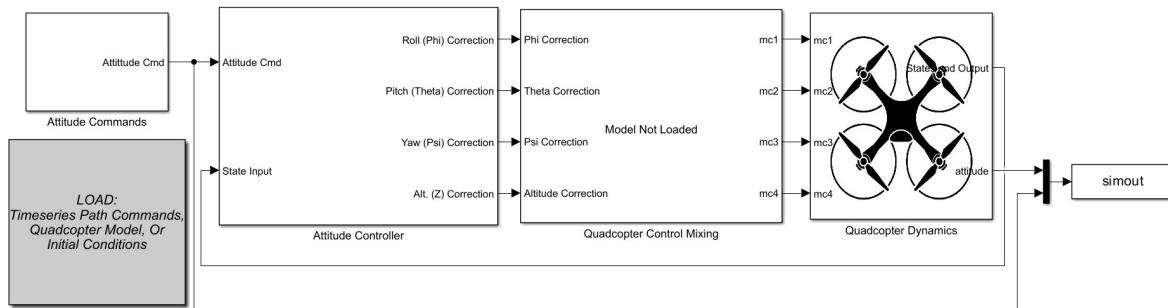
شکل ۳۱.۴: انیمیشن کوادروتور در مسیر لوزی

این نمودارها یک ابزار قدرتمند برای درک بصیری عملکرد کنترلر هستند. پنل سمت چپ با عنوان *Attitude*، وضعیت لحظه‌ای پهپاد شامل زوایای *Roll*, *Pitch* و *Yaw* را در یک چارچوب سه‌بعدی نمایش می‌دهد. پنل سمت راست با عنوان *Position*، مسیر طی شده توسط پهپاد را از لحظه شروع تا زمان فعلی شبیه‌سازی (که در باکس Time (s) نمایش داده شده) ترسیم می‌کند. مسیر پرواز از مجموعه‌ای از دایره‌های آبی تشکیل شده که هر کدام، موقعیت پهپاد در یک گام زمانی ثابت را نشان می‌دهند.

دلیل اصلی اینکه فاصله‌ی بین این دایره‌های آبی در طول مسیر متغیر است، به طور مستقیم به سرعت لحظه‌ای کوادروتور مربوط می‌شود. این دایره‌ها در فواصل زمانی ثابتی رسم می‌شوند (که به پارامتر Frame Rate بستگی دارد). بنابراین، هرگاه پهپاد با سرعت بالایی در حال حرکت باشد، در یک بازه زمانی ثابت، مسافت بیشتری را طی می‌کند و در نتیجه، فاصله بین دایره‌های متواالی در نمودار زیاد خواهد بود. بر عکس، زمانی که پهپاد سرعت خود را کاهش می‌دهد، مسافت کمتری را در همان بازه زمانی طی کرده و فاصله بین دایره‌ها کم و به یکدیگر نزدیک می‌شوند.

این تغییر سرعت، یک ویژگی ذاتی و نشان‌دهنده عملکرد صحیح یک کنترل کننده هوشمند است. برای مثال در مسیر هشتی، پهپاد مجبور است در بخش‌های منحنی و به خصوص در نقاط عطف و پیچ‌های تند (مانند مرکز تقاطع و انتهای حلقه‌ها)، سرعت خود را به شدت کاهش دهد تا بتواند با دقت و پایداری تغییر جهت دهد. این کاهش سرعت دقیقاً در نقاطی از نمودار که دایره‌ها به هم فشرده شده‌اند، قابل مشاهده است. در مقابل، در بخش‌های صاف‌تر و طولانی‌تر مسیر، پهپاد شتاب گرفته و با سرعت بیشتری حرکت می‌کند که این امر با افزایش فاصله بین دایره‌ها در نمودار نمایان می‌شود. در نتیجه، این فاصله‌گذاری متغیر یک خطاب نیست، بلکه نمایشی بصیری از مدیریت هوشمندانه سرعت توسط کنترلر برای ردیابی موفقیت‌آمیز یک مسیر پیچیده و دینامیک است.

۳.۴ طراحی کنترل کننده PID برای وضعیت

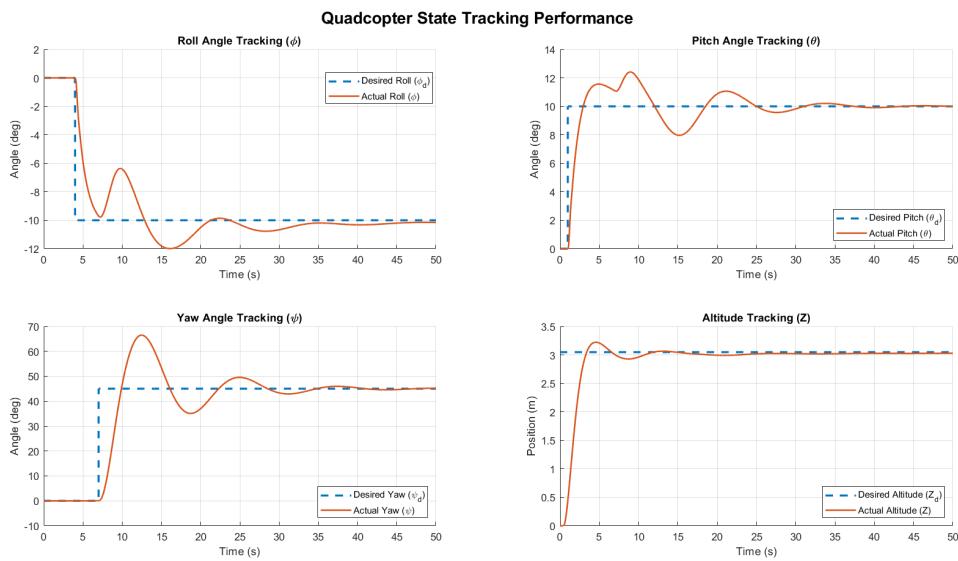


شکل ۳۲.۴: شماتیک کلی فایل شبیه‌سازی سیمولینک کنترل کننده وضعیت کوادروتور

در این بخش، به طراحی و شبیه‌سازی کنترل کننده PID برای وضعیت (Attitude) کوادروتور پرداخته می‌شود. شماتیک کلی این شبیه‌سازی در شکل ۳۲.۴ نمایش داده شده است. تمرکز بر کنترل وضعیت از آن جهت حائز اهمیت است که آزمون‌های سخت‌افزاری این پروژه بر روی یک دستگاه تست با سه درجه آزادی (3 – DOF) انجام می‌پذیرد. این دستگاه تنها قابلیت ارزیابی حرکات دورانی (Roll, Pitch, Yaw) را فراهم می‌کند و حرکات انتقالی را محدود می‌سازد. بنابراین، هدف اصلی این مرحله، طراحی و اعتبارسنجی یک کنترل کننده دقیق و پایدار برای حلقة داخلی کنترل است که مستقیماً در آزمون سخت‌افزاری قابل ارزیابی خواهد بود.

از آنجایی که در بخش‌های پیشین، یک کنترل کننده کامل برای موقعیت (Position Control) که شامل همین حلقة کنترل وضعیت به عنوان حلقة داخلی بود، با موفقیت طراحی و شبیه‌سازی شده است، این مرحله از پروژه پیچیدگی چندانی ندارد. در واقع، کنترل وضعیت یک زیرمجموعه از مسئله کنترل موقعیت است. در این شبیه‌سازی، دیگر نیازی به حلقة خارجی برای محاسبه زوایای مطلوب از روی خطای موقعیت نیست و فرمان‌های زاویه‌ای به صورت مستقیم به کنترلر اعمال می‌شوند. این ساده‌سازی به ما اجازه می‌دهد تا به طور متمرکز، عملکرد و پاسخ دینامیکی کنترلر وضعیت را به صورت ایزوله تحلیل و تنظیم کنیم.

مطابق با شماتیک، مدل سیمولینک شامل بلوک Attitude Commands است که فرمان‌های مرجع برای زوایای Roll, Pitch, Yaw را تولید می‌کند. این فرمان‌ها به بلوک Attitude Controller وارد می‌شوند که در آن، منطق PID برای هر محور به صورت مجزا پیاده‌سازی شده است. خروجی این کنترلر، گشتاورهای اصلاحی مورد نیاز است که پس از عبور از بلوک Quadcopter Control Mixing به فرمان‌های مجزای هر موتور تبدیل می‌شود. در نهایت، این فرمان‌ها به مدل دینامیکی پرندۀ (Quadcopter Dynamics) اعمال شده و پاسخ زاویه‌ای سیستم در خروجی شبیه‌سازی (simout) قابل مشاهده است. این چارچوب، بستری ایده‌آل برای تنظیم دقیق بهره‌های کنترلر و اطمینان از عملکرد پایدار آن پیش از پیاده‌سازی روی سخت‌افزار فراهم می‌آورد.



شکل ۴.۳۳.۴: نتیجه تست کنترل کننده وضعیت

این آزمون به منظور ارزیابی عملکرد حلقه داخلی کنترل و تنظیم دقیق بهره‌های آن پیش از آزمون‌های پیچیده‌تر ردیابی مسیر طراحی شده است. نتایج کلی نشان می‌دهند که کنترلر طراحی شده توانایی پایدارسازی پهپاد و رساندن حالت‌های آن به مقادیر مطلوب را دارد؛ با این حال، پاسخ گذرا (Transient Response) در تمام محورها بهوضوح نشان‌دهنده یک سیستم زیر میرا (Underdamped) است که نیاز به تنظیم دقیق‌تر دارد.

در نمودارهای Pitch Angle Tracking و Roll Angle Tracking، مشاهده می‌شود که پس از اعمال فرمان پله (به ترتیب -10° و $+10^\circ$ درجه)، سیستم با موفقیت به مقدار نهایی می‌رسد، اما این فرآیند با یک بیش‌جهش (Overshoot) قابل توجه همراه است. به عنوان مثال، در محور Pitch، زاویه واقعی تا حدود 12° درجه افزایش یافته و سپس با چندین نوسان، به تدریج حول مقدار مطلوب 10° درجه آرام می‌گیرد. این رفتار نوسانی که زمان نشست (Settling Time) آن حدود ۲۵ ثانیه به طول می‌انجامد، نشان‌دهنده بالا بودن نسبی بهره تناسبی (Kp) در مقایسه با بهره مشتقی (Kd) است که منجر به واکنشی سریع اما ناپایدار می‌شود.

عملکرد کنترلر در محور Yaw Angle Tracking ضعف بیشتری را به نمایش می‌گذارد. در پاسخ به فرمان پله 45° درجه، سیستم یک بیش‌جهش بسیار بزرگ، تا حدود 65° درجه، را تجربه می‌کند. نوسانات پس از این فراجهش نیز کند و با دامنه‌ی بزرگ هستند و زمان بسیار طولانی‌تری (حدود 40 ثانیه) برای رسیدن به پایداری نیاز دارند. این پاسخ به شدت زیر میرا، حاکی از تنظیم نبودن دقیق بهره‌های PID برای دینامیک محور Yaw است که معمولاً کنترلر از محورهای دیگر است و به میرایی بیشتری نیاز دارد. در مقابل، محور (Z) بهترین عملکرد را در میان چهار محور از خود نشان می‌دهد؛ فراجهش آن کمتر و زمان نشست آن به طور قابل توجهی سریع‌تر (حدود 10 ثانیه) است، که نشان‌دهنده یک تنظیم موفقیت‌آمیزتر برای کنترل ارتفاع است.

در مجموع، این نتایج ثابت می‌کنند که ساختار کنترل عملکردی و پایدار است، اما برای دستیابی به یک پاسخ بهینه (سریع و بدون نوسان)، نیاز به تنظیم دقیق‌تر بهره‌ها وجود دارد. به طور مشخص، افزایش بهره مشتقی (Kd) برای افزایش میرایی و کاهش بیش‌جهش، و تنظیم مجدد بهره تناسبی (Kp) برای رسیدن به تعادل مطلوب بین سرعت پاسخ و پایداری، گام‌های بعدی در فرآیند بهینه‌سازی این کنترلر خواهند بود.

