Adaptive Controller Development and Evaluation for a 6DOF Controllable Multirotor

Theresa C. W. Furgiuele

Dissertation submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy
in
Aerospace Engineering

Jonathan T. Black, Chair
Alexander Leonessa
Shane D. Ross
Kevin K. Schroeder
Craig A. Woolsey

September 2, 2022
Blacksburg, Virginia

Keywords: adaptive control, model reference adaptive control, adaptive backstepping control, UAV/UAS, omnicopter, 6DOF controllable multirotor, quadcopter, time delay margin estimation, control tuning, matrix measure, bounded linear stability analysis, Pixhawk, PX4 firmware

Copyright 2022, Theresa C. W. Furgiuele

Adaptive Controller Development and Evaluation for a 6DOF Controllable Multirotor

Theresa C. W. Furgiuele

ABSTRACT

The omnicopter is a small unmanned aerial vehicle capable of executing decoupled translational and rotational motion (six degree of freedom, 6DOF, motion). The development of controllers for various 6DOF controllable multirotors has been much more limited than development for quadrotors, which makes selecting a controller for a 6DOF multirotor difficult. The omnicopter is subject to various uncertainties and disturbances from hardware changes, structural dynamics, and airflow, making adaptive controllers particularly interesting to investigate. The goal of this research is to design and evaluate the performance of various position and attitude controller combinations for the omnicopter, specifically focusing on adaptive controllers. Simulations are first used to compare combinations of three position controllers, PID, model reference adaptive control, augmented model reference adaptive control (aMRAC), and four attitude controllers, PI/feedback linearization (PIFL), augmented model reference adaptive control, backstepping, and adaptive backstepping (aBack). For the simulations, the omnicopter is commanded to point at and track a stationary aim point as it travels along a C^0 continuous trajectory and a trajectory that is C^1 continuous. The controllers are stressed by random disturbances and the addition of an unaccounted for suspended mass. The augmented model reference adaptive controller for position control paired with the adaptive backstepping controller for attitude control is shown to be the best controller combination for tracking various trajectories while subject to disturbances. Based on the simulation results, the PID/PIFL and aMRAC/aBack controllers are selected

to be compared during three different flight tests. The first flight test is on a C^1 continuous trajectory while the omnicopter is commanded to point at and track a stationary aim point. The second flight test is a hover with an unmodeled added weight, and the third is a circular trajectory with a broken blade. As with the simulation results, the adaptive controller is shown to yield better performance than the nonadaptive controller for all scenarios, particularly for position tracking. With an added weight or a broken propeller, the adaptive attitude controller struggles to return to level flight, but is capable of maintaining steady flight when the nonadaptive controller tends to fail. Finally, while model reference adaptive controllers are shown to be effective, their nonlinearity can make them difficult to tune and certify via standard certification methods, such as gain and phase margin. A method for using time delay margin estimates, a potential certification metric, to tune the adaptive parameter tuning gain matrix is shown to be useful when applied to an augmented MRAC controller for a quadrotor.

Adaptive Controller Development and Evaluation for a 6DOF Controllable Multirotor

Theresa C. W. Furgiuele

GENERAL AUDIENCE ABSTRACT

The omnicopter is a small unmanned aerial vehicle capable of executing decoupled translational and rotational motion. The development of controllers for these types of vehicles has been limited, making controller selection difficult. The omnicopter is subject to variations in hardware and airflow, making adaptive controllers particularly interesting to investigate. The goal of this research is to design and compare the performance of various position and attitude controller combinations for the omnicopter, specifically focusing on adaptive controllers. Simulations are first used to compare combinations of several position and attitude controllers on various trajectories and disturbances. Simulation results showed that a fully adaptive controller combination produced the best trajectory tracking while subject to disturbances. As with the simulation results, flight tests showed the adaptive controller yields better performance than the nonadaptive controller for all scenarios, particularly for position tracking. Finally, while the adaptive position controller was shown to be effective, it is difficult to tune and certify for widespread use. A method for using time delay margin estimates, a potential certification metric, to tune the adaptive controller is shown to be useful when applied to an adaptive controller for a quadrotor.

To my husband, Sean, for his never ending love and support.

To Ginny, for being the friend I needed through this whole process.

Acknowledgments

This dissertation would never have been possible without the help of many students. First and foremost, I am immensely grateful to Patrick Thomas, without whom the omnicopter would never have gotten off the ground and been able to accomplish the flights I needed. I would also like to thank Collin Deans for laying the foundation for the omnicopter at Virginia Tech upon which I could develop my research. I am grateful to Gustavo Gargioni, Marco Peterson, and Minzhen Du for developing the SpaceDrones lab and helping along the way as we tried to get the omnicopter in the air.

Several professors have also helped shape the course of my studies. I would like to thank my advisor, Dr. Black, for allowing me into a lab where I could learn about space and UAVs and for giving me the flexibility I needed to be able to see my husband these past few years. I would like to thank Dr. Schroeder for his advice and mentorship. I would also like to thank Dr. Leonessa for introducing me to adaptive controls and for providing guidance as I learned how to apply adaptive controllers to the omnicopter.

Throughout my time as a graduate student, I have also maintained an internship at NSWC Dahlgren Division. The advice and experience I have gained over the years there have shaped this dissertation. I am extremely grateful to all the engineers who gave me their time and wisdom, especially Dan Parks. Your mentorship has been invaluable as I've navigated this last year of grad school.

Finally, I am grateful to all my fellow graduate students at the Hume Center and Space@VT for their support, aid, and friendship. Jake Viar, thank you for the climbing and yoga sanity breaks. Virginia Smith, Camille Samulski, thank you for allowing me to rant, encouraging me to bake, and providing a hand up whenever I needed one.

Contents

List of Figures xii

List of Tables xx

List of Abbreviations xxii

1 Introduction 1

1.1 Motivation 1

1.2 Research Questions 4

1.3 Contributions 4

1.4 Summary of Chapters 5

2 Literature Review 6

2.1 Adaptive Control of 4DOF Controllable Multirotors 6

2.2 Adaptive Control of 6DOF Controllable Multirotors 8

2.3 Time Delay Margin Estimation of Adaptive Systems 9

3 Time Delay Margin Tuning of an Adaptive Controller for a Quadrotor 12

3.1 Introduction 12

3.2 Quadrotor Dynamics and Controller Development 13

3.2.1 Quadrotor Dynamics . . . 13
3.2.2 Baseline Control: LQR with Integral Feedback . . . 17
3.2.3 Augmented Model Reference Adaptive Control (aMRAC) . . . 18
3.3 Time Delay Margin Estimation . . . 21
3.3.1 Bounded Linear Stability Analysis Introduction . . . 21
3.3.2 BLSA modified for aMRAC . . . 22
3.3.3 Matrix Measure . . . 24
3.3.4 Matrix Measure Modification . . . 27
3.4 Simulation Results . . . 28
3.4.1 aMRAC MIMO Results . . . 31
3.4.2 LTI Baseline Controller Results . . . 35
3.4.3 Quadrotor Flight Example . . . 36
3.5 Discussion . . . 37

4 Simulation Methodology **39**
4.0.1 Disturbances . . . 40
4.0.2 Test Trajectories . . . 43

5 Position Dynamics and Control **46**
5.1 Dynamics . . . 46
5.2 PID . . . 46

5.3 Model Reference Adaptive Control with Input Saturation (MRAC) 47

5.3.1 PID Augmented Model Reference Adaptive Control with Input Saturation (aMRAC) 51

6 Attitude Dynamics and Control **56**

6.1 Dynamics and Kinematics 56

6.2 Proportional Integral/Feedback Linearization Control (PI/FL) 57

6.3 Augmented Model Reference Adaptive Control (aMRAC) 59

6.4 Backstepping 62

6.5 Adaptive Backstepping 64

7 Simulation Results **68**

7.1 Error Computation 68

7.2 Error Results 69

7.2.1 Nominal Mass Results 69

7.2.2 Increased Mass Results 76

7.3 Controller Commands 82

7.3.1 Force Commands 82

7.3.2 Torque Commands 88

7.3.3 Motor Commands 94

7.3.4 Command Implications 100

7.3.5 Position Adaptive Parameter Performance . . . 102
7.3.6 Attitude Adaptive Parameter Performance . . . 110
7.4 Discussion . . . 123

8 Flight Tests **126**

8.1 Omnicopter Hardware Overview . . . 126
8.2 Custom Firmware Implementation . . . 130
8.3 Controller Adjustments for Flight . . . 131
8.4 Flight Test Results . . . 134
8.4.1 Attitude Observations . . . 134
8.4.2 Smooth Trajectory . . . 135
8.4.3 Hover Tests with Added Weight . . . 147
8.4.4 Circular Trajectory with Broken Propeller . . . 162
8.5 Discussion . . . 172

9 Conclusions and Future Work **174**

9.1 Proposed Attitude and Motor Control Improvements . . . 175
9.2 Modeling and Tuning . . . 177
9.3 Position Control Comments . . . 178
9.4 Research Question Conclusions . . . 180
9.5 Broader Applications . . . 180

Appendix A Projection Operator **183**

Appendix B Instructions for Implementing Custom Controllers in PX4 Firmware **184**

Appendix C Matrix Measure Computation **188**

C.1 Standard MRAC [1] 188

C.1.1 Set-Up 188

C.1.2 Error Equation Development 190

C.1.3 Matrix Measure Method 193

C.2 MRAC Augmentation of an Optimal Baseline Controller (aMRAC) 197

C.2.1 Set-Up 197

C.2.2 Error Equation Development 198

C.2.3 Matrix Measure Method 201

Bibliography **202**

List of Figures

3.1 Quadrotor body frame and motors 14

3.2 Control Overview 16

3.3 aMRAC Overview 19

3.4 aMRAC Transient State Response 32

3.5 aMRAC ϕ Transient State Response for Reduced Thruster Capability 32

3.6 Surface plots comparing time delay margin with Gamma and Lambda 33

3.7 aMRAC True TDM vs TDM estimate using nominal MM method 34

3.8 aMRAC True TDM vs TDM estimate using modified MM method 35

3.9 Nominal thrust controller comparison 36

3.10 Reduced thrust controller comparison 37

4.1 Omnicopter system overview 39

4.2 Example Force and Torque Disturbances 43

4.3 Trajectories Created from Waypoints 44

5.1 Model reference adaptive control with input saturation overview 48

5.2 PID augmented model reference adaptive control with input saturation overview 52

7.1 Jagged trajectory mean error for position (left) and attitude (right) averaged over 10 runs for the omnicopter at nominal mass. 74

7.2 Smooth trajectory mean error for position (left) and attitude (right) averaged over 10 runs for the omnicopter at nominal mass. 75

7.3 Jagged trajectory mean error for position (left) and attitude (right) averaged over 10 runs for the omnicopter with increased mass. 80

7.4 Smooth trajectory mean error for position (left) and attitude (right) averaged over 10 runs for the omnicopter with increased mass. 81

7.5 Jagged trajectory, nominal mass omnicopter force command comparison . . 84

7.6 Smooth trajectory, nominal mass omnicopter force command comparison . . 85

7.7 Jagged trajectory, increased mass omnicopter force command comparison . . 86

7.8 Smooth trajectory, increased mass omnicopter force command comparison . 87

7.9 Jagged trajectory, nominal mass omnicopter torque command comparison. Note that transient peaks have been attenuated to preserve visualization of steady state responses. 90

7.10 Smooth trajectory, nominal mass omnicopter torque command comparison. Note that some peaks have been attenuated to preserve visualization of steady state responses. 91

7.11 Jagged trajectory, increased mass omnicopter torque command comparison. Note that some peaks have been attenuated to preserve visualization of steady state responses. 92

7.12 Smooth trajectory, increased mass omnicopter torque command comparison. Note that some peaks have been attenuated to preserve visualization of steady state responses. 93

7.13 Jagged trajectory, nominal mass omnicopter torque command comparison . . 96

7.14 Smooth trajectory, nominal mass omnicopter torque command comparison . 97

7.15 Jagged trajectory, increased mass omnicopter torque command comparison . 98

7.16 Smooth trajectory, increased mass omnicopter torque command comparison 99

7.17 Jagged trajectory, nominal mass omnicopter $\hat{\boldsymbol{K}}_x$ for MRAC position control comparison 103

7.18 Jagged trajectory, nominal mass omnicopter $\hat{\boldsymbol{K}}_r$ for MRAC position control comparison 103

7.19 Jagged trajectory, nominal mass omnicopter $\hat{\boldsymbol{\Theta}}$ for aMRAC position control comparison 104

7.20 Smooth trajectory, nominal mass omnicopter $\hat{\boldsymbol{K}}_x$ for MRAC position control comparison 105

7.21 Smooth trajectory, nominal mass omnicopter $\hat{\boldsymbol{K}}_r$ for MRAC position control comparison 105

7.22 Smooth trajectory, nominal mass omnicopter $\hat{\boldsymbol{\Theta}}$ for aMRAC position control comparison 106

7.23 Jagged trajectory, increased mass omnicopter $\hat{\boldsymbol{K}}_x$ for MRAC position control comparison 107

7.24 Jagged trajectory, increased mass omnicopter $\hat{\boldsymbol{K}}_r$ for MRAC position control comparison 107

7.25 Jagged trajectory, increased mass omnicopter $\hat{\boldsymbol{\Theta}}$ for aMRAC position control comparison 108

7.26 Smooth trajectory, increased mass omnicopter $\hat{\boldsymbol{K}}_x$ for MRAC position control comparison 109

7.27 Smooth trajectory, increased mass omnicopter $\hat{\boldsymbol{K}}_r$ for MRAC position control comparison 109

7.28 Smooth trajectory, increased mass omnicopter $\hat{\boldsymbol{\Theta}}$ for aMRAC position control comparison 110

7.29 Jagged trajectory, nominal mass omnicopter $\hat{\boldsymbol{\Theta}}$ for aMRAC attitude control comparison 112

7.30 Jagged trajectory, nominal mass omnicopter$\hat{\boldsymbol{J}}^{-1}$ for aMRAC attitude control comparison 112

7.31 Jagged trajectory, nominal mass omnicopter $\hat{\boldsymbol{t}}_d$ for aMRAC attitude control comparison 113

7.32 Jagged trajectory, nominal mass omnicopter$\hat{\boldsymbol{\Theta}}$ for aBack attitude control comparison 113

7.33 Jagged trajectory, nominal mass omnicopter$\hat{\boldsymbol{\Psi}}$ for aBack attitude control comparison 114

7.34 Smooth trajectory, nominal mass omnicopter $\hat{\boldsymbol{\Theta}}$ for aMRAC attitude control comparison 115

7.35 Smooth trajectory, nominal mass omnicopter $\hat{\boldsymbol{J}}^{-1}$ for aMRAC attitude control comparison 115

7.36 Smooth trajectory, nominal mass omnicopter $\hat{\boldsymbol{t}}_d$ for aMRAC attitude control comparison 116

7.37 Smooth trajectory, nominal mass omnicopter $\hat{\boldsymbol{\Theta}}$ for aBack attitude control comparison 116

7.38 Smooth trajectory, nominal mass omnicopter $\hat{\boldsymbol{\Psi}}$ for aBack attitude control comparison 117

7.39 Jagged trajectory, increased mass omnicopter $\hat{\boldsymbol{\Theta}}$ for aMRAC attitude control comparison 118

7.40 Jagged trajectory, increased mass omnicopter $\hat{\boldsymbol{J}}^{-1}$ for aMRAC attitude control comparison 118

7.41 Jagged trajectory, increased mass omnicopter $\hat{\boldsymbol{t}}_d$ for aMRAC attitude control comparison 119

7.42 Jagged trajectory, increased mass omnicopter $\hat{\boldsymbol{\Theta}}$ for aBack attitude control comparison 119

7.43 Jagged trajectory, increased mass omnicopter $\hat{\boldsymbol{\Psi}}$ for aBack attitude control comparison 120

7.44 Smooth trajectory, increased mass omnicopter $\hat{\boldsymbol{\Theta}}$ for aMRAC attitude control comparison 121

7.45 Smooth trajectory, increased mass omnicopter $\hat{\boldsymbol{J}}^{-1}$ for aMRAC attitude control comparison 121

7.46 Smooth trajectory, increased mass omnicopter $\hat{\boldsymbol{t}}_d$ for aMRAC attitude control comparison 122

7.47 Smooth trajectory, increased mass omnicopter $\hat{\boldsymbol{\Theta}}$ for aBack attitude control comparison 122

7.48 Smooth trajectory, increased mass omnicopter $\hat{\boldsymbol{\Psi}}$ for aBack attitude control comparison 123

8.1 Omnicopter CAD Rendering (credit: Patrick Thomas) 127

8.2 Omnicopter Hardware and Center Mount Diagram 128

8.3 Laboratory C2 architecture, adapted from [2] 130

8.4 Smooth Trajectory for Flight Tests. All units are in meters 136

8.5 Controller error variance comparison for the smooth trajectory. Dots represent mean error, and error bars represent one standard deviation above and below the mean error. 138

8.6 PID/PIFL 3D position tracking for smooth trajectory. The orange line is the actual trajectory flown, and the blue line is the desired trajectory. 139

8.7 aMRAC/aBack 3D position tracking for smooth trajectory. The orange line is the actual trajectory flown, and the blue line is the desired trajectory. . . 140

8.8 PID/PIFL position and attitude error for smooth trajectory 141

8.9 PID/PIFL position and attitude error for smooth trajectory, light battery . 142

8.10 PID/PIFL position and attitude error for smooth trajectory, standard battery 142

8.11 aMRAC/aBack position and attitude error for smooth trajectory, light battery 143

8.12 PID/PIFL force, torque, and motor commands for smooth trajectory 144

8.13 aMRAC/aBack force, torque, and motor commands for smooth trajectory . 145

8.14 aMRAC adaptive parameter column norms for smooth trajectory 146

8.15 aBack adaptive parameter column norms for smooth trajectory 147

8.16 Diagram of omnicopter with 9 inch blades and attached hook 148

8.17 Controller error variance comparison for hover with added weight. Dots represent mean error, and error bars represent one standard deviation above and below the mean error. 150

8.18 PID/PIFL position and attitude error for hover with added weight 151

8.19 aMRAC/aBack position and attitude error for hover with added weight . . . 152

8.20 aMRAC adaptive parameter column norms for hover with added weight . . . 153

8.21 aBack adaptive parameter column norms for hover with added weight 154

8.22 PID/PIFL force, torque, and motor commands for hovering with added weight 156

8.23 aMRAC/aBack force, torque, and motor commands for hovering with added weight 157

8.24 PID/PIFL position and attitude error for hover with added weight and corrected parameters 159

8.25 aMRAC/aBack position and attitude error for hover with added weight and corrected parameters 160

8.26 aBack adaptive parameter column norms for hover with added weight and correct parameters 161

8.27 Diagram of omnicopter with 8 inch blades and broken propeller 163

8.28 Controller error variance comparison for circular trajectory with broken propeller 164

8.29 PID/PIFL position and attitude error for circular trajectory with broken propeller 165

8.30 aMRAC/aBack position and attitude error for circular trajectory with broken propeller 166

8.31 PID/PIFL 3D position tracking for circular trajectory with broken propeller 167

8.32 aMRAC/aBack 3D position tracking for circular trajectory with broken propeller 168

8.33 PID/PIFL force, torque, and motor commands for circling with a broken propeller 169

8.34 aMRAC/aBack force, torque, and motor commands for circling with a broken propeller 170

8.35 aMRAC adaptive parameter column norms for circling with a broken propeller 171

8.36 aBack adaptive parameter column norms for circling with a broken propeller 172

List of Tables

3.1 Quadrotor Parameters 30

3.2 Controller Tuning Parameters 30

4.1 Omnicopter Parameters 40

4.2 Load Parameters 41

4.3 Disturbances 42

4.4 Trajectory Creation Points 44

5.1 PID Parameters 47

5.2 MRAC Parameters 51

5.3 aMRAC Parameters 54

6.1 PI/FL Parameters 58

6.2 aMRAC Attitude Parameters 62

6.3 Back/aBack Parameters 67

7.1 Mean Position Error of 10 Runs for the Omnicopter at Nominal Mass. Asterisks indicate the lowest position error amongst all controller combinations for a disturbance magnitude. Circles indicate the error associated with the best position/attitude controller combination. 72

7.2 Mean Attitude Error of 10 Runs for the Omnicopter at Nominal Mass. Asterisks indicate the lowest attitude error amongst all controller combinations for a disturbance magnitude. Circles indicate the error associated with the best position/attitude controller combination. 73

7.3 Mean position error of 10 Runs for the Omnicopter lifting a 0.25 kg unmodeled mass. Asterisks indicate the lowest position error amongst all controller combinations for a disturbance magnitude. Circles indicate the error associated with the best position/attitude controller combination. 78

7.4 Mean Attitude Position Error of 10 Runs for the Omnicopter Lifting a 0.25 kg Panel. Asterisks indicate the lowest attitude error amongst all controller combinations for a disturbance magnitude. Circles indicate the error associated with the best position/attitude controller combination. 79

8.1 PID/PIFL Parameters 132

8.2 aMRAC/aBack Parameters 132

8.3 Flight Test Trajectory Creation Points 135

8.4 Smooth Trajectory Flight Test Position Errors 137

8.5 Smooth Trajectory Flight Test Attitude Errors 137

8.6 Hover Flight Test Position Errors 149

8.7 Hover Flight Test Attitude Errors 149

8.8 Hover Flight Test Attitude Errors 160

8.9 Circular Flight Test Position Errors 163

8.10 Circular Flight Test Attitude Errors 164

List of Abbreviations

BLSA/MM

$\boldsymbol{C}$ error plant matrix

$\boldsymbol{D}$ error delay matrix

$\boldsymbol{S}$ optimal state transformation matrix

t_d,TDM time delay margin

BLSA Bounded Linear Stability Analysis

MM Matrix Measure

Controls

$\boldsymbol{K}$, k control gain matrix/scalar, subscripts indicate the value or vector being multiplied to the gain

$\boldsymbol{P}$ positive definite solution to the Lyapunov equation

$\boldsymbol{Q}_{lqr}$, $\boldsymbol{R}_{lqr}$ tuning matrices for LQR control

$\boldsymbol{Q}_{lyap}$ symmetric positive definite Lyapunov equation tuning matrix

Γ, γ symmetric positive definite adaptive parameter tuning gain matrix

τ time constant, s

ε positive projection tolerance

$\boldsymbol{\alpha}$ stabilizing function

$\boldsymbol{\psi}$ disturbance torque bound

$\boldsymbol{G}$ backstepping error matrix

$\boldsymbol{z}$ backstepping state

Dynamics

$\boldsymbol{\Lambda}$ positive, diagonal control effectiveness matrix

$\boldsymbol{\omega}$, Ω angular velocity of the omnicopter expressed in the body frame and of a rotor, respectively, rad/s

$\boldsymbol{\Phi}$ regressor vector

$\boldsymbol{\Theta}$ constant parametric matrix

$\boldsymbol{A}$ constant plant matrix

$\boldsymbol{a}$ attitude and altitude of the vehicle

$\boldsymbol{B}$ constant control matrix

$\boldsymbol{d}$ location of part CM with respect to the omnicopter body frame, m

$\boldsymbol{f}$ force, N

$\boldsymbol{g}$ gravitational acceleration, m/s^2

$\boldsymbol{p}$ position of the vehicle in the inertial frame, m

$\boldsymbol{q}$ unit quaternion

$\boldsymbol{r}$ reference input

$\boldsymbol{R}(\boldsymbol{q})$ rotation matrix from inertial to body frame

$\boldsymbol{t}$ torque applied to the omnicopter, Nm

$\boldsymbol{x}$ state vector

Matrix Operations

$[\cdot]^*$ true parameter value

$[\cdot]^\dagger$ matrix pseudoinverse

$[\cdot]_\times$ skew symmetric matrix

$\hat{[\cdot]}$ estimate of the parameter

$\overline{[\cdot]}$ augmented matrix, only used in Ch. 3

$\tilde{[\cdot]}$ indicates the difference between the estimated and true parameter value

Omnicopter/Quadrotor Parameters

$\boldsymbol{\kappa}$ diagonal matrix of motor thrust-to-drag ratios, Nm/N

$\boldsymbol{J}$ moment of inertia of the respective part, unbolded value indicates a scalar, $kg \cdot m^2$

$\boldsymbol{L}$ matrix of motor positions relative to omnicopter CM whose columns are $\boldsymbol{l}_i$, m

$\boldsymbol{N}$ matrix of all motor normal vectors whose columns are $\boldsymbol{n}_i$

$\boldsymbol{T}$ thrust and torque map

c yaw torque constant, Nm/N

c_f motor thrust coefficient, $N/(rad/s)^2$

m mass of the part indicated in the subscript, kg

Subscripts/Superscripts

0	initial condition
ω	angular velocity
a	augmented
ad, bl	adaptive, baseline
att, pos	attitude, position
CM	center of mass
cmd	commanded value from controller
d	desired trajectory value
$dist$	disturbance
err	error
est	estimate
i	motor identifier, 1...total number of motors
j	indicates axis of disturbance torque (x,y,z), 1...3
k	point in time, t_0 ... t_{end}, T is total number of timesteps
l	states, 1...L
mot	motor
n	run number, 1...N
p	unaugmented plant
P, I, D	proportional, integral, derivative when modifying a gain matrix

ref reference

sat, $unsat$ saturation filter, unsaturated value

t transformed

x, y, z x,y,and z axes, respectively

1,2 1st, 2nd step of backstepping design

Chapter 1

Introduction

1.1 Motivation

The omnicopter, designed by Brescianini and D'Andrea [3], is a small unmanned aerial vehicle (UAV) capable of executing decoupled translational and rotational motion. Additionally, its thrust and torque capabilities are the same for any vehicle attitude allowing for a full six degree of freedom (6DOF) range of motion with no preferred orientation, setting it apart from other 6DOF controllable multirotor designs [3]. Standard multirotor vehicles, which only enable four degree of freedom (4DOF) control (three translational and one rotational), commonly attach sensors (e.g., a camera) to an undermounted gimbal to provide the additional DOF(s) needed for precise pointing. As the gimbal is mounted under the multi-rotor, the pointing domain is typically limited to around 2π steradians. In contrast, the 6DOF omnicopter provides a full 4π steradians pointing domain without the need for an added gimbal. Thus the omnicopter lends itself well to tasks such as inspection of structures, but can also be leveraged in formation flying applications where specific orientations between nodes must be maintained. It can also be used to emulate the dynamics of free-flying satellites for algorithm testing purposes, and is one of the only options for emulating tumbling. The omnicopter can also be used to help offload gravity for large space structures testing. To perform any of these tasks the omnicopter needs position and attitude control schemes that allow it to hover, follow specified paths, and point at desired locations.

Multirotors have been a popular UAV platform for conducting controls research. A wide range of linear, nonlinear, and intelligent controllers have been designed and implemented on these vehicles, as seen in these survey papers [4, 5, 6, 7]. Therefore, when building a typical multirotor and considering its intended use, there is an extensive selection of pre-developed controllers from which to choose. Additionally, controller performance comparisons exist to aid with controller selection [4, 6, 8, 9, 10, 11]. However, typical multirotors are only 4DOF controllable, so controllers that have been designed for these multirotors will require modification to apply to 6DOF controllable multirotors, and the performance of the 4DOF controller will not be applicable to 6DOF scenarios and use cases.

The development of controllers for various 6DOF controllable multirotors has been much more limited, which makes selecting a controller for a 6DOF controllable multirotor difficult. Typical 6DOF controllable multirotor control schemes that have been simulated and/or implemented on hardware are either PID-based [3, 12, 13, 14, 15, 16, 17, 18, 19, 20] or some type of nonlinear controller [3, 18, 21, 22, 23, 24, 25, 26, 27, 28]. Typically, feedback linearization is used [3, 18, 21, 22, 23]. Other nonlinear controllers include geometric control [24, 25], sliding mode control [27], adaptive finite-time sliding mode control [28], and adaptive backstepping [26]. Additionally, the omnicopter utilizes quaternions to characterize its attitude to avoid singularities that occur with Euler angles. References [3, 20, 21, 29] present attitude controllers for 6DOF controllable multirotors that use quaternions. Identifying which position and attitude controller combination yields the best uncoupled rotational and translational motion is necessary for achieving the best tracking performance and task execution. Only one controller comparison study exists for 6DOF controllable multirotors [29]. However, it is for tilt-rotor quadrotors instead of fixed-rotor UAVs such as the omnicopter. It compares feedback linearization, quaternion based P/PID control (CPID), and quasi-time optimal position tracking/PID attitude tracking (IQTO), determining that feed-

back linearization and IQTO exhibit better pose tracking while IQTO can handle larger initial errors.

Depending on the task, modifications may be made to the omnicopter that change its mass and moment of inertia. For example, adding a camera, switching out a companion computer, or lifting an object will increase its mass and affect its center of mass and moment of inertia. Flying in formations may also subject the omnicopters to varying external airflow. Within the omnicopter, because the blades are angled instead of planar, which is the case for 4DOF controllable multirotors, the downwash of each blade interacts with the other blades causing unknown aerodynamic effects. Additionally, the omnicopter is subject to the same challenges posed by 4DOF controllable multirotor dynamics, especially when tracking trajectories. Their dynamics are nonlinear, open loop unstable [30], and subject to rotor blade aerodynamics, such as blade flapping [31]. Additionally, initial computations of their mass and moment of inertia are subject to uncertainty. During operation, multirotors can be subject to decreases in total thrust, loss of effectiveness in one rotor, and partial damage of a propeller, posing challenges to controller design and effectiveness [31, 32, 33, 34]. Additionally, flying close to structures, such as walls, can cause changes in airflow. Therefore, control schemes that can adapt to parametric uncertainty and be robust to disturbances without sacrificing performance are desirable. One potential solution to address these challenges is the use of adaptive controllers. The purpose of adaptive control is to allow control parameters to change during flight to cancel out or minimize the effects of uncertainties that are present in the system via errors in vehicle modeling or unexpected changes to the vehicle.

1.2 Research Questions

1. What is the best control scheme combination for omnicopter trajectory tracking; specifically, do adaptive controllers perform better than nonadaptive controllers?

2. Is there an intuitive, methodical way to tune MRAC-based adaptive control gains?

3. Were the Simulink simulations used to initially tune and gauge controller performance effective on the hardware implementation of the omnicopter?

1.3 Contributions

1. Design of MRAC and aMRAC controllers for position control of the omnicopter

2. Design of a quaternion-based adaptive backstepping controller for attitude control of the omnicopter

3. A method for tuning an aMRAC quadrotor controller that considers both time delay margin and desired transient response

4. C++ code for the aMRAC/adaptive backstepping and aMRAC/aMRAC controller combinations that can be used in PX4 Firmware. The code can be accessed at https://code.vt.edu/space_at_vt/space_drones/04---rpo-drone-to-space-simulator/omnicopter

5. Comparison of omnicopter controller performance for various uncertainties and disturbances via Simulink simulations and flight tests

1.4 Summary of Chapters

Chapter 2 provides a summary of the current state of adaptive control for 4DOF and 6DOF controllable multirotors as well as of the methods for time delay margin estimation of adaptive systems. Chapter 3 discusses a method for tuning an aMRAC quadrotor controller that considers both time delay margin and desired transient response. Chapter 4 provides an overview of the simulations used to evaluate the controller pairs used in this parametric study. Chapters 5 and 6 discuss the theory of the position controllers and attitude controllers, respectively. For this work, three position controllers (PID, MRAC, and aMRAC) and four attitude controllers (PI/FL, aMRAC, backstepping, and adaptive backstepping) are considered. Chapter 7 provides an analysis and discussion of the simulation results for position and attitude tracking on each combined controller pair under various perturbations. Chapter 8 provides flight test results comparing the PID/PIFL and aMRAC/adaptive backstepping controller combinations. Finally, Chapter 9 presents the conclusions of this work.

Chapter 2

Literature Review

2.1 Adaptive Control of 4DOF Controllable Multirotors

Various adaptive control schemes have been shown to be effective on 4DOF controllable multirotors subject ro parametric uncertainty, wind disturbances, and actuator failures. Model refence adaptive controllers (MRAC) have been investigated by several researchers [31, 32, 35, 36, 37]. MRAC, modified MRAC (MMRAC), and MIT rule MRAC were implemented and compared to the performance of a nominal LQR controller in [32]. For actuator failures, the adaptive controllers outperformed the baseline controllers, with MRAC resulting in the lowest tracking errors for a hover trajectory. MRAC is also shown to outperform a baseline LQR [31] and PD [36, 37] controllers in simulations and flight tests. Additionally, [31] implements modifications to MRAC and combined/composite MRAC (CMRAC) to make them robust to time delays. In [38], MRAC and CMRAC were used to augment a baseline LQR controller for a quadrotor, and both showed improved hover and altitude tracking performance after a decrease in thrust capability, with CMRAC exhibiting the best performance.

Several types of adaptive backstepping controllers [30, 39, 40, 41, 42] and sliding mode controllers [43, 44] have also been designed for quadrotors. An adaptive integral backstepping

controller outperformed a nonadaptive backstepping controller as well as a PID controller in [41]. A finite-time adaptive integral backstepping fast terminal sliding mode controller outperformed nonadaptive and adaptive backstepping controllers as well as an adaptive robust backstepping controller in [45]. [46] develops an adaptive fuzzy global sliding mode controller that outperforms PD, backstepping, boundary layer sliding mode, and active disturbance rejection controllers.

A linear matrix inequality-based adaptive robust controller was shown to have robust transient performance and improved steady state tracking when faced with wind gusts and mass uncertainty and changes during operations such as building inspection and package delivery [47]. An L1 Adaptive controller was used to augment baseline PD/PID controllers to improve trajectory tracking performance of a small quadrotor [48]. In [49], L1 adaptive control was used to augment backstepping as well as to be implemented directly into backstepping with euler angle and quaternion formulations. The direct use of adaptation instead of augmenting a baseline backstepping controller was shown to send a smoother signal. An adaptive feedback linearization controller was shown to improve trajectory tracking of a quadrotor in the presence of parametric uncertainty, mass changes, and disturbances such as wind gusts [50]. To execute aggressive maneuvers, a geometric adaptive controller [51], an adaptive super twisting controller [52], and indirect MRAC augmentation of a baseline geometric controller [53] have been developed. Adaptive controllers using neural networks combined with dynamic inversion have been designed as well [54, 55]. The neural network controller outperforms baseline PD and dynamic model inversion controllers [54].

It appears the most popular adaptive controllers designed for quadrotors are MRAC-based, backstepping, sliding mode, and dynamic inversion. The performance improvements achieved by using adaptive controllers on 4DOF controllable multirotors faced with uncertainties and disturbances, which the omnicopter can also experience, indicates that adaptive control could

also be effective for the omnicopter.

2.2 Adaptive Control of 6DOF Controllable Multirotors

Compared to 4DOF controllable multirotors, little research has been done on the adaptive control of 6DOF controllable multirotors. An adaptive backstepping controller was designed in [26] for position and attitude control of the end effector on a 6DOF controllable aerial manipulator. Additionally, an adaptive finite-time sliding mode controller was designed for the AirBall, a 6DOF controllable vehicle with six propellers on rotating shafts [27]. Specifically designed for the omnicopter, a MRAC and an augmented MRAC (aMRAC) controller were designed in [56] and compared to the PID position controller from [3]. The adaptive position controllers were shown to yield better tracking performance under large disturbances. Additionally, in [57] a backstepping controller, adaptive backstepping controller, and an augmented MRAC (aMRAC) controller were compared to the PI/Feedback linearization (PI/FL) attitude controller from [3]. The PI/FL and aMRAC controllers showed the best attitude tracking performance under large disturbances and moment of inertia changes. The results from [56] and [57] separated position and attitude control. The performance of the controllers may be affected when force and torque commands have to compete for the limited force output of the eight omnicopter motors. Therefore, this research pairs the position controllers that were investigated in [56] with the attitude controllers from [57] to determine which combination yields the best overall position and attitude trajectory tracking for the omnicopter.

2.3 Time Delay Margin Estimation of Adaptive Systems

Despite the reported performance benefits, a problem with MRAC is a lack of guidance for choosing adaptive gains and an ability to quantify robustness with metrics. For MRAC, two open problems are the need to develop guidelines for tuning parameters, and the need for a method to provide insight into the transient performance of the adaptive controller [58]. It is widely acknowledged that the lack of standard certification metrics and methodology is a barrier to the wide-spread implementation of adaptive controllers [59, 60, 61, 62, 63]. Adaptive controllers are inherently nonlinear and therefore the standard metrics for certifying linear controllers, namely gain and phase margin, are not applicable without approximations, which have been explored in [64, 65, 66]. Time delay margins have been identified as a suitable metric for evaluating adaptive controllers because they are inherent to any control system, are related to phase margins, and robustness to time delay also indicates robustness against unmodeled dynamics [60]. It has been shown that for standard MRAC, as the adaptive parameter tuning matrix ($\mathbf{\Gamma}$) increases to infinity, the time delay margin decreases to zero. Increasing $\mathbf{\Gamma}$ also decreases settling time but increases transient oscillations and overshoot of the desired trajectory [67]. However, although the relationship between the adaptive parameter and the transient response can be qualitatively characterized, there is no method for finding the best tradeoff between fast convergence and smoothness [67] for standard MRAC. Guaranteed transient performance of output and input signals while preventing the high frequency oscillations that occur with large adaptation rates can be achieved by implementing a modification to MRAC that augments the reference model to include error feedback [68]. This method also provides guidance on how to select the adaptive parameter and the feedback gain to ensure desired tracking performance. The modification

is shown to apply to combined MRAC (CMRAC) with similar benefits [69], and improved bounds on the transient performance of modified MRAC (MMRAC) have been developed using the projection operator [70]. The methods presented here offer insight into the transient performance of MRAC-based controllers, and in some cases, provide guidance for selecting the adaptive parameter. However, there is no discussion of tuning with regards to time delay margins. Since adaptive rate gain and time-delay margin are related, if a time-delay margin is required to ensure system stability, there will be a maximum adaptive rate gain correlating to that time delay margin. Selecting tuning parameters based on time delay margin has been previously explored in [71, 72].

The challenge with using time delay margin to help tune parameters is the lack of an easy way to analytically compute time delay margins for nonlinear systems. Therefore time delay margins need to be computed via simulations, analytical estimations, or a hybrid combination of simulations and analytic equations [73]. Simulations, particularly monte carlo simulations can be used to compute upper bounds on time delay margins [61, 62, 74]. However, they can be time-consuming and prone to missing important dynamics depending on the reference trajectories and initial conditions selected [61]. For analytical analysis, freezing the system at steady state allows for LTI system analysis, but the results are not suitable for fast maneuvers that transition rapidly through flight regimes [75]. Time delay margin estimation using sum of squares polynomial optimization has been explored but is limited in the number of variables and polynomial degree that can be feasibly computed [74, 76]. Other time delay margin estimation methods include Padé approximations [77, 78], and Razumikin's method [72]. It has also been shown that using the projection operator allows for the analytical computation of the time delay margin that yields global boundedness for the adaptive system [79]. Hybrid methods combine simulation or recorded signals with analytical methods to estimate time delay margins [73]. For example, bounded linear stability

analysis locally estimates the adaptation based on time data before using the estimate in the matrix measure method, Padé approximations, or Lyapunov-Krasovskii with sum of squares optimization to compute time delay margin estimates for the selected time window [1]. Estimates computed with the matrix measure method are shown to be the easiest to compute and the least conservative. Since the estimates are conservative, they can serve as lower bounds on time delay margins.

Chapter 3

Time Delay Margin Tuning of an Adaptive Controller for a Quadrotor

3.1 Introduction

The contribution of this research is the design of an augmented MRAC controller for a quadrotor, considering controller effectiveness uncertainty, and the application of time delay margin estimates to ensure system stability and to help tune $\mathbf{\Gamma}$. The bounded linear stability analysis (BLSA) combined with matrix measure (MM) method for time delay margin estimation previously developed [1] is slightly modified to account for controller effectiveness before being applied to the quadrotor. The BLSA/MM method was chosen as the basis for this work because of its computational ease, especially for a MIMO system. Results in [1] were for SISO and SIMO systems, so this work evaluates its effectiveness for a MIMO quadrotor, a system in wide use. Initial results yielded very conservative time delay margin estimates. Therefore, the modification from [73] is implemented. The analysis required to compute the matrix measure of the system was easily extended to help select a $\mathbf{\Gamma}$ that was an appropriate trade-off between time delay margin and desired transient response. The augmented MRAC controller was chosen for controlling the quadrotor because the LQR controller with integral feedback provides a good baseline for performance comparisons as well as for the reference model. While modifying the reference model can yield desired transients and the use of a

projection operator is a common way to improve robustness, this work is meant focus on how this method works for a basic augmented MRAC controller while considering controller effectiveness.

The rest of this Chapter is structured as follows. Section 3.2 develops the quadrotor dynamics and controllers. Section 3.3 discusses BLSA and MM methods for estimating TDM and their modifications to apply to the quadrotor controllers. Section 3.4 presents simulation results for the quadrotor. Finally, Section 3.5 discusses the simulation results and presents conclusions and future work.

3.2 Quadrotor Dynamics and Controller Development

This section introduces the quadrotor dynamics, deriving the multi-input multi-output (MIMO) linearized dynamics from the original nonlinear position and attitude dynamics. Baseline linear quadratic regulator (LQR) with integral feedback controllers are designed for position and attitude control. The attitude and altitude controller is then made adaptive using the augmented MRAC (aMRAC) method.

3.2.1 Quadrotor Dynamics

The body frame and motor orientation for the quadrotor are shown in Figure 3.1.