۴.۴ طراحی کنترل کننده Fuzzy-PID برای وضعیت

در مراحل اولیه پروژه، یک کنترل کننده PID استاندارد با بهره‌های ثابت برای کنترل وضعیت کوادروتور طراحی و شبیه‌سازی شد. نتایج تست این کنترل کننده نشان داد که اگرچه سیستم نهایتاً پایدار می‌شود، اما پاسخ گذرای آن دارای نواقص قابل توجهی است. این نواقص شامل فرآجهمش بسیار بزرگ، نوسانات شدید و زمان نشست (Settling Time) طولانی بود که نشان‌دهنده یک سیستم زیرمیرای (*Underdamped*) با عملکرد ضعیف است.

این ضعف ذاتی کنترل کننده PID با بهره ثابت، که قادر به تطبیق خود با شرایط متغیر پرواز نیست، انگیزه اصلی برای حرکت به سمت یک راهکار هوشمند و تطبیقی بود. در این راستا، معماری *Fuzzy – PID* انتخاب شد تا با استفاده از منطق فازی، بهره‌های کنترل کننده به صورت آنلاین و هوشمند تنظیم شوند و عملکرد دینامیکی سیستم بهبود یابد.

متدولوژی طراحی: از PID دستی تا محدوده دینامیک فازی

رویکرد اتخاذ شده در این پروژه، یک روش مهندسی عملی و مؤثر برای طراحی سیستم Fuzzy-PID است. به جای تعریف محدوده بهره‌ها به صورت کاملاً تجربی، فرآیند در دو مرحله کلیدی انجام شد:

تنظیم دستی کنترل کننده PID به عنوان نقطه شروع

ابتدا، یک کنترل کننده PID استاندارد برای هر یک از محورهای وضعیت (Roll, Pitch, Yaw) به صورت دستی و با دقت تنظیم (*Tune*) شد. هدف از این مرحله، یافتن یک مجموعه بهره بهینه بود که بهترین عملکرد ممکن را با یک کنترل کننده خطی ارائه دهد. این بهره‌های بهینه به عنوان نقطه مرکزی یا مبدا (*Baseline*) برای سیستم فازی در نظر گرفته شدند. مقادیر نهایی به دست آمده از این فرآیند به شرح زیر است:

:*Roll* ۰

$$2.15 = kp_roll \quad ۰$$

$$2.5 = kd_roll \quad ۰$$

:*Pitch* ۰

$$2.9 = kp_pitch \quad ۰$$

$$3.1 = kd_pitch \quad ۰$$

:*Yaw* ۰

$$1 = kp_yaw \quad ۰$$

$$2.5 = kd_yaw \quad ۰$$

تعریف محدوده دینامیک برای تنظیم فازی

ایده اصلی در این مرحله، تبدیل بهره‌های ثابت به محدوده‌های دینامیک بود. به جای استفاده از یک مقدار ثابت، به سیستم فازی اجازه داده شد تا بهره‌ها را در یک بازه مشخص در اطراف نقطه بهینه دستی، تغییر دهد. این بازه‌ها به صورت درصدی از مقادیر مبدا تعریف شدند:

■ برای محورهای *Pitch* و *Roll*: یک محدوده نسبتاً دقیق ۵٪ بالاتر و ۱۰٪ پایین‌تر از مقدار بهینه انتخاب شد. این انتخاب نشان می‌دهد که تنظیم دستی برای این دو محور به نقطه بهینه نزدیک بوده و تنها به تنظیمات دقیق و جزئی نیاز دارد.

$$2.2575 = \text{Kpmax_roll} \quad 1.935 = \text{Kpmin_roll}$$

$$2.625 = \text{Kdmax_roll} \quad 2.25 = \text{Kdmin_roll}$$

$$3.045 = \text{Kpmax_pitch} \quad 2.61 = \text{Kpmin_pitch}$$

$$3.255 = \text{Kdmax_pitch} \quad 2.79 = \text{Kdmin_pitch}$$

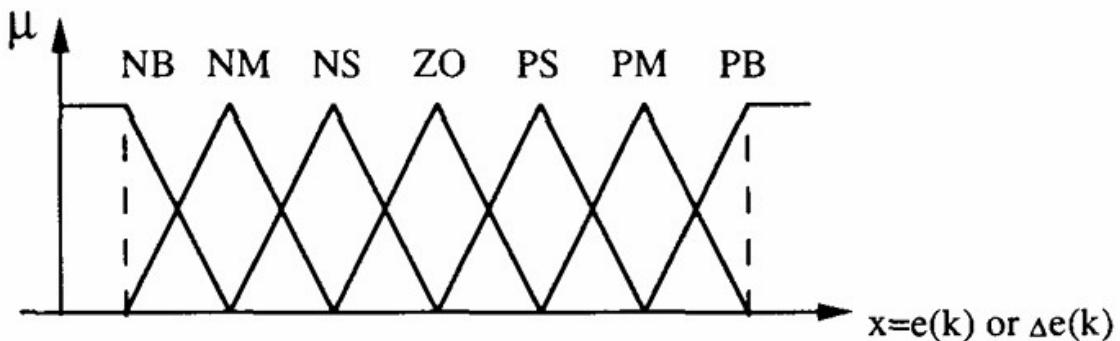
■ برای محور *Yaw*: با توجه به دینامیک کنترل و پاسخ ضعیفتر این محور در حالت *PID* دستی، یک محدوده وسیع‌تر ۲۰٪ بالاتر و ۲۰٪ پایین‌تر از مقدار بهینه در نظر گرفته شد. این کار به سیستم فازی اجازه می‌دهد تا تطبیق‌پذیری و اصلاحات بزرگ‌تری را برای پایدارسازی این محور اعمال کند.

$$1.2 = \text{Kpmax_yaw} \quad 0.8 = \text{Kpmin_yaw}$$

$$3.0 = \text{Kdmax_yaw} \quad 2.25 = \text{Kdmin_yaw}$$

این متدولوژی، پایداری حاصل از یک کنترل کننده خوش‌تنظیم را با انعطاف‌پذیری و هوشمندی منطق فازی ترکیب می‌کند.

توابع عضویت برای ورودی‌ها (خطا و مشتق خطای)



شکل ۳۴.۴: توابع عضویت برای خطای و مشتق خطای

این نمودار، نحوه **فازی‌سازی** (*Fuzzification*) ورودی‌های سیستم را تعریف می‌کند. ورودی‌های این کنترل کننده، خطای فعلی سیستم ($e(k)$) و تغییرات خطای ($\Delta e(k)$) هستند.

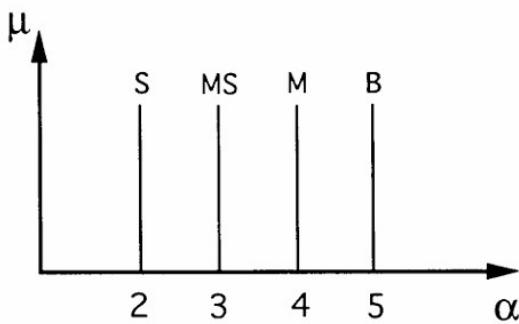
■ هدف: هدف این بخش، تبدیل یک مقدار عددی و دقیق (مثلاً خطای . درجه) به مفاهیم کیفی و زبانی است که برای انسان قابل درک باشد.

■ اجزا: این نمودار از هفت تابع عضویت مثلثی تشکیل شده است که هر کدام یک مفهوم زبانی را نمایندگی می‌کنند:

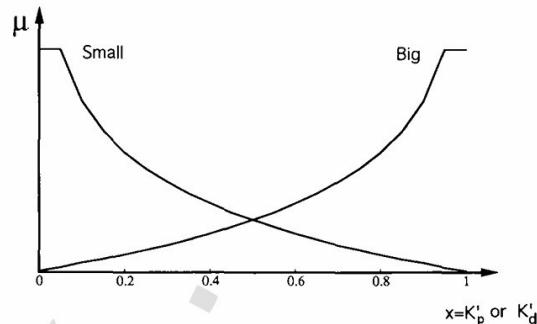
- منفی بزرگ (NB)
- منفی متوسط (NM)
- منفی کوچک (NS)
- تقریباً صفر (ZO)
- مثبت کوچک (PS)
- مثبت متوسط (PM)
- مثبت بزرگ (PB)

□ نحوه کار: هر مقدار ورودی می‌تواند به یک یا چند مجموعه از این مفاهیم تعلق داشته باشد. برای مثال، یک مقدار خطای کوچک ممکن است ۷۰ درصد به مجموعه "ZO" و ۳۰ درصد به مجموعه "PS" تعلق داشته باشد. این فرآیند به سیستم اجازه می‌دهد تا با عدم قطعیت کار کرده و استنتاج‌های نرمی انجام دهد.

توابع عضویت برای خروجی‌ها



شکل ۳۶.۴: توابع عضویت برای α



شکل ۳۵.۴: توابع عضویت برای K'_p و K'_d

این دو نمودار، مفاهیم زبانی را به مقادیر عددی و دقیق برای خروجی کنترل کننده تبدیل می‌کنند. این فرآیند بخشی از مرحله فازی‌زدایی (*Defuzzification*) است.

توابع عضویت برای بهره‌های K'_p و K'_d

این نمودار برای تعیین مقادیر بهره تناسبی (K'_p) و بهره مشتقی (K'_d) به کار می‌رود.

□ هدف: پس از اینکه موتور استنتاج فازی بر اساس قوانین تصمیم گرفت که یک بهره باید «کوچک» یا «بزرگ» باشد، این نمودار مقدار عددی دقیق آن را مشخص می‌کند.

□ اجزا: این نمودار شامل دو تابع عضویت غیرخطی به نام‌های *Small* (کوچک) و *Big* (بزرگ) است. این دو تابع مکمل یکدیگر هستند؛ یعنی در هر نقطه، مجموع درجات عضویت آنها تقریباً برابر یک است.

توابع عضویت برای پارامتر α

این نمودار برای تعیین مقدار پارامتر α استفاده می‌شود که برای تنظیم بهره انتگرال به کار می‌رود.

■ هدف: این نمودار یک روش بسیار ساده و محاسباتی برای تعیین خروجی ارائه می‌دهد.

■ اجزا: این نمودار از توابع عضویت تک‌مقداری (*Singleton*) تشکیل شده است. در این حالت، هر مفهوم زبانی به یک مقدار عددی واحد و مشخص نگاشت داده می‌شود:

S (کوچک): مقدار ۲

MS (متوسط کوچک): مقدار ۳

M (متوسط): مقدار ۴

B (بزرگ): مقدار ۵

■ نحوه کار: اگر قوانین فازی نتیجه بگیرند که α باید "متوسط" باشد، خروجی بدون هیچ محاسبه پیچیده‌ای مستقیماً برابر با ۴ خواهد بود. این روش به دلیل سادگی و کارایی بالا در سیستم‌های کنترلی بسیار رایج است.

جداول قوانین فازی: مغز متفکر کنترل کننده

		$\Delta e(k)$						
		NB	NM	NS	ZO	PS	PM	PB
$e(k)$	NB	B	B	B	B	B	B	B
	NM	S	B	B	B	B	B	S
	NS	S	S	S	B	B	S	S
	ZO	S	S	S	B	S	S	S
	PS	S	S	B	B	B	S	S
	PM	S	B	B	B	B	B	S
	PB	B	B	B	B	B	B	B

شکل ۴.۳۷: قوانین تنظیم فازی برای K'_p

		$\Delta e(k)$						
		NB	NM	NS	ZO	PS	PM	PB
$e(k)$	NB	S	S	S	S	S	S	S
	NM	B	B	S	S	S	B	B
	NS	B	B	B	S	B	B	B
	ZO	B	B	B	B	B	B	B
	PS	B	B	B	S	B	B	B
	PM	B	B	S	S	S	B	B
	PB	S	S	S	S	S	S	S

شکل ۳۸.۴: قوانین تنظیم فازی برای K'_d

		$\Delta e(k)$						
		NB	NM	NS	ZO	PS	PM	PB
$e(k)$	NB	2	2	2	2	2	2	2
	NM	3	3	2	2	2	3	3
	NS	4	3	3	2	3	3	4
	ZO	5	4	3	3	3	4	5
	PS	4	3	3	2	3	3	4
	PM	3	3	2	2	2	3	3
	PB	2	2	2	2	2	2	2

شکل ۳۹.۴: قوانین تنظیم فازی برای α

این سه جدول، در واقع دانش و استراتژی یک اپراتور متخصص را در قالب مجموعه‌ای از قوانین "اگر-آنگاه" پیاده‌سازی می‌کنند. هر خانه از جدول نماینده یک قانون برای یک وضعیت مشخص از سیستم است.

منطق تنظیم بهره تناسبی (K'_p) و مشتقی (K'_d)

این دو جدول یک استراتژی کنترلی مکمل را اجرا می‌کنند:

■ جدول K'_p : منطق این جدول تهاجمی و سریع است. هدف اصلی آن، رساندن سریع سیستم به نقطه تنظیم است. به همین دلیل، هرگاه خطاب بزرگ باشد (سطرهای NB و PB) یا سیستم در حال دور شدن از نقطه تنظیم باشد، بهره تناسبی را بزرگ (B) انتخاب می‌کند تا نیروی اصلاحی قوی اعمال شود.

■ جدول K'_d : منطق این جدول پایدارکننده و میراکننده است. هدف آن جلوگیری از نوسان و فراجهش است. بنابراین، هرگاه سیستم به نقطه تنظیم نزدیک می‌شود (خانه‌های مرکزی جدول)، بهره مشتقی را بزرگ (B) انتخاب می‌کند تا مانند یک ترمز عمل کرده و سیستم را آرام کند. بر عکس، زمانی که خطاب بزرگ است، بهره مشتقی را کوچک (S) نگه می‌دارد تا مانع از حرکت سریع اولیه سیستم نشود.

این دو جدول با همکاری یکدیگر، تعادلی هوشمند بین سرعت پاسخ و پایداری ایجاد می‌کنند.

منطق تنظیم پارامتر α

این جدول وظیفه مدیریت خطا حالت ماندگار و جلوگیری از انباشت خطا (**Integral Windup**) را بر عهده دارد. پارامتر α با زمان انتگرال رابطه معکوس دارد (هرچه α کوچکتر، انتگرال گیری قوی‌تر). استراتژی جدول به این صورت است:

■ **انتگرال گیری قوی (مقدار α کم):** زمانی که خطا وجود دارد اما سیستم تقریباً آرام است (مثلاً $e(k) = NS$ و $ZO = e(k)\Delta$ ، مقدار α کوچک (مثلاً ۲) است تا هرگونه خطای باقیمانده به سرعت حذف شود).

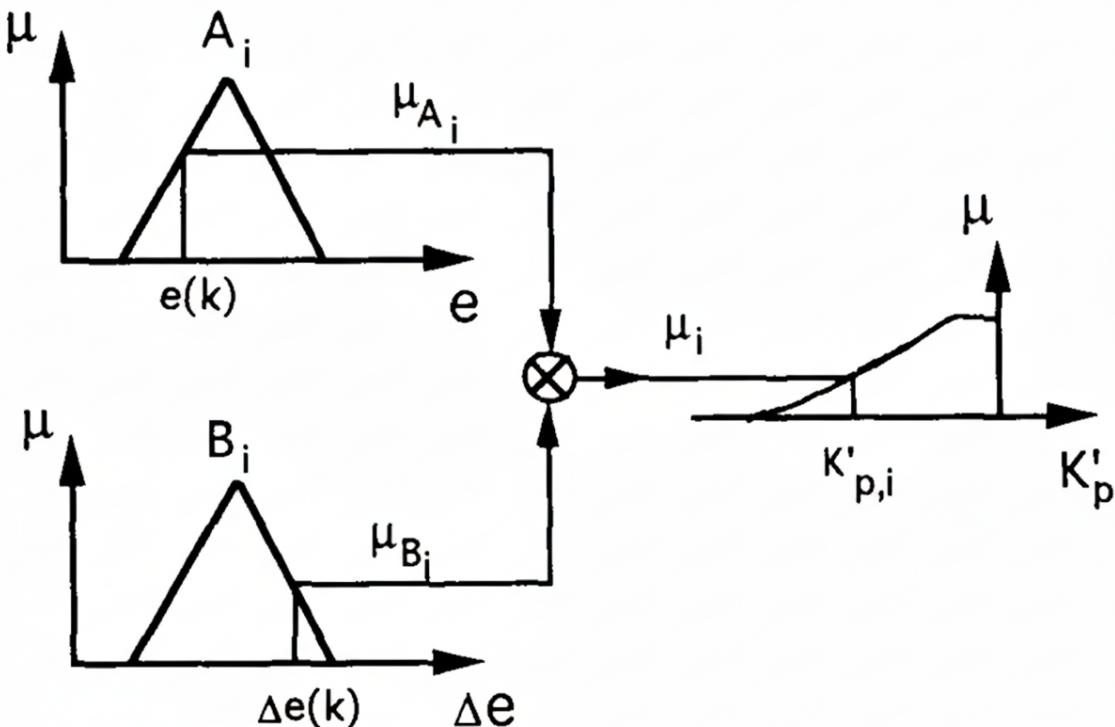
■ **انتگرال گیری ضعیف (مقدار α زیاد):** زمانی که خطا صفر است اما سیستم با سرعت در حال حرکت است ($e(k) \neq 0$ و $NB/PB = e(k)\Delta$)، مقدار α بزرگ (۵) انتخاب می‌شود. این یک حرکت هوشمندانه برای جلوگیری از فرجهش است؛ زیرا از جمع شدن بی‌مورد خطای انتگرال در زمانی که سیستم در حال عبور سریع از نقطه تنظیم است، جلوگیری می‌کند.

فرآیند استنتاج: نحوه اجرای یک قانون

این نمودار نشان می‌دهد که چگونه یک قانون واحد از جداول بالا در عمل اجرا می‌شود. این فرآیند قلب محاسباتی منطق فازی است:

۱. **اندازه‌گیری:** مقادیر عددی و دقیق خطا (e) و تغییرات خطا (Δe) از سیستم خوانده می‌شوند.
۲. **تعیین درجه تعلق:** با استفاده از توابع عضویت ورودی، مشخص می‌شود که این مقادیر عددی تا چه حد (با درجه عضویت μ) به مفاهیم زبانی مانند "خطای مثبت کوچک" یا "تغییر خطای صفر" تعلق دارند.
۳. **محاسبه "قدرت آتش" قانون (μ):** درجه‌های تعلق ورودی‌ها با هم ترکیب می‌شوند (معمولًاً با عملگر AND که با ضرب یا مینیمم پیاده‌سازی می‌شود). خروجی این مرحله (μ_i) نشان می‌دهد که قانون فعلی تا چه اندازه با وضعیت کنونی سیستم مرتبط است.
۴. **شكل دهی به خروجی:** در نهایت، "قدرت آتش" به دست آمده، خروجی آن قانون را شکل می‌دهد. برای مثال، اگر قدرت یک قانون ۰.۷ باشد و نتیجه آن "بهره بزرگ" باشد، این قانون یک نسخه محدود شده از تابع عضویت "بزرگ" با ارتفاع ۰.۷ را به خروجی نهایی اضافه می‌کند.

در نهایت، خروجی‌های تمام قوانینی که فعال شده‌اند با هم تجمع شده و به یک فرمان کنترلی واحد تبدیل می‌شوند.



شکل ۴۰.۴: فرآیند استنباط یک قانون فازی

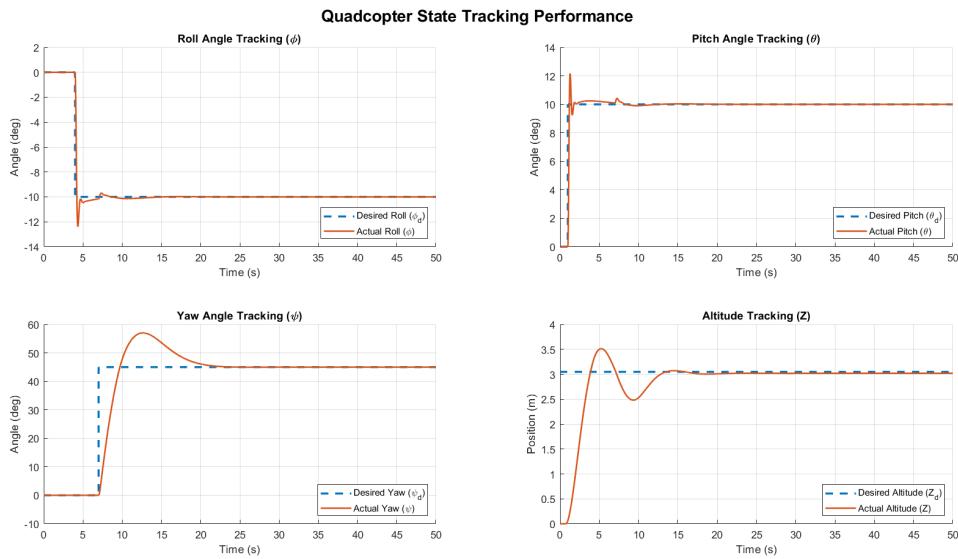
معماری و پیاده‌سازی سیستم استنتاج فازی

سیستم کنترل کننده Fuzzy-PID طراحی شده بر اساس یک موتور استنتاج فازی (Fuzzy Inference System) عمل می‌کند که ساختار آن در گزارش به تفصیل شرح داده شده است.

■ **ورودی‌های سیستم:** خطای وضعیت ($e(k)$) و تغییرات خطای ($\Delta e(k)$) به عنوان ورودی‌های عددی به سیستم داده می‌شوند.

■ **مотор استنتاج:** این موتور با استفاده از یک پایگاه دانش شامل توابع عضویت پایگاه قوانین اگر-آنگاه ($IF - THEN$)، ورودی‌ها را پردازش می‌کند.

■ **خروجی‌های سیستم:** خروجی موتور فازی، مقادیر بهینه و لحظه‌ای برای بهره‌های نرمال شده K'_d و K'_p و پارامتر α است. این مقادیر سپس به بهره‌های واقعی در محدوده‌های دینامیکی که در بخش قبل تعریف شد، نگاشت داده می‌شوند.



شکل ۴۱.۴: نتیجه تست کنترل کننده وضعیت برای کنترل کننده PID-Fuzzy

تحلیل نتایج و نتیجه‌گیری

نتایج شبیه‌سازی کنترل کننده Fuzzy-PID یک بهبود چشمگیر و بنیادین را نسبت به کنترل کننده PID استاندارد نشان می‌دهد.

□ کاهش چشمگیر فراجهش و نوسانات: پاسخ سیستم در تمام محورها به یک پاسخ بهینه و به خوبی میرا شده تبدیل شده است. فراجهش‌های بزرگ و لرزش‌های طولانی به طور کامل حذف شده‌اند.

□ افزایش سرعت پاسخ: زمان نشت (Settling Time) به شدت کاهش یافته و سیستم با سرعت بسیار بیشتری به مقدار مطلوب خود همگرا می‌شود.

این بهبود عملکرد، موفقیت رویکرد اتخاذ شده را اثبات می‌کند. سیستم Fuzzy-PID توانست با تنظیم هوشمند و لحظه‌ای بهره‌ها بر اساس شرایط خطأ، ضعف‌های کنترل کننده خطی را برطرف کرده و یک پاسخ کنترلی سریع، دقیق و پایدار را برای وضعیت کوادراتور فراهم آورد. این استراتژی، بسته قابل اعتماد برای آزمون‌های پیچیده‌تر و پیاده‌سازی نهایی روی سخت‌افزار واقعی ایجاد می‌کند.