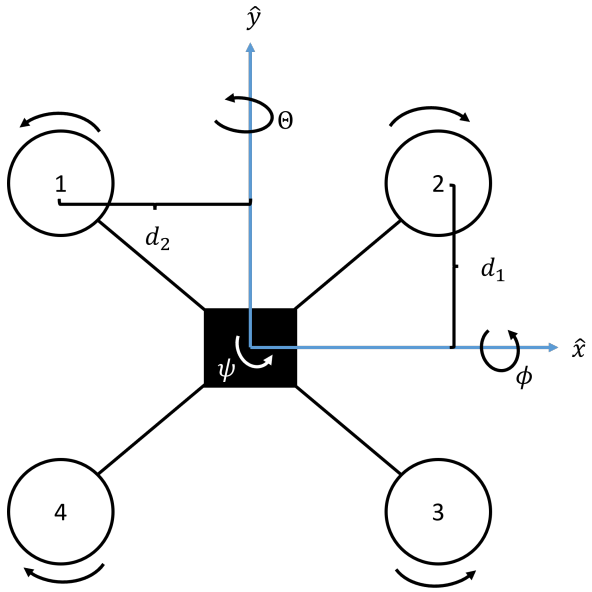


Figure 3.1: Quadrotor body frame and motors

The quadrotor attitude dynamics and kinematics are defined using the standard 3-2-1 Euler angles, where ϕ is roll about the x axis, θ is pitch about the y axis, and ψ is yaw about the z axis. The Euler angle time derivatives are related to the angular velocity, $\boldsymbol{\omega}$, of the quadrotor via the matrix in Eq. (3.1), where *cos*, *sin*, and *tan* are c, s, and t, respectively.

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 1 & \mathrm{s}\phi\mathrm{t}\theta & \mathrm{c}\phi\mathrm{t}\theta \\ 0 & \mathrm{c}\phi & -\mathrm{s}\phi \\ 0 & \frac{\mathrm{s}\phi}{\mathrm{c}\theta} & \frac{\mathrm{c}\phi}{\mathrm{c}\theta} \end{bmatrix} \omega \tag{3.1}$$

The attitude dynamics of the quadrotor expressed in the body frame are shown in Eq. (3.2), where the moment of inertia is $\boldsymbol{J} = diag(J_x, J_y, J_z)$.

$$\boldsymbol{J}\dot{\boldsymbol{\omega}} = \boldsymbol{t} - \boldsymbol{\omega} \times \boldsymbol{J}\boldsymbol{\omega} \tag{3.2}$$

With a small angle assumption, the attitude dynamics become Eq. (3.3).

$$\begin{bmatrix} \ddot{\phi} \\ \ddot{\theta} \\ \ddot{\psi} \end{bmatrix} = \begin{bmatrix} \dot{\theta}\dot{\psi}\frac{J_y - J_z}{J_x} + \frac{d_1}{J_x}t_x \\ \dot{\phi}\dot{\psi}\frac{J_z - J_x}{J_y} + \frac{d_2}{J_y}t_y \\ \dot{\phi}\dot{\theta}\frac{J_x - J_y}{J_z} + \frac{c}{J_z}t_z \end{bmatrix} \tag{3.3}$$

The forces (f) and torques ($\boldsymbol{t} = (t_x, t_y, t_z)^T$) created by the individual motor thrusts ($f_i, i = 1...4$) are mapped in Eq. (3.4) - (3.7), where c_t is the yaw torque constant and d_1 and d_2 are the distance from the quadrotor center of mass to the motor center in the y and x directions, respectively.

$$f = f_1 + f_2 + f_3 + f_4 \tag{3.4}$$

$$t_x = \frac{d_1}{J_x}(f_1 + f_2 - f_3 - f_4) \tag{3.5}$$

$$t_y = \frac{d_2}{J_y}(f_1 - f_2 - f_3 + f_4) \tag{3.6}$$

$$t_z = \frac{c_t}{J_z}(-f_1 + f_2 - f_3 + f_4) \tag{3.7}$$

The quadrotor position dynamics, expressed in the inertial frame, are shown in Eq. (3.8), where the gravitational acceleration is $9.81 m/s^2$ and m is the quadrotor mass.

$$\begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix} = \begin{bmatrix} (\mathrm{c}\phi\mathrm{s}\theta\mathrm{c}\psi + \mathrm{s}\phi\mathrm{s}\psi)\frac{f}{m} \\ (\mathrm{c}\phi\mathrm{s}\theta\mathrm{s}\psi - \mathrm{s}\phi\mathrm{c}\psi)\frac{f}{m} \\ \mathrm{c}\phi\mathrm{c}\theta\frac{f}{m} - g \end{bmatrix} \tag{3.8}$$

The position dynamics use the third column of the direction-cosine matrix in Eq. (3.9), which is the rotation from the body to inertial frame.

$$\boldsymbol{R}_z(\psi)\boldsymbol{R}_y(\theta)\boldsymbol{R}_x(\phi) = \begin{bmatrix} c\theta c\psi & s\theta s\phi c\psi - s\psi c\phi & s\theta c\phi c\psi + s\psi s\phi \\ c\theta s\psi & s\theta s\phi s\psi + c\phi c\psi & s\theta c\phi s\psi - s\phi c\psi \\ -s\theta & c\theta s\phi & c\theta c\phi \end{bmatrix} \tag{3.9}$$

For position and attitude controller development, which will occur in the following sections, the quadrotor dynamics need to be linearized. The chosen equilibrium point for the linearization is the hover point, where all angles are 0. Since the x and y position of the quadrotor are dependent on the roll and pitch, ϕ and θ, angles, the xy-dynamics are separated into an outer loop that feeds desired reference ϕ and θ into an inner loop. The inner loop contains attitude and altitude dynamics, which directly correlate to the control torques and forces that specify thruster force. Yaw angle, ψ, and altitude, z, are independent of roll and pitch and receive their own reference trajectories. Figure 3.2 provides an overview of the outer and inner loop dynamics and controllers.

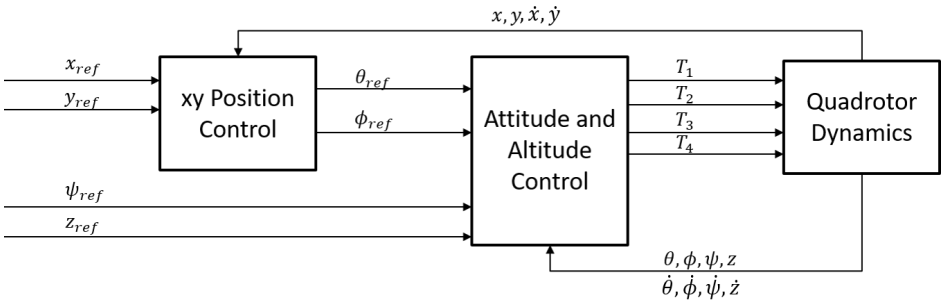


Figure 3.2: Control Overview

The outer loop dynamics for x and y are shown in Eq. (3.10) and take the form: $\dot{\boldsymbol{x}}_{p,pos} = \boldsymbol{A}_{p,pos}\boldsymbol{x}_{p,pos} + \boldsymbol{B}_{p,pos}\boldsymbol{u}_{pos}$.

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \ddot{x} \\ \ddot{y} \end{bmatrix} = \begin{bmatrix} \mathbf{0}_{2\times2} & \boldsymbol{I}_{2\times2} \\ \mathbf{0}_{2\times2} & \mathbf{0}_{2\times2} \end{bmatrix} \begin{bmatrix} x \\ y \\ \dot{x} \\ \dot{y} \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & g \\ -g & 0 \end{bmatrix} \begin{bmatrix} \phi_{ref} \\ \theta_{ref} \end{bmatrix} \tag{3.10}$$

The inner loop attitude and altitude dynamics are shown in Eq. (3.11) and take the form: $\dot{\boldsymbol{x}}_{p,att} = \boldsymbol{A}_{p,att}\boldsymbol{x}_{p,att} + \boldsymbol{B}_{p,att}\boldsymbol{u}_{att}$, where $\boldsymbol{x}_{p,att} = (\phi, \theta, \psi, z, \dot{\phi}, \dot{\theta}, \dot{\psi}, \dot{z})^T$.

$$\dot{\boldsymbol{x}}_{p,att} = \begin{bmatrix} \mathbf{0}_{4\times4} & \boldsymbol{I}_{4\times4} \\ \mathbf{0}_{4\times4} & \mathbf{0}_{4\times4} \end{bmatrix} \boldsymbol{x}_{p,att} + \begin{bmatrix} \mathbf{0}_{4\times1} & \mathbf{0}_{4\times1} & \mathbf{0}_{4\times1} & \mathbf{0}_{4\times1} \\ \frac{d_1}{J_x} & \frac{d_1}{J_x} & -\frac{d_1}{J_x} & -\frac{d_1}{J_x} \\ \frac{d_2}{J_y} & -\frac{d_2}{J_y} & -\frac{d_2}{J_y} & \frac{d_2}{J_y} \\ -\frac{c}{J_z} & \frac{c}{J_z} & -\frac{c}{J_z} & \frac{c}{J_z} \\ \frac{1}{m} & \frac{1}{m} & \frac{1}{m} & \frac{1}{m} \end{bmatrix} \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \end{bmatrix} \tag{3.11}$$

3.2.2 Baseline Control: LQR with Integral Feedback

LQR controllers with integral action were designed for both the inner and outer loops. To design these controllers, the linearized dynamics developed in the previous section were augmented with the trajectory error dynamics.

For the outer loop, the resulting dynamics are $\dot{\boldsymbol{x}}_{a,pos} = \boldsymbol{A}_{a,pos}\boldsymbol{x}_{a,pos} + \boldsymbol{B}_{a,pos}\boldsymbol{u}_{pos} + \boldsymbol{B}_{pos,ref}\boldsymbol{p}_{ref}$:

$$\begin{bmatrix} \dot{\boldsymbol{x}}_{p,pos} \\ \boldsymbol{e}_{pos} \end{bmatrix} = \begin{bmatrix} \boldsymbol{A}_{p,pos} & \mathbf{0}_{4\times2} \\ \boldsymbol{I}_{2\times2} & \mathbf{0}_{2\times4} \end{bmatrix} \begin{bmatrix} \boldsymbol{x}_{p,pos} \\ \int \boldsymbol{e}_p \end{bmatrix} + \begin{bmatrix} \boldsymbol{B}_{p,pos} \\ \mathbf{0}_{2\times2} \end{bmatrix} \begin{bmatrix} \phi_{ref} \\ \theta_{ref} \end{bmatrix} + \begin{bmatrix} \mathbf{0}_{4\times2} \\ -\boldsymbol{I}_{2\times2} \end{bmatrix} \begin{bmatrix} x_{ref} \\ y_{ref} \end{bmatrix} \tag{3.12}$$

where $\boldsymbol{e}_{pos} = \begin{bmatrix} x & y \end{bmatrix}^T - \begin{bmatrix} x_{ref} & y_{ref} \end{bmatrix}^T$ and $\boldsymbol{u}_{pos} = \begin{bmatrix} \phi_{ref} & \theta_{ref} \end{bmatrix}^T = -\boldsymbol{K}_{pos}\boldsymbol{x}_{a,pos}$. $\boldsymbol{K}_{pos}$ is computed via the LQR method. The final selected gains yield all eigenvalues of the closed loop system

$(\boldsymbol{A}_{a,pos} - \boldsymbol{B}_{a,pos}\boldsymbol{K}_{pos})$ with negative real parts.

For the inner loop, the resulting dynamics are $\dot{\boldsymbol{x}}_{a,att} = \boldsymbol{A}_{a,att}\boldsymbol{x}_{a,att} + \boldsymbol{B}_{a,att}\boldsymbol{u}_{att} + \boldsymbol{B}_{att,ref}\boldsymbol{a}_{ref}$:

$$\begin{bmatrix} \dot{\boldsymbol{x}}_{p,att} \\ \boldsymbol{e}_{att} \end{bmatrix} = \begin{bmatrix} \boldsymbol{A}_{p,att} & \boldsymbol{0}_{8\times4} \\ \boldsymbol{I}_{4\times4} & \boldsymbol{0}_{4\times8} \end{bmatrix} \begin{bmatrix} \boldsymbol{x}_{p,att} \\ \int \boldsymbol{e}_{att} \end{bmatrix} + \begin{bmatrix} \boldsymbol{B}_{p,att} \\ \boldsymbol{0}_{4\times4} \end{bmatrix} \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \end{bmatrix} + \begin{bmatrix} \boldsymbol{0}_{8\times4} \\ -\boldsymbol{I}_{4\times4} \end{bmatrix} \begin{bmatrix} \phi_{ref} \\ \theta_{ref} \\ \psi_{ref} \\ z_{ref} \end{bmatrix} \tag{3.13}$$

where $\boldsymbol{e}_{att} = \begin{bmatrix} \phi & \theta & \psi & z \end{bmatrix}^T - \begin{bmatrix} \phi_{ref} & \theta_{ref} & \psi_{ref} & z_{ref} \end{bmatrix}^T$ and $\boldsymbol{u}_{att} = \begin{bmatrix} f_1 & f_2 & f_3 & f_4 \end{bmatrix}^T = -\boldsymbol{K}_{att}\boldsymbol{x}_{a,att}$. $\boldsymbol{K}_{att}$ is computed via the LQR method. The final selected gains yield all eigenvalues of the closed loop system $(\boldsymbol{A}_{a,att} - \boldsymbol{B}_{a,att}\boldsymbol{K}_{att})$ with negative real parts.

For both position and attitude controller tuning, the constrained nonlinear optimization function in MATLAB, fmincon, was used to help tune the weights of $\boldsymbol{Q}_{lqr}$ for each controller. The goal of the optimization was to minimize steady state root mean square trajectory tracking error. The reference trajectories were step inputs that were on the extreme end for each state. Only the weights of $\boldsymbol{Q}_{lqr}$ associated with position or attitude and error were optimized upon in fmincon, not the velocity values. This reduced the computational effort of the optimization while still producing desired responses. $\boldsymbol{R}_{lqr}$ was kept as the identity matrix. It is important to note that the optimization could tune the weights too aggressively, and the results could be reduced by a couple orders of magnitude, improving stability margins while yielding acceptable tracking results.

3.2.3 Augmented Model Reference Adaptive Control (aMRAC)

Although outer and inner loop controllers were designed for the quadrotor, only the inner loop controller is made adaptive. Since the inner loop controller directly affects all control

thrusts, it was determined to be the best controller to adapt upon to affect total quadrotor motion. Although both the outer and inner loop controllers could have been adapted upon, it was decided that multiple adaptations in a system would be redundant and could negatively interfere with each other. Having designed an inner loop LQR controller, an augmented MRAC (aMRAC) controller was the chosen method to make the inner loop controller adaptive. The adaptive portion of the controller is designed to augment the baseline LQR controller, canceling out uncertainties that would prevent the baseline controller from performing with the nominal behavior it was designed to have.

An overview of the aMRAC control scheme is shown in Figure 3.3, and the equations that correspond to each block will be developed subsequently.

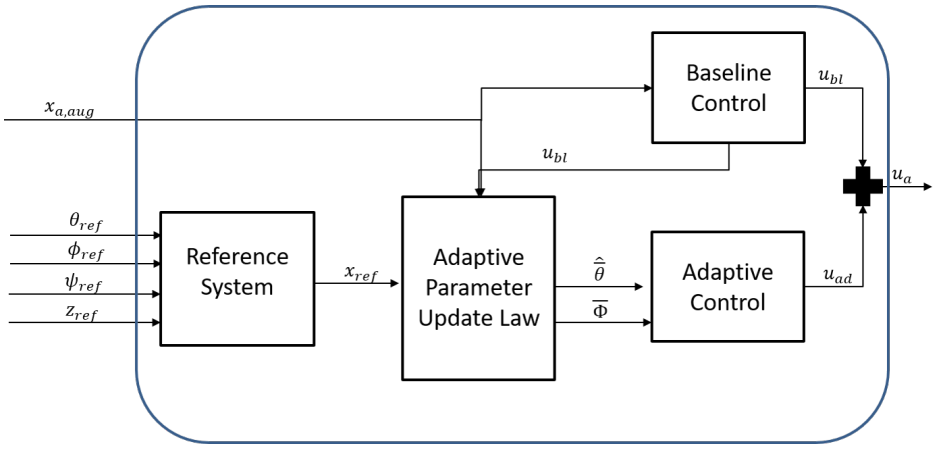


Figure 3.3: aMRAC Overview

The total control fed to the system is $\boldsymbol{u}_{att} = \boldsymbol{u}_{bl} + \boldsymbol{u}_{ad}$. The adaptive controller ($\boldsymbol{u}_{ad}$) augments the baseline controller ($\boldsymbol{u}_{bl}$) to yield the desired performance. The baseline control, $\boldsymbol{u}_{bl} = -\boldsymbol{K}_{att}\boldsymbol{x}_{a,att}$, is the LQR controller designed in the previous section. To develop the adaptive controller, the augmented dynamics that include matched uncertainties are required, shown in Eq. (3.14), where $\boldsymbol{A}_{ref} = \boldsymbol{A}_{a,att} - \boldsymbol{B}_{a,att}\boldsymbol{K}_{att}$, $\overline{\boldsymbol{\Phi}} = \left[\dot{\theta}\dot{\psi}\ \dot{\phi}\dot{\psi}\ \dot{\phi}\dot{\theta}\ \boldsymbol{u}_{bl}^T\right]^T$, $\overline{\boldsymbol{\Theta}}^T = \left[\boldsymbol{\Theta}^{*^T}\ (\boldsymbol{I}_{4\times4} - \boldsymbol{\Lambda}^{-1})\right]$,

and $\boldsymbol{\Lambda}$ is the diagonal controller effectiveness matrix . The matched uncertainties, $\overline{\boldsymbol{\Theta}}^T\overline{\boldsymbol{\Phi}}$ are derived from the small angle assumption version of the attitude dynamics in Eq. (3.3), and the corresponding $\boldsymbol{\Theta}^{*^T}$ is shown in Eq. (3.15). The adaptive controller is chosen to be $\boldsymbol{u}_{ad} = -\hat{\overline{\boldsymbol{\Theta}}}^T\overline{\boldsymbol{\Phi}}$, such that it cancels out the matched uncertainties in the dynamics. The $\hat{}$ indicates an estimate of the parameter. The resulting closed loop system has the same dynamics as the reference model, which has been chosen to have the desired nominal behavior of the baseline controller: $\dot{\boldsymbol{x}}_{ref} = \boldsymbol{A}_{ref}\boldsymbol{x}_{ref} + \boldsymbol{B}_{att,ref}\boldsymbol{a}_{ref}$.

$$\dot{\boldsymbol{x}}_{a,att} = \boldsymbol{A}_{ref}\boldsymbol{x}_{a,att} + \boldsymbol{B}_{a,att}\boldsymbol{\Lambda}(\boldsymbol{u}_{ad} + \overline{\boldsymbol{\Theta}}^T\overline{\boldsymbol{\Phi}}) + \boldsymbol{B}_{att,ref}\boldsymbol{a}_{ref} \tag{3.14}$$

$$\boldsymbol{\Theta}^{*^T} = \begin{bmatrix} \frac{J_y - J_z}{4d_1} & \frac{J_z - J_x}{4d_2} & -\frac{J_x - J_y}{4c} \\ \frac{J_y - J_z}{4d_1} & -\frac{J_z - J_x}{4d_2} & \frac{J_x - J_y}{4c} \\ -\frac{J_y - J_z}{4d_1} & -\frac{J_z - J_x}{4d_2} & -\frac{J_x - J_y}{4c} \\ -\frac{J_y - J_z}{4d_1} & \frac{J_z - J_x}{4d_2} & \frac{J_x - J_y}{4c} \end{bmatrix} \tag{3.15}$$

The adaptive parameter update law is chosen to be $\dot{\hat{\overline{\boldsymbol{\Theta}}}} = \boldsymbol{\Gamma}\overline{\boldsymbol{\Phi}}\boldsymbol{e}^T\boldsymbol{P}\boldsymbol{B}_{a,att}$, where $\boldsymbol{e} = \boldsymbol{x}_{a,att} - \boldsymbol{x}_{ref}$ and $\boldsymbol{\Gamma}$ is the adaptive rate gain matrix. The chosen update law yields a negative Lyapunov function derivative, $\dot{V} = -\boldsymbol{e}^T\boldsymbol{Q}_{lyap}\boldsymbol{e} \leq 0$, where the Lyapunov function is $V = \boldsymbol{e}^T\boldsymbol{P}\boldsymbol{e} + trace(\Delta\overline{\boldsymbol{\Theta}}^T\boldsymbol{\Gamma}^{-1}\Delta\overline{\boldsymbol{\Theta}}\boldsymbol{\Lambda})$ and $\Delta\overline{\boldsymbol{\Theta}} = \hat{\overline{\boldsymbol{\Theta}}} - \overline{\boldsymbol{\Theta}}$. $\boldsymbol{P}$ is the symmetric positive definite solution to the algebraic Lyapunov equation $\boldsymbol{P}\boldsymbol{A}_{ref} + \boldsymbol{A}_{ref}^T\boldsymbol{P} = -\boldsymbol{Q}_{lyap}$, where $\boldsymbol{Q}_{lyap} = \boldsymbol{Q}_{lyap}^T > 0$ is chosen.

3.3 Time Delay Margin Estimation

Having designed adaptive controllers, the challenge becomes tuning the controllers to yield desired tracking performance while maintaining acceptable stability margins, defined by time delay margins. Running multiple simulations to determine the true time delay margin is time consuming. The hybrid method of bounded linear stability analysis (BLSA) and matrix measure (MM) requires only one simulation run to generate data for computing a TDM estimate of an adaptive system. That same simulation run can also provide insight into the transient performance of the selected adaptive tuning parameters.

BLSA is the local approximation for adaptation that allows for linear estimation methods of TDM to be applied to the adaptive system. In this case, the MM method is used because it was shown to be a conservative method for estimating TDM [1]. From a validation standpoint, conservativeness is safer than nonconservativeness. However, it is not beneficial to have TDM estimates that are too conservative, so modifications to the MM method are necessary to yield more realistic estimates.

The BLSA/MM method for computing TDM estimates for MIMO systems that use standard MRAC control and only consider parametric uncertainty in Θ have been explored [1]. This section extends the previously developed theory to aMRAC control of MIMO systems that have uncertainty in controller effectiveness as well as parametric uncertainty.

3.3.1 Bounded Linear Stability Analysis Introduction

Bounded linear stability analysis is introduced in [1] to locally approximate the MIMO adaptive system for standard MRAC within a specified time window. The estimation allows local linear stability analysis to be performed on the system. The plant used is $\dot{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x} +$

$\boldsymbol{B}[\boldsymbol{u} + \boldsymbol{\Theta}^{*T}\boldsymbol{\Phi}]$, only considering parametric uncertainty in $\boldsymbol{\Theta}$, not accounting for controller effectiveness. The $*$ indicates that the true value of $\boldsymbol{\Theta}$ is used, while the $\hat{}$ indicates an estimate of the uncertain parameter is used. The stable reference model used is $\dot{\boldsymbol{x}}_m = \boldsymbol{A}_m\boldsymbol{x}_m + \boldsymbol{B}_m\boldsymbol{r}$. The total control used in the system is $\boldsymbol{u}(t) = -\boldsymbol{K}_x\boldsymbol{x}(t) + \boldsymbol{K}_r\boldsymbol{r}(t) - \boldsymbol{u}_{ad}(t)$, where $\boldsymbol{u}_{ad} = \hat{\boldsymbol{\Theta}}^T\boldsymbol{\Phi}$.

Knowing the adaptive parameter update law is $\dot{\hat{\boldsymbol{\Theta}}} = -\boldsymbol{\Gamma}\boldsymbol{\Phi}\boldsymbol{e}^T\boldsymbol{P}\boldsymbol{B}$, the local bound on the adaptation is approximated by $\dot{\hat{\boldsymbol{\Theta}}}^T\boldsymbol{\Phi} = -\gamma\boldsymbol{B}^T\boldsymbol{P}\boldsymbol{e}$, where $\gamma = \frac{1}{T_0}\int_{t_k - T_0}^{t_k} \boldsymbol{\Phi}^T\boldsymbol{\Gamma}\boldsymbol{\Phi}\, d\tau$ for the time window specified by $t \in [t_k - T_0, t_k)$, $t_0 = 0$, $t_k = t_{k-1} + T_0$, $k = 1, 2, ..., T$ and T_0 is the length of the time window.

The error dynamics for $\boldsymbol{e}_m = \boldsymbol{x}_m - \boldsymbol{x}$ can be estimated as Eq. (3.16) or Eq. (3.17), where t_d is the input time delay, $\boldsymbol{\Theta}_k = \boldsymbol{\Theta}(t_k)$, $(\boldsymbol{\Phi}'_k)_l = \frac{\partial \boldsymbol{\Phi}(\boldsymbol{x}(t_k))}{\partial \boldsymbol{x}_l}$, k denotes the point in time, and $l = 1, ..., L$ denotes the state. Higher order terms are ignored. Assuming step inputs for the reference model, $\dot{\boldsymbol{x}}_{ref}$ and $\dot{\boldsymbol{r}}$ go to 0 over time and terms related to them can be ignored.

$$\ddot{\boldsymbol{e}}_m(t) = [\boldsymbol{A} + \boldsymbol{B}\boldsymbol{\Theta}^{*T}\boldsymbol{\Phi}'_k]\dot{\boldsymbol{e}}_m(t) - [\gamma\boldsymbol{B}\boldsymbol{B}^T\boldsymbol{P}]\boldsymbol{e}_m(t - t_d) - [\boldsymbol{B}\boldsymbol{K}_x + \boldsymbol{B}\hat{\boldsymbol{\Theta}}_k^T\boldsymbol{\Phi}'_k]\dot{\boldsymbol{e}}_m(t - t_d) \quad (3.16)$$

$$\begin{bmatrix} \dot{\boldsymbol{e}}_m(t) \\ \ddot{\boldsymbol{e}}_m(t) \end{bmatrix} = \begin{bmatrix} \boldsymbol{0} & \boldsymbol{I} \\ \boldsymbol{0} & [\boldsymbol{A} + \boldsymbol{B}\boldsymbol{\Theta}^{*T}\boldsymbol{\Phi}'_k] \end{bmatrix} \begin{bmatrix} e_m(t) \\ \dot{e}_m(t) \end{bmatrix} - \begin{bmatrix} \boldsymbol{0} & \boldsymbol{0} \\ \gamma\boldsymbol{B}\boldsymbol{B}^T\boldsymbol{P} & [\boldsymbol{B}\boldsymbol{K}_x + \boldsymbol{B}\hat{\boldsymbol{\Theta}}_k^T\boldsymbol{\Phi}'_k] \end{bmatrix} \begin{bmatrix} \boldsymbol{e}_m(t - t_d) \\ \dot{\boldsymbol{e}}_m(t - t_d) \end{bmatrix} \quad (3.17)$$

3.3.2 BLSA modified for aMRAC

The local bound used for MRAC is applicable for aMRAC, and the error dynamics only require slight modification. The plant used is $\dot{\boldsymbol{x}} = \boldsymbol{A}_a\boldsymbol{x}_a + \boldsymbol{B}_a\boldsymbol{\Lambda}[\boldsymbol{u} + \boldsymbol{\Theta}^{*T}\boldsymbol{\Phi}] + \boldsymbol{B}_{ref}r$, including

uncertainty in the control effectiveness, $\boldsymbol{\Lambda}$. The subscript a indicates the plant dynamics have been augmented with integral error. The stable reference model is $\dot{\boldsymbol{x}}_{ref} = \boldsymbol{A}_{ref}\boldsymbol{x}_{ref} + \boldsymbol{B}_{ref}\boldsymbol{r}$. The total control is $\boldsymbol{u}(t) = \boldsymbol{u}_{bl}(t) + \boldsymbol{u}_{ad}(t) = -\boldsymbol{K}\boldsymbol{x}_a(t) - \hat{\overline{\boldsymbol{\Theta}}}^T\overline{\boldsymbol{\Phi}}$. When there is input time delay, the plant can be rewritten as Eq. (3.18), where $\overline{\boldsymbol{\Theta}}^{*T} = [\,\boldsymbol{\Theta}^* \;(\boldsymbol{I}-\boldsymbol{\Lambda}^{-1})\,]^T$ and $\overline{\boldsymbol{\Phi}} = [\,\boldsymbol{\Phi}(t)^T \;\boldsymbol{u}_{bl}(t)^T\,]^T$.

$$\begin{aligned}\dot{\boldsymbol{x}}_a(t) &= \boldsymbol{A}_a\boldsymbol{x}_a(t) + \boldsymbol{B}_a\boldsymbol{\Lambda}[\boldsymbol{u}(t-t_d) + \boldsymbol{\Theta}^{*T}\boldsymbol{\Phi}] + \boldsymbol{B}_{ref}r(t)\\ &= \boldsymbol{A}_a\boldsymbol{x}_a(t) - \boldsymbol{B}_a\boldsymbol{K}\boldsymbol{x}_a(t-t_d) + \boldsymbol{B}_a\boldsymbol{\Lambda}[\boldsymbol{u}_{ad}(t-t_d) + \overline{\boldsymbol{\Theta}}^{*T}\overline{\boldsymbol{\Phi}}(t)]\end{aligned} \tag{3.18}$$

The local bound on the adaptation remains the same as it was from [1]: $\dot{\hat{\overline{\boldsymbol{\Theta}}}}^T\boldsymbol{\Phi} = -\gamma\boldsymbol{B}^T\boldsymbol{P}\boldsymbol{e}$. The computation of γ also remains the same, now using $\overline{\boldsymbol{\Phi}}$ instead of $\boldsymbol{\Phi}$.

The error dynamics for $\boldsymbol{e} = \boldsymbol{x}_a - \boldsymbol{x}_{ref}$ can be estimated as Eq . (3.19) or Eq. (3.20), essentially the same as Eq . (3.16) or Eq. (3.17), but with the inclusion of $\boldsymbol{\Lambda}$ where applicable.

$$\ddot{\boldsymbol{e}}(t) = [\boldsymbol{A}_a + \boldsymbol{B}_a\boldsymbol{\Lambda}\overline{\boldsymbol{\Theta}}^{*T}\overline{\boldsymbol{\Phi}}_k']\dot{\boldsymbol{e}}(t) - [\gamma\boldsymbol{B}_a\boldsymbol{\Lambda}\boldsymbol{B}_a^T\boldsymbol{P}]\boldsymbol{e}(t-t_d) - [\boldsymbol{B}_a\boldsymbol{K} + \boldsymbol{B}_a\boldsymbol{\Lambda}\hat{\overline{\boldsymbol{\Theta}}}_k^T\overline{\boldsymbol{\Phi}}_k']\dot{\boldsymbol{e}}(t-t_d) \tag{3.19}$$

$$\begin{bmatrix}\dot{\boldsymbol{e}}(t)\\ \ddot{\boldsymbol{e}}(t)\end{bmatrix} = \begin{bmatrix}\boldsymbol{0} & \boldsymbol{I}\\ \boldsymbol{0} & [\boldsymbol{A}_a + \boldsymbol{B}_a\boldsymbol{\Lambda}\overline{\boldsymbol{\Theta}}^{*T}\overline{\boldsymbol{\Phi}}_k']\end{bmatrix}\begin{bmatrix}\boldsymbol{e}(t)\\ \dot{\boldsymbol{e}}(t)\end{bmatrix} - \begin{bmatrix}\boldsymbol{0} & \boldsymbol{0}\\ \gamma\boldsymbol{B}_a\boldsymbol{\Lambda}\boldsymbol{B}_a^T\boldsymbol{P} & [\boldsymbol{B}_a\boldsymbol{K} + \boldsymbol{B}_a\boldsymbol{\Lambda}\hat{\overline{\boldsymbol{\Theta}}}_k^T\overline{\boldsymbol{\Phi}}_k']\end{bmatrix}\begin{bmatrix}\boldsymbol{e}(t-t_d)\\ \dot{\boldsymbol{e}}(t-t_d)\end{bmatrix} \tag{3.20}$$

3.3.3 Matrix Measure

The chosen TDM estimation method is the matrix measure method because it was shown to be conservative, but not excessively so, and easy to compute [1]. The local approximations of the error dynamics computed using BLSA are used by the matrix measure method to compute the TDM for the adaptive system. Although the TDM is estimated for a local time window, the TDM for the system is taken as the average of the estimates for the entire time.

The matrix measure (μ) is defined as Eq. (3.21), the eigenvalues of the symmetric part of a complex matrix [1, 80]. $\boldsymbol{C}^*$ is the complex conjugate transpose of a matrix $\boldsymbol{C}$.

$$\mu(\boldsymbol{C}) = \lambda\left(\frac{\boldsymbol{C}+\boldsymbol{C}^*}{2}\right) \tag{3.21}$$

The matrix measure properties pertinent to estimating time delay margin are in Eq. (3.22) -(3.27) [1].

$$\mu(\boldsymbol{C}) \in \mathbb{R} \tag{3.22}$$

$$\overline{\mu}(\boldsymbol{C}) = \max \mu(\boldsymbol{C}) \tag{3.23}$$

$$\underline{\mu}(\boldsymbol{C}) = \min \mu(\boldsymbol{C}) \tag{3.24}$$

$$\underline{\mu}(\boldsymbol{C}) \leq \mathrm{Re}\lambda(\boldsymbol{C}) \leq \overline{\mu}(\boldsymbol{C}) \tag{3.25}$$

$$\mathrm{Im}\lambda(\boldsymbol{C}) \leq \overline{\mu}(-j\boldsymbol{C}) \tag{3.26}$$

$$\overline{\mu}(\boldsymbol{C}) \leq ||\boldsymbol{C}|| \tag{3.27}$$

The TDM can be estimated using the matrix measure method on one of two forms of the error dynamics [1]. The first formulation uses the error dynamics written in matrix form, which is Eq. (3.17) for standard MRAC and Eq. (3.20) for aMRAC, represented as

$\dot{\boldsymbol{z}}(t) = \boldsymbol{C}_k \boldsymbol{z}(t) - \boldsymbol{D}_k \boldsymbol{z}(t - t_d)$. Therefore, $\boldsymbol{z}(t) = [\, \boldsymbol{e}_{m(t)^T}\ \dot{\boldsymbol{e}}_{m(t)^T}\,]^T$ for standard MRAC and $\boldsymbol{z}(t) = [\, \boldsymbol{e}_{(t)^T}\ \dot{\boldsymbol{e}}_{(t)^T}\,]^T$ for aMRAC , where $\boldsymbol{C}_k$ is the state matrix that corresponds with the non time delayed errors and $\boldsymbol{D}_k$ is the state matrix corresponding with the time delayed errors at time k. The resulting time delay margin is Eqs. (3.28) and (3.29). The full derivation of this equation can be seen in Appendix C.1.

$$TDM_k \leq \frac{1}{\omega_k} \cos^{-1}\left(\frac{\overline{\mu}(\boldsymbol{C}_k) + \overline{\mu}(j\boldsymbol{D}_k)}{||\boldsymbol{D}_k||}\right) \tag{3.28}$$

$$\omega_k = \overline{\mu}(-j\boldsymbol{C}_k) + ||\boldsymbol{D}_k|| \tag{3.29}$$

The second formulation derives the time delay margin from the characteristic equation of the error dynamics equation, which is Eq. (3.16) for the standard MRAC scenario. For standard MRAC, the resulting time delay margin is shown in Eqs. (3.30) and (3.31). The full derivation of this equation can be seen in Appendix C.1.

$$TDM_k \leq \frac{1}{\omega_k} cos^{-1}\left[\frac{\omega_k \overline{\mu}(\boldsymbol{A} + \boldsymbol{B}\boldsymbol{\Theta}^{*T}\boldsymbol{\Phi}'_k)[\omega_k||\boldsymbol{B}\boldsymbol{K}_x + \boldsymbol{B}\hat{\boldsymbol{\Theta}}_k^T\boldsymbol{\Phi}'_k|| - \overline{\mu}(j\gamma\boldsymbol{B}\boldsymbol{B}^T\boldsymbol{P})]}{\omega_k^2||\boldsymbol{B}\boldsymbol{K}_x + \boldsymbol{B}\hat{\boldsymbol{\Theta}}_k^T\boldsymbol{\Phi}'_k||^2 + ||\gamma\boldsymbol{B}\boldsymbol{B}^T\boldsymbol{P}||^2} + \frac{[\omega_k\overline{\mu}[j(\boldsymbol{B}\boldsymbol{K}_x + \boldsymbol{B}\hat{\boldsymbol{\Theta}}_k^T\boldsymbol{\Phi}'_k)] + ||\gamma\boldsymbol{B}\boldsymbol{B}^T\boldsymbol{P}||]}{\sqrt{\omega_k^2||\boldsymbol{B}\boldsymbol{K}_x + \boldsymbol{B}\hat{\boldsymbol{\Theta}}_k^T\boldsymbol{\Phi}'_k||^2 + ||\gamma\boldsymbol{B}\boldsymbol{B}^T\boldsymbol{P}||^2}}\right] \tag{3.30}$$

$$\omega_k = \frac{\overline{\mu}(-j(\boldsymbol{A}+\boldsymbol{B}\boldsymbol{\Theta}^{*^T}\boldsymbol{\Phi}'_k)) + ||\boldsymbol{B}\boldsymbol{K}_x + \boldsymbol{B}\hat{\boldsymbol{\Theta}}_k^T\boldsymbol{\Phi}'_k||}{2} \left(1+\sqrt{1+\frac{4||\gamma\boldsymbol{B}\boldsymbol{B}^T\boldsymbol{P}||}{\left(\overline{\mu}(-j(\boldsymbol{A}+\boldsymbol{B}\boldsymbol{\Theta}^{*^T}\boldsymbol{\Phi}'_k)) + ||\boldsymbol{B}\boldsymbol{K}_x + \boldsymbol{B}\hat{\boldsymbol{\Theta}}_k^T\boldsymbol{\Phi}'_k||\right)^2}}\right) \quad (3.31)$$

For aMRAC, because the error equation is slightly modified to include controller effectiveness uncertainty (Λ) as well as the augmented matrices ($\overline{\boldsymbol{\Theta}}$, $\overline{\boldsymbol{\Phi}}$) associated with the closed loop plant model, the time delay margin estimation equations are slightly changed to include these terms as well, see Eqs. (3.32) and (3.33). $\boldsymbol{\Lambda}$ is not present with every $\boldsymbol{B}$ matrix because those values represent the baseline controller portion of the closed loop plant model, which does not associate with controller effectiveness uncertainty. For readability, the a subscripts have been left off the $\boldsymbol{A}$ and $\boldsymbol{B}$ matrices in Eqs. (3.32) and (3.33). The full derivation of this equation can be seen in Appendix C.2.

$$TDM_k \leq \frac{1}{\omega_k} cos^{-1}\left[\frac{\omega_k\overline{\mu}(\boldsymbol{A}+\boldsymbol{B}\boldsymbol{\Lambda}\overline{\boldsymbol{\Theta}}^{*^T}\overline{\boldsymbol{\Phi}}'_k)[\omega_k||\boldsymbol{B}\boldsymbol{K}+\boldsymbol{B}\boldsymbol{\Lambda}\hat{\overline{\boldsymbol{\Theta}}}_k^T\overline{\boldsymbol{\Phi}}'_k|| - \overline{\mu}(j\gamma\boldsymbol{B}\boldsymbol{\Lambda}\boldsymbol{B}^T\boldsymbol{P})]}{\omega_i^2||\boldsymbol{B}\boldsymbol{K}+\boldsymbol{B}\boldsymbol{\Lambda}\hat{\overline{\boldsymbol{\Theta}}}_k^T\overline{\boldsymbol{\Phi}}'_k||^2 + ||\gamma\boldsymbol{B}\boldsymbol{\Lambda}\boldsymbol{B}^T\boldsymbol{P}||^2} + \frac{[\omega_k\overline{\mu}[j(\boldsymbol{B}\boldsymbol{K}+\boldsymbol{B}\boldsymbol{\Lambda}\hat{\overline{\boldsymbol{\Theta}}}_k^T\overline{\boldsymbol{\Phi}}'_k)] + ||\gamma\boldsymbol{\Lambda}\boldsymbol{B}\boldsymbol{B}^T\boldsymbol{P}||]}{\sqrt{\omega_k^2||\boldsymbol{B}\boldsymbol{K}_x+\boldsymbol{B}\boldsymbol{\Lambda}\hat{\overline{\boldsymbol{\Theta}}}_k^T\overline{\boldsymbol{\Phi}}'_k||^2 + ||\gamma\boldsymbol{B}\boldsymbol{\Lambda}\boldsymbol{B}^T\boldsymbol{P}||^2}}\right] \quad (3.32)$$

$$\omega_k = \frac{\overline{\mu}[-j(\boldsymbol{A}+\boldsymbol{B}\boldsymbol{\Lambda}\overline{\boldsymbol{\Theta}}^{*^T}\overline{\boldsymbol{\Phi}}'_k)] + ||\boldsymbol{B}\boldsymbol{K}+\boldsymbol{B}\boldsymbol{\Lambda}\hat{\overline{\boldsymbol{\Theta}}}_k^T\overline{\boldsymbol{\Phi}}'_k||}{2} \left(1+\sqrt{1+\frac{4||\gamma\boldsymbol{B}\boldsymbol{\Lambda}\boldsymbol{B}^T\boldsymbol{P}||}{(\overline{\mu}(-j(\boldsymbol{A}+\boldsymbol{B}\boldsymbol{\Lambda}\overline{\boldsymbol{\Theta}}^{*^T}\overline{\boldsymbol{\Phi}}'_k)) + ||\boldsymbol{B}\boldsymbol{K}+\boldsymbol{B}\boldsymbol{\Lambda}\hat{\overline{\boldsymbol{\Theta}}}_k^T\overline{\boldsymbol{\Phi}}'_k||)^2}}\right) \quad (3.33)$$

For a quadrotor, the $\overline{\mathbf{\Phi}}_k'$ matrix is shown in Eq. (3.34)

$$\overline{\mathbf{\Phi}}_k' = \begin{bmatrix} \mathbf{0}_{1\times4} & 0 & \dot{\psi} & \dot{\theta} & 0 & 0_{1\times4} \\ \mathbf{0}_{1\times4} & \dot{\psi} & 0 & \dot{\phi} & 0 & \mathbf{0}_{1\times4} \\ \mathbf{0}_{1\times4} & \dot{\theta} & \dot{\phi} & 0 & 0 & \mathbf{0}_{1\times4} \\ -\boldsymbol{K}_{att,(:,1:4)} & -\boldsymbol{K}_{att,(:,5)} & -\boldsymbol{K}_{att,(:,6)} & -\boldsymbol{K}_{a,(:,7)} & -\boldsymbol{K}_{att,(:,8)} & -\boldsymbol{K}_{att,(:,9:12)} \end{bmatrix} \tag{3.34}$$

3.3.4 Matrix Measure Modification

In some scenarios, the matrix measure can yield results that are too conservative to be useful. This occurs because the matrix measure and norms are not invariant to similarity transformations [73]. An optimal matrix transformation for the state matrices, $\boldsymbol{C}$ and $\boldsymbol{D}$, can be found that makes the results of matrix measure TDM estimates less conservative [73]. The TDM estimation considered is in the form from Eq. (3.28) and Eq. (3.29). The transformed state matrices are $\boldsymbol{C}_{t,k} = \boldsymbol{S}^{-1}\boldsymbol{C}_k\boldsymbol{S}$ and $\boldsymbol{D}_{t,k} = \boldsymbol{S}^{-1}\boldsymbol{D}_k\boldsymbol{S}$, where $\boldsymbol{S}$ is the optimal state transformation matrix to be found.

Recalling from Eq. (3.27) that $\overline{\mu}(\boldsymbol{C}) \leq ||\boldsymbol{C}||$, the equations can be written in terms of norms as Eq. (3.35).

$$\omega \leq ||\boldsymbol{C}_k||_2 + ||\boldsymbol{D}_k||_2 \tag{3.35}$$

The cost function (J) used to optimize the transformation matrix is selected as Eq. (3.36), minimizing the matrix norms.

$$J = ||\boldsymbol{S}^{-1}\boldsymbol{C}_k\boldsymbol{S}||_2 + ||\boldsymbol{S}^{-1}\boldsymbol{D}_k\boldsymbol{S}||_2 \tag{3.36}$$

The gradient of the cost function is Eq. (3.37), where F indicates the Frobenius norm.

$$\frac{\partial J}{\partial \boldsymbol{S}} = \boldsymbol{S}^{-T}\left(\frac{\boldsymbol{C}_{t,k}^{T}\boldsymbol{C}_{t,k} - \boldsymbol{C}_{t,i}\boldsymbol{C}_{t,k}^{T}}{||\boldsymbol{C}_{t,k}||_{F}} + \frac{\boldsymbol{D}_{t,k}^{T}\boldsymbol{D}_{t,k} - \boldsymbol{D}_{t,k}\boldsymbol{D}_{t,k}^{T}}{||\boldsymbol{D}_{t,k}||_{F}}\right) \tag{3.37}$$

The optimization problem is solved using an interior-point method, fmincon in MATLAB. The constraint is chosen such that the value within $\cos^{-1}$ in Eq. (3.28) is less than or equal to one, preventing imaginary numbers. The resulting constraint is Eq. (3.38). For the nonmodified matrix measure method, imaginary TDM estimates are prevented by using the absolute value of the estimate.

$$0 > -||\boldsymbol{D}_{t,k}|| + \overline{\mu}(\boldsymbol{C}_{t,k}) + \overline{\mu}(j\boldsymbol{D}_{t,k}) \tag{3.38}$$

This matrix measure modification is used in conjunction with the BLSA estimates to compute the "modified MM" time delay margin estimates of aMRAC control of a quadrotor in the next section. The TDM estimates computed with Eqs. (3.32) and (3.33) provide the "nominal MM" results.

3.4 Simulation Results

For TDM estimation, only the inner loop attitude and altitude controller is considered. Step inputs are fed directly into ϕ and θ instead of being computed from the outer loop position controller because the inner control loop has to run faster and can therefore tolerate less time delay margin than the outer loop controller. Therefore, the inner loop is the limiting factor for TDM in the system.

A fourth order Runge-Kutta solver (ODE4 in Simulink) is used to compute simulations at

a time step of 0.001 seconds. The trajectories tested are 5 degree step inputs for ϕ, θ,and ψ, within the limits of the linearization. z is commanded to rise to 2.25 m from a 1 m initial position. A change of 1.25 m is approximately half the maximum height a quadrotor could rise in an indoor laboratory. A starting point of 1 m above the floor prevents the need to impose a floor constraint within the dynamics. The trajectories were run for 10 seconds, approximately 3 times the maximum settling time of 4 seconds for the z dynamics. Although the trajectories are run for 10 seconds, the figures in this section only show the first 3 seconds of simulation to highlight the transient response of the states.

For the modified matrix measure method, the initial guess at the first simulation time step is the identity matrix. The optimization for the subsequent time steps is initialized with the optimized $\boldsymbol{S}$ matrix from the previous time step. The maximum function evaluations and maximum iterations are both set to $1e-6$.

The true TDM are computed to within +/- 0.001 seconds using simulations to find the time delay at which the system becomes unstable, when error grows unbounded.. The same amount of time delay was input into each thrust command when computing the true TDM. It was also assumed that the controllers had unlimited control authority.

For the time delay margin estimates, a time window of $T_0 = 0.1$ seconds was used to compute γ and the resulting adaptation estimate. 0.1 is approximately $\frac{1}{3}$ the fastest rise time of about 0.3 seconds for ϕ and θ, using the guidelines from [73].

The quadrotor parameters used for simulations are shown in Table 3.1, representative of an indoor research quadrotor. The tuning parameters used for the controllers are shown in Table 3.2. Although the position controller was not used in simulations for TDM computation, the values are included here for completeness.

The controllers were stressed by reducing the effectiveness of the first thruster, trying 0.75,

Table 3.1: Quadrotor Parameters

Parameter	Value	Unit
m	1.121	kg
$\boldsymbol{J}$	$diag$[0.01 0.0082 0.0148]	Nm
d_1	0.1068	m
d_2	0.0879	m
c	0.01	$\frac{Nm}{N}$

Table 3.2: Controller Tuning Parameters

Position LQR Control	
$\boldsymbol{Q}_{lqr,pos}$	diag(0.1, 0, 0, 0, 0.3, 0.2)
$\boldsymbol{R}_{lqr,pos}$	$\boldsymbol{I}_{2\times2}$
Attitude LQR Control	
$\boldsymbol{Q}_{lqr,att}$	diag(2.1, 0.1, 7.7, 22.9, 0, 0, 0, 0, 200.4, 200.5, 200, 98.5)
$\boldsymbol{R}_{lqr,att}$	$\boldsymbol{I}_{4\times4}$
Attitude aMRAC Control	
$\boldsymbol{Q}_{lyap}$	$\boldsymbol{I}_{12\times12}$

0.5, and 0.25 effectiveness, affecting the first diagonal value of Λ. Note that Λ is a diagonal matrix, but since controller effectiveness is only reduced for the first thruster, the reduced value is the only value presented subsequently when discussing Λ. All other diagonal values are 1.

The process for selecting $\boldsymbol{\Gamma}$ is typically iterative, involving trial and error [81]. Since the matrix measure method requires the running of simulations, minimal effort is required to include varying $\boldsymbol{\Gamma}$ along with the simulations that vary $\boldsymbol{\Lambda}$. For the quadrotor, all diagonals of $\boldsymbol{\Gamma}$ were set to the same value as a simple starting point. The magnitude of Γ can then be changed to span feasible choices for the system. For the quadrotor, $\boldsymbol{\Gamma}$ was varied across three orders of magnitude: from 1-10 in steps of 1, 10-100 in steps of 10, and 100-1000 in steps of 100. Based on the state responses, the appropriate order of magnitude can be narrowed down. The variation from 1-10 for each order of magnitude allows for further narrowing down of the appropriate $\boldsymbol{\Gamma}$ value. Further tuning, varying the diagonals, can be performed

after that if desired. Since $\mathbf{\Gamma}$ is a matrix with the same diagonal values, only the diagonal value will be referred to in subsequent discussions of Γ. It can also be referred to as the Γ multiplier.

3.4.1 aMRAC MIMO Results

The transient state responses for varying $\mathbf{\Lambda}$ and $\mathbf{\Gamma}$ for the MIMO system are shown in Figure 3.4. When the quadrotor has full controller effectiveness ($\mathbf{\Lambda} = \boldsymbol{I}_{4\times4}$), there is not a significant difference in controller performance for all $\mathbf{\Gamma}$ values. Only the z reponse indicates a difference, with larger drops for smaller Γ values, which is seen across all Λ values. The differences between the Γ values become more pronounced as thruster effectiveness is decreased, especially for ϕ, θ, and ψ, with ϕ and θ exhibiting similar behavior. Figure 3.5 portrays the transient ϕ response at each magnitude of Γ. Γ values in the 1-10 range yield higher overshoots, longer settling times, and larger, slower transient oscillations than in the 10-1000 range. While the 100-1000 Γ values generally yield a smoother transient response, there is an initial rapid oscillation that does not appear in lower magnitude Γ values. Additionally, for $\Lambda = 0.25$, oscillations make a reappearance in the ϕ and θ terms around 1 second into flight. For Γ values in the 10-100 range, oscillations appear in the rise, as with Γ = 1-10, but overall the rise time is much quicker and the oscillations are smaller in magnitude. For values greater than 20, it becomes hard to distinguish individual responses, indicating those values yield similar transient responses. The significant differences in oscillation magnitude and settling time can also be seen in the ψ responses in Figure 3.4. These characteristics indicate that the ideal Γ for quadrotor control lies in the 30-100 range.

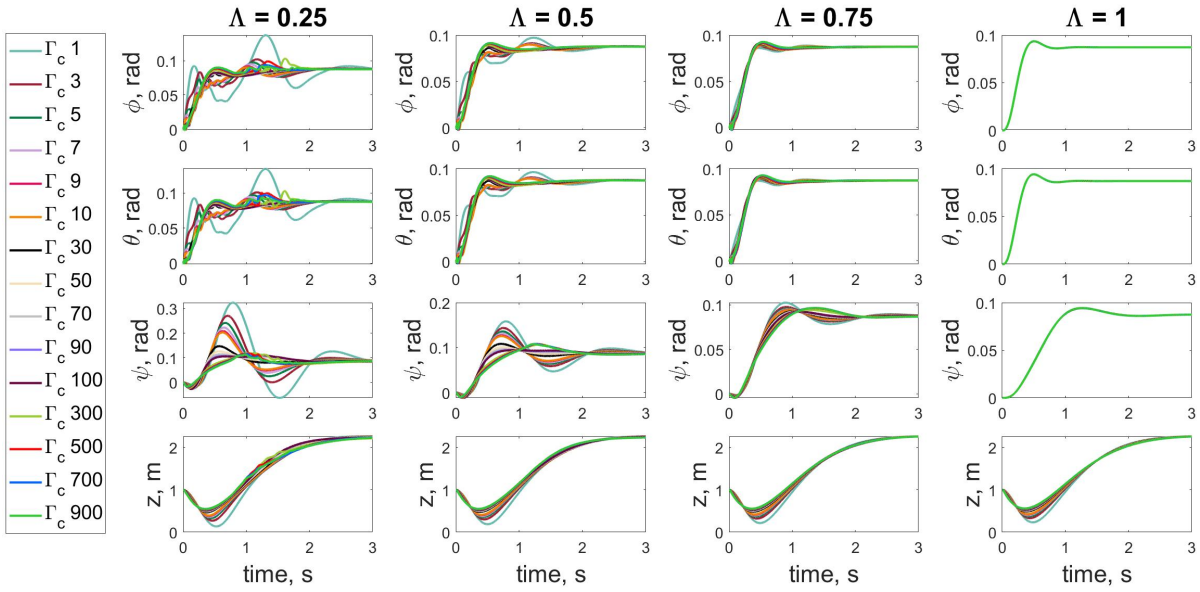


Figure 3.4: aMRAC Transient State Response

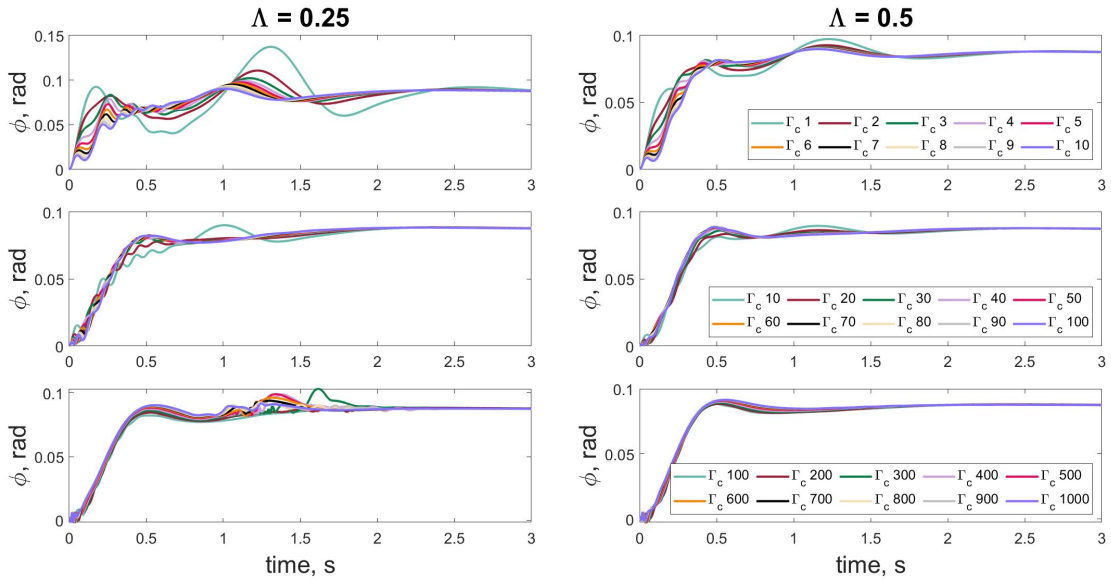
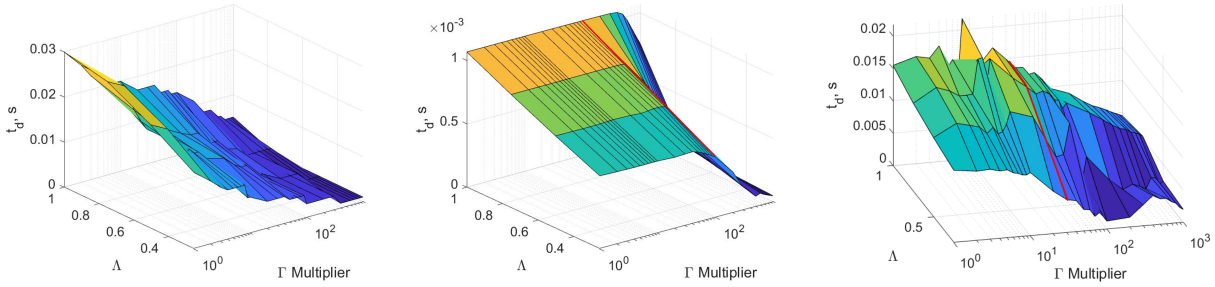


Figure 3.5: aMRAC ϕ Transient State Response for Reduced Thruster Capability

However, as was discussed before, there is a tradeoff between Γ and TDM, and the goal is to

find a balance between controller effectiveness and necessary TDM. The surface plots in Figure 3.6 show how the TDM truth and estimates vary with Γ and Λ. The true TDM indicates that TDM steadily decreases with increasing Γ, and that decreasing controller effectiveness causes slight decreases in TDM. The TDM estimates tend to have a slight plateau before a sharp drop in the TDM estimates, which occurs around $\Gamma = 100$ for the nominal matrix measure method and around $\Gamma = 30$ for the modified matrix measure method. If there is no specific TDM that must be achieved, the ideal Γ value would be at the end of the plateau, the highest Γ for an acceptable TDM. For this quadrotor, the value would be 30 for a true TDM of 0.006 - 0.017 seconds for minimum and maximum Λ values. Red lines in Figure 3.6 indicate the edge of the plateaus.



(a) True (b) Nominal MM (c) Modified MM

Figure 3.6: Surface plots comparing time delay margin with Gamma and Lambda

Looking at TDM estimate comparisons to the true TDM values in Fig. 3.7, the unmodified matrix measure estimates are several orders of magnitude too conservative, on the order of 10^{-3} or 10^{-4}, especially for small Γ values. The disparity decreases as Γ increases to the 100 order of magnitude.

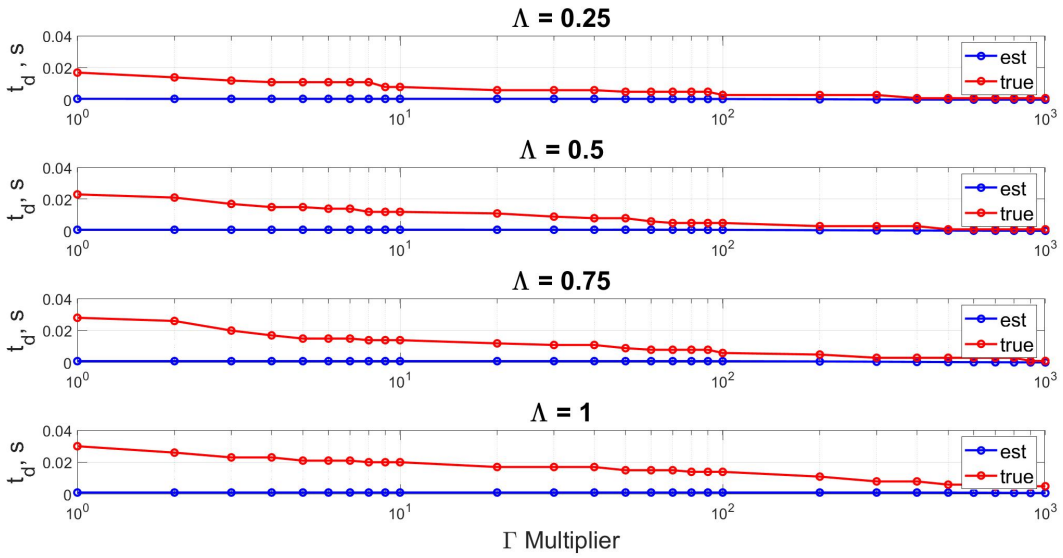


Figure 3.7: aMRAC True TDM vs TDM estimate using nominal MM method

The modified matrix measure method was effective at decreasing the conservativeness of the TDM estimates, very closely approximating the truth for most values, as seen in Figure 3.8. However, as with the nonmodified MM method, estimates for low Γ values are not as accurate as those for higher Γ values. Additionally, some of the estimates became nonconservative, particularly for the $\Lambda = 0.5$ and $\Lambda = 0.75$ scenarios. It is also important to note that the optimizations required for the modified MM method take a longer time to run than the unmodified MM method. However, the estimate improvements are worth the additional computation time because they provide more realistic insight into the truth values when tuning gains. Unfortunately, the results cannot be guaranteed to be conservative, which hinders the usefulness for validation and verification purposes.

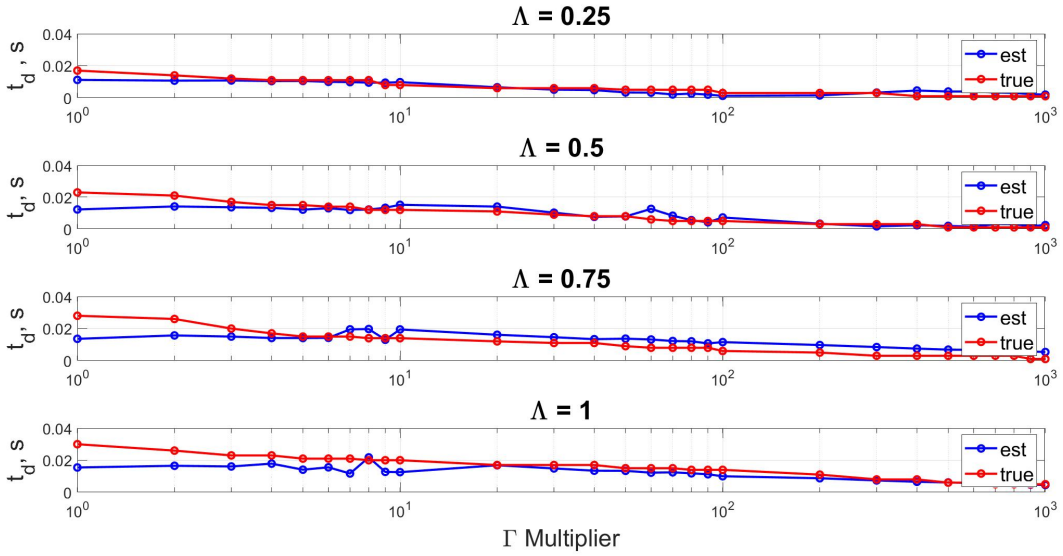


Figure 3.8: aMRAC True TDM vs TDM estimate using modified MM method

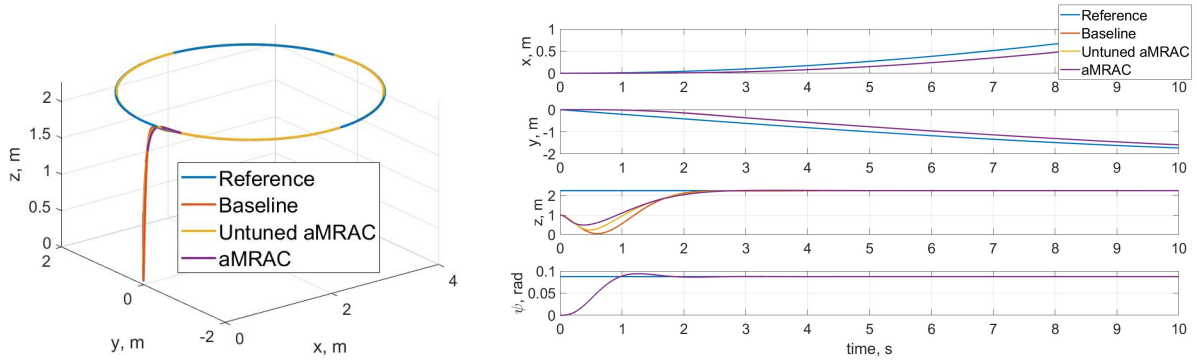
3.4.2 LTI Baseline Controller Results

The time delay margin estimate for the baseline controller applied to the LTI system without uncertainty can be estimated with the matrix margin method using Eqs. (3.28) and (3.29), where $\boldsymbol{C} = \boldsymbol{A}_{a,att}$ and $\boldsymbol{D} = \boldsymbol{B}_{a,att}\boldsymbol{K}_{att}$. The estimate using the non-modified MM method is 0.001 seconds, whereas with the optimization, the estimate is 0.017 seconds, one order of magnitude larger. The true time delay margin for this system is 0.053 seconds. As with the nonlinear system, the true time delay margin is computed via simulations with the same amount of delay in each input. As was seen with the adaptive system, the nominal MM method produced a very conservative result. In this case, it was only one order of magnitude too small, instead of the two or three order of magnitude disparity for the adaptive systems. Also as with the adaptive system, the optimization yields a TDM estimate that is the correct order of magnitude. The optimization for the LTI system appears to yield more conservative

TDM estimates than some of the adaptive system estimates, especially when large $\mathbf{\Gamma}$ are used in the adaptive system.

3.4.3 Quadrotor Flight Example

The selected Γ with 30 as the diagonal value was tested within the full quadrotor control scheme with unlimited control authority and no time delay. Its performance was compared to the baseline LQR with integral feedback controllers and a poorly tuned aMRAC controller ($\mathbf{\Gamma} = \boldsymbol{I}_{7\times7}$). The selected test trajectory was a circle with a radius of 2 meters centered at (2,0,2.25) meters that took 60 seconds to complete. The quadrotor started at (0,0,1) and moved counterclockwise around a circle while maintaining a constant yaw angle of 5 degrees. The results for full controller effectiveness are shown in Figure 3.9. All the controllers have similar x, y, and ψ transient and steady state tracking. However, the tuned aMRAC controller exhibits less of a drop in the z trajectory transient, while the baseline controllers exhibit the greatest decrease in z.

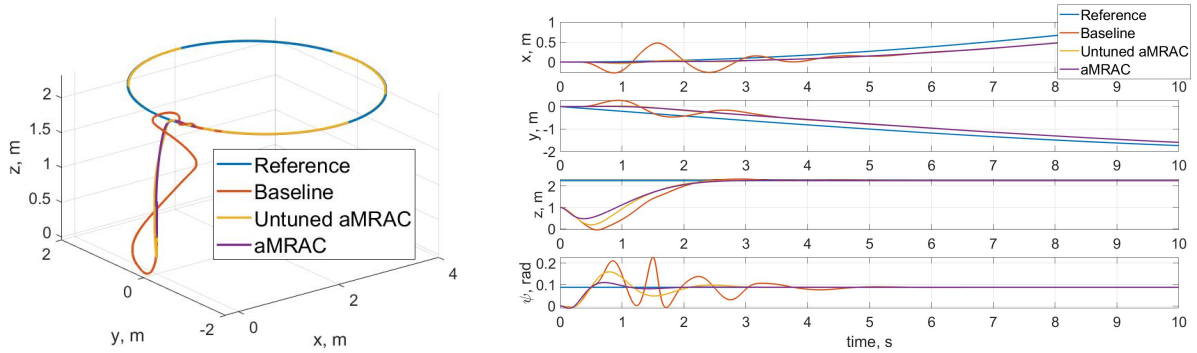


(a) Position comparison (b) Individual state comparison, transient response

Figure 3.9: Nominal thrust controller comparison

The results for 0.5 controller effectiveness of the first thruster are shown in Figure 3.10. Reducing controller effectiveness did not cause any of the controllers to fail completely. They

were all capable of achieving steady state tracking after some amount of time. However, the controllers had significantly different transient responses. The tuned aMRAC controller had the fastest settling time and least transient oscillations for all the states, maintaining performance similar to that exhibited with full controller effectiveness. The baseline controller performed the worst, with the largest transient oscillations and longest settling time.



(a) Position comparison

(b) Individual state comparison, transient response

Figure 3.10: Reduced thrust controller comparison

3.5 Discussion

The goal of this research was to develop an augmented model reference adaptive controller for a quadrotor and to modify, asses, and utilize the bounded linear stability analysis and matrix measure methods for TDM estimation and for selecting an appropriate adaptive parameter gain matrix. The standard matrix measure method was shown to yield results that were too conservative for a quadrotor modeled as a multi-input, multi-output system. A modified version of the matrix measure method yielded more accurate estimates of the TDM but sometimes produced nonconservative estimates with no clear source for the overestimate beyond the choice of optimization method, constraints, or the initial guess. Therefore, the method would not be suitable for validation and verification purposes based on stability margins. In

the future, it will be important to understand why some estimates were nonconservative and explore what, if anything, can be done to prevent overestimates of TDM. However, since the estimates closely follow the values and overall trends of the true TDM, the values are useful for tuning $\mathbf{\Gamma}$ via a methodical approach that does not require the multiple simulation runs necessary for determining the true TDM. The choice of $\mathbf{\Gamma}$ is then also linked to the state response and TDM for multiple parameter values. The selected choice for $\mathbf{\Gamma}$ was shown to yield better trajectory tracking than the baseline controller and a poorly tuned aMRAC controller. The results in this Chapter indicate that the methodology presented should be applicable for various controller effectiveness scenarios, particularly for MIMO systems using aMRAC control, such as the omnicopter.

Future work should extend this method to include common modifications to MRAC and aMRAC, such as closed loop reference models and the projection operator. Additionally, the results presented assumed the quadrotor had unlimited control authority. Therefore, initial spikes in thruster commands along with transient command peaks that correspond to signal overshoots are assumed feasible. In reality, the motors are limited to the amount of thrust they can produce, so analyses should be run that consider input saturation. Additionally, only the effects of reduced controller effectiveness on the first thruster were considered when designing $\mathbf{\Gamma}$. In the future, for full characterization of the controller, various combinations of thruster effectiveness would need to be tried.

Chapter 4

Simulation Methodology

Figure 4.1 depicts how the position and attitude controllers are integrated within the full omnicopter system, where the control allocation algorithm and motor dynamics equations are from [3]. Although the motors are bidirectional, for simplicity, the dynamics associated with a change of rotation direction are not simulated. This simplification may have impacts on the dynamics of the omnicopter while changing directions, but should not significantly affect the qualitative comparison between controller pairs. If necessary for hardware implementation, a control allocation algorithm can be designed to avoid rapid motor direction changes and has been shown to work effectively during hardware flight tests [3].

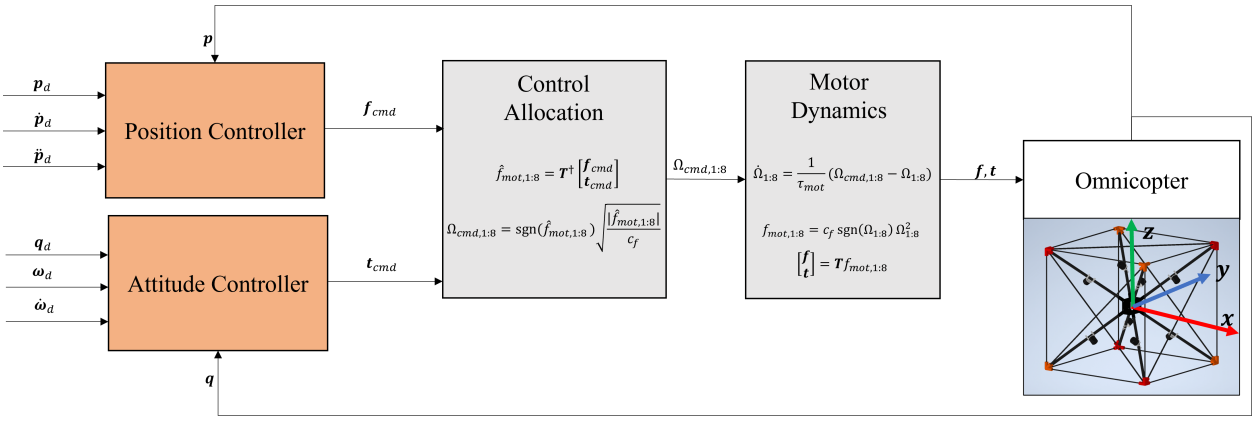


Figure 4.1: Omnicopter system overview

The motor is modeled as a first order system, shown in Eq. (4.1), with time constant $\tau_{mot} = 0.081$ seconds [3].

$$\dot{\Omega}_i = \frac{1}{\tau_{mot}}(\Omega_{cmd,i} - \Omega_i) \tag{4.1}$$

Continuous time simulations of the controllers were performed using ODE4 in MATLAB/Simulink, a fourth order Runge-Kutta solver, with a 0.001 second timestep. Physical parameter constants used from [3] are shown in Table 4.1.

Table 4.1: Omnicopter Physical Parameters [3]

Parameter	**Value**
m	$0.892\ kg$
$\boldsymbol{J}$	$diag(1.16 \times 10^{-2}, 1.13 \times 10^{-2}, 1.13 \times 10^{-2})\ kg \cdot m^2$
J_{mot}	$3.41 \times 10^{-5}\ kg \cdot m^2$
c_f	$4.721 \times 10^{-6}\ \frac{N}{(rad/s)^2}$
$\|\|l\|\|$	$0.184\ m$
$\boldsymbol{L}_{mot}$	$\frac{\|\|l\|\|}{\sqrt{3}}\begin{bmatrix} 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \end{bmatrix}$
$\boldsymbol{N}_{mot}$	$\begin{bmatrix} -a & b & -b & a & a & -b & b & -a \\ b & b & -a & -b & -b & -a & a & b \\ c & -c & -c & c & c & -c & -c & c \end{bmatrix}$
a, b, c	$\frac{1}{2} + \frac{1}{\sqrt{12}}, \frac{1}{2} - \frac{1}{\sqrt{12}}, \frac{1}{\sqrt{3}}$
$\boldsymbol{\kappa}_{mot}$	$1.478 \times 10^{-2} I_{8\times 8}\ \frac{Nm}{N}$
$\boldsymbol{T}$	$\begin{bmatrix} \boldsymbol{N}_{mot} \\ \boldsymbol{L}_{mot} \times \boldsymbol{N}_{mot} + \boldsymbol{N}_{mot}\boldsymbol{\kappa}_{mot} \end{bmatrix}$

4.0.1 Disturbances

The effectiveness of the controllers against parametric uncertainty and random disturbances is tested by increasing the system mass and moment of inertia and introducing disturbances

of varying magnitude, as follows:

Mass and Moment of Inertia

If the omnicopter were to carry an object, its center of mass and moment of inertia would change. Using the body-fixed frame of the omnicopter, fixed at its geometric center, as the reference frame, the altered center of mass can be computed. It is assumed that the load and arm are centered about the -z axis, rigid, and have a fixed orientation that are the same as the omnicopter. The moment of inertia for the system can be computed using the parallel axis theorem.

With the center of mass and moment of inertia change, the applied torque from the rotors needs to be changed from acting about the omnicopter center of mass to acting around the system center of mass. The value of $\boldsymbol{L}_{mot}$ must be shifted to the correct distance from the center of mass, $\boldsymbol{L}_{adj} = \boldsymbol{L}_{mot} - \boldsymbol{d}_{CM}$, so $\boldsymbol{t}_{actual} = [\boldsymbol{L}_{adj} \times \boldsymbol{N}_{mot} + \boldsymbol{\kappa}_{mot}\boldsymbol{N}_{mot}]\boldsymbol{f}_{mot}$.

For simulations, a thin plate is suspended on a massless rod below the center of the omnicopter, emulating a free-flyer moving a panel around an assembly site. The plate parameters used to compute the adjusted moment of inertia and center of mass are shown in Table 4.2.

Table 4.2: Omnicopter Load Parameters

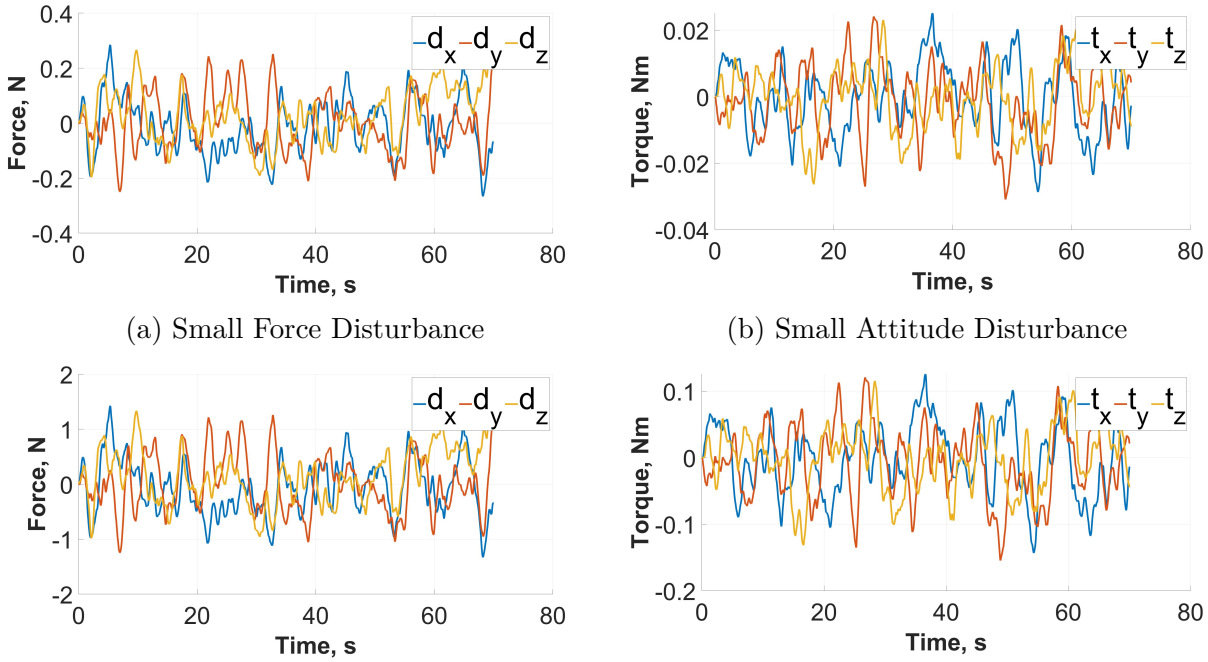
Parameter	**Value**
Parameter	**Value**
m_{load}	$0.25kg$
$\boldsymbol{J_{load}}$	$diag(2.6 \times 10^{-3}, 2.6 \times 10^{-3}, 5.1 \times 10^{-3})\ kg \cdot m^2$
$\boldsymbol{d}_{CM,load}$	$-0.31m$
load sidelength	$0.35m$

Random External Disturbance

To determine if the controllers are robust against perturbations, disturbances of various magnitudes are input in the force and torque dynamics. The types and magnitudes of disturbance that can/will act on the omnicopter are unknown. They could come from air currents within the lab, from the internal downwash-motor interactions, or from another omnicopter, if several are flying near each other. Thinking in terms of airflow disturbances, the exact correlation between a disturbance force vector and a disturbance torque vector is unclear because the omnicopter has an irregular shape with propellers at various angles and locations. Therefore, separate random forces and torques are input into the system. However, qualitatively, large forces should correlate with large torques and small forces with small torques, which is implemented. Random Gaussian white noise of various standard deviations, shown in Table 4.3, were filtered with a 0.25 Hz lowpass filter and input to the system position and attitude dynamics as disturbance. The small and large standard deviations were chosen to be approximately 5% and 25% the magnitudes of the maximum force ($19.4N$) and torque at hover ($1.9Nm$) [3]. Examples of disturbances for a single run (in this case run 1) can be seen in Figure 4.2. Each run was seeded with the same number to ensure disturbance continuity across trajectory and mass changes for a given run.

Table 4.3: Pre-filtered white noise disturbance values

Disturbance Magnitude	**Force Disturbance Standard Deviation (N)**	**Torque Disturbance Standard Deviation (Nm)**
Small	1.0	0.1
Large	5.0	0.5



(a) Small Force Disturbance

(b) Small Attitude Disturbance

(c) Large Force Disturbance

(d) Large Attitude Disturbance

Figure 4.2: Example Force and Torque Disturbances

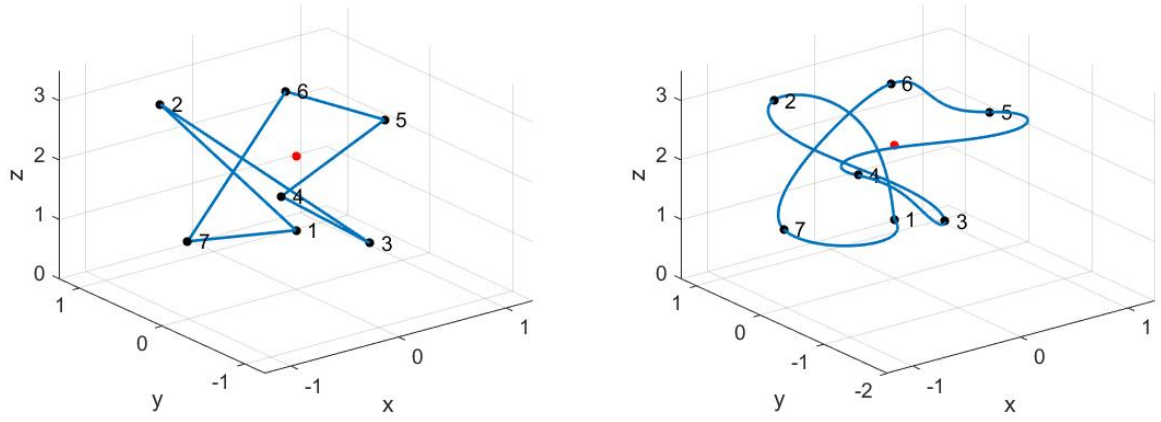
4.0.2 Test Trajectories

Two trajectories are used to test the controller capabilities. Six random points were generated in a space $2.5m$ x $2.5m$ x $2.25m$, a feasible lab space size for test flights. The points are shown in Table 4.4. The omnicopter has 10 seconds to travel between points. The first trajectory is "jagged" (i.e. C^0 continuous in $\boldsymbol{p}_d$), shown in Figure 4.3a. It commands the omnicopter to travel a direct line to each waypoint, maintaining a constant velocity for each segment. The second trajectory is a "smooth" trajectory (i.e. C^1 continuous in $\boldsymbol{p}_d$), shown in Figure 4.3b. It is computed using the waypointTrajectory.m function in MATLAB. The jagged trajectory is meant to stress the system be introducing discontinuities in $\dot{\boldsymbol{p}}_d$ that require sudden changes in forces and torques. It also demonstrates the controller capabilities for moving along straight paths. The smooth trajectory is meant to be less demanding of

Table 4.4: Trajectory Creation Points

Time (s)	**(x,y,z) Location** (m)
0	$(0, 0, 1)$
10	$(-0.65, 0.80, 2.93)$
20	$(-0.24, -1.21, 1.68)$
30	$(-1.01, -1.14, 2.80)$
40	$(1.11, 0.37, 2.08)$
50	$(0.19, 0.37, 3.01)$
60	$(-1.10, -0.12, 1.43)$
70	$(0, 0, 1)$

forces and torques but also to exhibit the controller performance along curved paths.



(a) Jagged (b) Smooth

Figure 4.3: Trajectories Created from Waypoints

For both trajectories, the omnicopter is commanded to fix its +x face on an aim point of (0,0,2.25) m while maintaining its +z face perpendicular to the velocity vector. The aim point (the red dot in Figures 4.3a and 4.3b) is roughly centered amongst the waypoints, and the resulting attitude trajectory forces the omnicopter to face a variety of different directions throughout time.

All simulations start with the same initial conditions. The omnicopter is given 40 seconds

to rise from the ground to the initial trajectory position and attitude before data recording begins. The omnicopter starts from rest at the origin, with its +x-face facing up ($\boldsymbol{q}_{init} = (cos(pi/4), 0, sin(pi/4), 0)$). All other initial conditions are zero except for $\boldsymbol{r}$ and $\overline{\boldsymbol{x}}_d$, which use the initial desired trajectory values. This period of initialization is used to prevent an initial error spike often seen in trying to achieve the commanded attitude associated with the first movement. The initial 40 seconds will not be represented in the data analysis, but will have an impact on the state of the adaptive parameters before starting the trajectory.