۵.۴ مقایسه بین PID و Fuzzy-PID

جدول ۱.۴: مقایسه معیارهای عملکرد برای کنترل کننده‌های Fuzzy – PID و PID

Fuzzy-PID		PID		
ITAE	MSE	ITAE	MSE	Axis
21.1218	0.2267	757.2181	2.4780	(ϕ) Roll
13.6834	0.2003	693.7909	2.0331	(θ) Pitch
1413.7	48.0384	6102.5	121.9768	(ψ) Yaw

این جدول به مقایسه عملکرد دو نوع کنترل کننده مختلف می‌پردازد: کنترل کننده (PID) و کنترل کننده فازی (Fuzzy-PID). هدف این مقایسه، ارزیابی دقت و کارایی این دو کنترل کننده در کنترل یک سیستم با سه محور حرکتی Roll و Pitch و Yaw است.

برای این ارزیابی از دو معیار سنجش خطأ استفاده شده است:

□ **ITAE (Integral of Time-weighted Absolute Error)**: انتگرال زمان در قدر مطلق خطأ

□ **MSE (Mean Squared Error)**: میانگین مربعات خطأ

تحلیل نتایج

همانطور که در جدول مشاهده می‌شود، مقادیر هر دو معیار خطأ (MSE و ITAE) برای کنترل کننده Fuzzy-PID و PID برابر هستند. در هر سه محور به طور چشمگیری کمتر از مقادیر مربوط به کنترل کننده PID است.

□ **مقادیر برای ITAE** نشان می‌دهد که کنترل کننده Fuzzy-PID نه تنها خطای کمتری دارد، بلکه این خطأ را در زمان بسیار کوتاه‌تری به صفر نزدیک می‌کند (پاسخ سریع‌تر و میرایی‌تر).

□ **مقادیر برای MSE** به این معناست که خطاهای بزرگ و نوسانات شدید در کنترل کننده Fuzzy-PID بسیار کمتر از نوع استاندارد آن است.

نتیجه‌گیری کلی از جدول این است که کنترل کننده Fuzzy-PID عملکرد بسیار بهتری، با دقت بالاتر، سرعت پاسخ سریع‌تر و پایداری بیشتر نسبت به کنترل کننده PID استاندارد در این کاربرد خاص از خود نشان داده است.

فرمول‌های معیارهای خطأ

در ادامه، فرمول‌های ریاضی برای دو معیار استفاده شده در جدول آورده شده است. در این فرمول‌ها، $e(t)$ نمایانگر تابع خطأ (اختلاف بین مقدار مطلوب و مقدار واقعی) در زمان t است.

میانگین مربعات خطأ (MSE)

این معیار، میانگین توان دوم خطاهای را محاسبه می‌کند. به دلیل وجود توان دوم، این معیار به خطاهای بزرگ حساسیت بیشتری نشان می‌دهد و جریمه سنگین‌تری برای آن‌ها در نظر می‌گیرد. مقدار کمتر MSE به معنای خطاهای کوچکتر و نوسانات کمتر است.

$$MSE = \frac{1}{T} \int_0^T [e(t)]^2 dt$$

در فرمول بالا، T بازه زمانی شبیه‌سازی یا عملکرد سیستم است.

انتگرال زمان در قدرمطلق خطأ ($ITAE$)

این معیار، انتگرال قدرمطلق خطأ را در زمان ضرب کرده و محاسبه می‌کند. ویژگی مهم $ITAE$ این است که به خطاهایی که برای مدت طولانی باقی می‌مانند، وزن بیشتری می‌دهد. بنابراین، یک کنترل کننده با $ITAE$ پایین، سیستمی است که خطای خود را به سرعت حذف می‌کند.

$$ITAE = \int_0^\infty t|e(t)|dt$$

فصل ۵

آزمون روی سخت افزار واقعی

۱.۵ معرفی کلی سنسور MPU-9250

سنسور $MPU - 9250$ یک واحد اندازه‌گیری اینرسی^۱ توسعه یافته توسط شرکت اینون‌سنس^۲ است که با ادغام سه نوع حسگر در یک تراشه، در بسیاری از کاربردهای ناوبری و ردیابی حرکتی به کار می‌رود. این تراشه دارای نه درجه آزادی ($9 - DOF$) است که شامل موارد زیر است:

■ شتاب‌سنجدۀ محوره^۳

■ ژیروسکوپ سه‌محوره^۴

■ مغناطیس‌سنجدۀ محوره^۵

ترکیب این سه حسگر امکان اندازه‌گیری شتاب خطی، نرخ چرخش زاویه‌ای و میدان مغناطیسی زمین را فراهم می‌سازد و اطلاعات جامعی درباره وضعیت حرکتی و جهت‌گیری جسم در فضای سه‌بعدی ارائه می‌دهد.

مشخصات اصلی

شتاب‌سنجد

■ محدوده دینامیکی: $\pm 2g, \pm 4g, \pm 8g, \pm 16g$

■ دقت خروجی: ۱۶ بیت

■ حساسیت در حالت LSB/g : معادل $\pm 2g$ 16384

¹Inertial Measurement Unit (IMU)

²InvenSense (TDK)

³Accelerometer

⁴Gyroscope

⁵Magnetometer, AK8963

ژیروسکوپ

■ محدوده دینامیکی: $\pm 250, \pm 500, \pm 1000, \pm 2000^{\circ}/s$

■ دقت خروجی: ۱۶ بیت

■ حساسیت در حالت $LSB//s ۱۳۱$: معادل $\pm 250^{\circ}/s$

مغناطیس سنج

■ محدوده اندازه گیری: $\pm 4800 \mu T$

■ دقت خروجی: ۱۶ یا ۱۴ بیت

■ حساسیت: حدود $LSB/\mu T 0.6$

سایر مشخصات

■ رابطه ای ارتباطی: گذرگاه I^2C^1 تا $kH z ۴۰۰$ و SPI^2 تا $MHz ۲۰$

■ ولتاژ تغذیه: ۳.۳ تا ۳.۶ ولت (ممکن است ۳.۳ ولت)

■ دمای کاری: $+85^{\circ}C$ تا $-40^{\circ}C$

■ جریان مصرفی: حدود 5.۳ میلی آمپر در حالت فعال

اجزای عملکردی

شتاپ سنج

وظیفه شتاب سنج، اندازه گیری شتاب خطی در راستای سه محور است. با توجه به وجود شتاب گرانش زمین، این حسگر امکان محاسبه زاویه های $Pitch^4$ و $Roll^3$ را فراهم می کند.

ژیروسکوپ

ژیروسکوپ نرخ تغییر زاویه حول سه محور را اندازه گیری می کند. با انتگرال گیری از این داده ها، تغییر زاویه نسبی محاسبه می شود. با این حال، به دلیل خطای تجمعی^۵ لازم است داده های ژیروسکوپ با اطلاعات شتاب سنج و مغناطیس سنج ترکیب گردد.

¹ Inter-Integrated Circuit (I^2C)

² Serial Peripheral Interface (SPI)

³ Roll

⁴ Pitch

⁵ Drift

مغناطیس سنج

مغناطیس سنج میدان مغناطیسی محیط را حس می کند و برای تعیین جهت شمال مغناطیسی به کار می رود. این خروجی مرجع مطلق برای زاویه ^1Yaw ^۱ محسوب می شود.

ترکیب داده ها

به منظور دستیابی به برآورد پایدار و دقیق از وضعیت و جهت گیری سه بعدی، داده های سه حسگر ترکیب می شوند. این فرآیند با عنوان «ترکیب داده ها»^۲ شناخته می شود.

■ شتاب سنج + ژیروسکوپ → ثبت Pitch و Roll

■ ژیروسکوپ + مغناطیس سنج → ثبت Yaw

الگوریتم های متداول در این حوزه شامل موارد زیر هستند:

■ فیلتر مکمل^۳

■ فیلتر ماهونی^۴

■ فیلتر مد گوییک^۵

■ فیلتر کالمان^۶

فیلتر مکمل

فیلتر مکمل^۷ یکی از ساده ترین روش ها برای ترکیب داده های شتاب سنج و ژیروسکوپ است. ایده اصلی این فیلتر بر این اساس است که ژیروسکوپ در کوتاه مدت دقیق ولی در بلند مدت دچار رانش می شود، در حالی که شتاب سنج در بلند مدت دقیق اما در کوتاه مدت نویزی است. بنابراین با ترکیب وزنی این دو منبع داده، زوایای پایدار و نسبتاً دقیق محاسبه می شوند. معادله کلی فیلتر مکمل به شکل زیر است:

$$\theta(t) = \alpha \left(\theta(t-1) + \dot{\theta}_{gyro} \cdot \Delta t \right) + (1 - \alpha) \theta_{acc}$$

که در آن θ زاویه تخمینی، $\dot{\theta}_{gyro}$ نرخ زوایه ای ژیروسکوپ و θ_{acc} زاویه محاسبه شده از شتاب سنج است. ضریب α معمولاً بین ۰.۹۸ تا ۰.۹۹ انتخاب می شود.

^۱Yaw

^۲Sensor Fusion

^۳Complementary Filter

^۴Mahony Filter

^۵Madgwick Filter

^۶Kalman Filter

^۷Complementary Filter

فیلتر ماهونی

فیلتر ماهونی^۱ یکی از الگوریتم‌های رایج در ترکیب داده‌های حسگرهای اینرسی است که به‌ویژه برای سنسورهای ۹ درجه آزادی (شتاب‌سنج، ژیروسکوپ و مغناطیس‌سنج) به کار می‌رود. اساس این فیلتر بر استفاده از کنترل کننده تناسبی^۲ انگرالی^۳ برای کاهش خطای ژیروسکوپ است. در این روش، جهت‌گیری جسم به صورت یک کواترنیون^۴ نمایش داده می‌شود. کواترنیون یک نمایش ریاضی از دوران سه‌بعدی است که نسبت به زاویه‌های اویلر پایدارتر بوده و مشکل قفل گیمبال^۵ در آن رخ نمی‌دهد. ایده اصلی فیلتر به این صورت است: ابتدا نرخ زاویه‌ای از ژیروسکوپ خوانده می‌شود و برای به‌روزرسانی وضعیت جسم به کار می‌رود. سپس بردارهای مرجع (مانند شتاب جاذبه یا میدان مغناطیسی زمین) با بردارهای اندازه‌گیری شده مقایسه می‌شوند و اختلاف آن‌ها به عنوان یک خطا محاسبه می‌گردد. این خطا در یک حلقه تناسبی^۶ انگرالی اعمال می‌شود و نرخ زاویه‌ای اصلاح شده به دست می‌آید. با استفاده از این نرخ اصلاح شده، کواترنیون وضعیت جسم به‌روزرسانی می‌شود.

معادله کلی این فیلتر به صورت زیر بیان می‌شود:

$$\dot{q} = \frac{1}{2}q \otimes \omega_{corr}$$

که در آن q کواترنیون وضعیت و \otimes ضرب کواترنیون است. نرخ زاویه‌ای اصلاح شده نیز از رابطه زیر به دست می‌آید:

$$\omega_{corr} = \omega_{gyro} + K_p e + K_i \int e dt$$

در این رابطه:

\square ω_{gyro} : نرخ زاویه‌ای خام ژیروسکوپ

\square e : خطای ناشی از اختلاف بردارهای اندازه‌گیری شده و بردارهای مرجع

\square K_p, K_i : ضرایب تناسبی و انگرالی برای تنظیم دقت و پایداری

خروجی نهایی فیلتر ماهونی عموماً به صورت زاویه‌های اویلر Roll، Pitch و Yaw ارائه می‌شود و در مقایسه با روش‌های ساده‌تر (مانند فیلتر مکمل) پایداری و دقت بیشتری دارد، در حالی که از نظر محاسباتی سبک‌تر از فیلتر کالمون است.