Chapter 5

Position Dynamics and Control

5.1 Dynamics

The nonlinear omnicopter position model from [3] is shown in Eq. (5.1), where $\boldsymbol{g} = (0, 0, 9.81)^T$.

$$m\ddot{\boldsymbol{p}} = \boldsymbol{R}(\boldsymbol{q})^{-1}\boldsymbol{f} - m\boldsymbol{g} \tag{5.1}$$

The linearized omnicopter position model, linearized about the omnicopter being aligned with the inertial frame, is shown in Eq. (5.2), where $\boldsymbol{x}_p = (\boldsymbol{p}^T, \dot{\boldsymbol{p}}^T)^T$, $\boldsymbol{p} = (p_x, p_y, p_z)^T$, and $\boldsymbol{f} = (f_x, f_y, f_z)^T$.

$$\dot{\boldsymbol{x}}_p = \begin{bmatrix} \boldsymbol{0_{3x3}} & \boldsymbol{I_{3x3}} \\ \boldsymbol{0_{3x3}} & \boldsymbol{0_{3x3}} \end{bmatrix} \boldsymbol{x}_p + \frac{1}{m} \begin{bmatrix} \boldsymbol{0_{3x3}} \\ \boldsymbol{I_{3x3}} \end{bmatrix} \boldsymbol{f} = \boldsymbol{A}_p\boldsymbol{x}_p + \boldsymbol{B}_p\boldsymbol{f} \tag{5.2}$$

5.2 PID

A PID controller with feedforward acceleration was originally designed in [3]. The force, $\boldsymbol{f_{cmd}}$, necessary to track $\boldsymbol{p_d}$ is shown in Eq. (5.3), where $\boldsymbol{p}_{err} = \boldsymbol{p}_d - \boldsymbol{p}$. The tuning gains used for simulations are from [3] and shown in Table 5.1.

$$\boldsymbol{f_{cmd}} = m\boldsymbol{R}(\boldsymbol{q})\left(k_{pos,P}\boldsymbol{p_{err}} + k_{pos,I}\int \boldsymbol{p_{err}}dt + k_{pos,D}\dot{\boldsymbol{p}}_{\boldsymbol{err}} + \ddot{\boldsymbol{p}}_{\boldsymbol{d}} + \boldsymbol{g}\right) \tag{5.3}$$

Table 5.1: PID Position Controller Parameters [3]

Parameter	**Value**
$k_{pos,P}$	14.09
$k_{pos,I}$	7.12
$k_{pos,D}$	6.91

5.3 Model Reference Adaptive Control with Input Saturation (MRAC)

The model reference adaptive controller is adapted from [82] and is shown in Fig. 5.1. The controller is designed to follow a linear reference model with desired closed loop system characteristics.

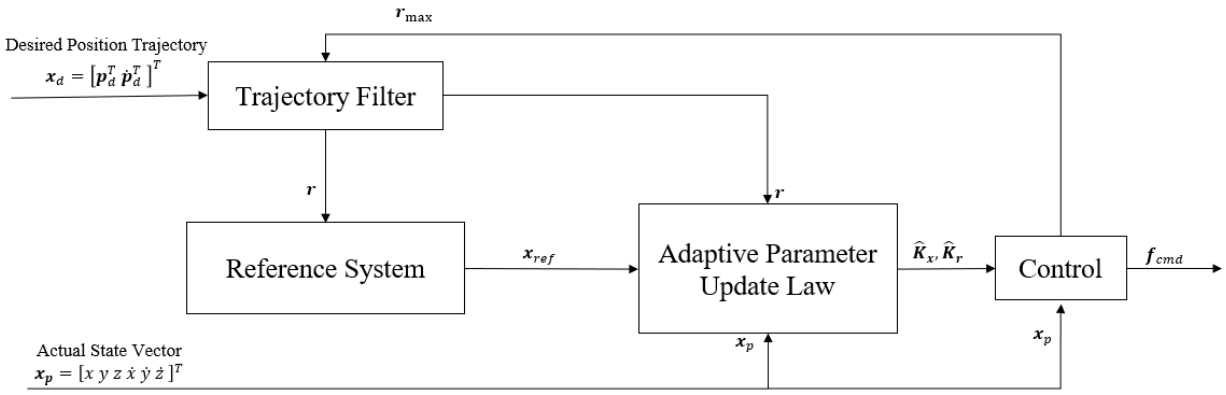


Figure 5.1: Model reference adaptive control with input saturation overview

To create the reference system for this controller, a LQR controller (see [82]) was used to stabilize the plant, and a reference input with gain was added to allow for trajectory tracking. The resulting reference system is shown in Eq. (5.4), where $\boldsymbol{A}_{ref} = \boldsymbol{A}_p - \boldsymbol{B}_p \boldsymbol{K}_{lqr}$ and $\boldsymbol{B}_{ref} = k\boldsymbol{B}_p$.

$$\dot{\boldsymbol{x}}_{ref} = \boldsymbol{A}_{ref}\boldsymbol{x}_{ref} + \boldsymbol{B}_{ref}\boldsymbol{r} \tag{5.4}$$

The reference input, $\boldsymbol{r}$, is selected such that the reference system will track the desired trajectory. Error between the reference and desired trajectory is defined as $\boldsymbol{e} = \boldsymbol{x}_{ref} - \boldsymbol{x}_d$, where $\boldsymbol{x}_d = [\boldsymbol{p}_d^T \ \dot{\boldsymbol{p}}_d^T]^T$. Taking the derivative with respect to time and substituting in relevant values, assuming the desired trajectory is also the reference trajectory ($\boldsymbol{e} = 0$), yields Eq. (5.5).

$$\dot{\boldsymbol{e}} = \dot{\boldsymbol{x}}_{ref} - \dot{\boldsymbol{x}}_d = \boldsymbol{A}_{ref}\boldsymbol{x}_d + \boldsymbol{B}_{ref}\boldsymbol{r}_d - \dot{\boldsymbol{x}}_d \tag{5.5}$$

Eq. (5.5) can be solved for $\boldsymbol{r}_d$, the desired reference trajectory, as seen in Eq. (5.6).

$$\boldsymbol{r}_d = \boldsymbol{B}_{ref}^{\dagger}(\dot{\boldsymbol{x}}_d - \boldsymbol{A}_{ref}\boldsymbol{x}_d) \tag{5.6}$$

The desired reference trajectory is then input into the differential equation in Eq. (5.7) that acts as a low pass filter with a fast time constant ($\tau_{sat} << 1$).

$$\dot{\boldsymbol{r}} = \frac{1}{\tau_{sat}}(\boldsymbol{r}_d - \boldsymbol{r}) \tag{5.7}$$

The plant model with parametric uncertainty for the system used for controller development is shown in Eq. (5.8).

$$\dot{\boldsymbol{x}}_p = \boldsymbol{A}_p\boldsymbol{x}_p + \boldsymbol{B}_p\boldsymbol{\Lambda}\boldsymbol{f} \tag{5.8}$$

The selected control for the plant to follow the reference system is $\boldsymbol{f}_{cmd} = \boldsymbol{R}(\boldsymbol{q})(\hat{\boldsymbol{K}}_x^T\boldsymbol{x}_p + \hat{\boldsymbol{K}}_r^T\boldsymbol{r})$. The $\hat{}$ symbol indicates estimated parameters that are adapted upon using the update laws (Eqs. (5.11) and (5.12)), which are computed using Lyapunov stability analysis. The Lyapunov function and its derivative are shown in Eqs. (5.9) and (5.10), where $\boldsymbol{e}_a = \boldsymbol{x}_p - \boldsymbol{x}_{ref}$, $\tilde{\boldsymbol{K}}_x = \hat{\boldsymbol{K}}_x - \boldsymbol{K}_x$, and $\tilde{\boldsymbol{K}}_r = \hat{\boldsymbol{K}}_r - \boldsymbol{K}_r$.

$$V = \frac{1}{2}\boldsymbol{e}_a^T\boldsymbol{P}\boldsymbol{e}_a + tr([\tilde{\boldsymbol{K}}^T\boldsymbol{\Gamma}_x^{-1}\tilde{\boldsymbol{K}}_x + \tilde{\boldsymbol{K}}^T\boldsymbol{\Gamma}_x^{-1}\tilde{\boldsymbol{K}}_x]\boldsymbol{\Lambda}) \tag{5.9}$$

$$\dot{V} = -\boldsymbol{e}_a^T\boldsymbol{Q}_{lyap}\boldsymbol{e}_a + 2tr(\tilde{\boldsymbol{K}}_x^T[\boldsymbol{\Gamma}_x^{-1}\dot{\hat{\boldsymbol{K}}}_x + \boldsymbol{x}_p\boldsymbol{e}_a^T\boldsymbol{P}\boldsymbol{B}_p]\boldsymbol{\Lambda}) + 2tr(\tilde{\boldsymbol{K}}_r^T[\boldsymbol{\Gamma}_r^{-1}\dot{\hat{\boldsymbol{K}}}_r + \boldsymbol{r}\boldsymbol{e}_a^T\boldsymbol{P}\boldsymbol{B}_p]\boldsymbol{\Lambda}) \tag{5.10}$$

The resulting stable adaptive laws that yield $\dot{V} \leq 0$ are shown in Eq. (5.11) and Eq. (5.12). These adaptive parameter update laws are used in conjunction with the projection operator in [82] and Appendix A, modified for the relevant adaptive parameter and variables.

$$\dot{\hat{\boldsymbol{K}}}_x = -\boldsymbol{\Gamma}_x \boldsymbol{x}_p \boldsymbol{e}_a^T \boldsymbol{P} \boldsymbol{B}_p \tag{5.11}$$

$$\dot{\hat{\boldsymbol{K}}}_r = -\boldsymbol{\Gamma}_r \boldsymbol{r} \boldsymbol{e}_a^T \boldsymbol{P} \boldsymbol{B}_p \tag{5.12}$$

To account for the input saturation due to the motors, a modification needs to be made to $\boldsymbol{r}$ and $\boldsymbol{f}_{cmd}$ based on the method presented in [83]. It is known that the maximum norm for the total control is limited to $||f||_{max}$. Let $\boldsymbol{f}_{unsat} = \hat{\boldsymbol{K}}_x^T \boldsymbol{x}_p + \hat{\boldsymbol{K}}_r^T \boldsymbol{r}$, the unrotated control with no saturation. If $||f_{unsat}|| > ||f||_{max}$, then the saturated force command can be computed as $\boldsymbol{f}_{cmd} = \boldsymbol{R}(\boldsymbol{q}) \frac{||f||_{max}}{||f_{unsat}||} \boldsymbol{f}_{unsat}$. Using the newly computed saturated force command, $\boldsymbol{f}_{cmd}$ before it has been rotated by $\boldsymbol{R}(\boldsymbol{q})$, an appropriate reference trajectory can now be back-calculated that will not cause the force command to oversaturate: $\boldsymbol{r}_{max} = (\hat{\boldsymbol{K}}_r^T)^\dagger (\boldsymbol{f}_{cmd} - \hat{\boldsymbol{K}}_x^T \boldsymbol{x}_p)$. The reference trajectory in Eq. (5.7) is now computed using $\boldsymbol{r}_d = \boldsymbol{r}_{max}$.

The tuning gains, projection operator parameters, and initial conditions shown in Table 5.2 were found via trial and error prior to the development of the method described in Ch. 3. Ideally, the gains would have been selected based on pre-defined step responses (e.g. rise time, overshoot, settling time) that would be similar to the baseline PID performance to keep the controller comparisons as uniform and fair as possible. Instead, visual inspection of the trajectory tracking was completed after each tuning change, and tuning was deemed acceptable once further attempted changes did not produce significantly improved tracking performance. This trial and error tuning process applies to every position and attitude controller except for the baseline controllers from [3]. For the adaptive parameters, the initial conditions are chosen to be the values based on the ideal omnicopter model, allowing the parameters to start as close to their true values as possible.

Table 5.2: MRAC Position Controller Parameters

Parameter	Value
$\boldsymbol{Q}_{lqr}$	diag([250 250 250 25 25 25])
$\boldsymbol{R}_{lqr}$	diag([1 1 1])
k	10
$\boldsymbol{Q}_{lyap}$	diag([1 1 10 1 1 1])
$\boldsymbol{\Gamma}_x$	diag([5 5 5 5 5 5])
$\boldsymbol{\Gamma}_r$	diag([5 5 10])
$\hat{\boldsymbol{K}}_{x,max}/\hat{\boldsymbol{K}}_{r,max}$	[50 50 50]
ε	[0.5 0.5 0.5]
$\|\|f\|\|_{max}$	26
$\hat{\boldsymbol{K}}_{x,0}$	$[(\boldsymbol{B}_{ref})^{\dagger}(\boldsymbol{A}_{ref} - \boldsymbol{A}_p)]^T$
$\hat{\boldsymbol{K}}_{r,0}$	$[(\boldsymbol{B}_{ref})^{\dagger}(\boldsymbol{B}_p)]^T$
τ	0.01

5.3.1 PID Augmented Model Reference Adaptive Control with Input Saturation (aMRAC)

A model reference adaptive controller is designed to augment a linearized version of the baseline PID controller from Eq. (5.3) and is shown in Fig. 5.2. The adaptive controller is used to cancel out uncertainties that prevent the baseline controller from performing with its desired, nominal behavior.

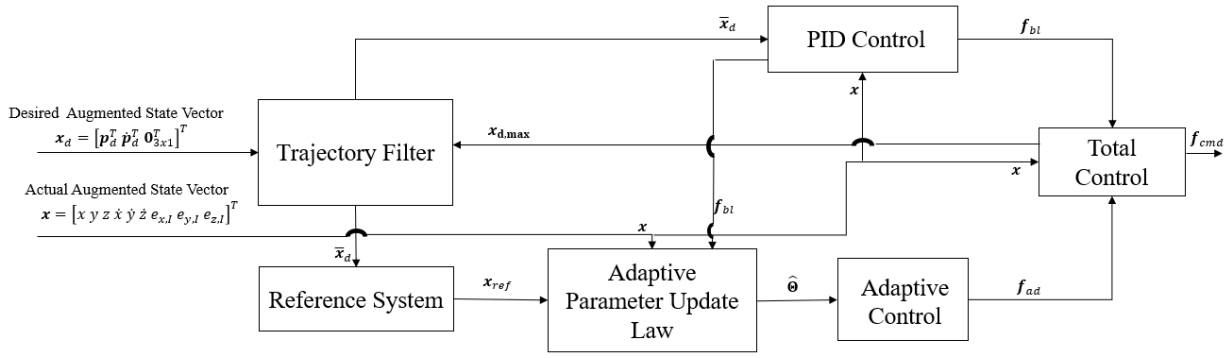


Figure 5.2: PID augmented model reference adaptive control with input saturation overview

To linearize the PID controller, the linearized dynamics shown in Eq. (5.2) need to be augmented with the position error dynamics. The augmented state vector is $\boldsymbol{x}_a = \begin{bmatrix} \boldsymbol{x}_p^T & \boldsymbol{e}_I^T \end{bmatrix}^T$, where $\boldsymbol{e}_I = \int_0^t (\boldsymbol{p} - \boldsymbol{p}_d) d\tau$. The resulting augmented linearized dynamics are shown in Eq. (5.13).

$$\dot{\boldsymbol{x}}_a = \begin{bmatrix} \boldsymbol{A}_p & \boldsymbol{0}_{6x3} \\ \boldsymbol{I}_{3x3} & \boldsymbol{0}_{3x6} \end{bmatrix} \begin{bmatrix} \boldsymbol{x}_p \\ \boldsymbol{e}_I \end{bmatrix} + \begin{bmatrix} \boldsymbol{B}_p \\ \boldsymbol{0}_{3x3} \end{bmatrix} \boldsymbol{f} + \begin{bmatrix} \boldsymbol{0}_{6x3} \\ -\boldsymbol{I}_{3x3} \end{bmatrix} \boldsymbol{p}_d = \boldsymbol{A}_a \boldsymbol{x}_a + \boldsymbol{B}_a \boldsymbol{f} + \boldsymbol{B}_{ref} \boldsymbol{p}_d \tag{5.13}$$

For aMRAC, the baseline controller is the linearized PID controller shown in Eq. (5.14), where $\boldsymbol{x}_d = [\boldsymbol{p}_d^T \ \dot{\boldsymbol{p}}_d^T \ \boldsymbol{0}_{1\times 3}]^T$ and $\boldsymbol{K}_{pid}^T = m \begin{bmatrix} k_{pos,P} \boldsymbol{I}_{3\times 3} & k_{pos,D} \boldsymbol{I}_{3\times 3} & k_{pos,I} \boldsymbol{I}_{3\times 3} \end{bmatrix}$.

$$\boldsymbol{f}_{bl} = -\boldsymbol{K}_{pid}^T (\boldsymbol{x}_a - \overline{\boldsymbol{x}}_d) \tag{5.14}$$

$\overline{\boldsymbol{x}}_d$ is computed by inputing the desired reference trajectory ($\boldsymbol{x}_d$) into the differential equation in Eq. (5.15), which acts as a low pass filter with a fast time constant ($\tau_{sat} << 1$).

$$\dot{\overline{\boldsymbol{x}}}_d = \frac{1}{\tau_{sat}} (\boldsymbol{x}_d - \overline{\boldsymbol{x}}_d) \tag{5.15}$$

When $\boldsymbol{f}_{bl}$ is input into the plant model, the resulting closed loop system and reference model is shown in Eq. (5.16).

$$\dot{\boldsymbol{x}}_{ref} = \boldsymbol{A}_{ref} x_{ref} + \boldsymbol{B}\boldsymbol{K}_{pid}^{T} \overline{\boldsymbol{x}}_d + \boldsymbol{B}_{ref} \overline{\boldsymbol{p}}_d \tag{5.16}$$

The total control for the system is $\boldsymbol{f}_{cmd} = R(\boldsymbol{q})(\boldsymbol{f}_{bl} + \boldsymbol{f}_{ad})$, where $\boldsymbol{f}_{ad}$ is designed in the following steps. The augmented plant system with parametric uncertainty is shown in Eq. (5.17). After inputting $\boldsymbol{f}_{cmd}$ for $\boldsymbol{f}$, combining terms, and defining $\boldsymbol{A}_{ref} = \boldsymbol{A}_a - \boldsymbol{B}_a \boldsymbol{K}_{pid}$, $\boldsymbol{\Theta}^T = (\boldsymbol{I}_{3x3} - \boldsymbol{\Lambda}^{-1})^T$ and $\boldsymbol{\Phi} = \boldsymbol{u}_{bl}$, the uncertain plant becomes Eq. (5.18).

$$\dot{\boldsymbol{x}}_a = \boldsymbol{A}_a \boldsymbol{x}_a + \boldsymbol{B}_a \boldsymbol{\Lambda} \boldsymbol{f} + \boldsymbol{B}_{ref} \overline{\boldsymbol{p}}_d \tag{5.17}$$

$$\dot{\boldsymbol{x}}_a = \boldsymbol{A}_{ref} \boldsymbol{x}_a + \boldsymbol{B}_a \boldsymbol{K}_{pid}^{T} \overline{\boldsymbol{x}}_d + \boldsymbol{B}_a \boldsymbol{\Lambda} (\boldsymbol{f}_{ad} + \Theta^T \Phi) + \boldsymbol{B}_{ref} \overline{\boldsymbol{p}}_d \tag{5.18}$$

The adaptive controller is chosen to be $\boldsymbol{f}_{ad} = -\hat{\boldsymbol{\Theta}}^{\boldsymbol{T}} \boldsymbol{\Phi}$ to match the uncertainties. The adaptive parameter update law is computed using Lyapunov stability analysis. The positive definite Lyapunov function and its derivative are shown in Eq. (5.20), where $\boldsymbol{e}_a = \boldsymbol{x}_a - \boldsymbol{x}_{ref}$ and $\tilde{\boldsymbol{\Theta}}^T = \hat{\boldsymbol{\Theta}}^T - \boldsymbol{\Theta}^T$.

$$V = \boldsymbol{e_a}^T \boldsymbol{P} \boldsymbol{e_a} + tr(\tilde{\boldsymbol{\Theta}}^{\boldsymbol{T}} \boldsymbol{\Gamma}^{-1} \tilde{\boldsymbol{\Theta}}^{\boldsymbol{T}} \boldsymbol{\Lambda}) \tag{5.19}$$

$$\dot{V} = -\boldsymbol{e}_a^T \boldsymbol{Q}_{lyap} \boldsymbol{e}_a + 2tr(\tilde{\boldsymbol{\Theta}}^T [\boldsymbol{\Gamma}^{-1} \dot{\hat{\boldsymbol{\Theta}}} - \boldsymbol{\Phi} \boldsymbol{e}_a^T \boldsymbol{P} \boldsymbol{B}_a] \boldsymbol{\Lambda}) \tag{5.20}$$

The resulting stable adaptive parameter update law is $\dot{\hat{\Theta}} = \boldsymbol{\Gamma} \boldsymbol{\Phi} \boldsymbol{e_a}^T \boldsymbol{P} \boldsymbol{B}_a$, which yields $\dot{V} \leq 0$,

and is used in conjunction with the projection operator in [82] and Appendix A.

To account for the input saturation due to the motors, a modification needs to be made to $\boldsymbol{x}_d$ and $\boldsymbol{f}_{cmd}$ based on the method presented in [83]. Let $\boldsymbol{f}_{unsat} = \boldsymbol{f}_{bl} + \boldsymbol{f}_{ad} = -\boldsymbol{K}_{pid}^T(\boldsymbol{x}_a - \overline{\boldsymbol{x}}_d) + \hat{\boldsymbol{\Theta}}^T \boldsymbol{K}_{pid}^T(\boldsymbol{x}_a - \overline{\boldsymbol{x}}_d)$, the unrotated control with no saturation. If $||f_{unsat}|| > ||f||_{max}$, then the saturated force command can be computed as $\boldsymbol{f}_{cmd} = \boldsymbol{R}(\boldsymbol{q})\frac{||f||_{max}}{||f_{unsat}||}\boldsymbol{f}_{unsat}$. Using the newly computed saturated force command, $\boldsymbol{f}_{cmd}$ before it has been rotated by $\boldsymbol{R}(\boldsymbol{q})$, an appropriate reference trajectory can now be back-calculated that will not cause the force command to oversaturate: $\boldsymbol{x}_{d,max} = (\boldsymbol{K}_{pid}^T - \hat{\boldsymbol{\Theta}}^T \boldsymbol{K}_{pid}^T)^\dagger(\boldsymbol{f}_{cmd} - (\hat{\boldsymbol{\Theta}}^T \boldsymbol{K}_{pid}^T - \boldsymbol{K}_{pid}^T)\boldsymbol{x}_a)$. The reference trajectory in Eq. (5.15) is now computed using $\overline{\boldsymbol{x}}_d = \boldsymbol{x}_{d,max}$.

The tuning gains, projection operator parameters, and initial conditions shown in Table 5.3 were found via trial and error prior to the development of the method in Ch. 3.

Table 5.3: aMRAC Position Controller Parameters

Parameter	**Value**
$\boldsymbol{Q}_{lyap}$	$\boldsymbol{I}_{9\times9}$
$\boldsymbol{\Gamma}$	diag([200 200 200])
$\hat{\boldsymbol{\Theta}}_{max}$	[50 50 50]
ε	[0.5 0.5 0.5]
$\|\|f\|\|_{max}$	26
$\hat{\boldsymbol{\Theta}}_0$	$\boldsymbol{I}_{3\times3}$
τ	0.01

The linearized omnicopter position dynamics were used to design the MRAC and aMRAC position controllers. While the MRAC controller required the determination of a new appro-

priate reference model, the aMRAC controller was used to augment the linearized original PID controller, so the closed-loop PID controller system served as its reference model. The three position controllers presented in this Chapter will be combined with the four attitude controllers presented in the next Chapter for performance comparisons using simulations.

Chapter 6

Attitude Dynamics and Control

6.1 Dynamics and Kinematics

The attitude kinematics for the omnicopter from [3] are shown in Eq. (6.1), where quaternion multiplication is defined in Eqs. (6.2) and (6.3). The unit quaternion is defined as $\boldsymbol{q} = (q_0, \tilde{\boldsymbol{q}})^T$ where q_0 and $\tilde{\boldsymbol{q}}$ are the scalar and vector components, respectively.

$$\dot{\boldsymbol{q}} = \frac{1}{2}\boldsymbol{q} \otimes \begin{bmatrix} 0 \\ \boldsymbol{\omega} \end{bmatrix} \tag{6.1}$$

$$\boldsymbol{q} \otimes \boldsymbol{r} = \begin{bmatrix} q_0 & -\tilde{\boldsymbol{q}}^T \\ \tilde{\boldsymbol{q}} & q_0 I_{3\times 3} - [\tilde{\boldsymbol{q}}]_\times \end{bmatrix} \boldsymbol{r} \tag{6.2}$$

$$[\tilde{\boldsymbol{q}}]_\times = \begin{bmatrix} 0 & -q_3 & q_2 \\ q_3 & 0 & -q_1 \\ -q_2 & q_1 & 0 \end{bmatrix} \tag{6.3}$$

The attitude dynamics for the omnicopter from [3] are shown in Eq.(6.4), where $\boldsymbol{t} = (t_x, t_y, t_z)^T$.

$$\boldsymbol{J}\dot{\boldsymbol{\omega}} + \sum_{i=1}^{8} J_{rot}\dot{\Omega}_i\boldsymbol{n}_i = \boldsymbol{t} - \boldsymbol{\omega} \times \left(\boldsymbol{J}\boldsymbol{\omega} + \sum_{i=1}^{8} J_{rot}\Omega_i\boldsymbol{n}_i\right) \tag{6.4}$$

The $\sum_{i=1}^{8} J_{rot}\dot{\Omega}_i\boldsymbol{n}_i$ term can be neglected because J_{rot} is much smaller than $\boldsymbol{J}$. The torque applied to the omnicopter by the rotor thrust as well as the torque acting in the opposite direction to rotor rotation is expressed in Eq. (6.5).

$$\boldsymbol{t} = [\boldsymbol{L}_{mot} \times \boldsymbol{N}_{mot} + \boldsymbol{N}_{mot}\boldsymbol{\kappa}_{mot}]\boldsymbol{f}_{mot} \tag{6.5}$$

The dynamics can be rewritten as Eq. (6.6) for developing the aMRAC, backstepping, and adaptive backstepping attitude controllers. The gyroscopic effect torque from the spinning rotors, $\boldsymbol{J}^{-1}[\omega]_\times \sum_{i=1}^{8} J_{rot}\Omega_i\boldsymbol{n}_i$, as well as all other disturbance torques are encompassed in $\boldsymbol{t}_{dist}$. The remaining terms are encompassed in $\boldsymbol{\Theta} = diag(\frac{J_z - J_y}{J_x}, \frac{J_x - J_z}{J_y}, \frac{J_y - J_x}{J_z})$ and $\boldsymbol{\Phi} = -(\omega_z\omega_y, \omega_x\omega_z, \omega_x\omega_y)^T$.

$$\dot{\boldsymbol{\omega}} = \boldsymbol{\Theta}\Phi + \boldsymbol{J}^{-1}\boldsymbol{t} + \boldsymbol{t}_{dist} \tag{6.6}$$

6.2 Proportional Integral/Feedback Linearization Control (PI/FL)

In the original omnicopter design [3], a PI controller, shown in Eq. (6.7), is used to compute desired angular velocity. $\boldsymbol{R}(\boldsymbol{q}_{err})^{-1}$ acts as the rotation between where the desired trajectory thinks the omnicopter is facing and its true orientation.

$$\boldsymbol{\omega_{cmd}} = k_{att,p}\widetilde{\boldsymbol{q}}_{err} + k_{att,i}\int \widetilde{\boldsymbol{q}}_{err}dt + \boldsymbol{R}(\boldsymbol{q}_{err})^{-1}\boldsymbol{\omega_d} \tag{6.7}$$

Desired torque is then computed using a feedback linearization controller, shown in Eq. (6.8). The tuning gains from [3] used in simulation are shown in Table 6.1.

$$\boldsymbol{t}_{cmd} = \frac{1}{\tau_\omega}\boldsymbol{J}(\boldsymbol{\omega_{cmd}} - \boldsymbol{\omega}) + \boldsymbol{\omega} \times \left(\boldsymbol{J}\boldsymbol{\omega} + \sum_{i=1}^{8} J_{rot}\Omega_i \boldsymbol{n}_i\right) \tag{6.8}$$

The quaternion error is defined as Eq. (6.9), where $\boldsymbol{q}^{-1} = [q_0, -\tilde{\boldsymbol{q}}^T]^T$.

$$\boldsymbol{q}_{err} = \boldsymbol{q}_d \otimes \boldsymbol{q}^{-1} = \begin{bmatrix} q_{err,0} \\ \tilde{\boldsymbol{q}}_{err} \end{bmatrix} = \begin{bmatrix} q_{0,d}q_0 + \tilde{\boldsymbol{q}}_d^T\tilde{\boldsymbol{q}} \\ q_0\tilde{\boldsymbol{q}}_d - q_{0,d}\tilde{\boldsymbol{q}} + [\tilde{\boldsymbol{q}}_d]_\times\tilde{\boldsymbol{q}} \end{bmatrix} \tag{6.9}$$

Table 6.1: PI/FL Attitude Controller Parameters [3]

Parameter	**Value**
$k_{att,P}$	6.46
$k_{att,I}$	3.42
τ_ω	0.044

6.3 Augmented Model Reference Adaptive Control (aMRAC)

A quaternion-based attitude controller that uses augmented model reference adaptive control concepts is developed for quadrotors in [53]. Adaptive control is used to augment a baseline nonlinear dynamic inversion controller with feed-forward terms. This controller design is directly applicable to the omnicopter attitude control, and the relevant equations from [53] are presented in this section.

Unlike the other controllers presented in this Section, quaternion error for aMRAC attitude control is defined as Eq. (6.10).

$$\boldsymbol{q}_{err} = \boldsymbol{q}_d^{-1} \otimes \boldsymbol{q} = \begin{bmatrix} q_{err,0} \\ \tilde{\boldsymbol{q}}_{err} \end{bmatrix} = \begin{bmatrix} q_{0,d}q_0 + \tilde{\boldsymbol{q}}_d^T \tilde{\boldsymbol{q}} \\ -q_0\tilde{\boldsymbol{q}}_d + q_{0,d}\tilde{\boldsymbol{q}} + [-\tilde{\boldsymbol{q}}_d]_\times \tilde{\boldsymbol{q}} \end{bmatrix} \tag{6.10}$$

Angular velocity error is defined as $\boldsymbol{\omega}_{err} = \boldsymbol{\omega} - \boldsymbol{\omega}_d$, where all angular velocities are expressed in the omnicopter body frame.

The baseline controller is defined as Eq. (6.11), where the quaternion logarithm is defined as $log_v(\boldsymbol{q}) = atan2(||\tilde{\boldsymbol{q}}||, q_0)\frac{\tilde{\boldsymbol{q}}}{||\tilde{\boldsymbol{q}}||}$.

$$\boldsymbol{t}_{bl} = [\boldsymbol{\omega}]_\times \boldsymbol{J}\boldsymbol{\omega} + \boldsymbol{J}\left(\dot{\boldsymbol{\omega}}_d + [\boldsymbol{\omega}]_\times \boldsymbol{\omega}_d - k_q log_v(\boldsymbol{q}_{err}) - k_\omega \boldsymbol{\omega}_{err}\right) \tag{6.11}$$

Since the baseline controller assumes all system parameters and dynamics are known and accounted for, it is augmented with an adaptive controller to negate uncertainties and restore desired closed loop system performance that reflects the ideal baseline controller performance. The total control is $\boldsymbol{t}_{cmd} = \boldsymbol{t}_{bl} + \boldsymbol{t}_{ad}$.

The desired closed loop error dynamics of the system is Eq. (6.12), where $\boldsymbol{A}_{ref} = -k_\omega \boldsymbol{I}_{3\times3} - [\boldsymbol{\omega}]_\times$ and $\boldsymbol{r} = k_\omega \boldsymbol{\omega}_d + \dot{\boldsymbol{\omega}}_d - k_q log_v(\boldsymbol{q}_{err})$.

$$\dot{\boldsymbol{\omega}} = \boldsymbol{A}_{ref}\boldsymbol{\omega} + \boldsymbol{r} \tag{6.12}$$

The reference model is then selected to be Eq. (6.13), where $\boldsymbol{e} = \boldsymbol{\omega}_{ref} - \boldsymbol{\omega}$ and $k_e\boldsymbol{e}$ is used to reduce oscillations in the transient response of the adaptive controller.

$$\dot{\boldsymbol{\omega}}_{ref} = \boldsymbol{A}_{ref}\boldsymbol{\omega}_{ref} + \boldsymbol{r} - k_e\boldsymbol{e} \tag{6.13}$$

The adaptive controller is chosen as Eq. (6.14).

$$\boldsymbol{t}_{ad} = \hat{\boldsymbol{J}}\left[-(\hat{\boldsymbol{\Theta}} - \boldsymbol{\Theta})\boldsymbol{\Phi} - \hat{\boldsymbol{t}}_{dist}\right] - (\boldsymbol{I}_{3\times3} - \hat{\boldsymbol{J}}\boldsymbol{J}^{-1})\boldsymbol{t}_{bl} \tag{6.14}$$

The adaptive parameter update laws are computed using the Lyapunov function in Eq. (6.15), where $\tilde{\boldsymbol{J}}^{-1} = \hat{\boldsymbol{J}}^{-1} - \boldsymbol{J}^{-1}$, $\tilde{\boldsymbol{\theta}} = \hat{\boldsymbol{\theta}} - \boldsymbol{\theta}$, and $\tilde{\boldsymbol{t}}_{dist} = \hat{\boldsymbol{t}}_{dist} - \boldsymbol{t}_{dist}$.

$$V = \boldsymbol{e}^T\boldsymbol{P}\boldsymbol{e} + tr\left(\tilde{\boldsymbol{J}}^{-1}\boldsymbol{\Gamma}_J^{-1}\tilde{\boldsymbol{J}}^{-T} + \tilde{\boldsymbol{\Theta}}\boldsymbol{\Gamma}_\Theta^{-1}\tilde{\boldsymbol{\Theta}}^T + \gamma^{-1}\tilde{\boldsymbol{t}}_{dist}\tilde{\boldsymbol{t}}_{dist}^T\right) \tag{6.15}$$

Selecting the adaptive parameter update laws to be Eqs. (6.16) - (6.18) yields $\dot{V} = -2k_\omega\boldsymbol{e}^T\boldsymbol{I}\boldsymbol{e} \leq 0$, ensuring stability. $\boldsymbol{P}$ is selected to be $\boldsymbol{I}_{3\times3}$ to yield the Lyapunov equation $\boldsymbol{A}_{ref}^T\boldsymbol{P} + \boldsymbol{P}\boldsymbol{A}_{ref} = -2k_\omega I$. Additionally, the parameter update laws are used in conjunction with the projection operator in Appendix A.

$$\dot{\hat{\boldsymbol{\Theta}}}^T = -\boldsymbol{\Gamma}_\theta \boldsymbol{\Phi} \boldsymbol{e}^T \boldsymbol{P} \tag{6.16}$$

$$\dot{\hat{\boldsymbol{J}}}^{-T} = -\boldsymbol{\Gamma}_J \boldsymbol{t}_{cmd} \boldsymbol{e}^T \boldsymbol{P} \tag{6.17}$$

$$\dot{\hat{\boldsymbol{t}}}_{dist}^T = -\gamma \boldsymbol{e}^T \boldsymbol{P} \tag{6.18}$$

Finally, for implementation, Eq. (6.19) is used as the reference model, which is the original reference model in Eq. (6.13) simplified after adding and subtracting $\hat{\boldsymbol{J}}^{-1}\boldsymbol{t}_{cmd}$.

$$\dot{\boldsymbol{\omega}}_{ref} = -k_e \boldsymbol{e} + \hat{\boldsymbol{J}}^{-1}\boldsymbol{t}_{cmd} + \hat{\boldsymbol{\Theta}}\boldsymbol{\Phi} + \hat{\boldsymbol{t}}_{dist} \tag{6.19}$$

Additionally, instead of using the computed $\boldsymbol{t}_{cmd}$ in Eq. (6.17) and Eq. (6.19), an estimate of the actual torque created by the motors is used. This estimated $\boldsymbol{t}_{est}$ is computed by inputting the desired $\boldsymbol{t}_{cmd}$ into the control allocation algorithm described in [3], which accounts for motor saturation and computes the desired force and angular velocity per motor. The commanded angular velocities are fed through a motor model, $\dot{\Omega}_i = \frac{1}{\tau_{mot}}(\Omega_{cmd,i} - \Omega_i)$, and the resulting angular velocity is used to calculate the actual torque generated by the motors via $f_{mot,i} = c_f \text{sgn}(\Omega_i)\Omega_i^2$. The estimated control, $\boldsymbol{t}_{est}$, is the last three elements of force and torque mapping, $\boldsymbol{T}\boldsymbol{f}_{mot}$.

Trial and error was used to determine the tuning gains, projection operator parameters, and initial conditions shown in Table 6.2.

Table 6.2: aMRAC Attitude Controller Parameters

Parameter	Value	Parameter	Value
$\boldsymbol{\Gamma}_\theta$	$\boldsymbol{I}_{3\times3}$	$\boldsymbol{\varepsilon}_{\Theta,J,t}$	[0.5 0.5 0.5]
$\boldsymbol{\Gamma}_J$	$\boldsymbol{I}_{3\times3}$	$\hat{\boldsymbol{\Theta}}_{max}$	[1 1 1]
γ	1	$\hat{\boldsymbol{J}}^{-1}_{max}$	[100 100 100]
k_e	10	$\hat{\boldsymbol{t}}_{dist,max}$	[1 1 1]
k_q	250	$\hat{\boldsymbol{\Theta}}_0$	$\boldsymbol{J}^{-1}\text{diag}([J_z - J_y, J_x - J_z, J_y - J_x])$
k_ω	50	$\hat{\boldsymbol{J}}^{-1}_0$	$\boldsymbol{J}^{-1}$

6.4 Backstepping

Designing a backstepping controller for the omnicopter is a 2-step process. The controller design in this section is formed upon the quaternion-based satellite attitude controller in [84]. The dynamics from Eq. (6.6) are used, however the negative sign is placed with $\boldsymbol{\Theta}$ instead of $\boldsymbol{\Phi}$: $\boldsymbol{\Theta} = -diag(\frac{J_z-J_y}{J_x}, \frac{J_x-J_z}{J_y}, \frac{J_y-J_x}{J_z})$ and $\boldsymbol{\Phi} = (\omega_z\omega_y, \omega_x\omega_z, \omega_x\omega_y)^T$.

Step 1

The first step designs the virtual control input : $\boldsymbol{\omega}_{err}$. Quaternion error between desired and actual attitude is defined as Eq. (6.9).

To allow for trajectory tracking, the necessary change of variables is defined as Eq. (6.20).

In the case of perfect tracking, when $\boldsymbol{q}_{err} = (\pm 1, \boldsymbol{0})^T$, $\boldsymbol{z}_1$ will go to 0.

$$\boldsymbol{z}_1 = \begin{bmatrix} 1 - |q_{err,0}| \\ \tilde{\boldsymbol{q}}_{err} \end{bmatrix} \tag{6.20}$$

Its derivative is computed in Eq. (6.21), where $[\cdot]_\times$ is the skew symmetric matrix.

$$\dot{\boldsymbol{z}}_1 = \frac{1}{2} \begin{bmatrix} \text{sgn}(q_{0,err})\tilde{\boldsymbol{q}}_{err}^T \\ q_{0,err}\boldsymbol{I}_{3\times 3} + [\tilde{\boldsymbol{q}}_{err}]_\times \end{bmatrix} (\boldsymbol{\omega}_{des} - \boldsymbol{\omega}) = \boldsymbol{G}\boldsymbol{\omega}_{err} \tag{6.21}$$

The virtual control input is chosen as $\boldsymbol{\omega}_{err} = \boldsymbol{\alpha}_1 + \boldsymbol{z}_2$, where $\boldsymbol{\alpha}_1$ is a stabilizing function that will be designed and $\boldsymbol{z}_2$ is a state that connects Step 1 with Step 2. Inserting the virtual control into Eq (6.21) yields $\dot{\boldsymbol{z}}_1 = \boldsymbol{G}(\boldsymbol{\alpha}_1 + \boldsymbol{z}_2)$.

Choosing $\boldsymbol{\alpha}_1 = -\boldsymbol{K}_1\boldsymbol{G}^T\boldsymbol{z}_1$ causes the first term in the Lyapunov function derivative in Eq. (6.23) to be negative definite. Step 2 will design a controller that negates the second term of $\dot{V}_1$ and yields a negative definite final Lyapunov function derivative, ensuring a stable controller for the entire system.

$$V_1 = \frac{1}{2}\boldsymbol{z}_1^T\boldsymbol{z}_1 \tag{6.22}$$

$$\dot{V}_1 = \boldsymbol{z}_1^T\dot{\boldsymbol{z}}_1 = \boldsymbol{z}_1^T\boldsymbol{G}(\boldsymbol{\alpha}_1 + \boldsymbol{z}_2) = -\boldsymbol{z}_1^T\boldsymbol{G}\boldsymbol{K}_1\boldsymbol{G}^T\boldsymbol{z}_1 + \boldsymbol{z}_1^T\boldsymbol{G}\boldsymbol{z}_2 \tag{6.23}$$

Step 2

The second step designs the actual control input for the system: $\boldsymbol{t}_{cmd}$. Rearranging the virtual control input from Step 1, $\boldsymbol{z}_2 = \boldsymbol{\omega}_{err} - \boldsymbol{\alpha}_1$. Its derivative is shown in Eq. (6.24), where $\dot{\boldsymbol{\alpha}}_1$ is computed using Eq. (6.25)-(6.27) and the dynamics from Eq. (6.4) are inserted

for $\dot{\boldsymbol{\omega}}$, ignoring disturbances.

$$\dot{\boldsymbol{z}}_2 = \dot{\boldsymbol{\omega}}_{des} - \dot{\boldsymbol{\omega}} - \dot{\boldsymbol{\alpha}}_1 = \dot{\boldsymbol{\omega}}_{des} - \boldsymbol{\Theta}\boldsymbol{\Phi} - \boldsymbol{J}^{-1}\boldsymbol{t} - \dot{\boldsymbol{\alpha}}_1 \tag{6.24}$$

$$\dot{\boldsymbol{\alpha}}_1 = -\boldsymbol{K}_1[\dot{\boldsymbol{G}}^T\boldsymbol{z}_1 + \boldsymbol{G}^T\dot{\boldsymbol{z}}_1] \tag{6.25}$$

$$\dot{\boldsymbol{G}} = \begin{bmatrix} \text{sgn}(q_{0,err})\dot{\tilde{\boldsymbol{q}}}_{err}^T \\ \dot{q}_{0,err}\boldsymbol{I}_{3\times 3} + [\dot{\tilde{\boldsymbol{q}}}_{err}]_\times \end{bmatrix} \tag{6.26}$$

$$\begin{bmatrix} \dot{q}_{0,err} \\ \dot{\tilde{\boldsymbol{q}}}_{err} \end{bmatrix} = \frac{1}{2}\begin{bmatrix} -\tilde{\boldsymbol{q}}_{err}^T \\ q_{0,err}\boldsymbol{I}_{3\times 3} + [\tilde{\boldsymbol{q}}_{err}]_\times \end{bmatrix}\boldsymbol{\omega}_{err} \tag{6.27}$$

The Lyapunov function and its derivative are chosen to be Eq. (6.29).

$$V_2 = V_1 + \frac{1}{2}\boldsymbol{z}_2^T\boldsymbol{z}_2 \tag{6.28}$$

$$\dot{V}_2 = \dot{V}_1 + \boldsymbol{z}_2^T\dot{\boldsymbol{z}}_2 = \dot{V}_1 + \boldsymbol{z}_2^T(\dot{\boldsymbol{\omega}}_{des} - \boldsymbol{\Theta}\boldsymbol{\Phi} - \boldsymbol{J}^{-1}\boldsymbol{t} - \dot{\boldsymbol{\alpha}}_1) \tag{6.29}$$

The final control design is chosen to be Eq. (6.30), which yields $\dot{V}_2 = -\boldsymbol{z}_1^T\boldsymbol{G}\boldsymbol{K}_1\boldsymbol{G}^T\boldsymbol{z}_1 - \boldsymbol{z}_2^T\boldsymbol{K}_2\boldsymbol{z}_2 \leq 0$.

$$\boldsymbol{t}_{cmd} = \boldsymbol{J}(\boldsymbol{G}^T\boldsymbol{z}_1 + \boldsymbol{K}_2\boldsymbol{z}_2 - \boldsymbol{\Theta}\boldsymbol{\Phi} + \dot{\boldsymbol{\omega}}_{des} - \dot{\boldsymbol{\alpha}}_1) \tag{6.30}$$

6.5 Adaptive Backstepping

The nonadaptive backstepping formulation assumes that the moment of inertia for the omnicopter is exactly known and that there are no disturbances to the system. The adaptive backstepping controller can adapt upon the terms that contain moment of inertia and account for the gyroscopic torques. For adaptive backstepping, the Step 1 design remains the

same because the kinematics equations are fully known and do not depend on moment of inertia. The method used in [26] is used to redesign Step 2 for adaptive backstepping control.