فیلتر مدگوییک

فیلتر مدگوییک^۷ یکی از روش‌های محبوب برای ترکیب داده‌های حسگرهای اینرسی است که به دلیل سرعت همگرایی بالا و دقت مناسب در بسیاری از پروژه‌های عملی استفاده می‌شود. ایده اصلی این فیلتر بر پایه‌ی یک بهینه‌سازی ریاضی قرار دارد. در این الگوریتم، وضعیت جسم به صورت یک کواترنیون نمایش داده می‌شود و یکتابع خطای تعریف می‌شود که نشان‌دهنده اختلاف بین بردارهای حسگر (شتاب‌سنج و مغناطیس‌سنج) و بردارهای پیش‌بینی شده بر اساس کواترنیون است. برای به حداقل رساندن این خطای فیلتر مدگوییک از روش گرادیان نزولی^۸ استفاده می‌کند. ایده این روش این است که اگر تابع خطای $f(q)$ داشته باشیم، جهت گرادیان $\nabla f(q)$ جهتی است که سریع‌ترین افزایش مقدار خطای را نشان می‌دهد.

¹Mahony Filter

²Proportional–Integral (PI) Controller

³Quaternion

⁴Gimbal Lock

⁵Madgwick Filter

⁶Gradient Descent

بنابراین با حرکت در خلاف جهت گرادیان، مقدار خطای کاهش می‌یابد. فیلتر مدگوییک با استفاده از این اصل، در هر مرحله تخمینی از گرادیان خطای را محاسبه کرده و یک تصحیح کوچک به نرخ زاویه‌ای ژیروسکوپ اعمال می‌کند.
معادله کلی به روزرسانی کواترنیون در این فیلتر به صورت زیر است:

$$\dot{q} = \frac{1}{2}q \otimes \omega_{gyro} - \beta \frac{\nabla f(q, a, m)}{\|\nabla f(q, a, m)\|}$$

در این رابطه:

q : کواترنیون وضعیت

ω_{gyro} : نرخ زاویه‌ای ژیروسکوپ

a, m : بردارهای شتاب‌سنج و مغناطیس‌سنج

$\nabla f(q, a, m)$: گرادیان تابع خطای

β : ضریب همگرایی که سرعت اصلاح خطای کنترل می‌کند

مزیت اصلی این روش آن است که حتی با نرخ نمونه‌برداری پایین (مثلاً $50 Hz$) نیز به سرعت همگرا می‌شود و رانش ژیروسکوپ را به خوبی کاهش می‌دهد. به همین دلیل، فیلتر مدگوییک انتخابی مناسب برای کاربردهای رباتیک سبک و دستگاه‌های پوشیدنی محسوب می‌شود.

فیلتر کالمن

فیلتر کالمن^۱ یک الگوریتم پیشرفته تخمین حالت است که بر پایه مدل دینامیکی سیستم و داده‌های اندازه‌گیری نویزی عمل می‌کند. این فیلتر در دو مرحله اجرا می‌شود: مرحله پیش‌بینی^۲ و مرحله به روزرسانی^۳. در مرحله پیش‌بینی، حالت بعدی بر اساس مدل دینامیکی تخمین زده می‌شود. در مرحله به روزرسانی، داده‌های حسگر وارد شده و تخمین اصلاح می‌گردد.
معادلات کلی فیلتر کالمن عبارت‌اند از:

$$\hat{x}_{k|k-1} = F\hat{x}_{k-1|k-1} + Bu_k$$

$$P_{k|k-1} = FP_{k-1|k-1}F^T + Q$$

$$K_k = P_{k|k-1}H^T(HP_{k|k-1}H^T + R)^{-1}$$

$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k(z_k - H\hat{x}_{k|k-1})$$

$$P_{k|k} = (I - K_k H)P_{k|k-1}$$

که در آن x بردار حالت، F ماتریس انتقال، Q کوواریانس نویز فرایند، R کوواریانس نویز اندازه‌گیری، H ماتریس مشاهده و K بهره کالمن است. این فیلتر با وجود دقت بالا، از نظر محاسباتی سنتگین‌تر از سایر روش‌هاست.

¹Kalman Filter

²Prediction

³Update

۲.۵ شرح کد MPU-9250

سنسور MPU-9250 به عنوان قلب سیستم ادراک در این پروژه عمل می‌کند. این واحد اندازه‌گیری اینرسی (IMU) وظیفه ارائه اطلاعات دقیق و لحظه‌ای از وضعیت زاویه‌ای کوادروتور را بر عهده دارد تا کنترل کننده بتواند تصمیمات اصلاحی صحیحی اتخاذ نماید. کد درایور این سنسور از دو فایل اصلی تشکیل شده است: فایل سرآیند (mpu9250.h) که ساختارها و توابع عمومی را تعریف می‌کند و فایل منبع (mpu9250.c) که منطق اصلی پیاده‌سازی را در خود جای داده است.

۱.۲.۵ فایل سرآیند (mpu9250.h)

این فایل به عنوان **واسطه عمومی (Public Interface)** کتابخانه عمل می‌کند و تعاریف کلیدی مورد نیاز برای استفاده از درایور را فراهم می‌آورد.

ساختارهای داده (Data Structures)

■ **MPU9250_t**: این ساختار اصلی، تمامی داده‌های مرتبه با سنسور را در خود تجمعی می‌کند. این داده‌ها شامل مقادیر خام ۱۶ بیتی از شتاب‌سنج و ژیروسکوپ، مقادیر تبدیل شده به واحدهای فیزیکی (g برای شتاب و Roll، Pitch برای سرعت زاویه‌ای)، دمای سنسور، و مهم‌تر از همه، خروجی‌های نهایی فیلتر شده یعنی زوایای Yaw و Gz_offset نیز برای نگهداری خطای بایاس کالیبره شده ژیروسکوپ محور Z در نظر گرفته شده است.

■ **Kalman_t**: این ساختار تمامی پارامترها و متغیرهای حالت مورد نیاز برای یک نمونه از **فیلتر کالمون** را تعریف می‌کند. این پارامترها شامل ماتریس‌های کوواریانس نویز (R_measure, Q_bias, Q_angle) برای تنظیم فیلتر، و متغیرهای حالت (bias, angle) هستند که توسط فیلتر تخمین زده می‌شوند.

اعلان توابع (Function Prototypes)

در این بخش، توابع عمومی که از خارج این فایل قابل فراخوانی هستند، اعلام شده‌اند. این توابع عبارتند از:

■ **MPU9250_Init**: برای راهاندازی اولیه و پیکربندی سنسور.

■ **MPU9250_Read_Gyro** و **MPU9250_Read_Accel**: برای خواندن داده‌های خام از هر سنسور به صورت مجزا.

■ **MPU9250_Calibrate_Gyro**: برای اجرای روتین کالیبراسیون ژیروسکوپ.

■ **MPU9250_Read_All**: تابع اصلی که تمام داده‌های سنسور را خوانده، فیلتر کالمون را اجرا کرده و زوایای نهایی را محاسبه می‌کند.

■ **Kalman_getAngle**: پیاده‌سازی الگوریتم فیلتر کالمون.

۲.۲.۵ فایل منبع (mpu9250.c)

این فایل شامل پیاده‌سازی و منطق اصلی توابع اعلام شده در فایل سرآیند است.

MPU9250_Init()

این تابع وظیفه آماده سازی سنسور برای کار را بر عهده دارد.

۱. بروزرسی شناسه دستگاه: ابتدا با خواندن رجیستر WHO_AM_I، از وجود و صحت ارتباط با سنسور اطمینان حاصل می شود.
۲. فعال سازی: سنسور با نوشتمن مقدار صفر در رجیستر مدیریت توان (PWR_MGMT_1_REG) از حالت خواب بیدار می شود.
۳. پیکربندی: نرخ نمونه برداری، و محدوده های اندازه گیری شتاب سنج ($\pm 2 \text{ g}$) و ژیروسکوپ ($\pm 250 \text{ dps}$) تنظیم می شوند. این مقادیر حساسیت حداکثری را برای سیستم فراهم می کنند.

MPU9250_Read_Gyro() و MPU9250_Read_Accel()

این توابع ۶ بایت داده خام را از طریق I2C از رجیستر های مربوطه می خوانند. سپس داده های خام ۱۶ بیتی را با تقسیم بر ضریب حساسیت (Scale Factor) به واحد های فیزیکی استاندارد تبدیل می کنند (۰.۱۶۳۸۴۰.۰ برای شتاب سنج و ۰.۱۳۱.۰ برای ژیروسکوپ که متناسب با محدوده های تنظیم شده در تابع Init است).

MPU9250_Calibrate_Gyro()

این تابع با دریافت تعداد مشخصی نمونه از ژیروسکوپ محور Z در حالت سکون و محاسبه میانگین آنها، خطای بایاس (Gz_offset) را به دست می آورد. این مقدار برای محاسبه دقیق تر زاویه Yaw ضروری است.

MPU9250_Read_All()

این تابع اصلی ترین بخش درایور است و در هر بار فراخوانی، یک چرخه کامل از خواندن تا پردازش را انجام می دهد:

۱. خواندن یک پارچه: ۱۴ بایت داده مربوط به شتاب سنج، دما و ژیروسکوپ را در یک تراکنش I2C می خواند تا سرعت و کارایی افزایش یابد.
۲. تبدیل واحد: داده های خام را به واحد های فیزیکی تبدیل می کند.
۳. محاسبه زوایای خام: با استفاده از داده های شتاب سنج، یک تخمین اولیه از زوایای Roll و Pitch از طریق روابط مثلثاتی (atan2 و atan) محاسبه می شود. این تخمین در بلند مدت دقیق اما در کوتاه مدت به نویز حساس است.
۴. اجرای فیلتر کالمن: برای هر یک از زوایای Roll و Pitch، تابع Kalman_getAngle فراخوانی می شود. این فیلتر، تخمین زاویه ای مبتنی بر شتاب سنج (که پایدار اما نویزی است) را با داده های سرعت زاویه ای ژیروسکوپ (که در کوتاه مدت دقیق اما در بلند مدت دچار دریفت می شود) ترکیب کرده و یک خروجی بهینه، پایدار و با پاسخ سریع تولید می کند.
۵. محاسبه زاویه Yaw: زاویه Yaw از طریق انتگرال گیری از سرعت زاویه ای کالیبره شده (Gz - Gz_offset) در طول زمان (dt) محاسبه می شود. این روش یک تخمین نسبی از جهت گیری ارائه می دهد که در طول زمان دچار دریفت خواهد شد، اما برای آزمون های کوتاه در سکوی تست کافی است.

Kalman_getAngle()

این تابع، الگوریتم فیلتر کالمن یک بعدی را پیاده‌سازی می‌کند. عملکرد آن در دو مرحله خلاصه می‌شود:

۱. مرحله پیش‌بینی (**Prediction**): حالت بعدی زاویه بر اساس مقدار قبلی و سرعت زاویه‌ای ژیروسکوپ پیش‌بینی می‌شود.
۲. مرحله بهروزرسانی (**Update**): خطای بین مقدار پیش‌بینی شده و مقدار اندازه‌گیری شده توسط شتاب‌سنج محاسبه می‌شود. سپس فیلتر با استفاده از بهره کالمن (**Kalman Gain**)، حالت خود (زاویه و بایاس ژیروسکوپ) را اصلاح می‌کند تا به تخمین بهینه‌تری دست یابد.

۳.۵ شرح کد basic

این مژول، پایه‌ای ترین و حیاتی ترین بخش نرم‌افزار کنترل کننده پرواز را تشکیل می‌دهد. وظیفه اصلی آن فراهم آوردن یک بستر زمانی دقیق (**Timing Foundation**) برای کل سیستم و مدیریت متغیرهای سراسری وضعیت پرواز است. بدون یک زمان‌بندی دقیق، اجرای صحیح الگوریتم‌های کنترل، خواندن منظم داده‌های سنسور و عملکرد پایدار سیستم غیرممکن خواهد بود. این مژول نیز از دو فایل سرآیند و منبع تشکیل شده است.

۱.۳.۵ فایل سرآیند (basic.h)

این فایل، تعاریف عمومی و متغیرهای سراسری را که باید در سایر بخش‌های پروژه قابل دسترسی باشند، مشخص می‌کند.

□ کتابخانه‌ها و تعاریف اولیه: این فایل کتابخانه‌های استاندارد STM32 HAL مانند main.h و tim.h را فراخوانی می‌کند. همچنین تایمر TIM2 به عنوان منبع زمان‌سنج سیستم با نام SYSTICK_TIMER تعریف شده است.

□ متغیرهای خارجی (extern)

□ این دو متغیر برای نگهداری زمان سیستم به ترتیب با دقت میکروثانیه و میلی‌ثانیه تعریف شده‌اند و در سراسر پروژه قابل دسترسی هستند.

□ Altitude, Yaw, Pitch, Roll: این متغیرهای سراسری، آخرین مقادیر محاسبه شده برای وضعیت و ارتفاع کوادروتور را نگهداری می‌کنند. با تعریف آن‌ها به صورت extern، مژول‌های دیگر (مانند کنترل کننده یا فرستنده تله‌متری) می‌توانند به راحتی به این داده‌های کلیدی دسترسی داشته باشند.

□ اعلام توابع (Function Prototypes)

□ module_00_basic_init: تابع راه‌اندازی اولیه مژول.

□ module_00_basic_loop_handler: تابعی که در حلقه اصلی برنامه برای بهروزرسانی زمان فراخوانی می‌شود.

□ module_00_basic_tim_interrupt_handler: تابع مدیریت وقفه‌های تایمر.

□ getSystickIn_ms و getSystickIn_us: توابعی برای دریافت زمان فعلی سیستم با دقت‌های مختلف.

۲.۳.۵ فایل منبع (basic.c)

این فایل، منطق اصلی مربوط به پیاده‌سازی زمان‌سنج دقیق سیستم را در خود دارد.

متغیرها

متغیرهای سراسری مربوط به زمان در این فایل مقداردهی اولیه می‌شوند. متغیر `us_dt` نیز برای محاسبه و نگهداری فاصله زمانی بین هر بار اجرای حلقه اصلی برنامه تعریف شده است که برای الگوریتم‌های انتگرال‌گیر و مشتق‌گیر در کنترل‌کننده بسیار حیاتی است.

`module_00_basic_init()`

این تابع، تایمر TIM2 را در حالت وقفه (*Interrupt*) فعال کرده و شمارنده آن را صفر می‌کند. از این لحظه به بعد، تایمر شروع به شمارش می‌کند و با هر بار سرریز شدن (*overflow*), یک وقفه ایجاد می‌نماید.

ساختار زمان‌سنج ۶۴ بیتی

از آنجایی که تایمرهای STM32 معمولاً ۱۶ یا ۳۲ بیتی هستند، برای جلوگیری از سرریز شدن شمارنده در مدت زمان کوتاه، یک زمان‌سنج ۶۴ بیتی با دقت میکروثانیه به صورت نرم‌افزاری پیاده‌سازی شده است:

▪ **`module_00_basic_tim_interrupt_handler()`**: این تابع در پاسخ به وقفه سرریز تایمر TIM2 اجرا می‌شود. با هر بار اجراء، متغیر `us_period_overflow` یک واحد افزایش می‌یابد. این متغیر تعداد دفعاتی که شمارنده ۳۲ بیتی سرریز شده است را ثبت می‌کند.

▪ **`htim2.Instance->getSystickIn_us()`**: این تابع زمان دقیق سیستم را با ترکیب مقدار فعلی شمارنده تایمر CNT که بخش کمارزش ۳۲ بیتی است) و تعداد سرریزها (`us_period_overflow`) که بخش پرالرزش ۳۲ بیتی است) محاسبه می‌کند. این کار از طریق عملیات شیفت بیتی انجام شده و یک شمارنده یکپارچه ۶۴ بیتی با دقت میکروثانیه را شبیه‌سازی می‌کند.

`module_00_basic_loop_handler()`

این تابع در هر تکرار از حلقه اصلی برنامه (main loop) فراخوانی می‌شود. وظیفه اصلی آن محاسبه دقیق فاصله زمانی بین اجرای فعلی و قبلی حلقه است. این مقدار که در متغیر `us_dt` ذخیره می‌شود، همان پارامتر Δt است که در معادلات دیفرانسیل کنترل‌کننده برای انتگرال‌گیری از خطای محاسبه مشتق آن به کار می‌رود و دقت آن تأثیر مستقیمی بر پایداری و عملکرد سیستم کنترلی دارد.

۴.۵ شرح کد controller

این مژول، مغز متفکر سیستم کنترل پرواز است. وظیفه اصلی آن، پیاده‌سازی الگوریتم‌های کنترل حلقه بسته برای پایدارسازی وضعیت کوادراتور، ترکیب خروجی‌های کنترلی (Motor Mixing) و ارسال فرمان نهایی به موتورها از طریق سیگنال PWM است. این بخش بر اساس داده‌های وضعیت دریافتی از مژول basic و سنسور MPU-9250، تصمیمات اصلاحی لازم را اتخاذ می‌کند.

۱.۴.۵ فایل سرآیند (`controller.h`)

این فایل، ساختار داده اصلی و توابع عمومی مژول کنترل‌کننده را تعریف می‌کند.

▪ **وابستگی‌ها**: این فایل با فراخوانی `basic.h`، به متغیرهای سراسری وضعیت پرواز (Yaw, Pitch, Roll) و توابع زمان‌بندی دقیق دسترسی پیدا می‌کند.

□ ساختار داده **ControlParam**: این ساختار به صورت یک قالب عمومی برای نگهداری تمام اطلاعات و پارامترهای یک حلقه کنترل PID طراحی شده است. اعضای کلیدی آن عبارتند از:

□ **k_p, k_i, k_d**: برهه های تناسبی، انتگرالی و مشتقی کنترل کننده.

□ **I_P, p_max_limit, i_max_limit, d_max_limit**: محدودیت های جداگانه برای هر یک از جملات P و I.

□ **D**: این قابلیت، یک روش پیشرفته برای جلوگیری از پدیده **بادشده گی انتگرال (Integral Windup)** و اشباع ناخواسته هر بخش از کنترل کننده است.

□ **set_point**: مقدار مطلوب یا نقطه تنظیم که کنترل کننده باید به آن برسد.

□ **point**: مقدار واقعی و اندازه گیری شده از سیستم (مثلاً زاویه Roll فعلی).

□ **d_error, last_error, error**: متغیرهای حالت برای محاسبه خطای فعلی، خطای قبلی و مشتق فیلتر شده خطای.

□ **control_effort**: خروجی نهایی و ترکیبی کنترل کننده که به بخش ترکیب موتورها ارسال می شود.

□ **اعلان توابع**: توابع عمومی **module_01_controller_init()** برای راه اندازی و **module_01_controller_loop_handler()** برای اجرا در حلقه اصلی برنامه، در این بخش اعلان شده اند.

۲.۴.۵ فایل منبع (**controller.c**)

این فایل شامل پیاده سازی منطق کنترل و الگوریتم های مربوطه است.

راه اندازی کنترل کننده ها

در ابتدای فایل، چهار نمونه از ساختار ControlParam برای هر یک از محورهای کنترلی (Altitude, Yaw, Pitch, Roll) ایجاد و با برهه های اولیه مقداردهی می شوند. متغیرهای سراسری برای فعال سازی هر کنترل کننده (Roll_enable, Pitch_enable) و فعال سازی کلی موتورها (all_enable) نیز به عنوان یک ویژگی ایمنی و آزمون تعریف شده اند.

module_01_controller_init()

این تابع، تایمر TIM3 را برای تولید سیگنال PWM بر روی هر چهار کanal راه اندازی می کند. این کانال ها مستقیماً به اسپید کنترلرهای چهار موتور متصل می شوند و وظیفه کنترل سرعت آن ها را بر عهده دارند.

module_01_controller_loop_handler()

این تابع، حلقه اصلی کنترل است که با فرکانس ثابت اجرا می شود.

□ **زمان بندی حلقه**: کد تنها در صورتی اجرا می شود که بیش از ۱۰۰۰۰ میکرو ثانیه (۱۰ میلی ثانیه) از آخرین اجرا گذشته باشد. این ساختار یک حلقه کنترل با فرکانس ثابت ۱۰۰ هرتز ایجاد می کند که برای عملکرد پایدار کنترل کننده های دیجیتال ضروری است.

□ **اجرای کنترل کننده ها**: در هر تکرار، مقادیر فعلی Roll, Pitch و Yaw از متغیرهای سراسری خوانده شده و به عنوان ورودی به تابع عمومی pidController ارسال می شوند. فاصله زمانی دقیق حلقه (dt) نیز محاسبه و به این تابع داده می شود.

□ الگوریتم ترکیب موتورها (**Motor Mixing**): پس از محاسبه خروجی (control_effort) هر یک از چهار کنترل کننده، این مقادیر بر اساس یک ماتریس ترکیب استاندارد با یکدیگر جمع و تفریق می‌شوند تا مقدار نیروی لازم برای هر موتور (mx1 تا mx4) مشخص گردد. این الگوریتم برای یک کوادراتور با پیکربندی X به شکل زیر است:

$$\begin{bmatrix} m_1 \\ m_2 \\ m_3 \\ m_4 \end{bmatrix} = \begin{bmatrix} Thrust - Pitch_{effort} - Roll_{effort} - Yaw_{effort} \\ Thrust - Pitch_{effort} + Roll_{effort} + Yaw_{effort} \\ Thrust + Pitch_{effort} + Roll_{effort} - Yaw_{effort} \\ Thrust + Pitch_{effort} - Roll_{effort} + Yaw_{effort} \end{bmatrix}$$

□ ارسال فرمان به موتورها: در نهایت، مقادیر محاسبه شده برای هر موتور پس از عبور از یکتابع محدود کننده، به رجیسترها TIM3 Capture/Compare اعمال می‌شوند. یک مقدار پایه 150 نیز به خروجی اضافه می‌شود که به عنوان حداقل مقدار PWM برای روشن ماندن موتورها عمل می‌کند.

پیاده‌سازی الگوریتم pidController()

این تابع یک پیاده‌سازی عمومی و مقاوم از الگوریتم PID را ارائه می‌دهد.

□ جمله مشتق فیلترشده: به جای محاسبه مستقیم مشتق خطای نویز سنسور بسیار حساس است، از یک فیلتر پایین‌گذر ساده بر روی تفاضل خطاهای استفاده شده است: $d_error = d_error + (delta_error - d_error) * 0.2$. این کار اثر نویز را به شدت کاهش داده و اجازه می‌دهد از بهره مشتقی (Kd) بالاتری برای میرایی نوسانات استفاده شود.

□ محدودیت ضد بادشده (**Anti-Windup**): پس از محاسبه هر یک از جملات P، I و D، مقدار آنها در محدوده تعریف شده در ساختار ControlParam محدود (clamped) می‌شود. این کار از انباست بیش از حد خطای انگرال در زمان‌هایی که عملگر (موتور) به اشباع رسیده است، جلوگیری کرده و پایداری سیستم را به شدت بهبود می‌بخشد.

این پیاده‌سازی هوشمندانه، کنترل کننده‌ای را فراهم می‌کند که نه تنها دقیق، بلکه در برابر نویز و محدودیت‌های فیزیکی سیستم نیز مقاوم است.