As in Step 2 of the nonadaptive backstepping controller, the adaptive backstepping controller uses $\boldsymbol{z}_2 = \boldsymbol{\omega}_{err} - \boldsymbol{\alpha}_1$. However, the derivative equation in Eq. (6.24) becomes Eq. (6.31) when accounting for disturbances, $\boldsymbol{t}_{dist}$.

$$\dot{\boldsymbol{z}}_2 = \dot{\boldsymbol{\omega}}_{des} - \boldsymbol{\Theta}\boldsymbol{\Phi} - \boldsymbol{J}^{-1}\boldsymbol{t} + \boldsymbol{t}_{dist} - \dot{\boldsymbol{\alpha}}_1 \tag{6.31}$$

The disturbance torques can be bounded using $|t_{dist,j}| \leq \psi_j^*$, where ψ_j is an unknown value that will be adapted upon. Additionally, the parameter matrix, $\boldsymbol{\Theta}$, containing moment of inertia values will be adapted upon. The error between estimated and true values are defined as $\tilde{\boldsymbol{\Theta}} = diag[(\hat{\Theta}_1 - \Theta_1^*), (\hat{\Theta}_2 - \Theta_2^*), (\hat{\Theta}_2 - \Theta_2^*)]$ and $\tilde{\boldsymbol{\psi}} = [(\hat{\psi}_1 - \psi_1^*), (\hat{\psi}_2 - \psi_2^*), (\hat{\psi}_2 - \psi_2^*)]^T$. Values with $\hat{}$ are estimates of the true values, denoted by *. The adaptive parameter update laws will be derived using the Lyapunov function in Eq. (6.32).

$$V_2 = V_1 + \frac{1}{2}\boldsymbol{z}_2^T\boldsymbol{z}_2 + \frac{1}{2}tr(\tilde{\boldsymbol{\Theta}}\boldsymbol{\Gamma}_\Theta^{-1}\tilde{\boldsymbol{\Theta}}) + \frac{1}{2}\tilde{\boldsymbol{\psi}}\boldsymbol{\Gamma}_\psi^{-1}\tilde{\boldsymbol{\psi}} \tag{6.32}$$

The derivative of Eq. (6.32) is Eq. (6.33).

$$\begin{aligned}\dot{V}_2 &= \dot{V}_1 + \boldsymbol{z}_2^T\dot{\boldsymbol{z}}_2 + tr(\tilde{\boldsymbol{\Theta}}\boldsymbol{\Gamma}_\Theta^{-1}\dot{\tilde{\boldsymbol{\Theta}}}) + \tilde{\boldsymbol{\psi}}^T\boldsymbol{\Gamma}_\psi^{-1}\dot{\tilde{\boldsymbol{\psi}}} \\ &= \dot{V}_1 + \boldsymbol{z}_2^T(\dot{\boldsymbol{\omega}}_{des} - \boldsymbol{\Theta}^*\boldsymbol{\Phi} - \boldsymbol{J}^{-1}\boldsymbol{t} - \dot{\boldsymbol{\alpha}}_1 + \boldsymbol{t}_{dist}) + tr(\tilde{\boldsymbol{\Theta}}\boldsymbol{\Gamma}_\Theta^{-1}\dot{\tilde{\boldsymbol{\Theta}}}) + \tilde{\boldsymbol{\psi}}^T\boldsymbol{\Gamma}_\psi^{-1}\dot{\tilde{\boldsymbol{\psi}}}\end{aligned} \tag{6.33}$$

Using the bounding inequality for disturbances, $\boldsymbol{z}_2^T\boldsymbol{t}_{dist}$ can be bounded by $\boldsymbol{z}_2^T\boldsymbol{t}_{dist} \leq \boldsymbol{z}_2^T\text{sgn}(\boldsymbol{z}_2)\boldsymbol{\psi}^*$, where $\text{sgn}(\boldsymbol{z}_2) = diag[\text{sgn}(z_{2,1}), \text{sgn}(z_{2,2}), \text{sgn}(z_{2,3})]$. Noting that $-\boldsymbol{z}_2^T\boldsymbol{\Theta}^*\boldsymbol{\Phi} = \boldsymbol{z}_2^T\tilde{\boldsymbol{\Theta}}\boldsymbol{\Phi} - \boldsymbol{z}_2^T\hat{\boldsymbol{\Theta}}\boldsymbol{\Phi}$

and $\boldsymbol{z}_2^T \text{sgn}(\boldsymbol{z}_2)\boldsymbol{\psi}^* = -\boldsymbol{z}_2^T \text{sgn}(\boldsymbol{z}_2)\tilde{\boldsymbol{\psi}} + \boldsymbol{z}_2^T \text{sgn}(\boldsymbol{z}_2)\hat{\boldsymbol{\psi}}$, Eq. (6.33) can be rewritten as Eq. (6.34).

$$\begin{aligned} \dot{V}_2 \leq \dot{V}_1 + \boldsymbol{z}_2^T(\dot{\boldsymbol{\omega}}_{des} - \hat{\boldsymbol{\Theta}}\boldsymbol{\Phi} - \boldsymbol{J}^{-1}\boldsymbol{t} - \dot{\boldsymbol{\alpha}}_1) + \boldsymbol{z}_2^T \text{sgn}(\boldsymbol{z}_2)\hat{\boldsymbol{\psi}} + \boldsymbol{z}_2^T\tilde{\boldsymbol{\Theta}}\boldsymbol{\Phi} \\ + tr(\tilde{\boldsymbol{\Theta}}\boldsymbol{\Gamma}_{\boldsymbol{\Theta}}^{-1}\dot{\hat{\boldsymbol{\Theta}}}) - \boldsymbol{z}_2^T \text{sgn}(\boldsymbol{z}_2)\tilde{\boldsymbol{\psi}} + \tilde{\boldsymbol{\psi}}^T\boldsymbol{\Gamma}_{\boldsymbol{\psi}}^{-1}\dot{\hat{\boldsymbol{\psi}}} \end{aligned} \tag{6.34}$$

The adaptive parameter update laws are shown in Eq. (6.35) and Eq. (6.36).

$$\dot{\hat{\boldsymbol{\Theta}}} = -\boldsymbol{\Gamma}_\Theta \boldsymbol{\Phi} \boldsymbol{z}_2^T \tag{6.35}$$

$$\dot{\hat{\boldsymbol{\psi}}} = \boldsymbol{\Gamma}_\psi \text{sgn}(\boldsymbol{z}_2)\boldsymbol{z}_2 \tag{6.36}$$

The update laws are computed by zeroing out like terms in Eq. (6.34), which are gathered in Eq. (6.37) and Eq. (6.38).

$$\boldsymbol{z}_2^T\tilde{\boldsymbol{\Theta}}\boldsymbol{\Phi} + tr(\tilde{\boldsymbol{\Theta}}\boldsymbol{\Gamma}_\Theta^{-1}\dot{\hat{\boldsymbol{\Theta}}}) = tr(\tilde{\boldsymbol{\Theta}}(\boldsymbol{\Gamma}_\Theta^{-1}\dot{\hat{\boldsymbol{\Theta}}} + \boldsymbol{\Phi}\boldsymbol{z}_2^T)) \tag{6.37}$$

$$-\boldsymbol{z}_2^T \text{sgn}(\boldsymbol{z}_2)\tilde{\boldsymbol{\psi}} + \tilde{\boldsymbol{\psi}}^T\boldsymbol{\Gamma}_\psi^{-1}\dot{\hat{\boldsymbol{\psi}}} = \tilde{\boldsymbol{\psi}}^T(\boldsymbol{\Gamma}_\psi^{-1}\dot{\hat{\boldsymbol{\psi}}} - \text{sgn}(\boldsymbol{z}_2)\boldsymbol{z}_2) \tag{6.38}$$

To make the update laws more robust, a projection operator will be used to determine $\dot{\hat{\boldsymbol{\Theta}}}$, and σ-modification will be used to compute $\dot{\hat{\boldsymbol{\psi}}}$, resulting in the final adaptive laws: $\dot{\hat{\boldsymbol{\Theta}}} = Proj(\dot{\hat{\boldsymbol{\Theta}}}, -\boldsymbol{\Gamma}_\Theta\boldsymbol{\Phi}\boldsymbol{z}_2^T)$ and $\dot{\hat{\boldsymbol{\psi}}} = \boldsymbol{\Gamma}_\psi[\text{sgn}(\boldsymbol{z}_2)\boldsymbol{z}_2 - \sigma\hat{\boldsymbol{\psi}}]$ [82].

The final control law for $\boldsymbol{t}$ in Eq. (6.39) is very similar to the nonadaptive backstepping controller designed previously. The key differences are the inclusion of a term to cancel out disturbances and the necessity to compute update laws for estimated parameters.

$$\boldsymbol{t}_{cmd} = \boldsymbol{J}(\boldsymbol{G}^T\boldsymbol{z}_1 + \boldsymbol{K}_2\boldsymbol{z}_2 - \hat{\boldsymbol{\Theta}}\boldsymbol{\Phi} + \text{sgn}(\boldsymbol{z}_2)\hat{\boldsymbol{\psi}} + \dot{\boldsymbol{\omega}}_{des} - \dot{\boldsymbol{\alpha}}_1) \tag{6.39}$$

Trial and error was used to determine the tuning gains, projection operator parameters, and initial conditions shown in Table 6.3.

Table 6.3: Backstepping and Adaptive Backstepping Parameters

Parameter	**Value**
$\boldsymbol{K}_1$	$15I_{3\times3}$
$\boldsymbol{K}_2$	$50I_{3\times3}$
$\boldsymbol{\Gamma}_\Theta$	$I_{3\times3}$
$\boldsymbol{\Gamma}_\psi$	$I_{3\times3}$
ε_Θ	[0.5 0.5 0.5]
$\hat{\boldsymbol{\Theta}}_{max}$	[1 1 1]
$\hat{\boldsymbol{\Theta}}_0$	$-\boldsymbol{J}^{-1}\text{diag}([J_z - J_y, J_x - J_z, J_y - J_x])$

The nonlinear attitude dynamics led to the development of four quaternion-based nonlinear attitude controllers for the omnicopter. In addition to the original nonadaptive PI/FL controller, a nonadaptive backstepping controller was also developed. The backstepping controller became the basis for the adaptive backstepping controller, and a nonlinear dynamic inversion controller was used to develop the reference model for the aMRAC attitude controller. Having developed the position and attitude controllers, the next Chapter presents the simulation results for each of the controller combinations.

Chapter 7

Simulation Results

The simulation results in this Chapter were computed using the set-up described in Chapter 4. Each of the 12 position/attitude controller combinations was simulated on a jagged and smooth trajectory. Each trajectory was flown with the nominal omnicopter mass and an increased omnicopter mass. Additionally, for each trajectory and mass flown, the omnicopter was subject to no, small, and large magnitude disturbances. For every mass and disturbance combination, 10 flights of the selected trajectory were simulated. The only difference between the 10 flights was the shape of the disturbances because they were randomly generated.

7.1 Error Computation

The position error for each point in time is computed as the distance between the true and desired location at the specified timestep, shown in Eq. (7.1).

$$e^n_{pos,k} = \sqrt{(x_k - x_{k,des})^2 + (y_k - y_{k,des})^2 + (z_k - z_{k,des})^2} \tag{7.1}$$

The attitude error for each point in time is computed as the angle associated with the geodesic distance between the true and desired attitude, shown in Eq. (7.2)

$$e^n_{att,k} = cos^{-1}(2(q_k \cdot q_{des,k})^2 - 1) \tag{7.2}$$

Figures 7.1 - 7.4 show the average position and attitude errors across all runs for each point in time, computed using Eq. (7.3). For the following equations, the subscript *pos* can be substituted with *att* to apply to attitude error.

$$e_{pos,plot,k} = \frac{\sum_{n=1}^{N} e_{pos,k}^{n}}{N} \tag{7.3}$$

Mean error is used to characterize position and attitude error across all time, seen in Eq. (7.4).

$$ME_{pos}^{n} = \frac{\sum_{k=t_0}^{t_{end}} e_{pos,k}^{n}}{T} \tag{7.4}$$

Displayed in Tables 7.1-7.4 are the position and attitude ME averaged over all runs, computed using Eq. (7.5).

$$ME_{pos,total} = \frac{\sum_{n=1}^{N} ME_{pos}^{n}}{N} \tag{7.5}$$

7.2 Error Results

7.2.1 Nominal Mass Results

The position and attitude simulation error results are shown in Figures 7.1 and 7.2 and Tables 7.1 and 7.2. Overall, the smooth trajectory yields smaller position and attitude tracking errors than the jagged trajectory, as expected. The jagged trajectory exhibits large peaks in error every 10 seconds that correlate with commanded maneuvers. Since the smooth trajectory prevents abrupt changes in direction, peaks in error tend to be smaller in magnitude, if they occur at all. For both trajectories and for all disturbances, all controllers are able to successfully control the omnicopter to the final position without becoming unstable. For

both trajectories, position error tends to increase with increasing disturbance. Disturbance increases do not appear to affect the tracking ability of the attitude controllers.

The aMRAC position controller consistently yields the smallest average position errors, regardless of attitude controller, trajectory, or disturbance. When there are no or small disturbances in the system, PID and MRAC tend to have similar average tracking errors. The adaptive backstepping attitude controller consistently yields the smallest attitude errors, regardless of position controller, trajectory, or disturbance. The average attitude tracking improvement compared to the other attitude controllers is about 0.3 - 1 degrees, as seen in Table 7.2. The aMRAC controller averages the largest attitude errors. For the jagged trajectory, the average backstepping error is smaller than the PI/FL error. For the smooth trajectory, however, PI/FL tends to have smaller average errors that tend to be within about 0.01-0.1 degrees. It should be noted that for the jagged trajectory in Figure 7.1, the PI/FL controller has a longer settling time than the other three controllers, although it converges to a smaller error than aMRAC and backstepping. Its lowest values achieve the error magnitude adaptive backstepping settles to. This transient response does not appear in the smooth trajectory, where PI/FL tends to maintain a constant error just above adaptive backstepping. It should also be noted that for both trajectories the aMRAC and backstepping attitude error lines tend to parallel each other, with the backstepping line running just below the aMRAC line.

Although there are twelve controller combinations, the position error results in Figures 7.1 and 7.2 tend to group together in three distinct error trends. The trends are denoted by similar point markers and line shading. The trends become more pronounced with increasing disturbance, distinguishing the controllers that are the best at handling disturbances (aMRAC followed by MRAC). For no/small disturbances, the PID and MRAC error lines tend to cross and remain closely grouped, reflecting the similarities in average error men-

tioned earlier. Close overlap within a band indicates that position controller performance is not significantly affected by attitude controller choice. For aMRAC position control error, the aMRAC/PIFL error line is separated from the aMRAC error group, indicating that the PI/FL attitude controller affects the aMRAC position controller performance more than the other 3 attitude controllers.

The attitude error results exhibit similar error groupings. These trends are grouped by color, indicating the choice of attitude control. Tight trends indicate that the choice of position controller does not affect the performance of the attitude controller. The high frequency oscillations seen in the PI/FL (green lines) and adaptive backstepping (orange lines) occur at the desired trajectory input frequency (50 Hz), which is less than the simulation solver frequency. Therefore, the actual values are being compared to a constant reference value for each input frequency cycle when computing error. These oscillations are negligible because they occur at the sub 0.5 degree scale and have a decreasing trend.

The aMRAC position controller combined with the PI/FL attitude controller yields the smallest average position tracking error for both trajectories and all disturbances (red asterisks in Table 7.1). However, the PID position controller combined with the adaptive backstepping attitude controller yields the smallest average attitude tracking error for the jagged trajectory while aMRAC combined with adaptive backstepping yields the smallest average errors for the smooth trajectory (blue asterisks). While these individually indicate the best average errors, the position/attitude errors associated with the preferred controller combination are circled in red/blue and will be discussed in Section 7.4.

Table 7.1: Mean Position Error of 10 Runs for the Omnicopter at Nominal Mass. Asterisks indicate the lowest position error amongst all controller combinations for a disturbance magnitude. Circles indicate the error associated with the best position/attitude controller combination.

Nominal Mass Position Mean Error (cm)									
		Jagged Trajectory				Smooth Trajectory			
Pos Ctrl \ Att Ctrl		PI/FL	aMRAC	Back	aBack	PI/FL	aMRAC	Back	aBack
No Disturbance	PID	1.78	2.05	2.27	2.37	1.32	1.37	1.39	1.39
	MRAC	2.04	2.19	2.33	2.37	1.77	1.80	1.81	1.81
	aMRAC	0.52*	0.76	0.80	(0.86)	0.27*	0.49	0.46	(0.47)
Small Disturbance	PID	2.34	2.58	2.79	2.87	1.85	1.89	1.91	1.91
	MRAC	2.14	2.29	2.43	2.47	1.85	1.88	1.89	1.89
	aMRAC	0.55*	0.83	0.85	(0.93)	0.3*	0.55	0.52	(0.53)
Large Disturbance	PID	6.14	6.31	6.48	6.52	5.71	5.75	5.76	5.76
	MRAC	3.46	3.60	3.74	3.76	3.13	3.16	3.17	3.17
	aMRAC	0.79*	1.17	1.19	(1.27)	0.62*	0.97	0.94	(0.96)

Table 7.2: Mean Attitude Error of 10 Runs for the Omnicopter at Nominal Mass. Asterisks indicate the lowest attitude error amongst all controller combinations for a disturbance magnitude. Circles indicate the error associated with the best position/attitude controller combination.

Nominal Mass Attitude Mean Error (deg)									
		Jagged Trajectory				Smooth Trajectory			
	Pos Ctrl \ Att Ctrl	PI/FL	aMRAC	Back	aBack	PI/FL	aMRAC	Back	aBack
No Disturbance	PID	3.43	3.61	2.82	2.59*	0.58	1.02	0.71	0.32
	MRAC	3.45	3.63	2.83	2.60	0.59	1.02	0.71	0.33
	aMRAC	3.74	3.82	3.00	(2.74)	0.69	1.03	0.71	(0.32*)
Small Disturbance	PID	3.43	3.61	2.82	2.59*	0.58	1.02	0.70	0.32
	MRAC	3.45	3.63	2.83	2.60	0.59	1.02	0.71	0.33
	aMRAC	3.74	3.82	2.99	(2.74)	0.69	1.03	0.71	(0.32*)
Large Disturbance	PID	3.43	3.62	2.81	2.59*	0.59	1.02	0.71	0.32
	MRAC	3.45	3.63	2.82	2.59	0.60	1.02	0.71	0.33
	aMRAC	3.71	3.81	2.98	(2.73)	0.70	1.03	0.71	(0.32*)

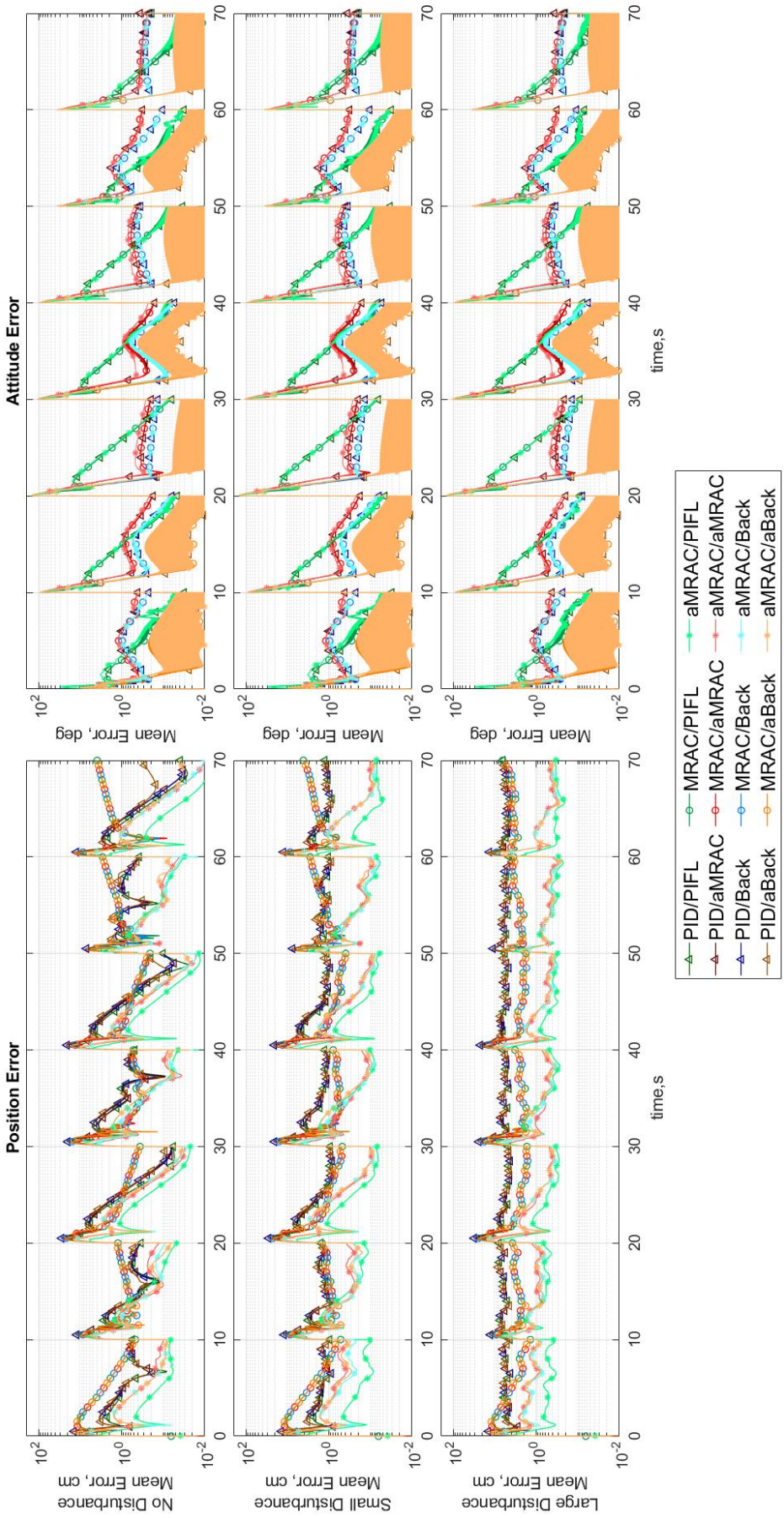


Figure 7.1: Jagged trajectory mean error for position (left) and attitude (right) averaged over 10 runs for the omnicopter at nominal mass.

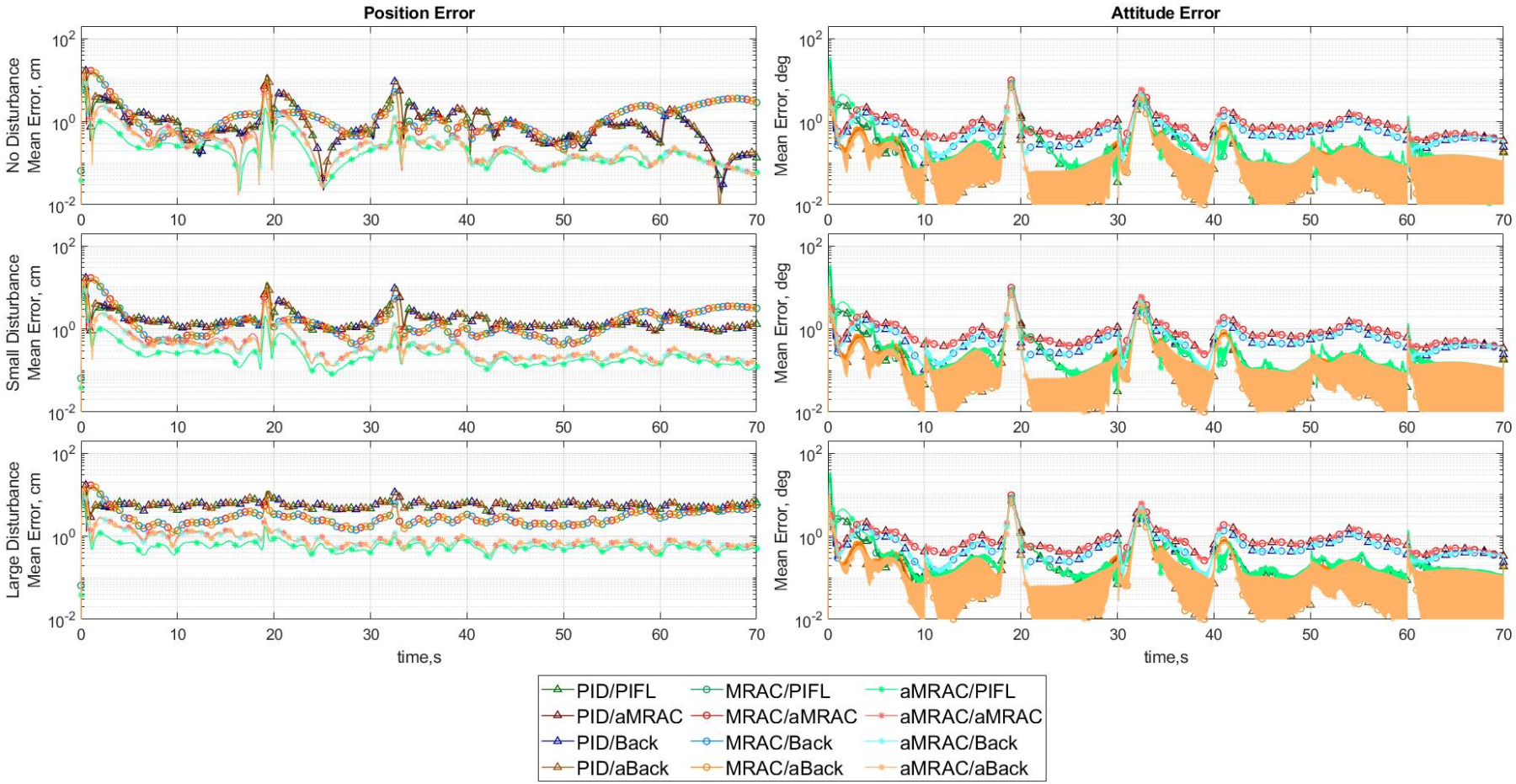


Figure 7.2: Smooth trajectory mean error for position (left) and attitude (right) averaged over 10 runs for the omnicopter at nominal mass.

7.2.2 Increased Mass Results

When mass has been added to the omnicopter, error magnitudes tend to increase compared to the nominal mass scenario for both position and attitude. Although the errors increase, several trends observed in the nominal mass case remain the same, see Figures 7.3 and 7.4 and Tables 7.3 and 7.4. The smooth trajectory yields smaller position and attitude tracking errors than the jagged trajectory, and all controllers are able to successfully control the omnicopter to the final position. For both trajectories, position error tends to increase with increasing disturbance.

The aMRAC position controller consistently yields the smallest position errors, regardless of attitude controller, trajectory, or disturbance. PID tends to have the largest position tracking errors, except for when there is no disturbance in the system. For no and small disturbances, the average error for MRAC and PID are similar, which can also be seen in the overlap of their error groups in Figures 7.3 and 7.4.

Unlike the nominal mass scenario, the adaptive backstepping attitude controller consistently yields the smallest attitude errors, regardless of position controller, trajectory, or disturbance. It provides about a 8-16 degree improvement over the PI/FL controller, which performs the worst. Backstepping performs similar or slightly better than PI/FL, and aMRAC performs slightly worse than aBack (about 2-3 degrees). These average trends are also reflected in the error groupings of Figures 7.3 and 7.4. Green (PI/FL) and blue (backstepping) lines parallel each other closely and stay above the parallel red (aMRAC) and orange (adaptive backstepping) error lines. Additionally, although disturbances do not significantly affect the attitude tracking of the aMRAC, backstepping, and adaptive backstepping controllers, the PI/FL controller is more sensitive to the disturbance magnitude than it was in the nominal mass scenario. Its average errors range by about 0.03-2.43 degrees, more than the 0.01-0.03

degrees for the nominal mass scenario.

Figures 7.3 and 7.4 exhibit error grouping behavior similar to the nominal mass scenario for both position and attitude error. Unlike the nominal mass scenario, the MRAC/PIFL combination results in significant position error increases that correlate with attitude error increases and oscillations after some direction changes. This occurs for both the jagged and smooth trajectories and can also be observed in the PID/PIFL combination, although with smaller error magnitudes. It is unclear why specific direction changes trigger the oscillations in attitude and position for these controllers. The decrease in the oscillations with the smooth trajectory indicates that smooth trajectories can alleviate undesired behavior that can be caused by discontinuities.

The aMRAC position controller combined with the PI/FL attitude controller yields the smallest average position tracking error for both trajectories and all disturbances (red asterisks in Table 7.3). However, the PID position controller combined with the adaptive backstepping attitude controller yields the smallest average attitude tracking error for the jagged trajectories, while the aMRAC position controller combined with the adaptive backstepping attitude controller yields the smallest errors for the smooth trajectory (blue asterisks). Circled in red/blue are the average position/attitude errors associated with the preferred controller combination, which will be discussed in Section 7.4.

Table 7.3: Mean position error of 10 Runs for the Omnicopter lifting a 0.25 kg unmodeled mass. Asterisks indicate the lowest position error amongst all controller combinations for a disturbance magnitude. Circles indicate the error associated with the best position/attitude controller combination.

Increased Mass Positon Mean Error (cm)									
		Jagged Trajectory				Smooth Trajectory			
	Att Ctrl / Pos Ctrl	PI/FL	aMRAC	Back	aBack	PI/FL	aMRAC	Back	aBack
No Disturbance	PID	2.35	4.91	3.56	4.88	1.67	2.04	1.72	2.24
	MRAC	15.74	4.24	3.72	4.52	2.79	2.58	2.40	2.58
	aMRAC	0.78*	2.81	1.86	(2.84)	0.37*	0.96	0.68	(0.83)
Small Disturbance	PID	3.06	5.62	4.30	5.40	2.43	2.76	2.46	2.85
	MRAC	15.63	4.40	3.87	4.69	2.94	2.70	2.52	2.69
	aMRAC	0.82*	2.89	1.93	(2.94)	0.41*	1.05	0.76	(0.90)
Large Disturbance	PID	8.38	10.74	9.30	10.33	7.64	7.85	7.60	7.83
	MRAC	18.35	6.35	5.72	6.70	7.15	4.40	4.22	4.36
	aMRAC	1.26*	3.63	2.47	(3.73)	0.77*	1.61	1.30	(1.40)

Table 7.4: Mean Attitude Position Error of 10 Runs for the Omnicopter Lifting a 0.25 kg Panel. Asterisks indicate the lowest attitude error amongst all controller combinations for a disturbance magnitude. Circles indicate the error associated with the best position/attitude controller combination.

Increased Mass Attitude Mean Error (deg)									
		Jagged Trajectory				Smooth Trajectory			
Pos Ctrl \ Att Ctrl		PI/FL	aMRAC	Back	aBack	PI/FL	aMRAC	Back	aBack
No Disturbance	PID	16.72	10.27	16.50	7.83*	16.21	7.70	13.53	5.66
	MRAC	24.88	10.29	16.66	8.05	17.28	7.76	13.57	5.78
	aMRAC	16.65	10.35	16.77	(8.01)	16.47	7.68	13.61	(5.64*)
Small Disturbance	PID	16.78	10.26	16.50	7.83*	16.24	7.70	13.53	5.66
	MRAC	24.79	10.29	16.66	8.04	17.51	7.76	13.57	5.78
	aMRAC	16.69	10.35	16.76	(8.00)	16.50	7.69	13.61	(5.63*)
Large Disturbance	PID	17.52	10.22	16.54	7.85*	16.64	7.70	13.57	5.67
	MRAC	24.95	10.25	16.70	8.05	19.71	7.75	13.60	5.79
	aMRAC	17.02	10.33	16.79	(8.00)	16.79	7.68	13.63	(5.63*)

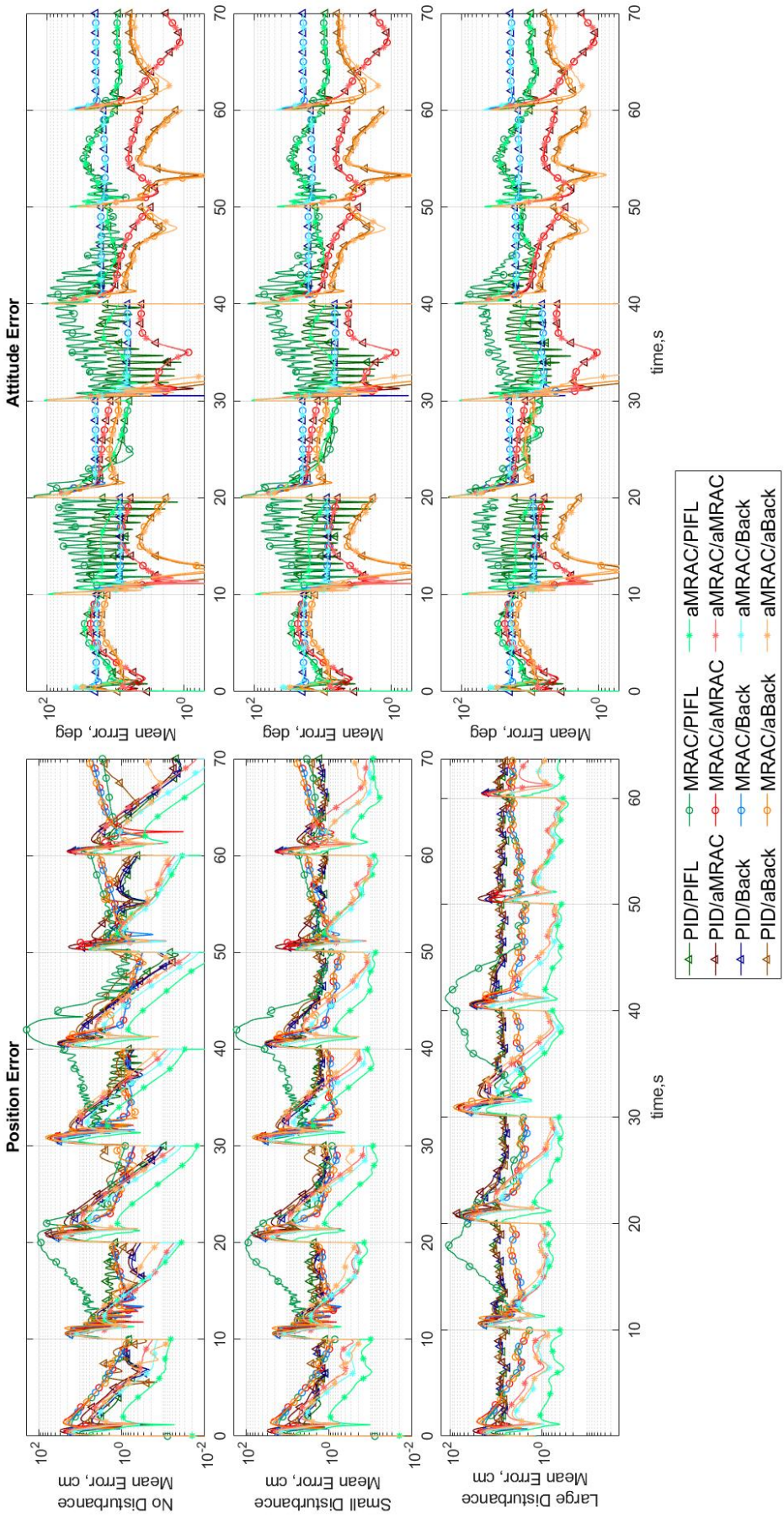


Figure 7.3: Jagged trajectory mean error for position (left) and attitude (right) averaged over 10 runs for the omnicopter with increased mass.

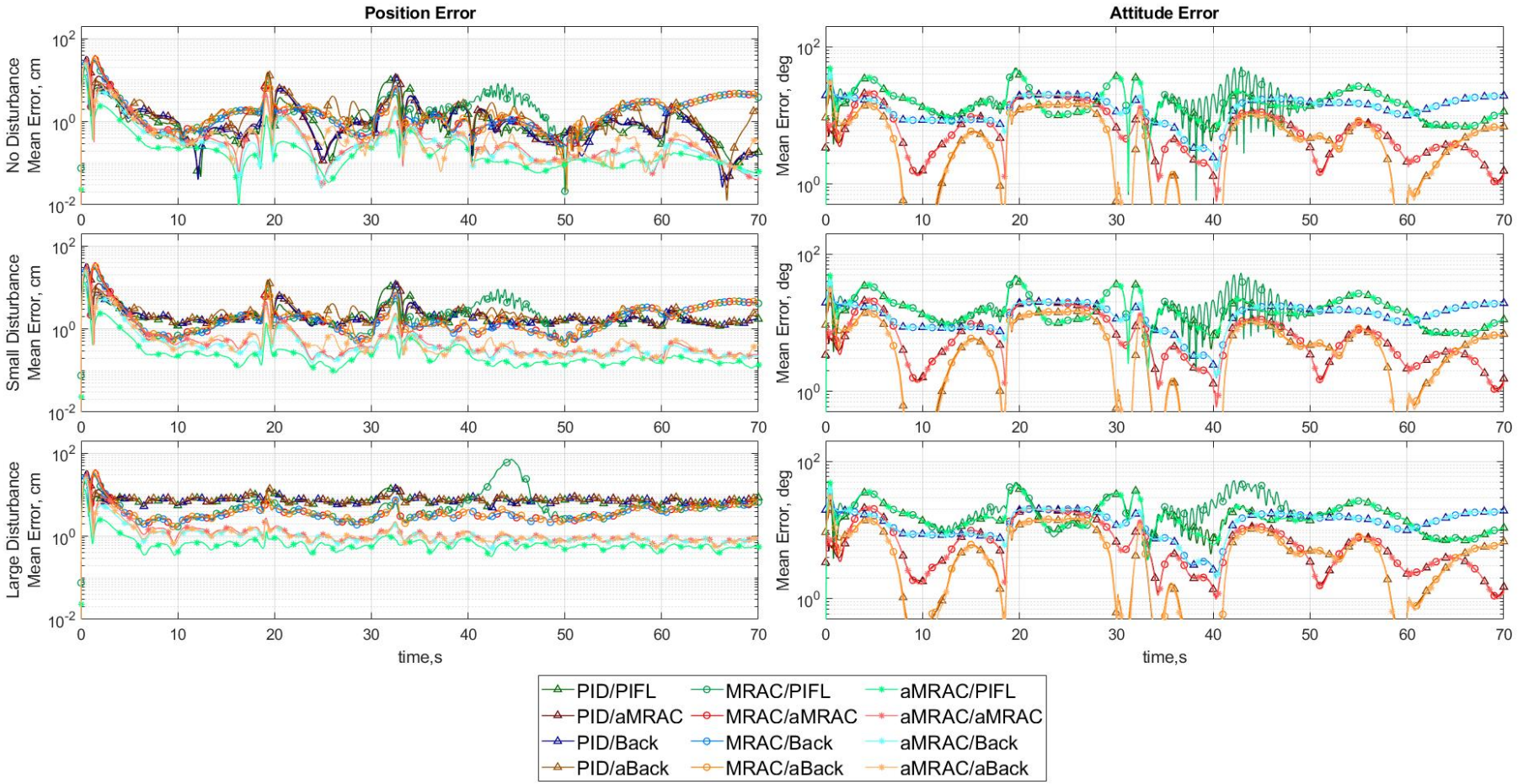


Figure 7.4: Smooth trajectory mean error for position (left) and attitude (right) averaged over 10 runs for the omnicopter with increased mass.

7.3 Controller Commands

In the following sections, only the results for the large disturbance scenario are shown. These results are representative of the general characteristics also seen in the no and small disturbance scenarios. Differences between these scenarios may be minor changes in magnitudes of values or slight differences in timing. The large disturbance scenario tends to have more variation in the steady state portion of the response than the no/small disturbance. This is expected because larger disturbances require more controller adjustments to compensate. However, as previously mentioned, these small changes do not affect the overall conclusions that can be drawn about the controller performance. The results shown are also for the first of 10 runs that were used to compute error. The only difference between runs is the disturbances the omnicopter experienced. These results are representative of trends that could be shown with any run.