5.5 main کد شرح

فایل main.c نقطه ورود و مرکز هماهنگی کل پروژه نرم‌افزاری است. این فایل وظیفه دارد تا سختافزار میکروکنترلر را راهاندازی کرده، تمام مأژول‌های نرم‌افزاری (شامل basic و controller) را به ترتیب صحیح مقداردهی اولیه نماید و در نهایت، حلقه بینهایت کنترلی را اجرا کند که وظیفه پایدارسازی کوادراتور را بر عهده دارد.

1.5.5 فاز راهاندازی (Initialization)

پیش از ورود به حلقه اصلی، مجموعه‌ای از دستورات برای آماده‌سازی سیستم اجرا می‌شود. این فرآیند که تنها یک بار در هنگام روشن شدن دستگاه انجام می‌گیرد، شامل مراحل زیر است:

۱. راه اندازی سخت افزار HAL: ابتدا توابع استاندارد HAL_Init() و HAL_Clock_Config() مانند STM32 HAL فراخوانی می شوند. سپس، تمام پریفراں های سخت افزاری مورد نیاز پروژه که از طریق CubeMX پیکربندی شده اند، راه اندازی می گردند. این پریفراں ها شامل موارد زیر هستند:

- I2C1: برای برقراری ارتباط با سنسور MPU-9250
- TIM2: به عنوان زمان سنج اصلی سیستم با دقت میکرو ثانیه (مورد استفاده در مژول basic).
- TIM3: برای تولید چهار کanal سیگنال PWM جهت کنترل موتورها (مورد استفاده در مژول controller).

۲. راه اندازی مژول های نرم افزاری: پس از آماده شدن سخت افزار، مژول های نرم افزاری به ترتیب فراخوانی می شوند:

- module_00_basic_init(): تایمர سیستم را برای زمان بندی دقیق فعال می کند.
 - module_01_controller_init(): تایمر PWM را برای ارسال سیگنال به موتورها فعال می کند.
 - MPU9250_Init(): سنسور IMU را راه اندازی کرده و از صحت ارتباط با آن اطمینان حاصل می کند.
۳. کالیبراسیون سنسور: در صورتی که راه اندازی سنسور موفقیت آمیز باشد، تابع MPU9250_Calibrate_Gyro با استفاده از ۵۰۰ نمونه فراخوانی می شود. این مرحله برای حذف خطای بایاس ژیروسکوپ و دستیابی به تخمین دقیق زاویه Yaw بسیار حیاتی است.

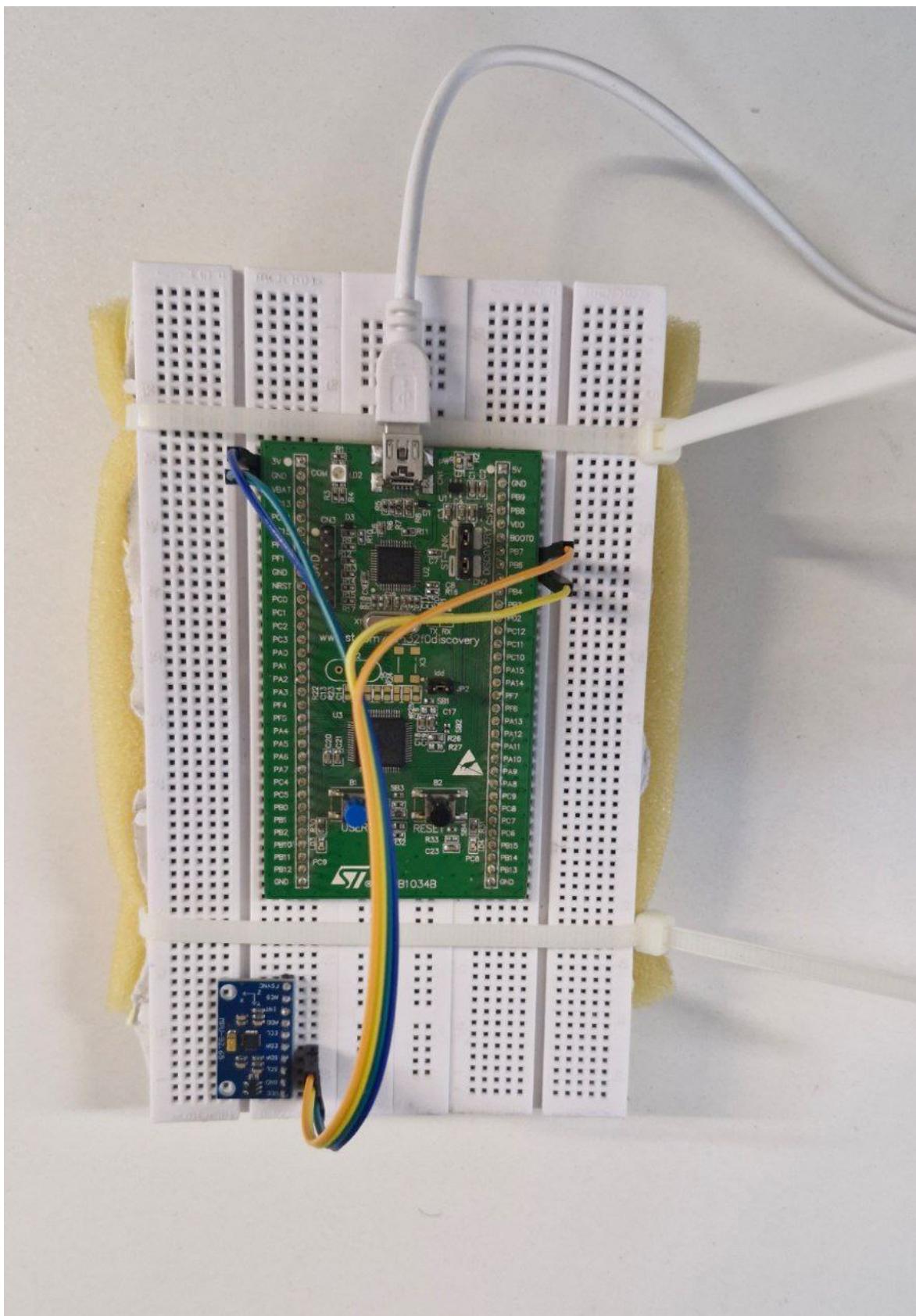
۲.۵.۵ حلقه کنترل اصلی (The Main Control Loop)

پس از اتمام فاز راه اندازی، برنامه وارد یک حلقه بی نهایت (while(1) می شود. این حلقه، قلب تپنده سیستم کنترل پرواز است و وظایف خود را به صورت پیوسته و با سرعت بالا تکرار می کند. ترتیب اجرای توابع در این حلقه برای عملکرد صحیح سیستم بسیار مهم است:

۱. module_00_basic_loop_handler(): در ابتدای هر تکرار، این تابع فراخوانی می شود تا زمان سیستم به روز شده و فاصله زمانی از حلقه قبلی (dt) محاسبه شود.
 ۲. module_01_controller_loop_handler(): در این مرحله، الگوریتم های کنترل PID اجرا می شوند. این مژول بر اساس آخرین داده های معتبر سنسور (که در تکرار قبلی حلقه خوانده شده اند) و با استفاده از dt محاسبه شده خروجی های کنترلی را محاسبه کرده و فرمان PWM جدید را به موتورها ارسال می کند.
 ۳. MPU9250_Read_All(): پس از اجرای منطق کنترل، داده های جدید از سنسور IMU خوانده شده و با اجرای فیلتر کالمن، مقادیر جدید و به روز شده برای زوایای Roll، Pitch و Yaw محاسبه می شوند.
 ۴. به روز رسانی متغیرهای سراسری: در انتهای حلقه، مقادیر جدید محاسبه شده از سنسور، در متغیرهای سراسری Roll، Pitch و Yaw کپی می شوند. این کار تضمین می کند که در تکرار بعدی حلقه، مژول کنترل کننده به جدید ترین اطلاعات وضعیت دسترسی خواهد داشت.
- این چرخه که به آن «حس کن - فکر کن - عمل کن» (Sense-Think-Act) گفته می شود، اساس کار تمام سیستم های رباتیک و کنترلی بلاذرنگ را تشکیل می دهد.

۳.۵.۵ مدیریت وقفه‌ها (Interrupt Handling)

تابع HAL_TIM_PeriodElapsedCallback یک تابع بازخوانی (Callback) است که به صورت خودکار توسط سخت‌افزار و در زمان وقوع یک وقفه (*Interrupt*) فراخوانی می‌شود. در این پروژه، این تابع زمانی اجرا می‌شود که تایمر module_00_basic_tim_interrupt_handler را به ماژول basic (از طریق تابع TIM2 سرریز شود. این تابع، رویداد سرریز را برای پیاده‌سازی یک زمان‌سنج دقیق و بدون اطلاع می‌دهد تا شمارنده نرم‌افزاری ۶۴ بیتی به درستی کار کند. این مکانیزم برای پیاده‌سازی یک زمان‌سنج دقیق و بدون خطا ضروری است.



شکل ۱.۵: برد استفاده شده برای راهاندازی



شکل ۲.۵: کواد به همراه برد استفاده شده

فصل ۶

نتیجه‌گیری

در این پژوهه، یک کنترل کننده مقاوم Fuzzy-PID برای پایدارسازی و ردیابی مسیر یک کوادراتور طراحی، شبیه‌سازی و ارزیابی گردید. هدف اصلی، غلبه بر چالش‌های ذاتی سیستم‌های کوادراتور، شامل دینامیک شدیداً غیرخطی، کوپلینگ بین حرکات دورانی و انتقالی، و ناپایداری حلقه-باز بود.

تحقیقات اولیه با مرور جامع ادبیات و تحلیل سیر تکامل روش‌های کنترلی آغاز شد. این بررسی نشان داد که کنترل کننده‌های خطی کلاسیک نظری PID ، علی‌رغم سادگی پیاده‌سازی، در مواجهه با مانورهای تهاجمی و تغییرات دینامیکی سیستم، عملکرد مطلوبی ندارند و دچار ناپایداری می‌شوند. این ضعف ذاتی، انگیزه‌ی اصلی برای انتخاب یک معماری کنترلی هوشمند و تطبیقی بود. بر این اساس، ساختار Fuzzy-PID به دلیل قابلیت تنظیم آنلاین و هوشمند بهره‌های کنترلی بر اساس شرایط لحظه‌ای پرواز انتخاب شد تا پاسخی سریع، دقیق و در عین حال پایدار فراهم آورد.

برای اعتبارسنجی عملی، یک محیط شبیه‌سازی جامع در بستر MATLAB/Simulink توسعه یافت. ابتدا، مدل دینامیکی کوادراتور با استفاده از یک رابط گرافیکی سفارشی که پارامترهای فیزیکی دقیق قطعات را دریافت می‌کرد، ایجاد شد. سپس، یک ساختار کنترلی آبشاری، متشکل از حلقه داخلی کنترل وضعیت و حلقه خارجی کنترل موقعیت، پیاده‌سازی شد. عملکرد کنترل کننده در سه سناریوی آزمون استاندارد ارزیابی شد: مسیر دایره‌ای برای سنجش پایداری پایه، مسیر هشتی برای آزمون مانور پذیری، و مسیر لوزی برای تحلیل پاسخ گذرا به تغییرات ناگهانی.

نتایج شبیه‌سازی کنترل کننده PID استاندارد، ضعف‌های پیش‌بینی شده را تأیید کرد؛ سیستم در پاسخ به فرمان‌های پله‌ای دچار فرجهش (*Overshoot*) بزرگ، نوسانات شدید و زمان نشست طولانی می‌شد که نشان‌دهنده یک عملکرد زیرمیرا (*Underdamped*) و نامطلوب بود. در مقابل، کنترل کننده Fuzzy-PID که با استفاده از یک سیستم استنتاج فازی بهره‌های خود را به صورت پویا تنظیم می‌کرد، بهبود عملکردی چشمگیری را به نمایش گذاشت. فرجهش و نوسانات به طور کامل حذف شدند، زمان نشست به شدت کاهش یافت و دقت ردیابی مسیر به طور قابل توجهی افزایش پیدا کرد. مقایسه‌ی کمی با استفاده از معیارهای خطای $ITAE$ و MSE ، برتری قاطع کنترل کننده Fuzzy-PID را در تمامی محورها به اثبات رساند و نشان داد که این رویکرد توانسته است تعادل موفقی بین سرعت پاسخ و پایداری ایجاد کند.