7.3.1 Force Commands

The commanded forces by each controller combination to execute the jagged trajectory when the omnicopter is at its nominal mass are shown in Figure 7.5. The overall shape of the commanded force is the same for each controller combination, with slight differences in peak values and transient oscillations. All combinations that use aMRAC for position control tend to have the highest peaks when the omnicopter changes direction. Combinations that use the MRAC and aMRAC position controllers tend to have transient oscillations, although the MRAC oscillations take a longer time to dissipate.

The commanded forces by each controller combination to execute the smooth trajectory when the omnicopter is at its nominal mass are shown in Figure 7.6. As with the jagged trajectory, each controller combination commands similar force profiles. While transient responses

were able to differentiate controllers in the jagged trajectory, the lack of discontinuities in the smooth trajectory yields smaller command spikes and prevents transient oscillations in the force commands. When overshoots due to direction changes occur, aMRAC position controllers tend to have the highest overshoot. However, these still tend to be small, staying within about +/- 10 N.

The commanded forces by each controller combination to execute the jagged trajectory when the omnicopter is carrying a 0.25 kg mass are shown in Figure 7.7. The forces for the increased mass omnicopter are similar to the nominal mass omnicopter commands. Changes in direction cause a transient response characterized by large command overshoots and oscillations. The most significant changes are the commanded forces and overshoots are increased due to carrying a larger mass. Additionally, the PID/PIFL combination and MRAC/PIFL combination exhibit large transient oscillations when direction changes occur at 10, 30 and 40 seconds that last at least until the next direction change. MRAC/PIFL exhibits the largest magnitude oscillations.

The commanded forces by each controller combination to execute the smooth trajectory when the omnicopter is carrying a 0.25 kg mass are shown in Figure 7.8. Once again, the most significant change due to adding mass to the omnicopter is increasing force commands. The use of the smooth trajectory, again, appears to have improved controller performance. The large oscillations in PID/PIFL that occured with the jagged trajectory do not exist anymore, and the oscillations associated with MRAC/PIFL are significantly reduced in magnitude.

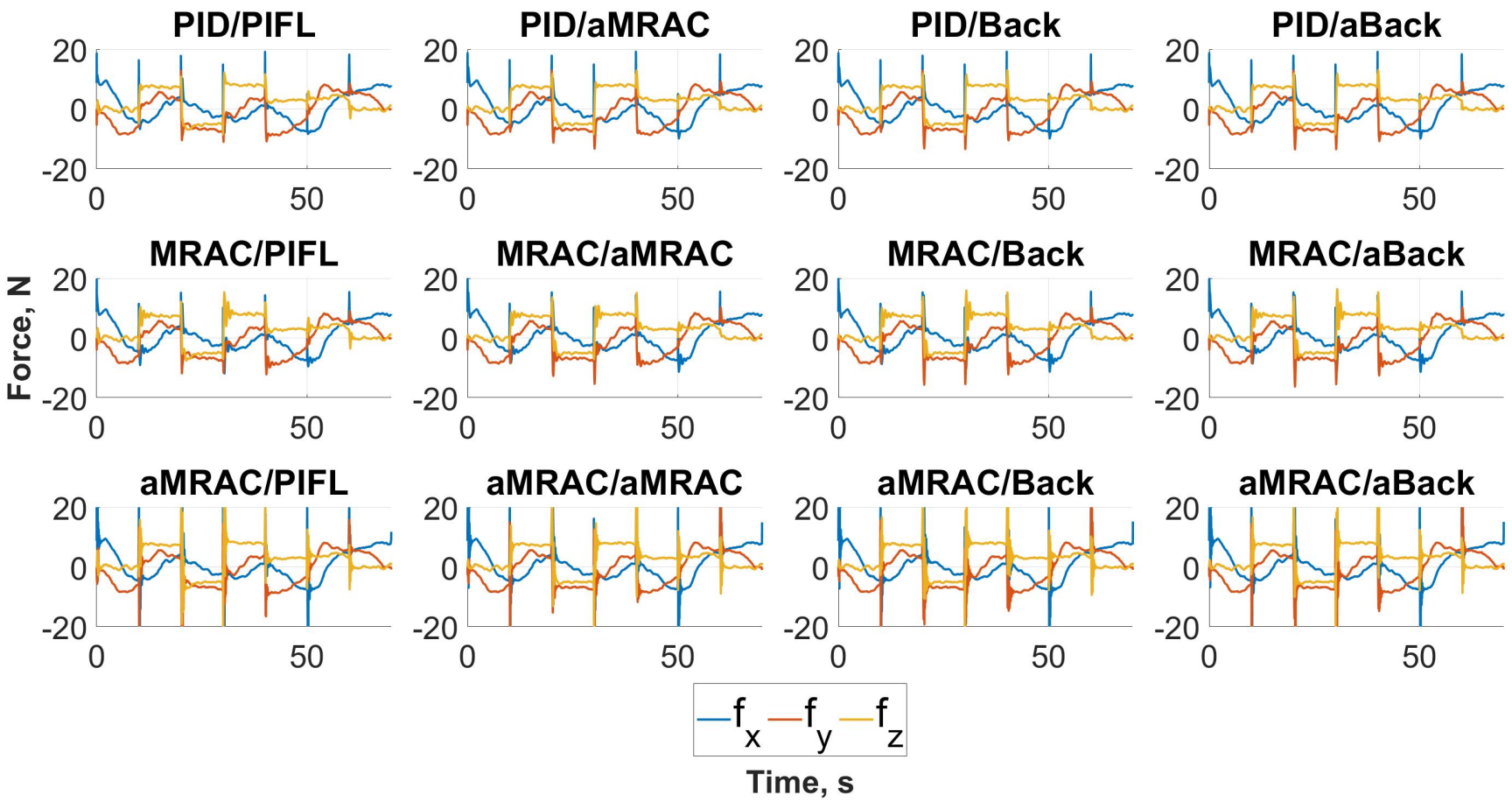


Figure 7.5: Jagged trajectory, nominal mass omnicopter force command comparison

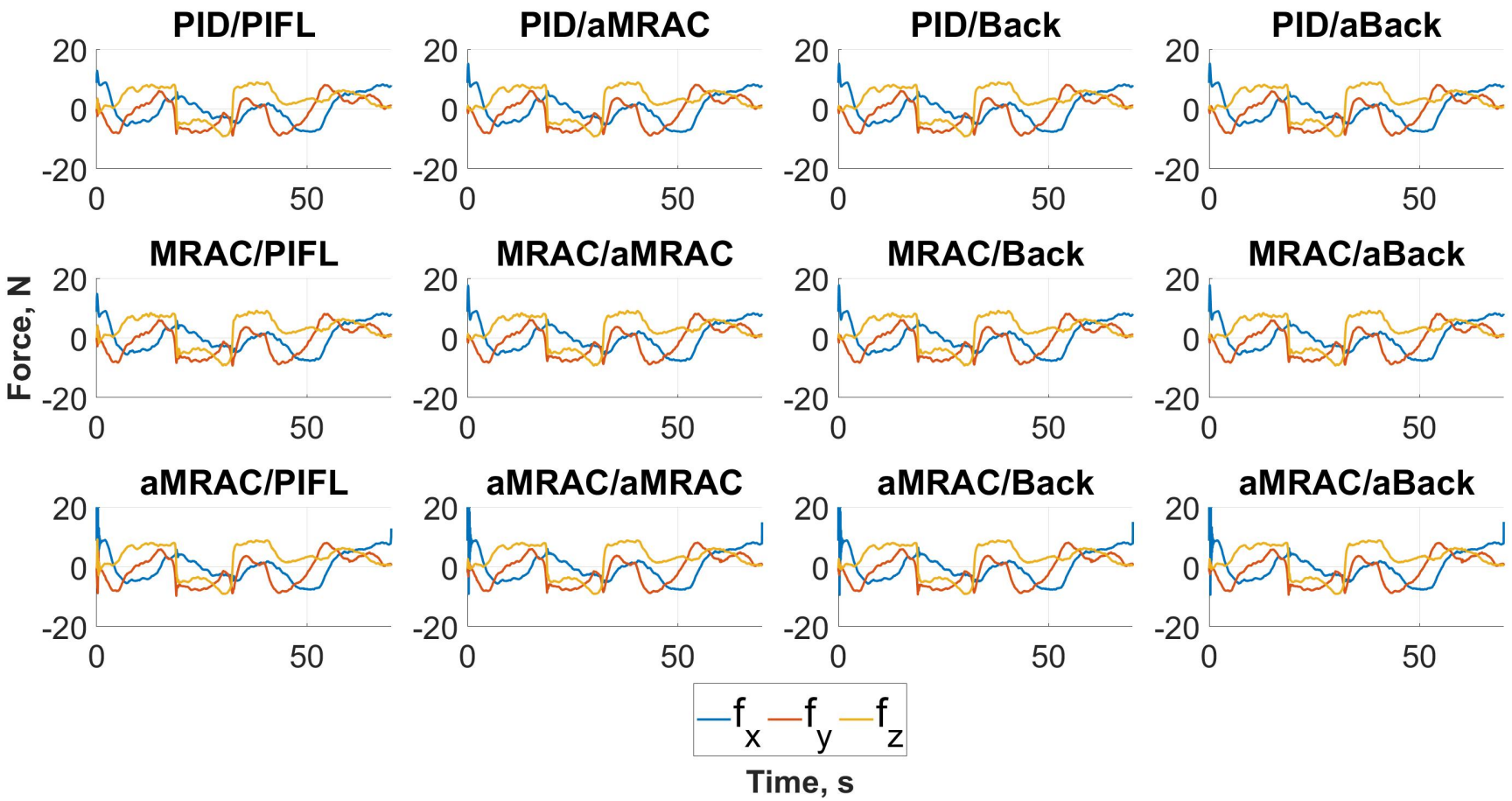


Figure 7.6: Smooth trajectory, nominal mass omnicopter force command comparison

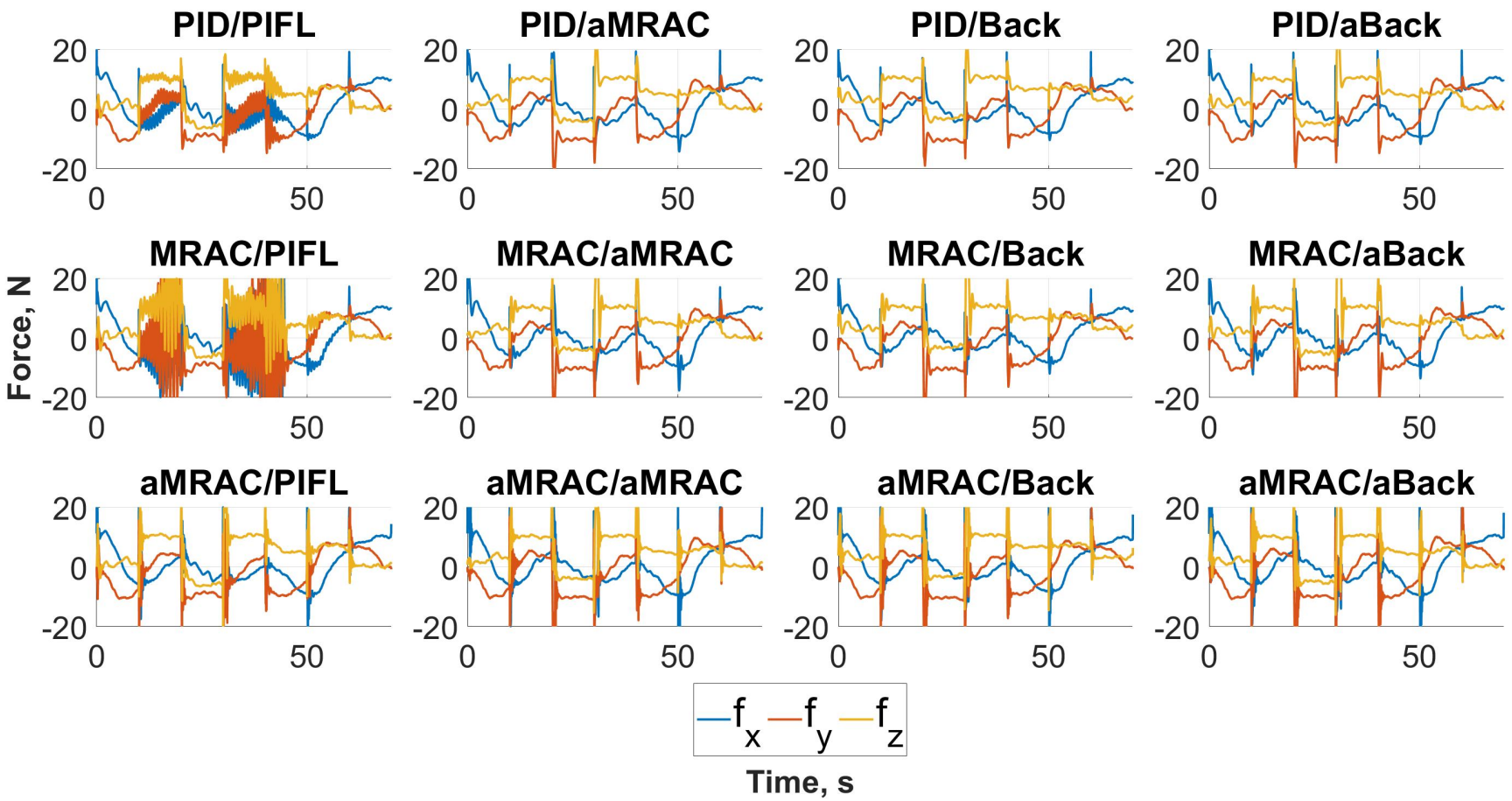


Figure 7.7: Jagged trajectory, increased mass omnicopter force command comparison

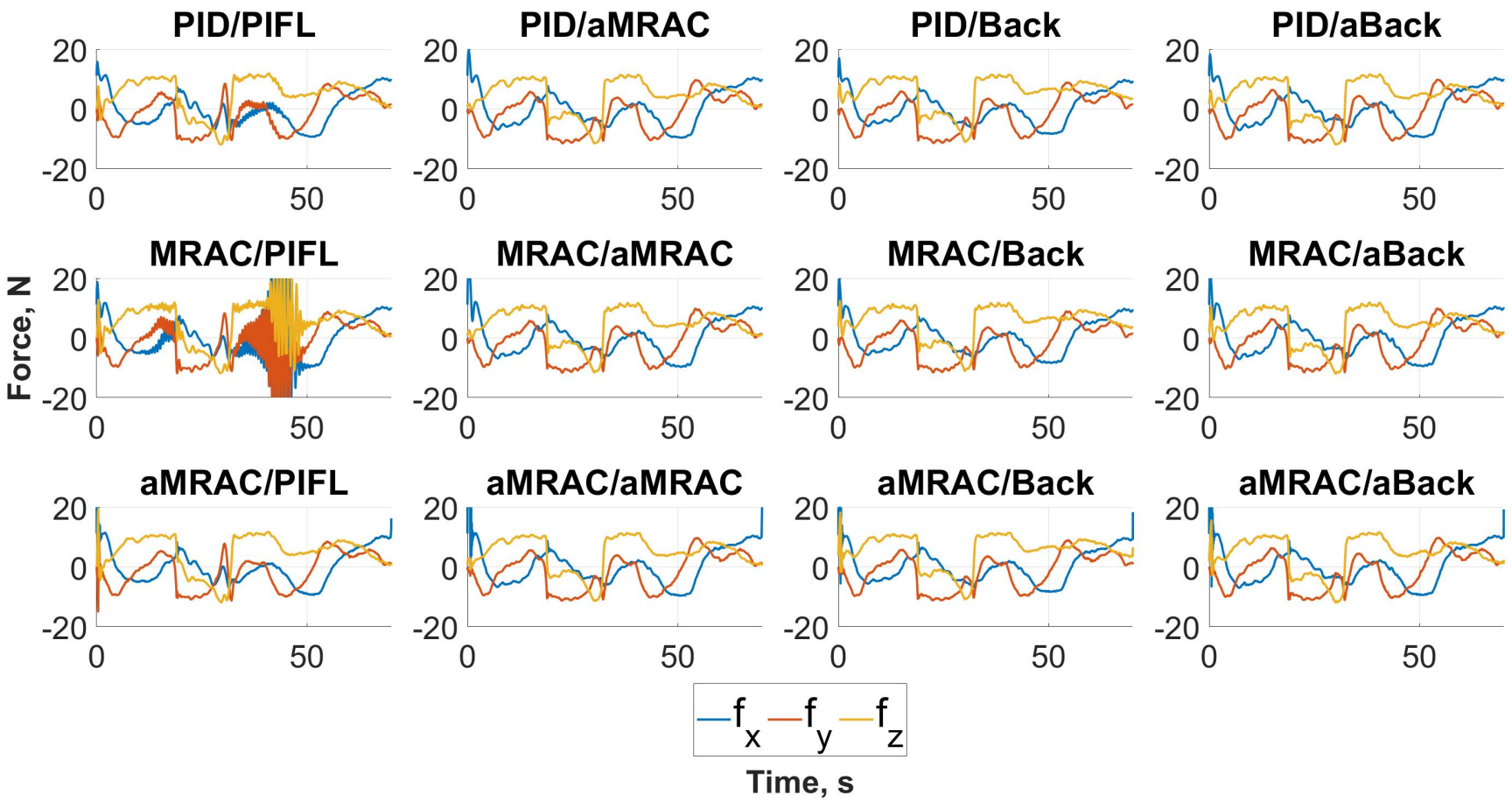


Figure 7.8: Smooth trajectory, increased mass omnicopter force command comparison

7.3.2 Torque Commands

The commanded torques by each controller combination to execute the jagged trajectory when the omnicopter is at its nominal mass are shown in Figure 7.9. Similar to the force commands, the overall shape of the commanded torques are the same for each controller combination, with slight differences in peak values and transient oscillations. Whenever the attitude controller is adaptive backstepping, the commanded torques appear to be marginally stable oscillations that grow with each direction change where torques about the x axis have the largest magnitude.

The commanded torques by each controller combination to execute the smooth trajectory when the omnicopter is at its nominal mass are shown in Figure 7.10. For the smooth trajectory, the overshoots that occur with direction changes are smaller in magnitude than for the jagged trajectory and do not occur in every torque command depending on the direction change. Once again, the adaptive backstepping controller exhibits marginally stable oscillations. For the smooth trajectory, the oscillations do not increase in magnitude after direction changes. Instead, they return to constant limits after the direction change. As seen in the jagged trajectory, commanded torques in the x axis are larger than those about the y and z axes.

The commanded torques by each controller combination to execute the jagged trajectory when the omnicopter is carrying a 0.25 kg mass are shown in Figure 7.11. The increase in mass causes commanded torques to have larger magnitudes than the nominal mass scenario. For the jagged trajectory, transient peaks and oscillations still exist. The oscillations associated with adaptive backstepping are still present, though they now consistently stay within the range of +/- 0.5 Nm instead of growing with each direction change. This time, the torque commands about the y axis match the magnitude of those about the x axis, both of which

are larger than commands about the z axis. The PID/PIFL and MRAC/PIFL controller combinations show large transient oscillations in the same locations that oscillations occur in the force commands. As before the magnitude of the PID/PIFL oscillations are smaller than the MRAC/PIFL oscillations.

The commanded torques by each controller combination to execute the smooth trajectory when the omnicopter is carrying a 0.25 kg mass are shown in Figure 7.12. As with the nominal mass scenario, the smooth trajectory yields smaller transient overshoots and reduces the transient oscillations compared to the jagged trajectory. The magnitudes of the commands are larger than they were for the nominal mass scenario. The increase in mass may have also damped out some of the transient oscillations that were present in the nominal mass scenario. Using aMRAC for attitude control can cause transient peaks to be larger than those commanded by the other attitude controllers. Using PI/FL as the attitude control does not appear to cause transient peaks when oscillations are not prevailing. The adaptive backstepping torque commands are similar to the jagged trajectory commands, hovering in +/- 0.5 Nm with commands about x and y being larger than those about z. As with the jagged trajectory and in the corresponding force commands, oscillations can be seen in the PID/PIFL and MRAC/PIFL torque commands. However, the oscillations in PID/PIFL are much smaller than for the jagged trajectory. There is also a decrease in oscillation magnitude for MRAC/PIFL torque commands.

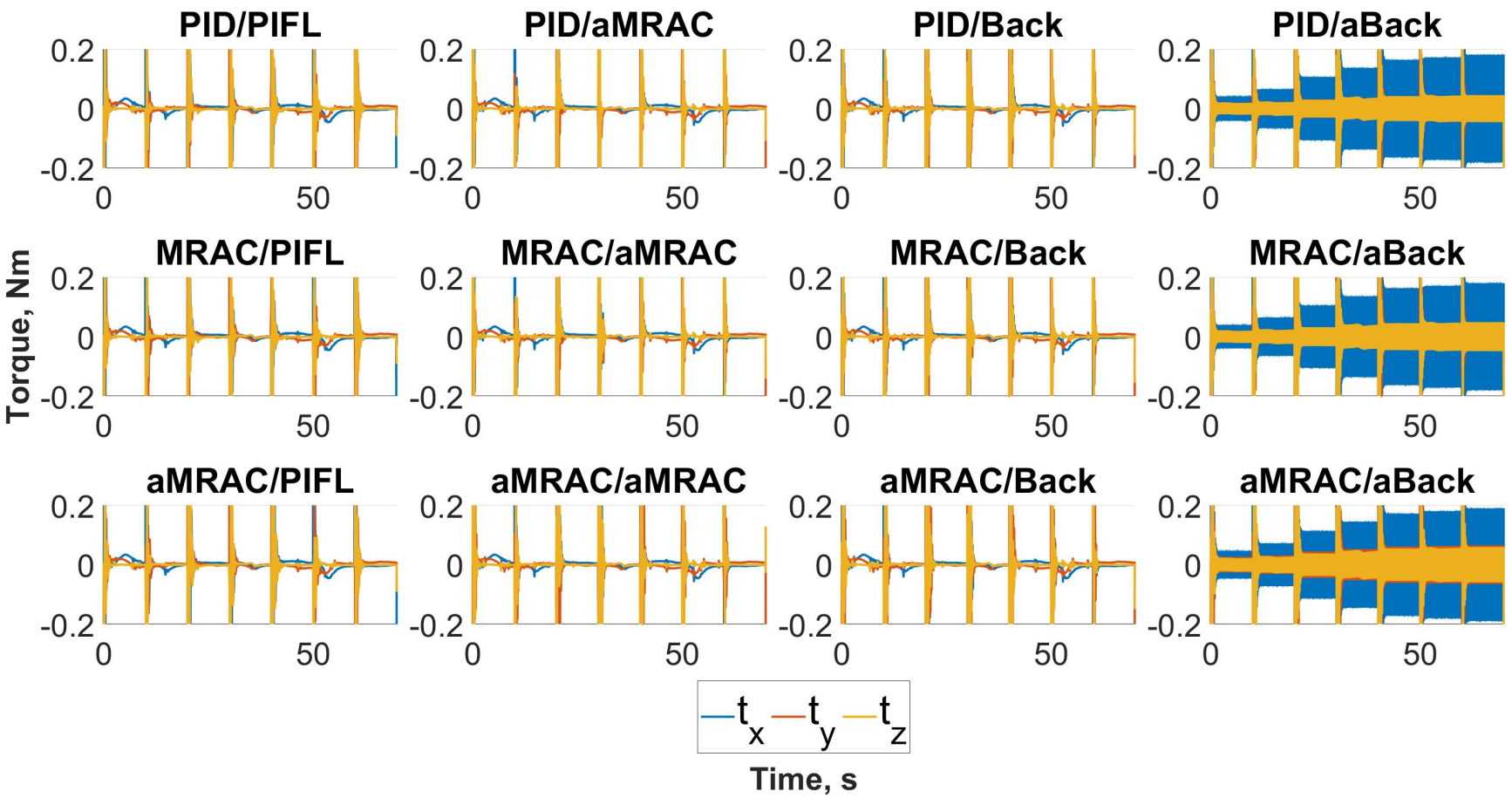


Figure 7.9: Jagged trajectory, nominal mass omnicopter torque command comparison. Note that transient peaks have been attenuated to preserve visualization of steady state responses.

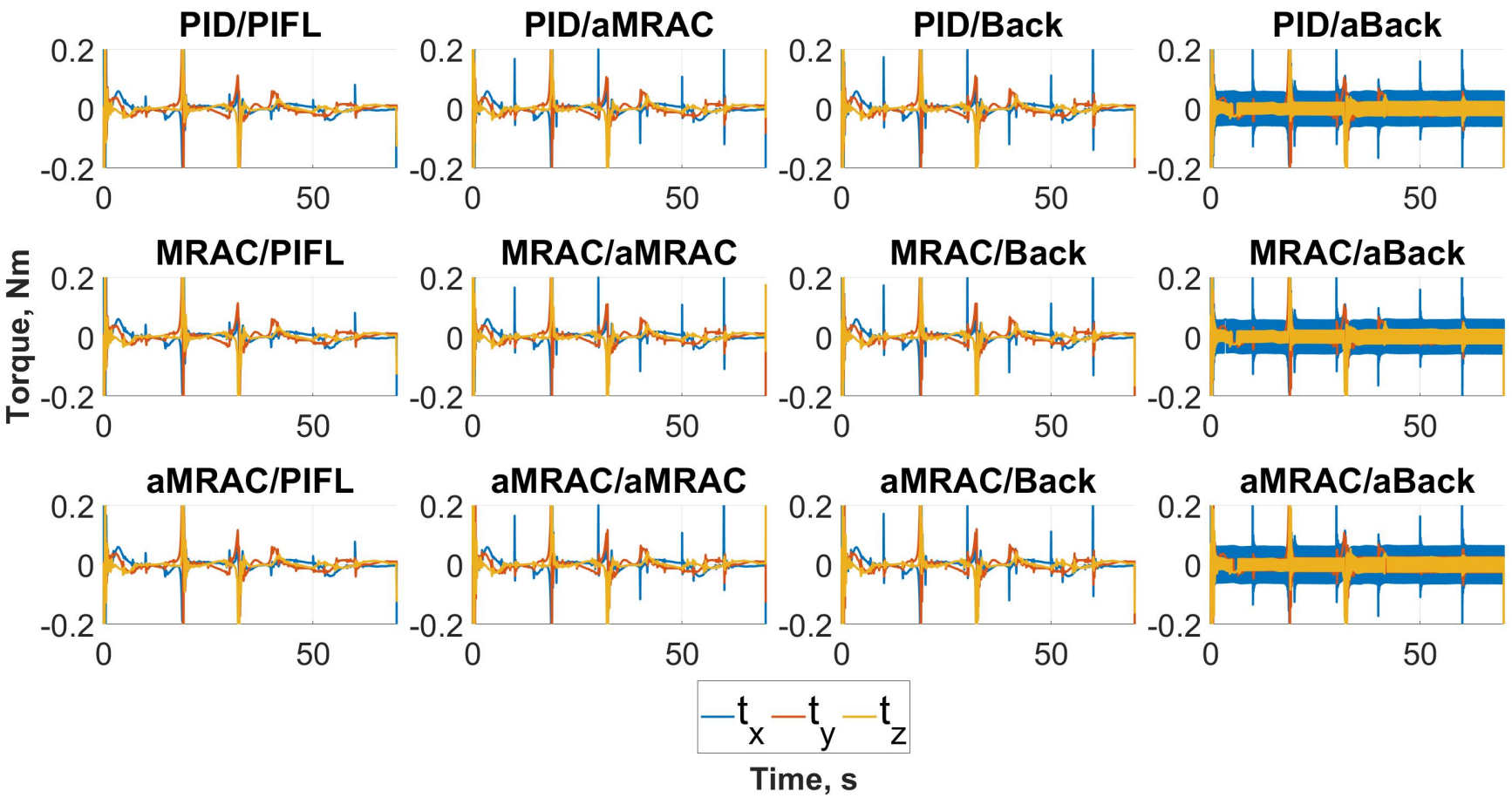


Figure 7.10: Smooth trajectory, nominal mass omnicopter torque command comparison. Note that some peaks have been attenuated to preserve visualization of steady state responses.

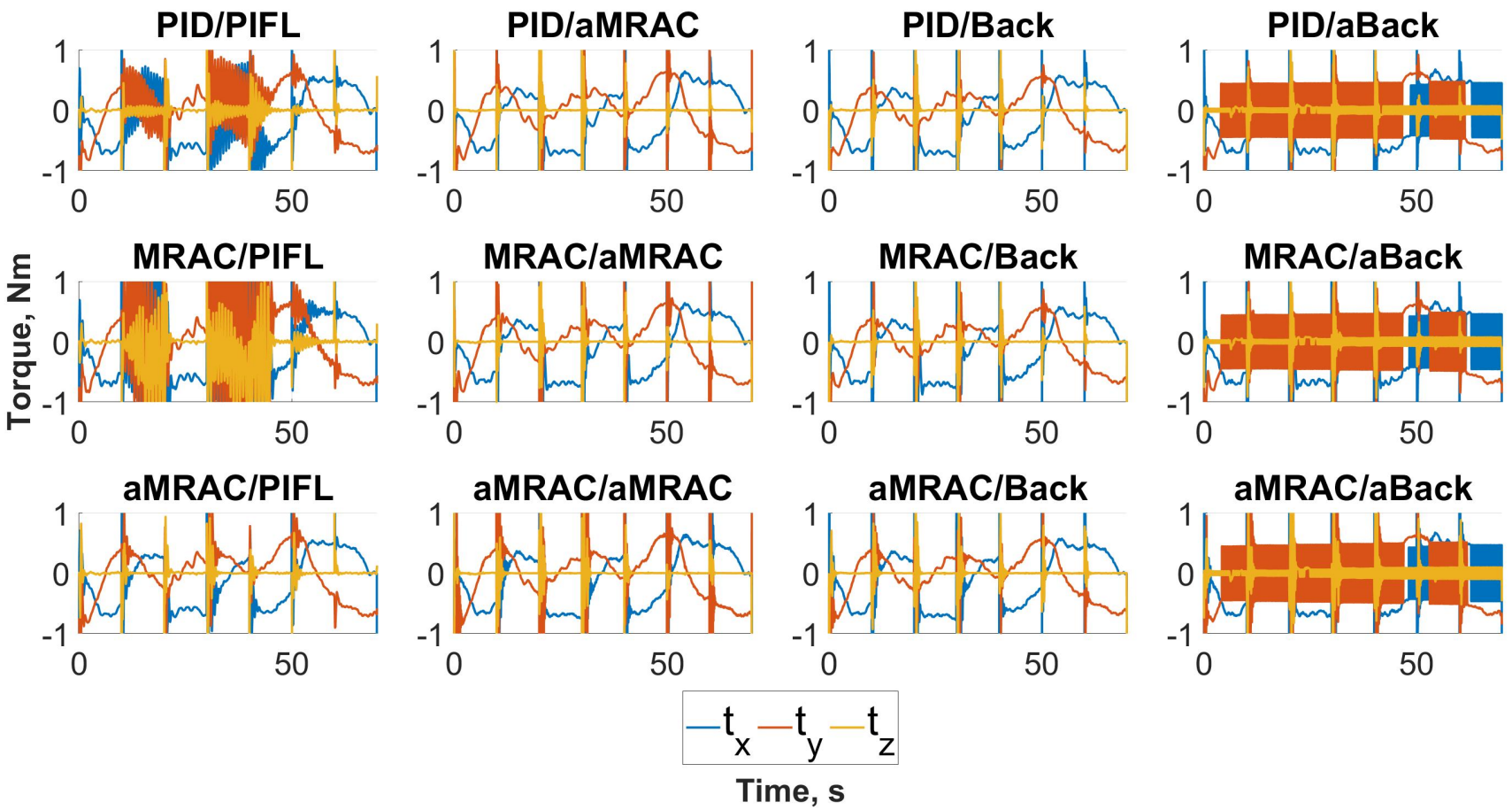


Figure 7.11: Jagged trajectory, increased mass omnicopter torque command comparison. Note that some peaks have been attenuated to preserve visualization of steady state responses.

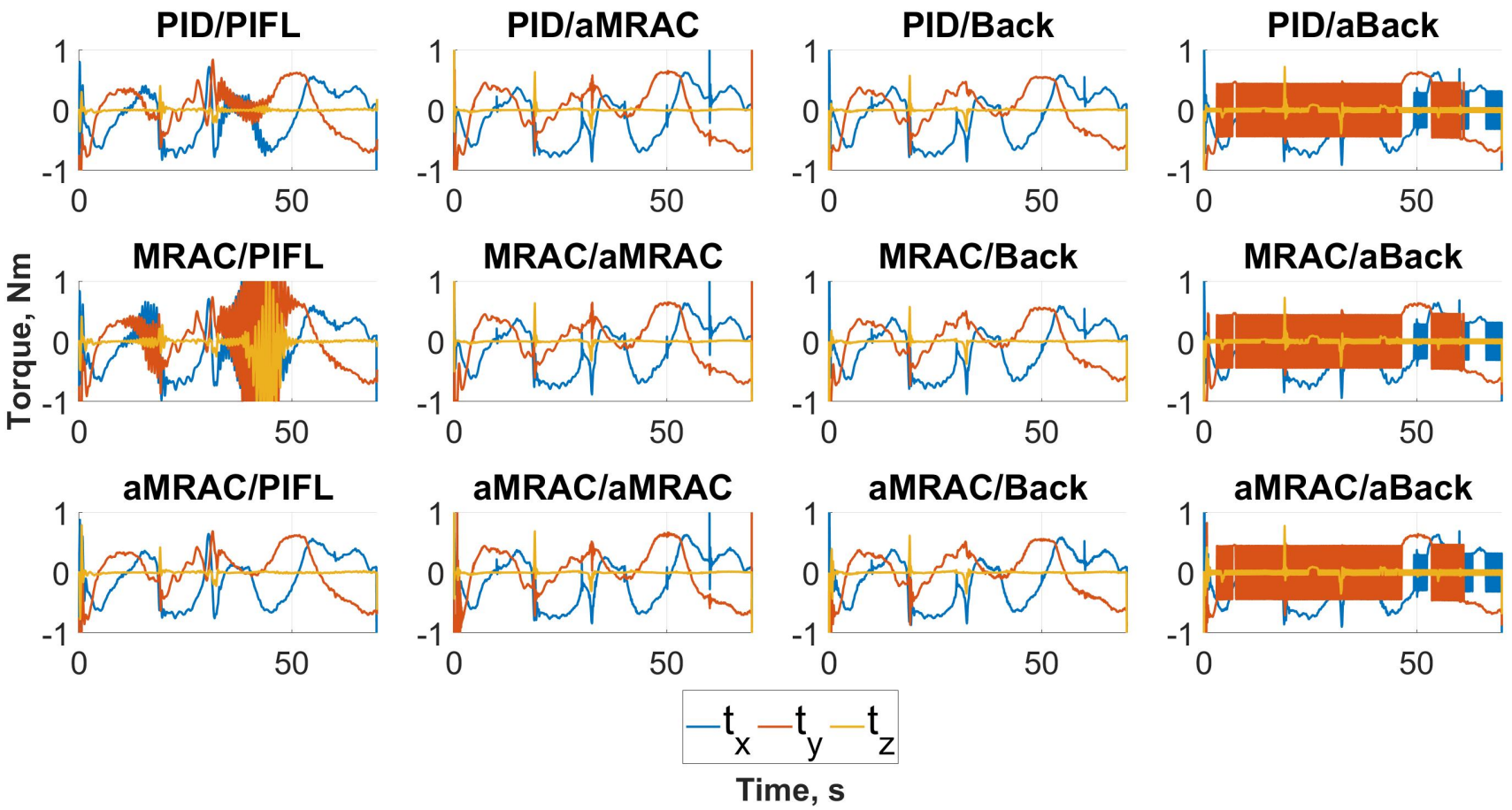


Figure 7.12: Smooth trajectory, increased mass omnicopter torque command comparison. Note that some peaks have been attenuated to preserve visualization of steady state responses.

7.3.3 Motor Commands

The motor outputs for the jagged and smooth trajectories at nominal and increased mass are shown in Figures 7.13-7.16. The input saturation was working because the motor angular velocities are bounded between +/- 1200 rad/s. For all trajectories, motor direction changes (when the lines cross 0 rad/s) typically occur around the time the commanded trajectory calls for a direction change. Typically, multiple motors change direction around the same time. Usually, multiple motors will change direction at the same time, but not all of them. The smooth trajectory motor results show when the commanded trajectory requires a pseudo-permanent motor direction change. For the jagged trajectories, motor direction changes sometimes occur during the transient peaks that are associated with trajectory direction changes. These motor direction changes occur quickly and are temporary, which is concerning because the physical hardware may not be capable of switching and re-switching direction as quickly as the transients command. These jagged trajectory results are particularly applicable for a takeoff specified by a single setpoint instead of a gradual increase to the desired location.

For all scenarios, the oscillations that appeared in the adaptive backstepping torque commands are not present in the motor responses. The lack of oscillations in the motor responses seems to indicate that the oscillations occurred rapidly enough for the motor response time to filter out.

The motor outputs used to execute the jagged trajectory when the omnicopter is at its nominal mass are shown in Figure 7.13. Using MRAC for position control tends to cause more transient oscillations than using PID or aMRAC.

The motor outputs used to execute the smooth trajectory when the omnicopter is at its nominal mass are shown in Figure 7.14. As seen with the force and torque commands, using

a smooth trajectory reduces transient oscillations and overshoots within the motor responses. Otherwise, no significant characteristics distinguish the controller combinations from each other.

The motor outputs used to execute the jagged trajectory when the omnicopter is lifting a 0.25 kg panel are shown in Figure 7.15. The increase in mass causes an increase in motor angular velocity and transient oscillations. Using PID for position control tends to yield fewer transient oscillations. Although the oscillations associated with adaptive backstepping are not present in the motor responses, the ones associated with MRAC/PIFL and PID/PIFL do appear. These oscillations cause rapid motor direction changes that would probably cause the omnicopter to crash. These oscillations correlate with attitude error oscillations as well as position error oscillations to a lesser extent.

The motor outputs used to execute the smooth trajectory when the omnicopter is lifting a 0.25 kg panel are shown in Figure 7.16. The increase in mass, as with the jagged trajectory, causes an increase in motor angular velocity. However, the use of the smooth trajectory reduces transient oscillations. Using PID for position control tends to yield fewer transient oscillations. The most prominent oscillations appear in the MRAC/PIFL and PID/PIFL motor responses, with MRAC/PIFL having the worse results. The MRAC/PIFL oscillations cause rapid motor direction changes that would probably cause the omnicopter to crash. These oscillations correlate with attitude error oscillations as well as position error oscillations to a lesser extent.

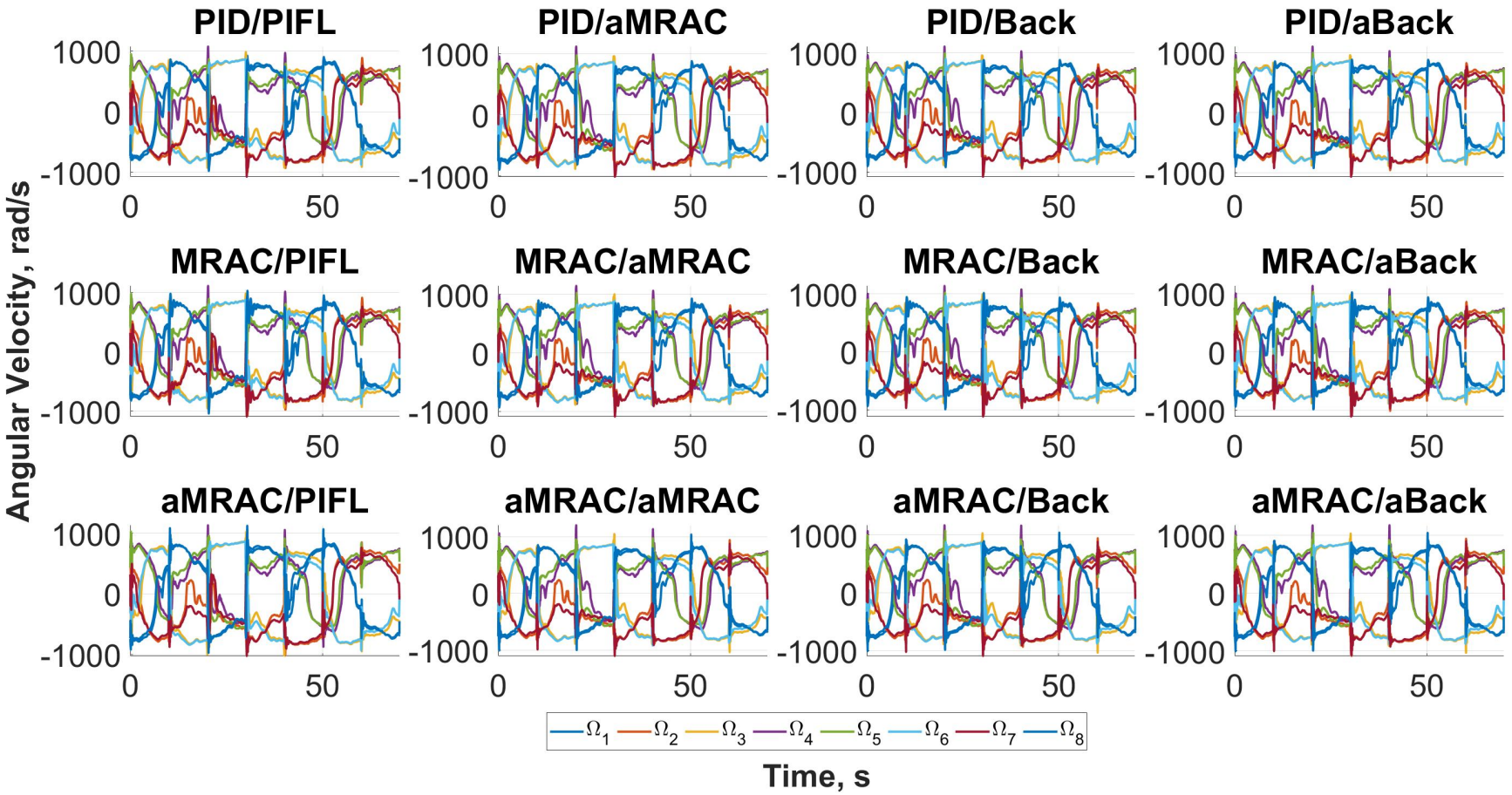


Figure 7.13: Jagged trajectory, nominal mass omnicopter torque command comparison

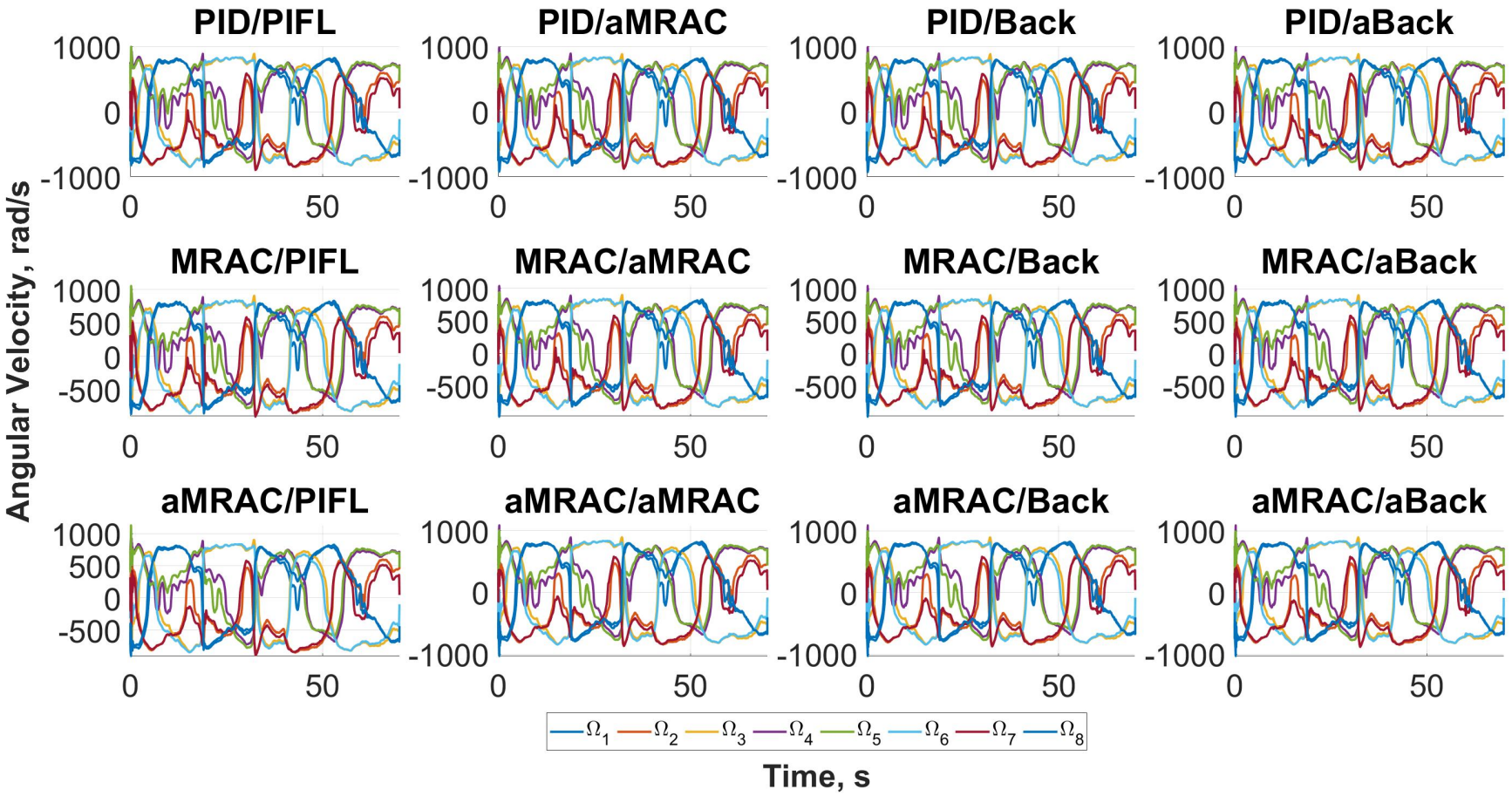


Figure 7.14: Smooth trajectory, nominal mass omnicopter torque command comparison

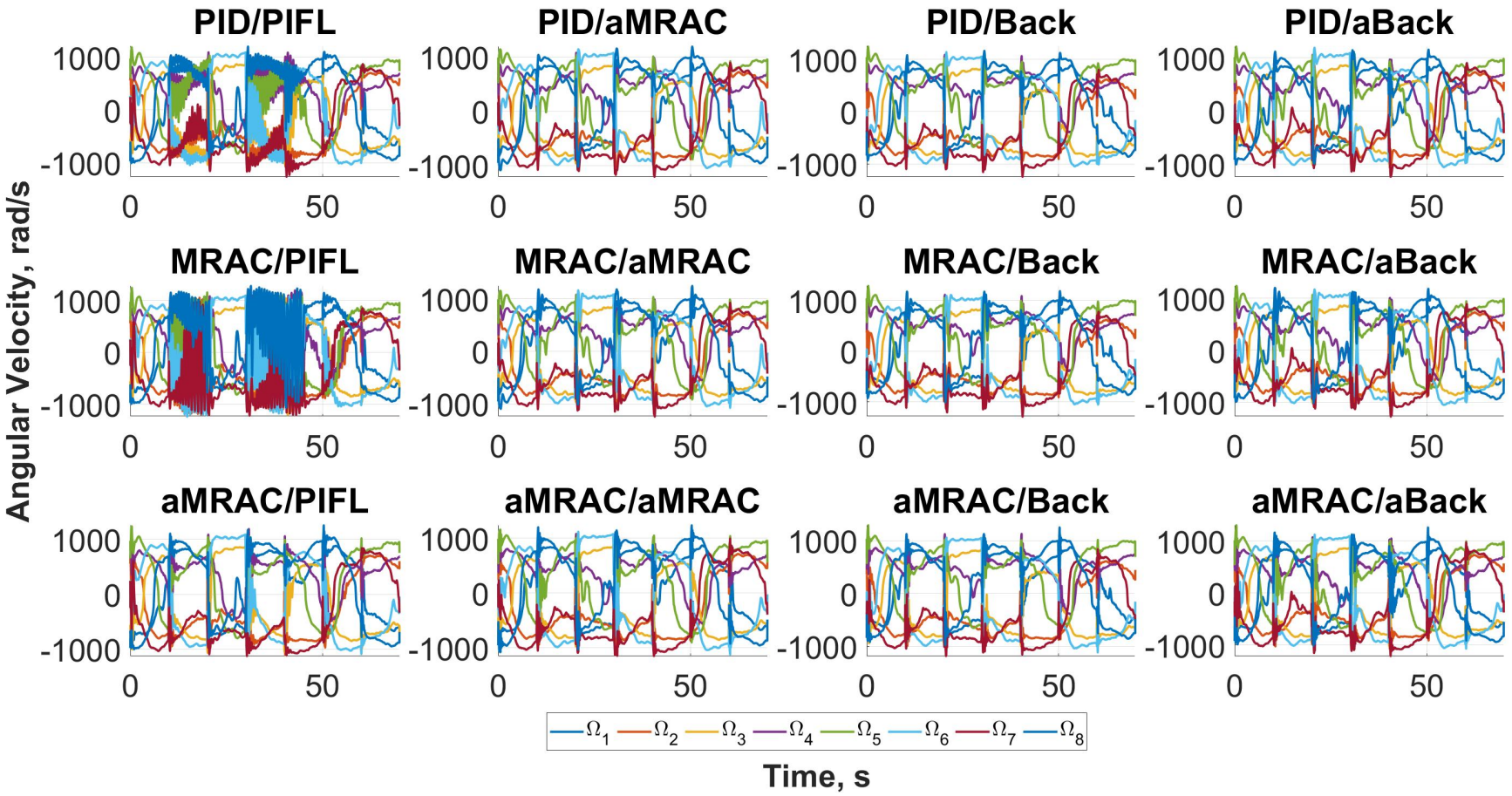


Figure 7.15: Jagged trajectory, increased mass omnicopter torque command comparison

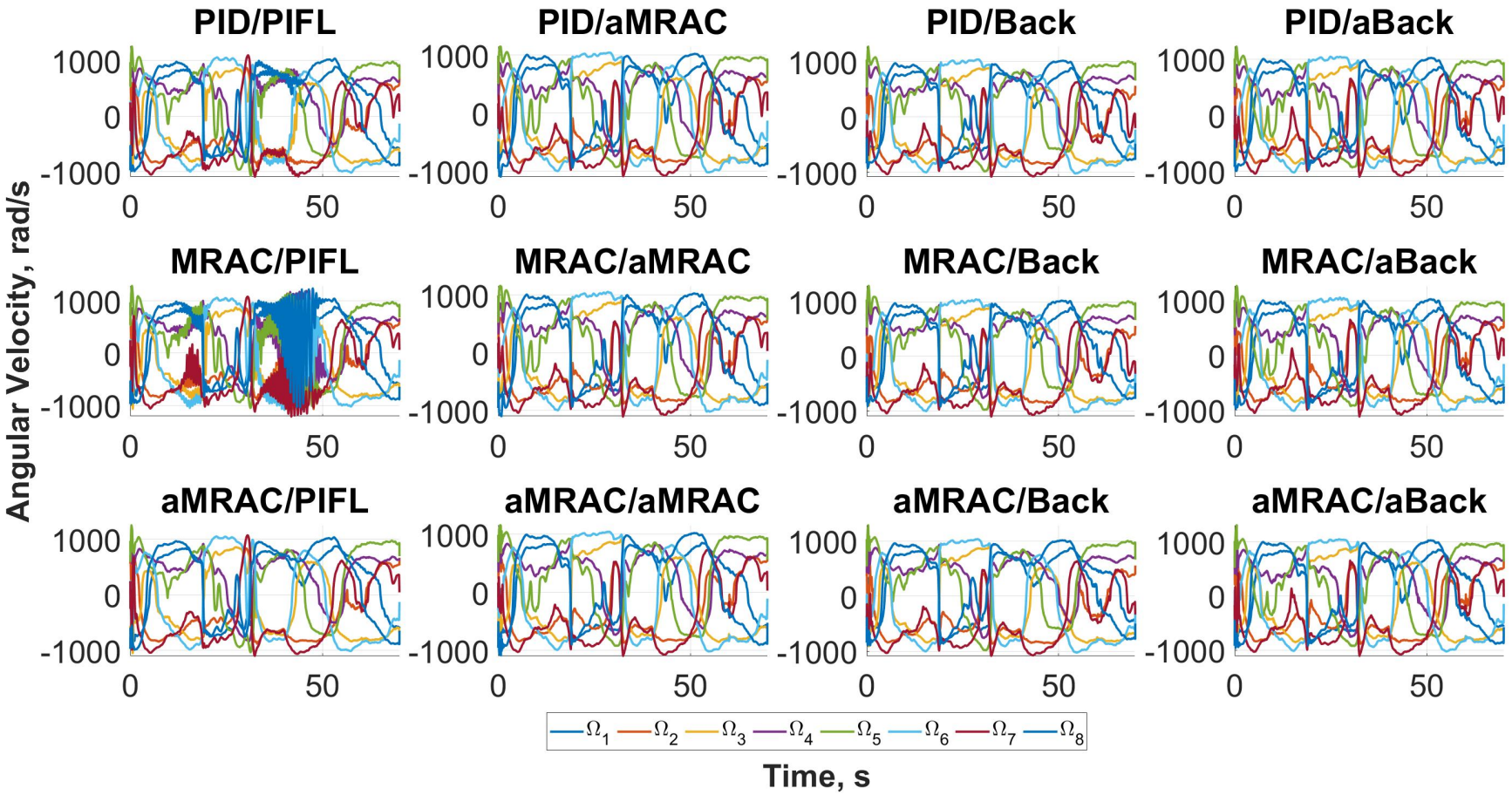


Figure 7.16: Smooth trajectory, increased mass omnicopter torque command comparison

7.3.4 Command Implications

While error trajectory tracking error comparisons are important to consider when selecting a controller for the omnicopter, it is also important to analyze force, torque, and motor commands to make sure good tracking does not correlate with undesirable controller phenomena, such as prolonged saturation or frequent oscillations. It is also important to check that the commanded forces and torques have reasonable magnitudes and expected behavior because a controller that exhibits steady state behavior very different from other controllers executing the same trajectory may be a poor choice to use.

Indicating that there is the expected correlation between commands and omnicopter performance, several traits that appeared in the trajectory error analyses of Section 7.2 were seen to appear in the force, torque, and motor commands. For example, the oscillations in MRAC/PIFL and PID/PIFL for the increased mass scenario appear in the trajectory error as well as the force, torque, and motor commands. These controller combinations are therefore undesirable choices for omnicopter control because of the poor attitude and position tracking as well as the severe motor oscillations. Additionally, for trajectory direction changes, particularly for the jagged trajectory, the spikes in error correlate with the command spikes as the controllers try to adjust to the new setpoint.

Some aspects of the commands can also confirm that the controllers are working properly. A combination of low tracking error and low motor saturation, which is desirable for motor and flight longevity, indicates a controller that could be desirable to put on the omnicopter. For most of the controllers, the motors do not appear to saturate for long periods of time. The lack of motor saturation means the control allocation portion of the code is working properly, and/or none of the force or torque commands from any of the controllers combine in a way that regularly saturate the motors. In fact, overall the steady state portion of

the force, torque, and motor commands appear similar in shape and magnitude for every scenario (PID/PIFL and MRAC/PIFL being the occasional exception). Therefore, motor saturation and steady state patterns do not substantially distinguish any controllers.

Sometimes undesirable characteristics can appear in force and torque commands that do not appear or are not obvious in the tracking error plots. Commands that cause the motors to frequently switch direction, as is common with the jagged trajectory (Figures 7.13 and 7.15) , are an example of an undesirable command response that is not obvious in the error plots. Because tracking error is expected to initially increase with a commanded direction change, the spikes in the error plots do not explicitly indicate that motors are switching direction. While the rapid/frequent motor switching tends to be a characteristic of the jagged trajectory instead of correlated with a specific controller, aMRAC tends to have the highest peaks and MRAC is the most prone to prolonged transient oscillations. While these step responses may be feasible in simulation, this transient behavior may be an indication that these controllers are too aggressively tuned to be used on the physical omnicopter. To avoid these characteristics, smooth trajectories should be used whenever possible. Another example of undesirable characteristics in commands that do not appear in dynamics are the rapid oscillations from the adaptive backstepping controller. Since the tracking error for this controller makes it a desirable choice for omnicopter control, it is worth the effort to determine the source of the oscillations and try to eliminate them without negatively impacting the good tracking performance. Being an adaptive controller, it is beneficial to analyze the adaptive parameter norms to see if the adaptation is the source of the oscillations. Any time an adaptive controller is exhibiting undesirable behavior, it is worth analyzing the adaptive parameter norms to ensure nothing in the adaptation itself is leading to instability (e.g. MRAC).

7.3.5 Position Adaptive Parameter Performance

It should be noted that parameter adaptation began upon takeoff, which occurred 40 seconds before these results are shown. Therefore, the initial conditions shown in Figures 7.17 - 7.28 at $t = 0sec$ for these adaptive parameters may not be uniform and do not match the initial conditions presented in Tables 5.2 and 5.3, which are for $t = -40sec$.

The column-wise norms of the adaptive parameters for the adaptive position controllers (MRAC and aMRAC) used to execute the jagged trajectory when the omnicopter is at its nominal mass are shown in Figures 7.17 - 7.19. The norm of the third column tends to change the most and have the highest values compared to the first and second columns, probably because the adaptive controllers do not explicitly account for gravity when computing the commanded force. Jumps in the parameters tend to occur when the trajectory commands direction changes. For aMRAC, $\hat{\Theta}$ tends to gradually increase or level off between jumps. For MRAC, $\hat{K}_x$ and $\hat{K}_r$ tend to alternate between increasing and decreasing with each direction change, acting as inversions of each other. Additionally, these estimates tend to be prone to transient oscillations.

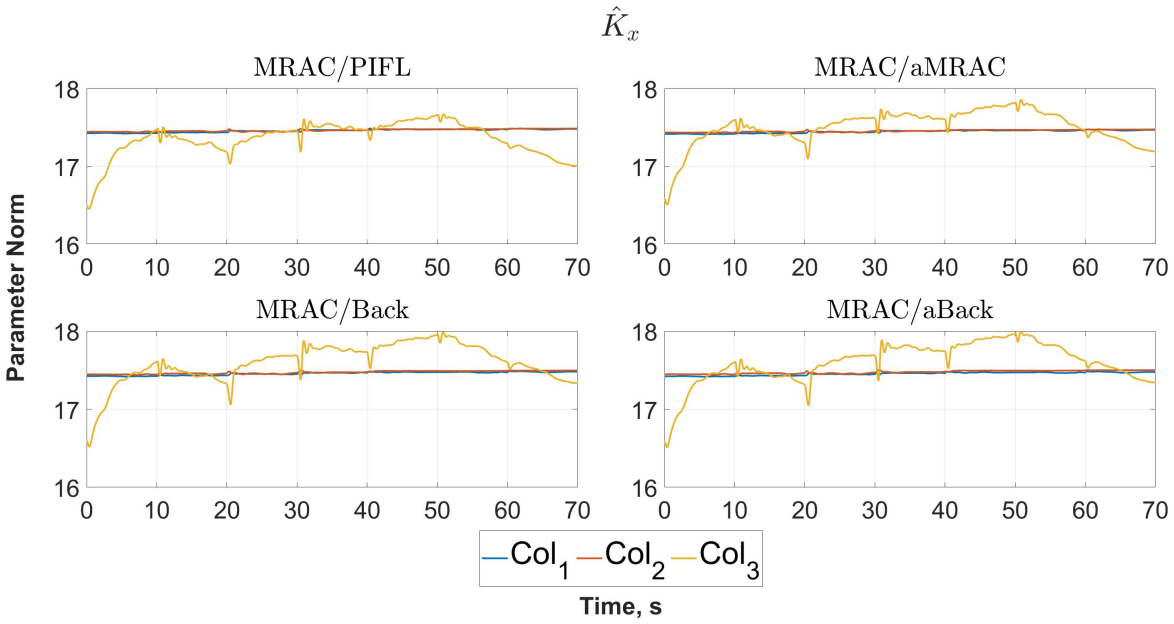


Figure 7.17: Jagged trajectory, nominal mass omnicopter $\hat{\boldsymbol{K}}_x$ for MRAC position control comparison

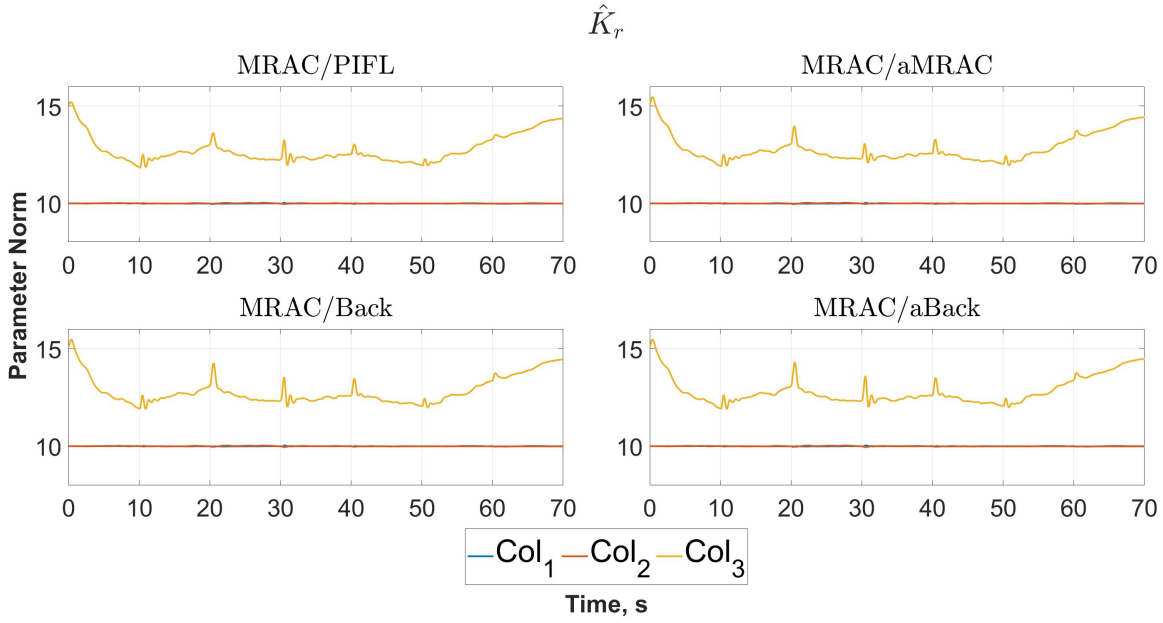


Figure 7.18: Jagged trajectory, nominal mass omnicopter $\hat{\boldsymbol{K}}_r$ for MRAC position control comparison

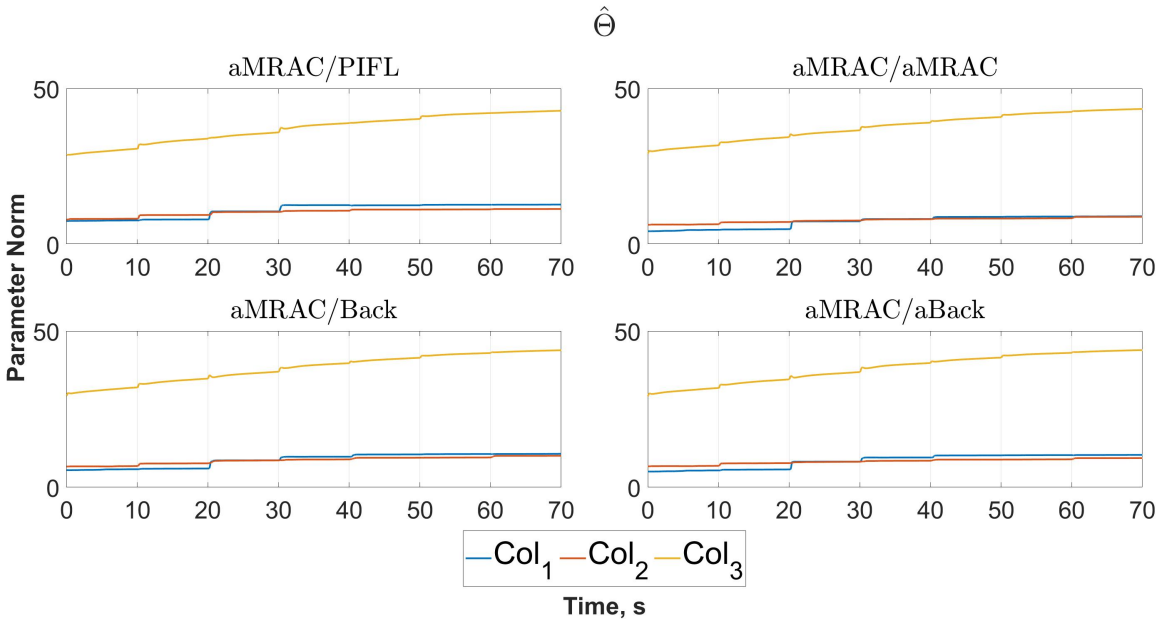


Figure 7.19: Jagged trajectory, nominal mass omnicopter $\hat{\mathbf{\Theta}}$ for aMRAC position control comparison

The column-wise norms of the adaptive parameters for the adaptive position controllers used to execute the smooth trajectory when the omnicopter is at its nominal mass are shown in Figures 7.20 - 7.22. The adaptive parameters exhibit similar trends to those associated with the jagged trajectory. However, the signals are smoother, lacking sudden jumps and transient oscillations. For aMRAC, there tends to be greater separation between the first and second column norms than there was for the jagged trajectory.

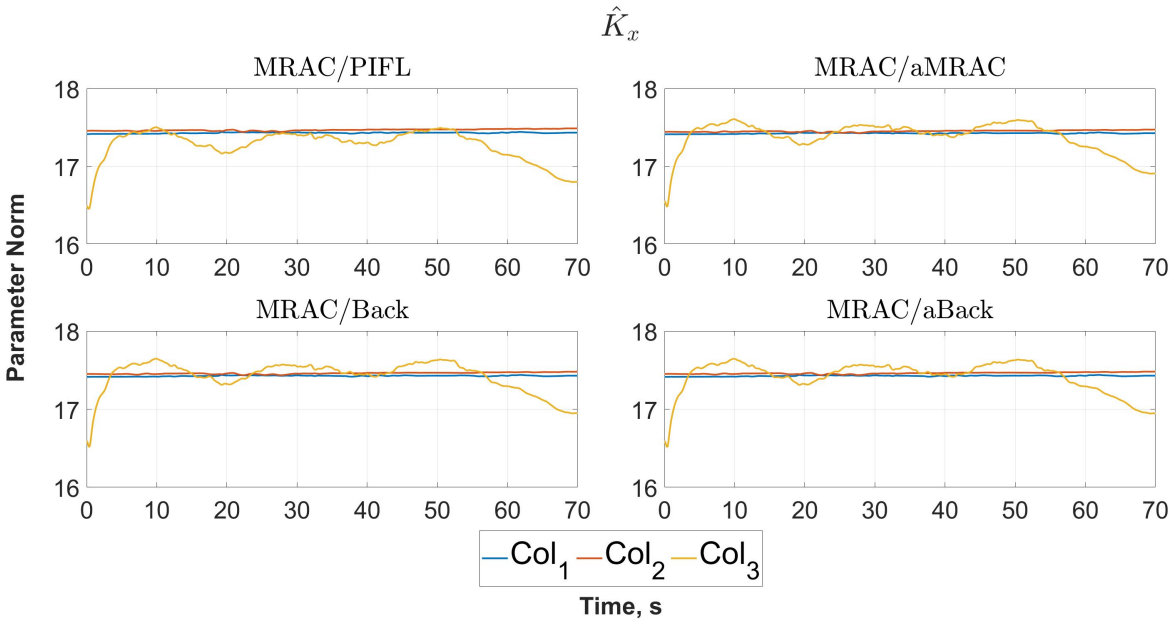


Figure 7.20: Smooth trajectory, nominal mass omnicopter $\hat{\boldsymbol{K}}_x$ for MRAC position control comparison

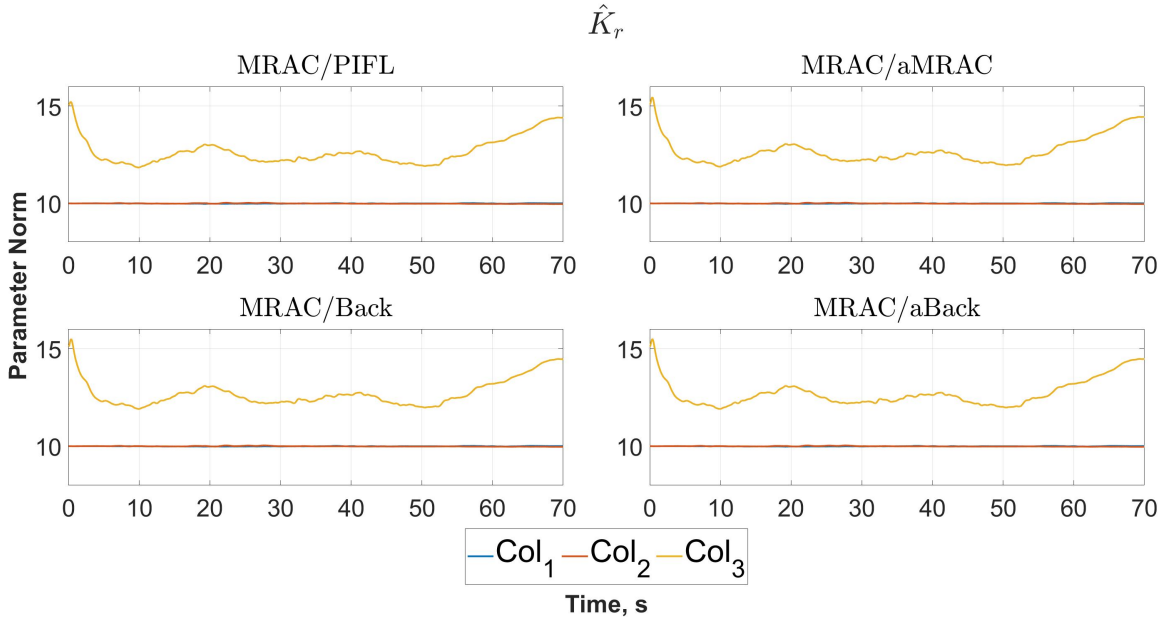


Figure 7.21: Smooth trajectory, nominal mass omnicopter $\hat{\boldsymbol{K}}_r$ for MRAC position control comparison

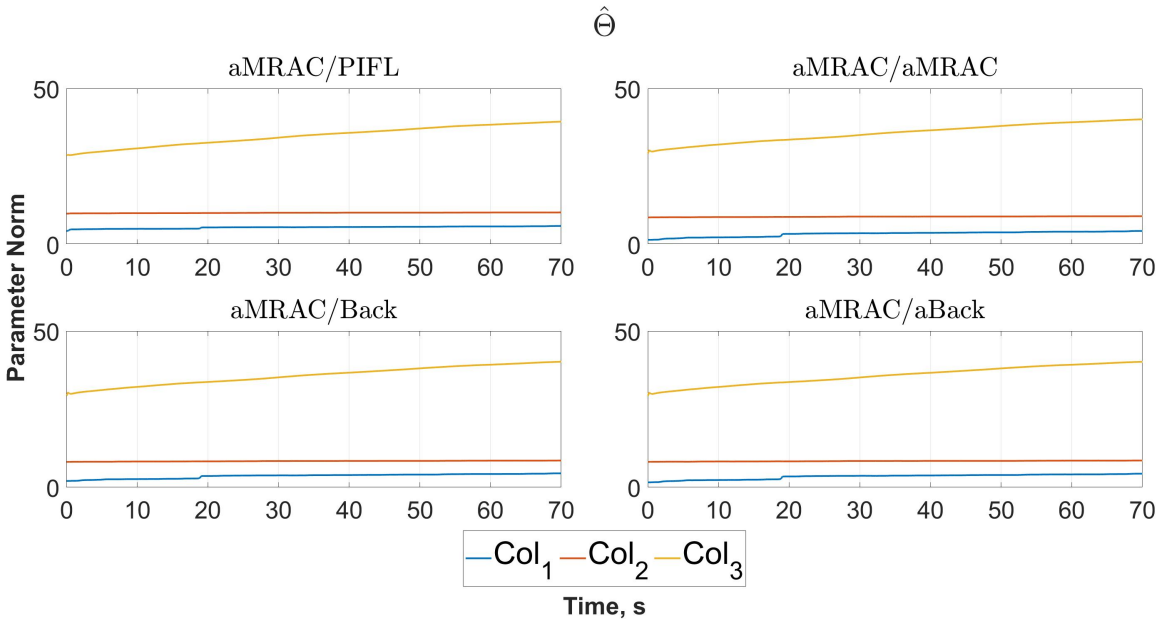


Figure 7.22: Smooth trajectory, nominal mass omnicopter $\hat{\boldsymbol{\Theta}}$ for aMRAC position control comparison

The column-wise norms of the adaptive parameters for the adaptive position controllers used to execute the jagged trajectory when the omnicopter is lifting a 0.25 kg panel are shown in Figures 7.23 - 7.25. The increase in omnicopter mass causes an increase in the magnitude of the third column norm. The increase in the third column norm is expected because increased mass requires increased force in the z direction to maintain flight. The increase in mass causes the transient peaks to jump higher and take longer to settle for $\hat{\boldsymbol{K}}_x$ and $\hat{\boldsymbol{K}}_r$. However, the transient oscillations are also reduced. The shape of values associated with MRAC/PIFL changes the most compared to the other controller combinations. The change in shape may be due to the oscillations seen in the force commands and motor responses. However, since the adaptive parameters do not exhibit oscillations, it would seem the oscillations in the force commands would enter from the states.

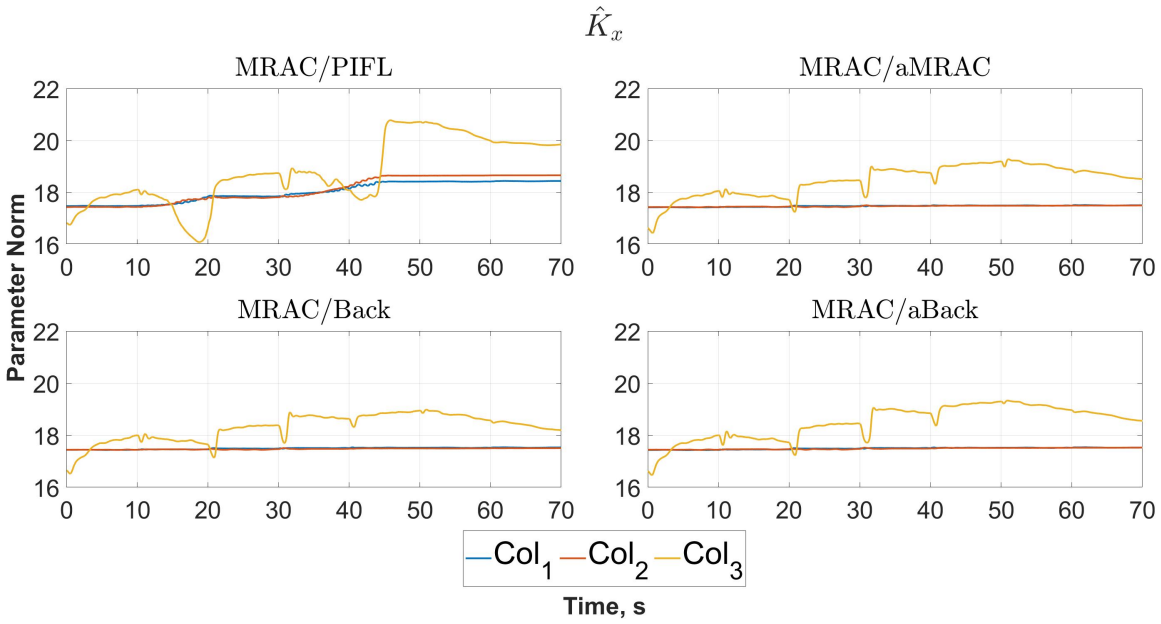


Figure 7.23: Jagged trajectory, increased mass omnicopter $\hat{\boldsymbol{K}}_x$ for MRAC position control comparison

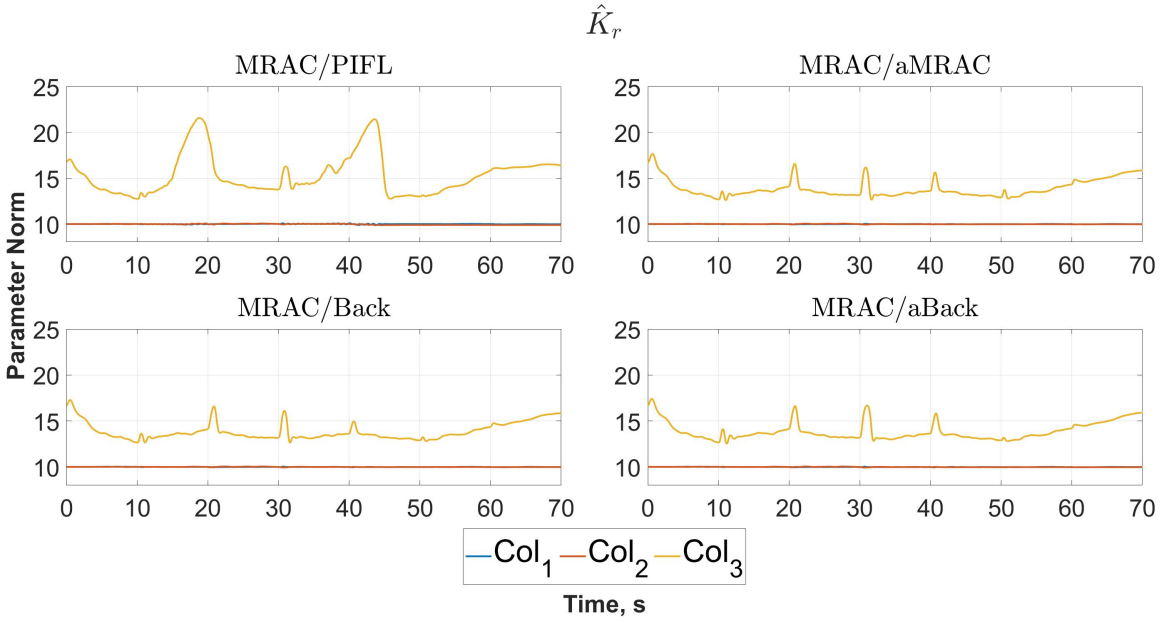


Figure 7.24: Jagged trajectory, increased mass omnicopter $\hat{\boldsymbol{K}}_r$ for MRAC position control comparison

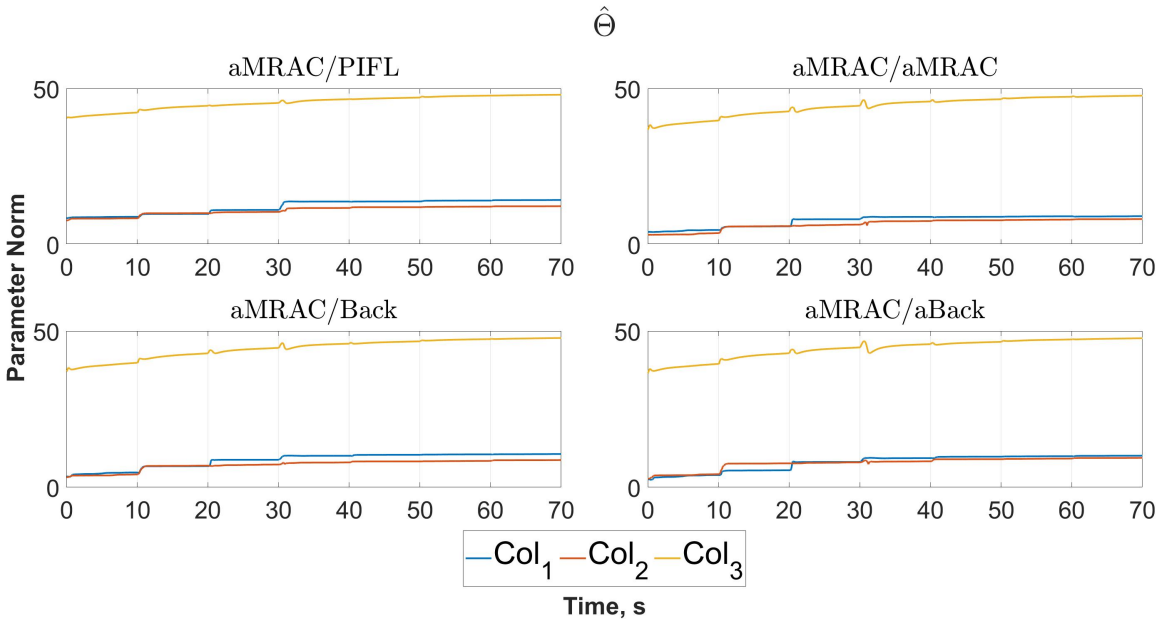


Figure 7.25: Jagged trajectory, increased mass omnicopter $\hat{\mathbf{\Theta}}$ for aMRAC position control comparison

The column-wise norms of the adaptive parameters for the adaptive position controllers used to execute the smooth trajectory when the omnicopter is lifting a 0.25 kg panel are shown in Figures 7.26 - 7.28. As with the jagged trajectory, the column norms for the adaptive parameters tend to maintain shapes similar to when the omnicopter was at its nominal mass, except with higher magnitudes. Once again, the exceptions are the parameters associated with MRAC, where the greatest changes in shape are associated with the greatest oscillations in the force commands. However, the lack of oscillation indicates that, as with jagged trajectory, the oscillations entered via the states.