در مجموع، این پژوهه با موفقیت نشان داد که ترکیب منطق فازی با ساختار کلاسیک PID یک راهکار مؤثر برای کنترل مقاوم و با عملکرد بالای کوادراتورها است. این کنترل کننده هوشمند با تطبیق‌پذیری خود، ضعف‌های ذاتی کنترل کننده‌های با بهره ثابت را برطرف کرده و یک بستر قابل اعتماد برای پیاده‌سازی نهایی بر روی سخت‌افزار واقعی فراهم می‌آورد.

كتاب نامه

- [1] Zulu, Andrew and John, Samuel. A review of control algorithms for autonomous quadrotors. *PLOS ONE*, 2024.
- [2] Shraim, Hassan, Awada, Ali, and Younes, Rafic. A survey on quadrotors: Configurations, modeling and identification, control, collision avoidance, fault diagnosis and tolerant control. *IEEE Aerospace and Electronic Systems Magazine*, 33(7):14–33, 2018.
- [3] Mian, Ahsan Abbas and Daobo, Wang. A review on comparative remarks, performance evaluation and improvement strategies of quadrotor controllers. *Applied Sciences*, 9(2):37, 2021.
- [4] Salih, Ahmed and Saleh, Mohammed. Modeling and control of x-shape quadcopter. *International Journal of Engineering Research and*, 11(2), 2022.
- [5] Abdulmajeed, Ali, Zhao, Long, and Al-mafrachi, Ali. A review of quadrotor unmanned aerial vehicles: Applications, architectural design and control algorithms. *Journal of Intelligent & Robotic Systems*, 104(2):1–33, 2022.
- [6] Zhang, Y., Chen, G., and Li, J. A review on quadrotor control methods. *AETR*, 2022.
- [7] Maaruf, Muhammad, Mahmoud, Muath, and Ma’arif, Aris. A survey of control methods for quadrotor uav. *International Journal of Robotics and Control Systems*, 2(3):652–665, 2022.
- [8] Ghadiok, Chayapol, Tan, Chee-Meng, and Wang, Danwei. Historical and current landscapes of autonomous quadrotor control: An early-career researchers’ guide. *Drones*, 8(3):72, 2023.
- [9] Zulu, Andrew and John, Samuel. A review of the prominent controllers for the quadrotor uav. *Intelligent Control and Automation*, 5(4):271, 2014.
- [10] Tran, Dong LT, Vo, Thanh C, Tran, Hoang T, Nguyen, Minh T, and Do, Hai T. Design optimal backstepping controller for quadrotor based on lyapunov theory for disturbances environments. *arXiv preprint arXiv:2503.06824*, 2025.
- [11] Zhang, J. et al. Modeling and control of 6-dof uav quadcopter using pid controllers according to ziegler-nichols. *International Journal of Mechanical and Production Engineering Research and Development*, 13(1):484–495, 2023.

- [12] Nemati, Hamidreza, Kumar, M, and Nagaty, A. Dynamic modeling and control techniques for a quadrotor. *International Journal of Computer Applications*, 98(13), 2014.
- [13] Madani, Tarek and Benallegue, Abdelaziz. Backstepping control for a quadrotor helicopter. in *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 3255–3260, 2006.
- [14] Reyes-Valeria, E., Enriquez-Caldera, Rogerio, Camacho, Sergio, and Guichard, Jose S. Lqr control for a quadrotor using unit quaternions: Modeling and simulation. in *2013 International Conference on Electronics, Communications and Computing (CONIELECOMP)*, pp. 155–160, 2013.
- [15] Mahfouz, Ahmed Alaa, Akram, Rizwan, and Omar, Hanafy M. Robust tracking control for quadrotor uav with external disturbances and uncertainties using neural network based mrac. *arXiv preprint arXiv:2403.18101*, 2024.
- [16] Lone, Ahtisham Aziz, Mercorelli, Paolo, Nemati, Hamidreza, and Zhu, Quanmin. Improved nonsingular adaptive super twisting sliding mode control for tracking of a quadrotor system in the presence of external disturbances and uncertainty. *PLOS ONE*, 19(10):e0309098, 2024.
- [17] Islam, S, Faraz, M, Ashour, RK, Dias, J, and Seneviratne, LD. Robust adaptive control of quadrotor unmanned aerial vehicle with uncertainty. in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4445–4450, 2015.
- [18] Vaidya, Varad and Keshavan, Jishnu. Dynamics-invariant quadrotor control using scale-aware deep reinforcement learning. *The Moonlight*, 2023.
- [19] Mahfouz, Ahmed Alaa, Omar, Hanafy M, and Akram, Rizwan. Adaptive controller for a quadrotor uav for carrying unknown payloads while tracking any trajectory. *Drones*, 6(9):251, 2022.
- [20] Omar, Hanafy M, Akram, Rizwan, Mukras, Saad MS, and Mahfouz, Ahmed Alaa. Recent advances and challenges in controlling quadrotors with suspended loads. *Progress in Aerospace Sciences*, 135:100859, 2022.
- [21] Islam, M. et al. A review of control methods for quadrotor uav. *International Journal of Current Science Research and Review*, 6(10), 2023.
- [22] Benevides, João RS, Ishihara, João Y, and Borges, Gleison A. Disturbance observer-based robust control of a quadrotor subject to parametric uncertainties and wind disturbance. *IEEE Access*, 10:8359–8372, 2022.
- [23] Li, J and Li, Y. Dynamic analysis and pid control for a quadrotor. in *2011 IEEE International Conference on Mechatronics and Automation*, pp. 573–578, 2011.
- [24] Ercan, H and Can, M. S. Lqr control of a quadrotor helicopter. in *2015 9th International Conference on Electrical and Electronics Engineering (ELECO)*, pp. 856–860, 2015.

- [25] Foehn, Philipp, Brescianini, Dario, and Scaramuzza, Davide. On-board state-dependent lqr for agile quadrotors. in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 7267–7274, 2018.
- [26] Reyes-Valeria, E, Enriquez-Caldera, Rogerio, Camacho-Lara, Sergio, and Guichard-Romero, Jose S. Lqr control for a quadrotor using unit quaternions: Modeling and simulation. *IEEE*, 2021.
- [27] Suresh, Harikrishnan, S, Aswin, and V, Sreekanth. Attitude control of a quadcopter. *International Journal of Scientific & Engineering Research*, 8(4):68–72, 2017.
- [28] Castillo-Zamora, Jorge, Garcia-Beltran, Carlos Daniel, Perez-Patricio, Miguel, and Hernandez-Gonzalez, Octavio. Discrete lqr plus integral action for quadrotor attitude and altitude control. *Applied Sciences*, 13(16):9293, 2023.
- [29] Sabatino, Francesco. Modeling and control of a quadrotor. tech. rep., KTH, 2015.
- [30] Guerrero-Sánchez, M.E., Martins, L., Cardeira, C., Hernández-González, O., and Lozano, R. Feedback linearization with zero dynamics stabilization for quadrotor control. *Journal of Intelligent & Robotic Systems*, 100(3-4):1247–1261, 2020.
- [31] Dierks, Travis and Jagannathan, Sarangapani. Backstepping and neural networks for control of a quadrotor uav. in *2009 American Control Conference*, pp. 1446–1451, 2009.
- [32] Mian, Ahsan Abbas and Daobo, Wang. A review on comparative remarks, performance evaluation and improvement strategies of quadrotor controllers. *Applied Sciences*, 11(10):4589, 2021.
- [33] Chen, Y. et al. Super twisting sliding mode control with a novel fuzzy pid surface for improved trajectory tracking of quadrotor unmanned aerial vehicles under external disturbances. *PLOS ONE*, 19(11):e0308997, 2024.
- [34] Benallegue, Abdelaziz, Mokhtari, A, and Fridman, L. Sliding mode control with a high order sliding mode observer for a quadrotor uav. in *2008 47th IEEE Conference on Decision and Control*, pp. 5594–5599, 2008.
- [35] Dydek, Z. T., Annaswamy, A. M., and Lavretsky, E. Adaptive control of quadrotor uavs: A design trade study with flight evaluations. *IEEE Transactions on Control Systems Technology*, 21(2):552–558, 2013.
- [36] Islam, S, Faraz, M, Ashour, RK, Dias, J, and Seneviratne, LD. Adaptive control of quadrotor unmanned aerial vehicle with time-varying uncertainties. *Journal of Intelligent & Robotic Systems*, 107(2):25, 2023.
- [37] Castillo, Pedro, Garcia, P, and Lozano, Rogelio. Robust control of quadrotors based on an uncertainty and disturbance estimator. *Journal of Intelligent & Robotic Systems*, 79(3-4):435–451, 2015.

- [38] Chen, Feng, Jiang, Rui, Zhang, Keqi, Jiang, Bin, and Tao, Gang. Trajectory tracking control for quadcopter unmanned aerial vehicle based on a nonlinear robust backstepping algorithm and extended state/disturbance observer. *Sensors*, 22(14):5263, 2022.
- [39] Sönmez, Serhat, Montecchio, Luca, Martini, Simone, Rutherford, Matthew J, Rizzo, Alessandro, Stefanovic, Margareta, and Valavanis, Kimon P. Reinforcement learning based prediction of pid controller gains for quadrotor uavs. *arXiv preprint arXiv:2502.04552*, 2025.
- [40] Vaidya, Varad and Keshavan, Jishnu. Dynamics-invariant quadrotor control using scale-aware deep reinforcement learning. *arXiv preprint arXiv:2303.05835*, 2023.
- [41] Song, Y, Naji, A, El-aasser, M, and Kumar, V. Can we level the playing field in learning-based quadrotor control? in *Robotics: Science and Systems*, 2021.
- [42] Dallagi, A. et al. Comparative study of control methods for quadrotor uav. *Applied Sciences*, 13(6):3464, 2023.
- [43] Dallagi, A. et al. Modeling and controlling highly nonlinear, multivariable, unstable, coupled and underactuated systems: A comparative study of quadrotor control. *Applied Sciences*, 13(6):3464, 2023.
- [44] Hanover, D, Loquercio, A, and Scaramuzza, D. L1-nmpc: An l1-adaptive nonlinear model predictive control for agile quadrotor flight. *IEEE Robotics and Automation Letters*, 7(2):2029–2036, 2021.
- [45] Pounds, P, Mahony, R, and Corke, P. A detailed dynamic and aerodynamic model of a quadrotor for control design. in *2012 Australasian Conference on Robotics and Automation*, 2012.
- [46] Li, J and Li, Y. Second order sliding mode control for a quadrotor uav. *ISA transactions*, 125:468–477, 2022.
- [47] Li, Y. et al. A novel hybrid control strategy for quadrotor uavs inspired by neural dynamics. *Applied Sciences*, 14(20):9592, 2024.
- [48] Zhang, J. Y. et al. Robust h-infinity dual cascade mpc-based attitude control study of a quadcopter uav. *Sensors*, 24(10):392, 2024.
- [49] Falanga, Davide, Mueggler, Elias, Faessler, Matthias, and Scaramuzza, Davide. Aggressive quadrotor flight through narrow gaps with onboard sensing and computing using active vision. in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4826–4833, 2017.
- [50] Tranzhao, B and Scaramuzza, D. Whole-body control through narrow gaps from pixels to action. *The Moonlight*, 2022.
- [51] Lee, Taeyoung. Dynamics and control of quadrotor with robotic manipulator. Seoul National University.