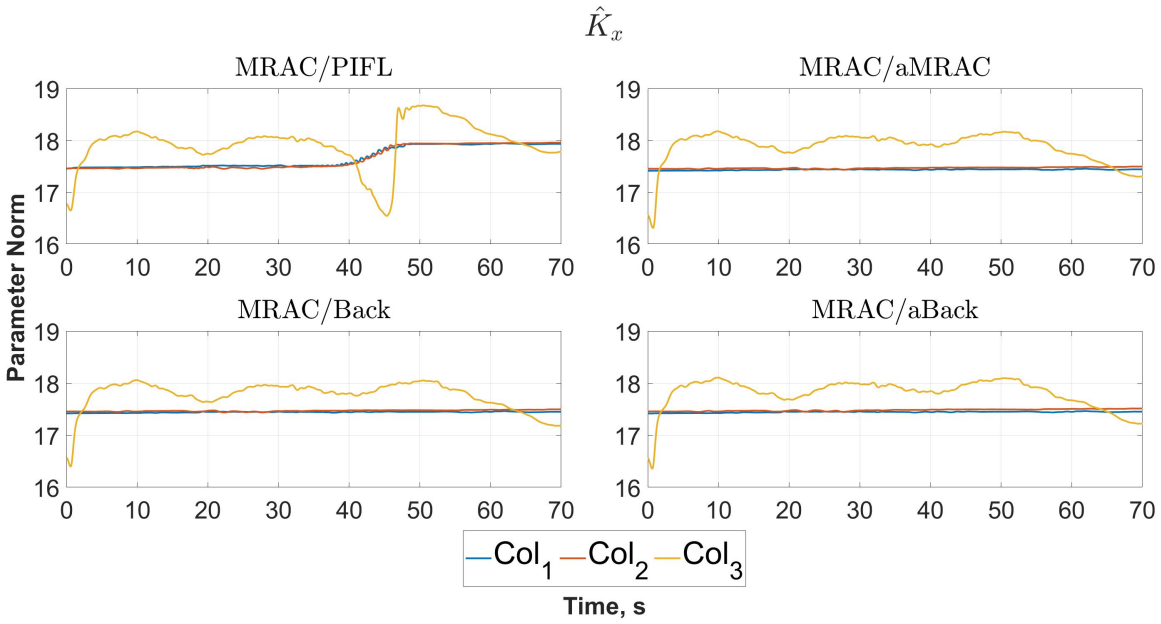


Figure 7.26: Smooth trajectory, increased mass omnicopter $\hat{\boldsymbol{K}}_x$ for MRAC position control comparison

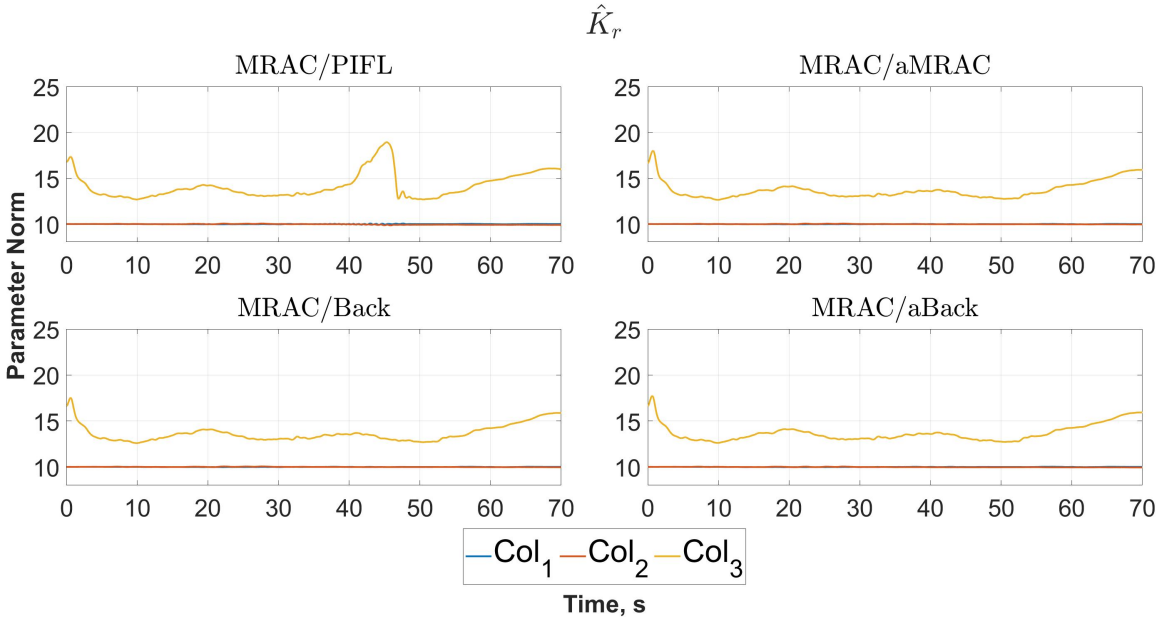


Figure 7.27: Smooth trajectory, increased mass omnicopter $\hat{\boldsymbol{K}}_r$ for MRAC position control comparison

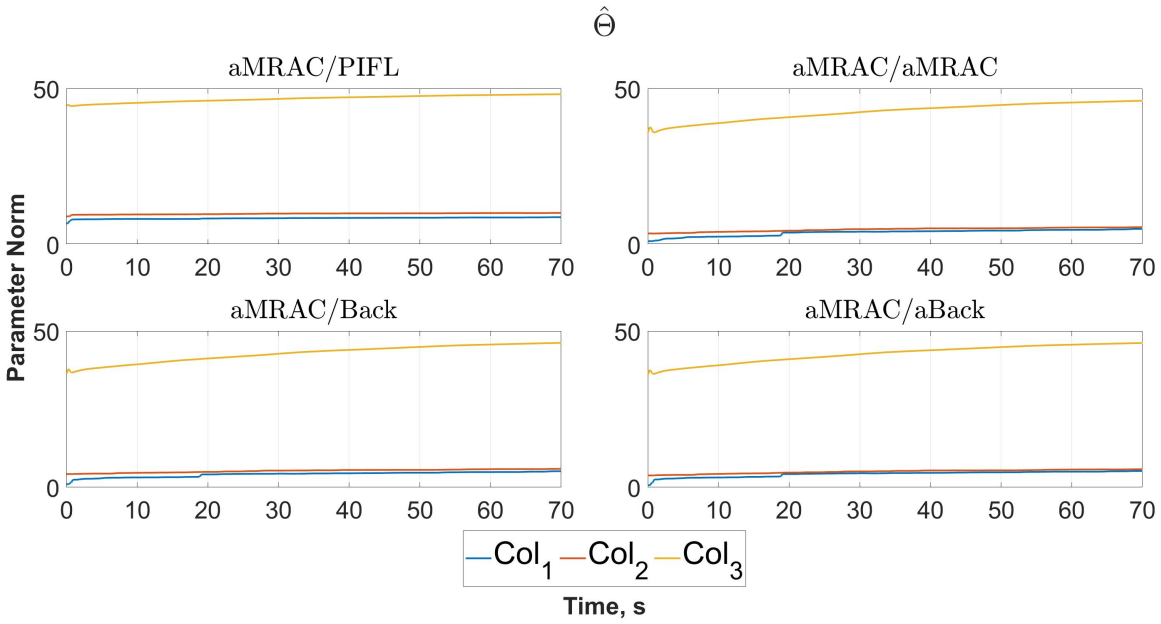


Figure 7.28: Smooth trajectory, increased mass omnicopter $\hat{\mathbf{\Theta}}$ for aMRAC position control comparison

7.3.6 Attitude Adaptive Parameter Performance

As with the position control adaptive parameters, it should be noted that attitude parameter adaptation began upon takeoff ($t = -40sec$) instead of at the start of the trajectory ($t = 0sec$ in Figures 7.29 - 7.48). The initial conditions for $t = -40sec$ are shown in Tables 6.2 and **??**.

The column-wise norms of the adaptive parameters for the adaptive attitude controllers (aMRAC and aBack) executing the trajectories at nominal and increased mass are seen in Figures 7.29 - 7.48. Overall, these parameters tend to be more dynamic than the position adaptive parameters, which were dominated by the column three norms. Although adaptive backstepping commanded oscillatory torques, these oscillations are not seen in any of the adaptive parameters for adaptive backstepping. The position control adaptive parameters

could be prone to transient oscillations. However, none of the attitude control adaptive parameters exhibit this trait.

Another interesting characteristic is that for $\hat{\Theta}$ for aMRAC attitude control, the relative values of the column norms change noticeably depending on the selected position controller. The variation occurs for both trajectories and both masses and is not seen in other adaptive parameters for position or attitude control. None of these controllers exhibited oscillations in the torque commands, so that is not the cause.

The column-wise norms of the adaptive parameters for the adaptive attitude controllers used to execute the jagged trajectory when the omnicopter is at its nominal mass are shown in Figure 7.29 - 7.33. Sudden changes in parameter values coincide with trajectory direction changes. Except for the disturbance torque estimation for aMRAC, most parameter estimates remain constant after trajectory changes. Most column norms are around the same value, except for $\hat{\Psi}$ for adaptive backstepping. For $\hat{\Psi}$, the first column tends to grow large than the other two values. The $\hat{\Theta}$ parameter for adaptive backstepping hits its projection boundaries, showing that the projection operator was implemented correctly.

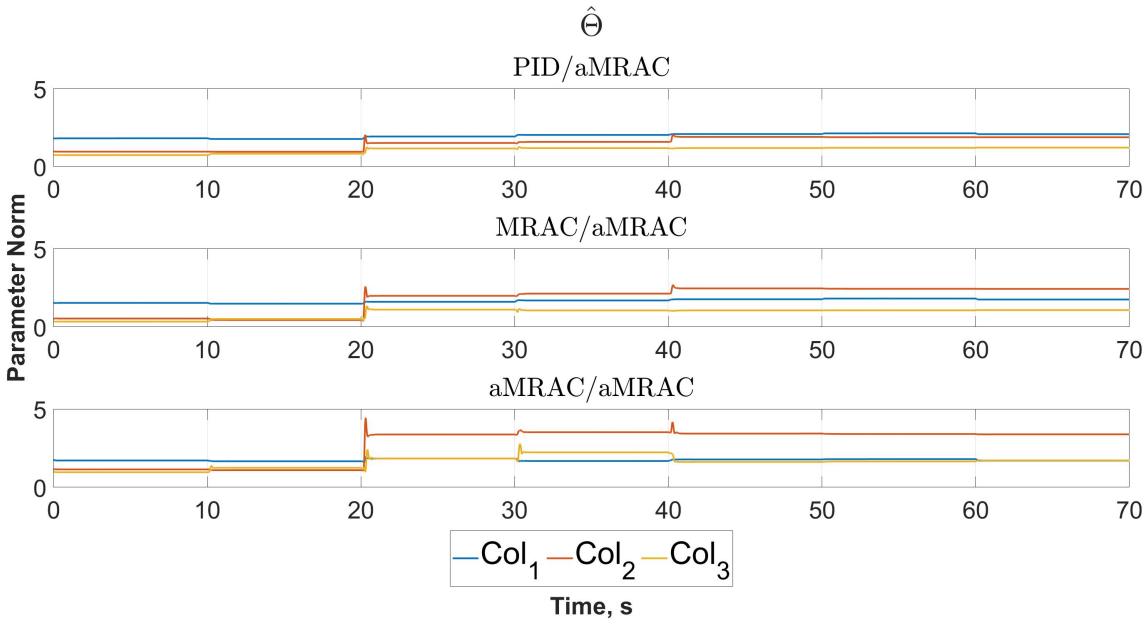


Figure 7.29: Jagged trajectory, nominal mass omnicopter $\hat{\boldsymbol{\Theta}}$ for aMRAC attitude control comparison

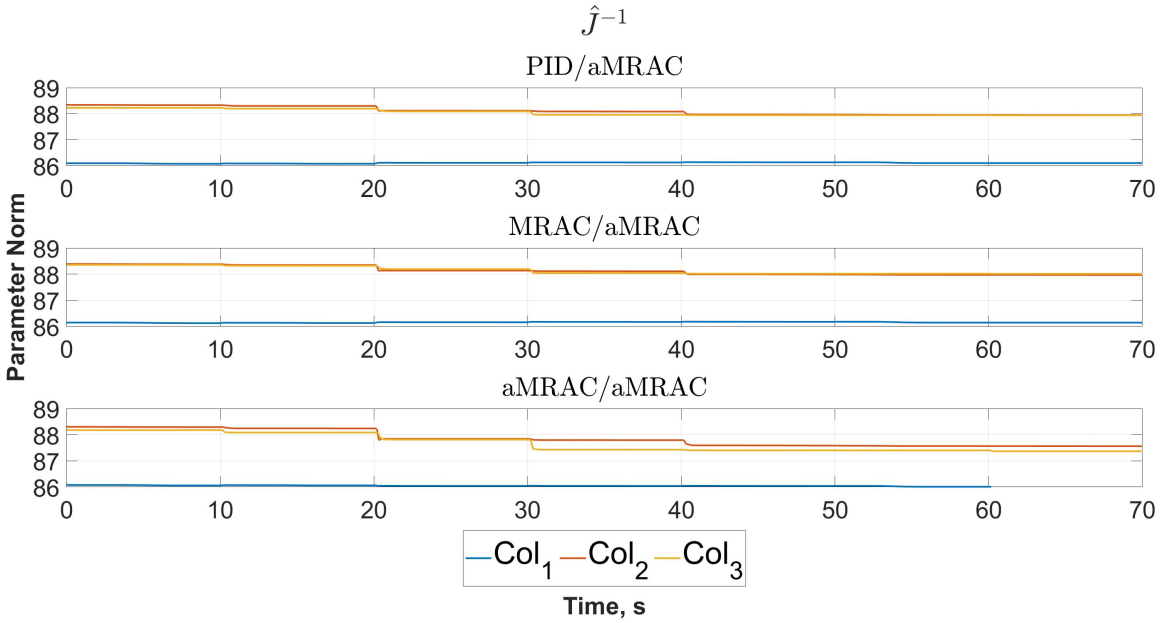


Figure 7.30: Jagged trajectory, nominal mass omnicopter$\hat{\boldsymbol{J}}^{-1}$ for aMRAC attitude control comparison

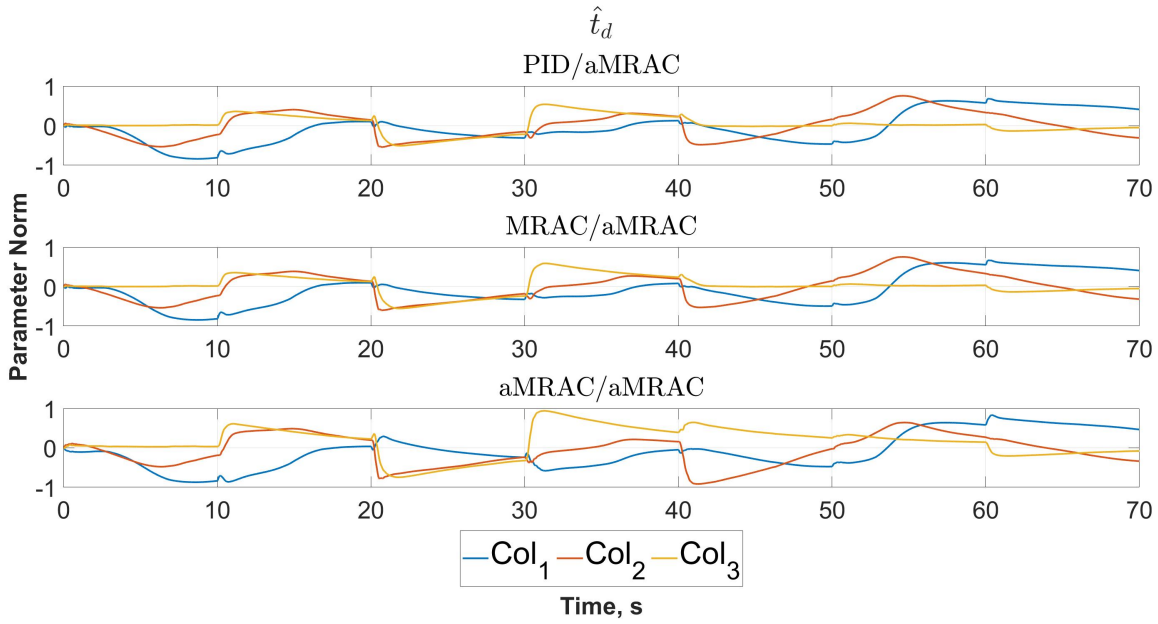


Figure 7.31: Jagged trajectory, nominal mass omnicopter $\hat{\boldsymbol{t}}_d$ for aMRAC attitude control comparison

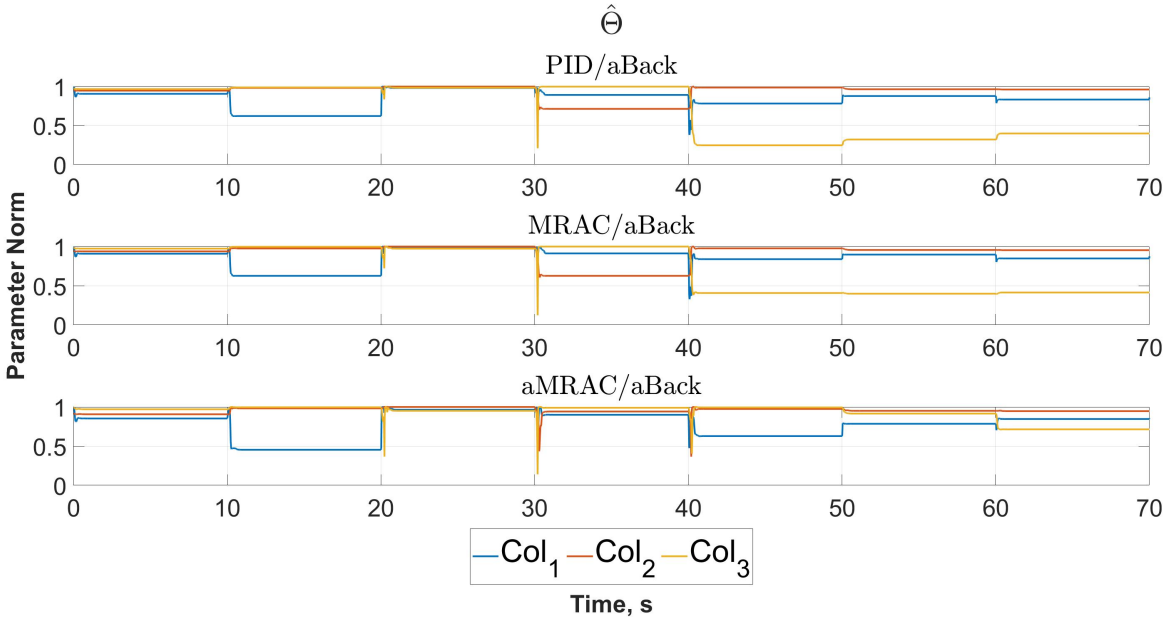


Figure 7.32: Jagged trajectory, nominal mass omnicopter$\hat{\boldsymbol{\Theta}}$ for aBack attitude control comparison

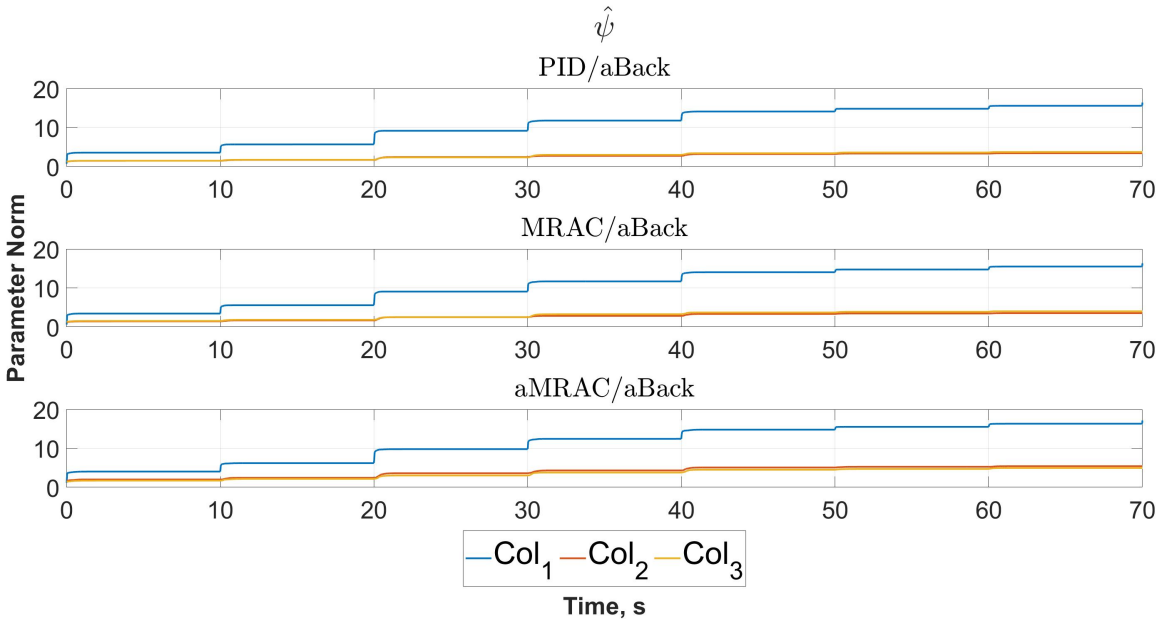


Figure 7.33: Jagged trajectory, nominal mass omnicopter$\hat{\mathbf{\Psi}}$ for aBack attitude control comparison

The column-wise norms of the adaptive parameters for the adaptive attitude controllers used to execute the smooth trajectory when the omnicopter is at its nominal mass are shown in Figure 7.34 - 7.38. As usual, the smooth trajectory yields fewer discontinuities in signals. Particularly for the adaptive backstepping parameters, the signals tend to remain more constant than for the jagged trajectory.

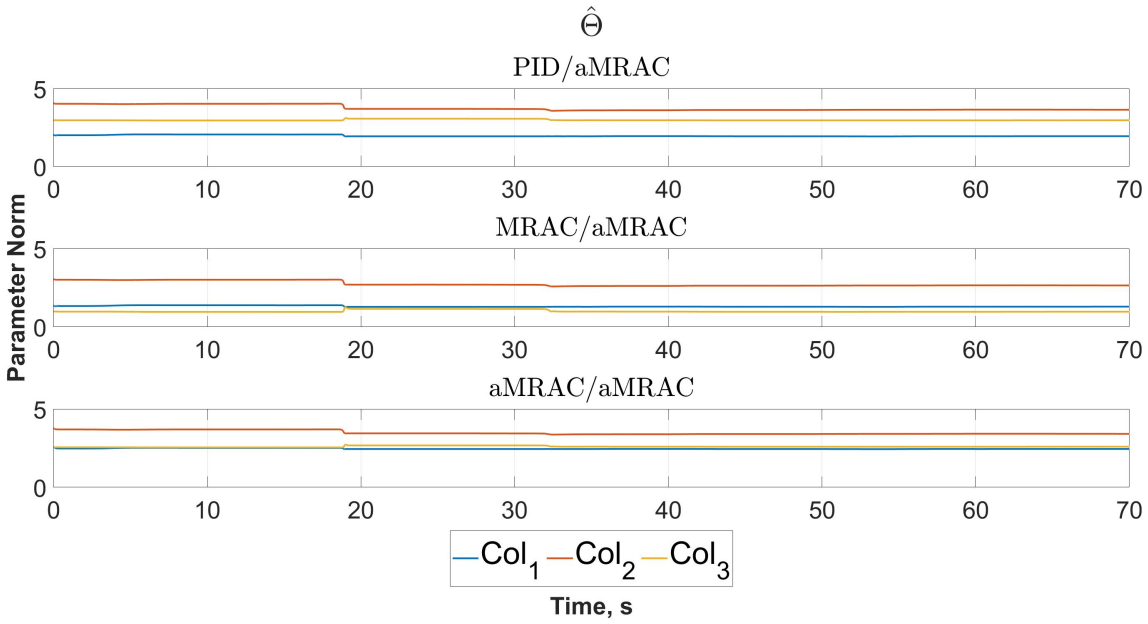


Figure 7.34: Smooth trajectory, nominal mass omnicopter $\hat{\mathbf{\Theta}}$ for aMRAC attitude control comparison

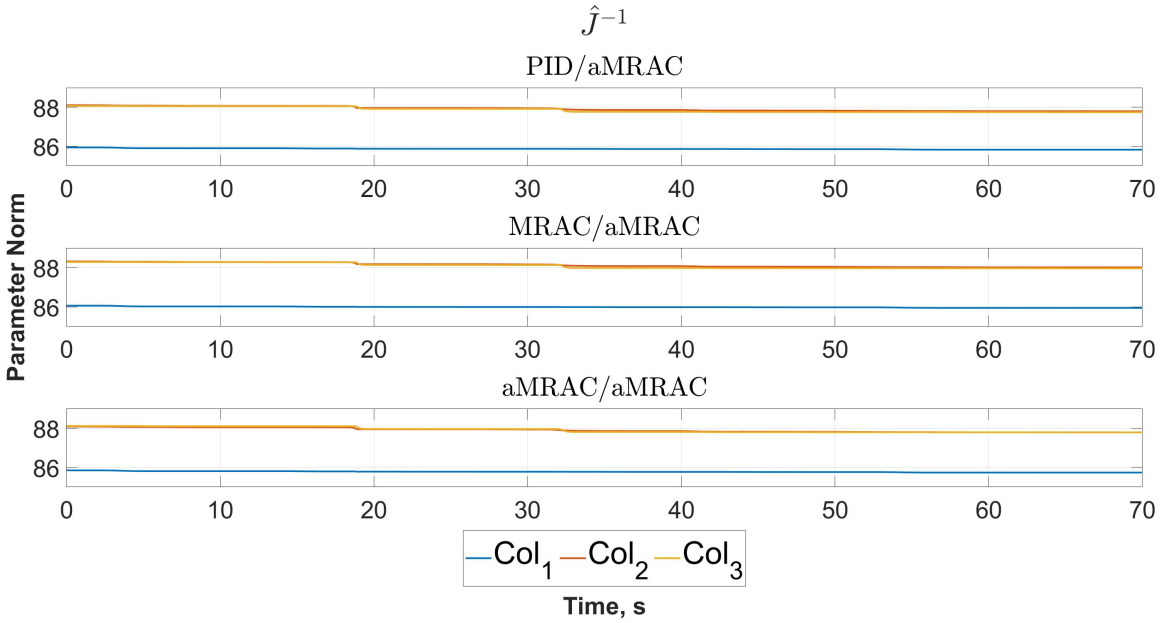


Figure 7.35: Smooth trajectory, nominal mass omnicopter $\hat{\boldsymbol{J}}^{-1}$ for aMRAC attitude control comparison

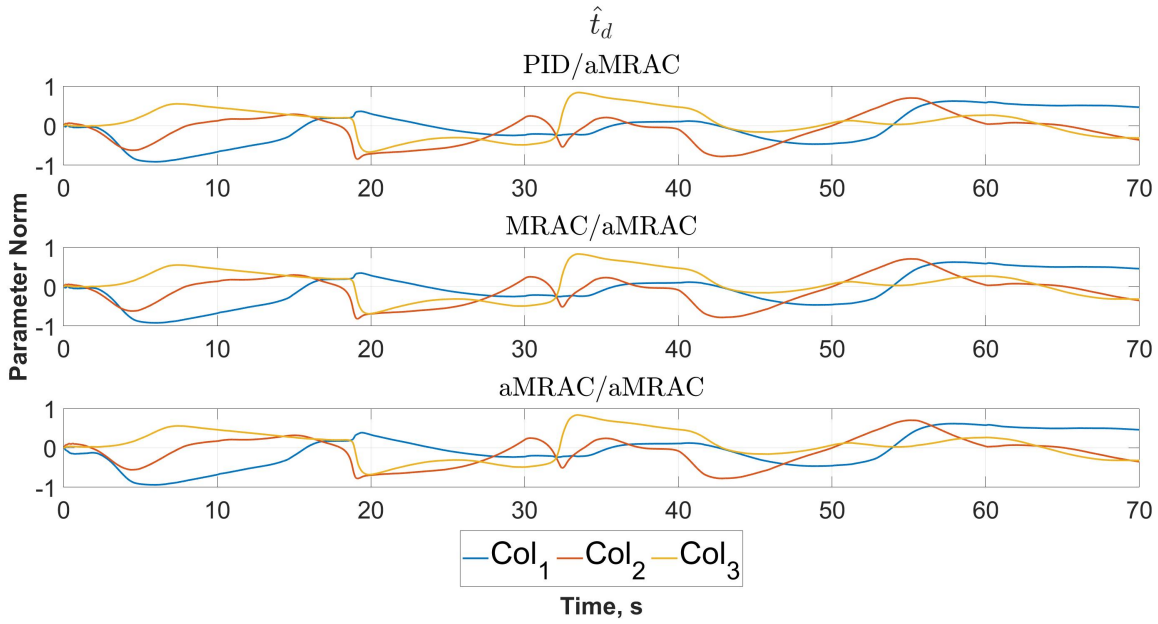


Figure 7.36: Smooth trajectory, nominal mass omnicopter $\hat{\boldsymbol{t}}_d$ for aMRAC attitude control comparison

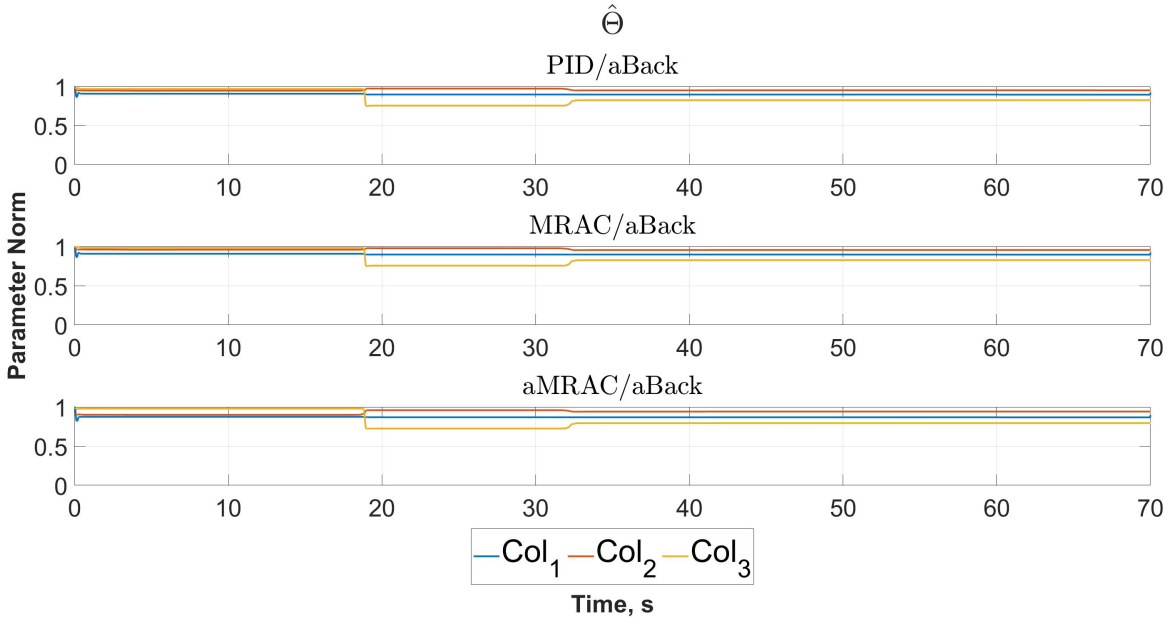


Figure 7.37: Smooth trajectory, nominal mass omnicopter $\hat{\boldsymbol{\Theta}}$ for aBack attitude control comparison

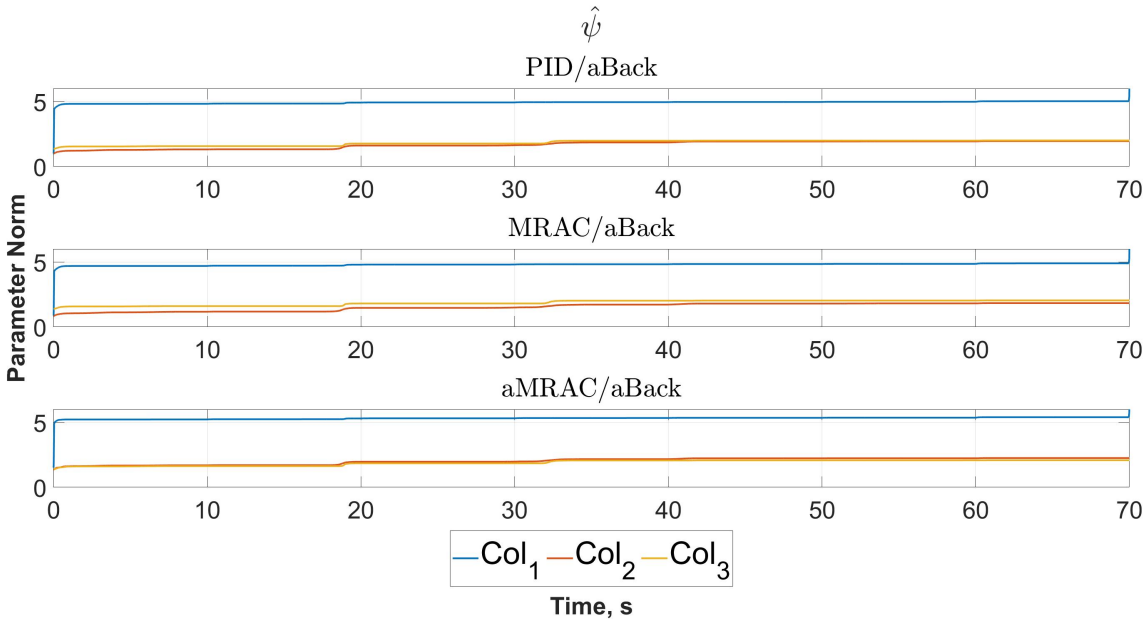


Figure 7.38: Smooth trajectory, nominal mass omnicopter $\hat{\mathbf{\Psi}}$ for aBack attitude control comparison

The column-wise norms of the adaptive parameters for the adaptive attitude controllers used to execute the jagged trajectory when the omnicopter is carrying a 0.25 kg panel are shown in Figures 7.39 - 7.43. The increase in mass causes the parameters to have larger magnitudes. For all parameters except the torque disturbance, the norm of the second column shows significant increases or decreases compared to the nominal mass scenario, particularly visible in $\hat{\Lambda}$ for aMRAC and $\hat{\psi}$ for adaptive backstepping. Additionally, for those same parameters, the norm of the first column tends to change the most from its initial value, especially compared to the nominal mass scenario. For the disturbance torque estimate in aMRAC, the first and second columns consistently hit the projection boundaries, proving that the projection operator was implemented correctly. For adaptive backstepping, once again the $\hat{\Theta}$ values hit their projection boundaries.

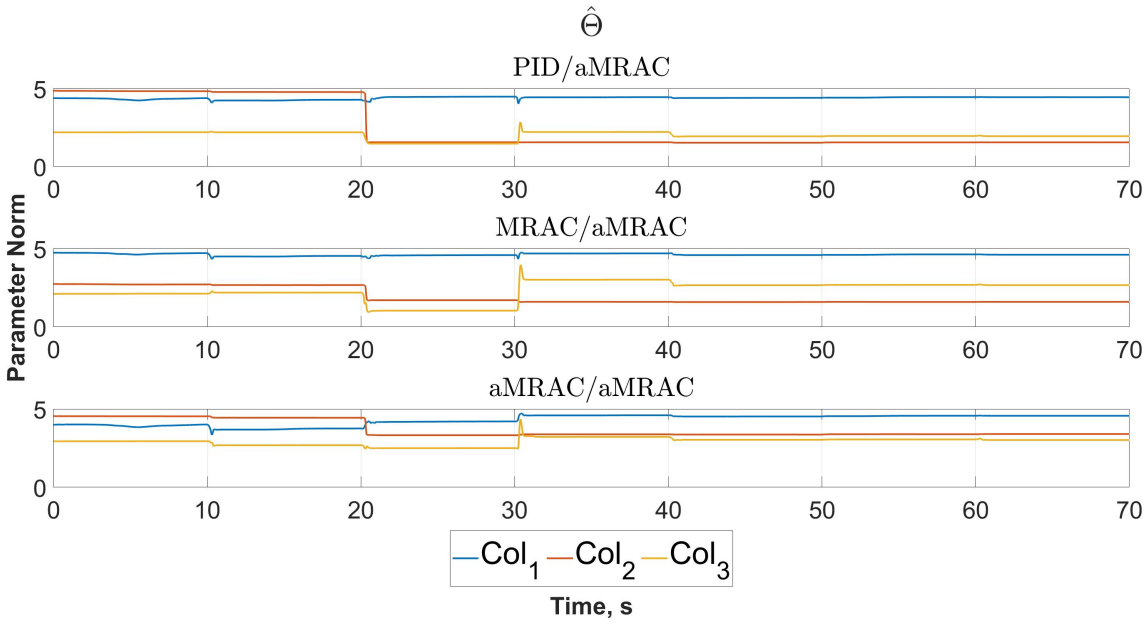


Figure 7.39: Jagged trajectory, increased mass omnicopter $\hat{\mathbf{\Theta}}$ for aMRAC attitude control comparison

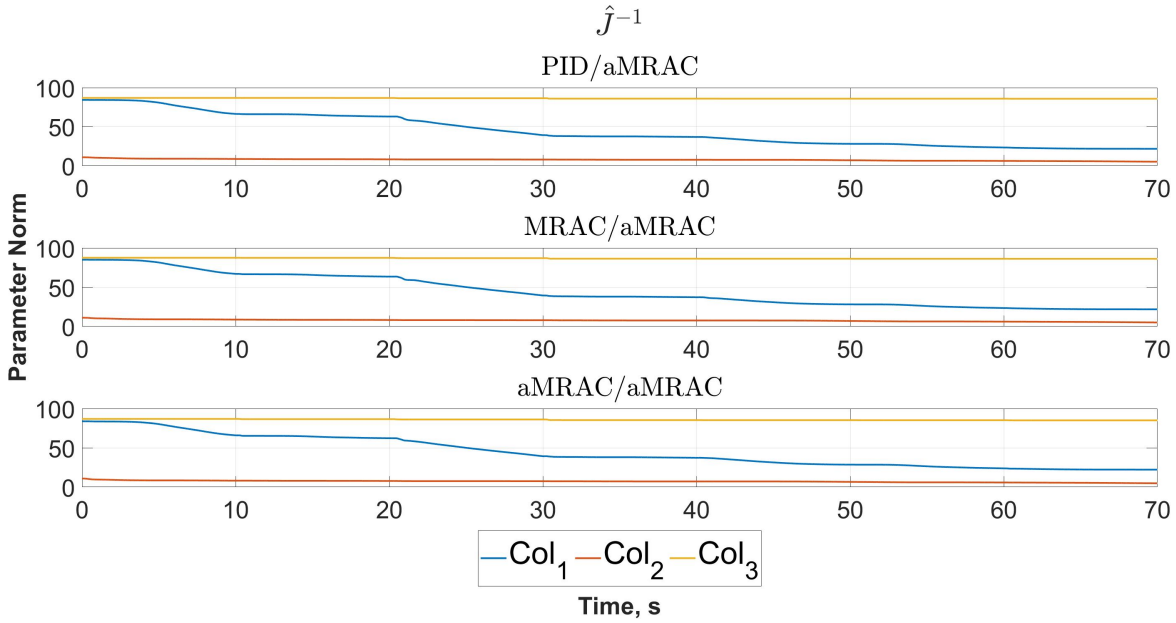


Figure 7.40: Jagged trajectory, increased mass omnicopter $\hat{\boldsymbol{J}}^{-1}$ for aMRAC attitude control comparison

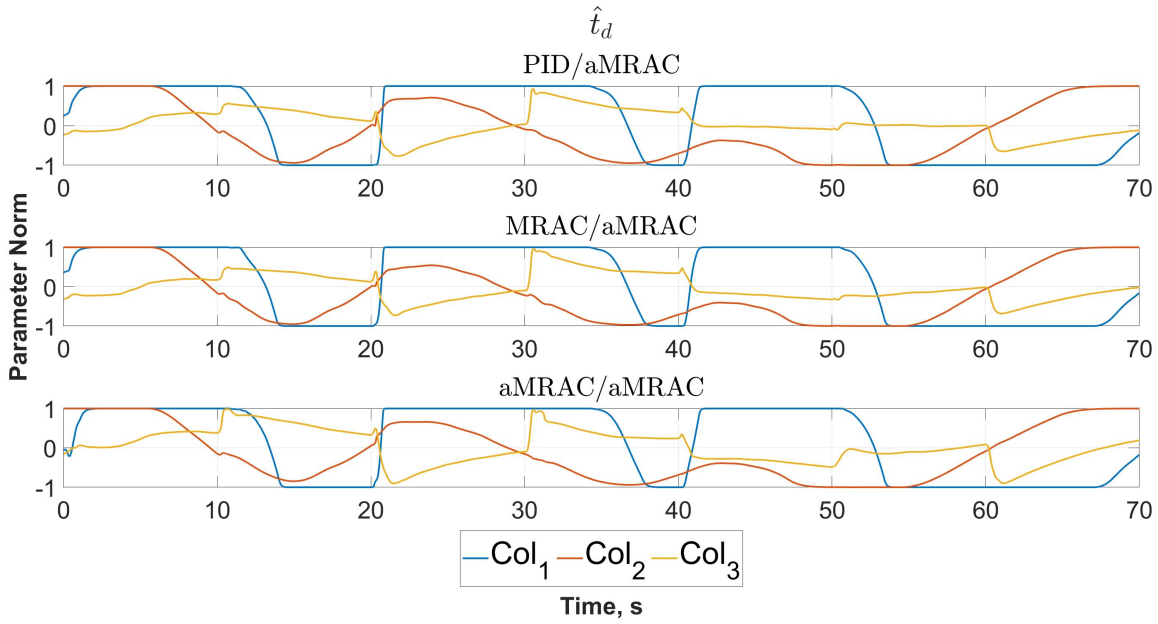


Figure 7.41: Jagged trajectory, increased mass omnicopter $\hat{\boldsymbol{t}}_d$ for aMRAC attitude control comparison

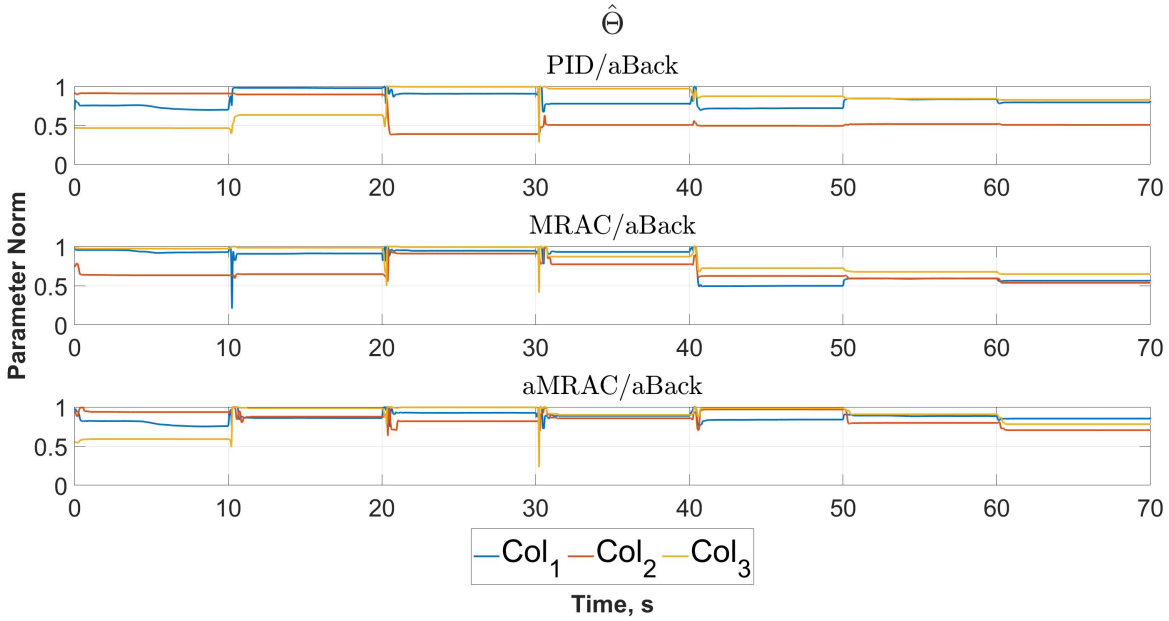


Figure 7.42: Jagged trajectory, increased mass omnicopter $\hat{\boldsymbol{\Theta}}$ for aBack attitude control comparison

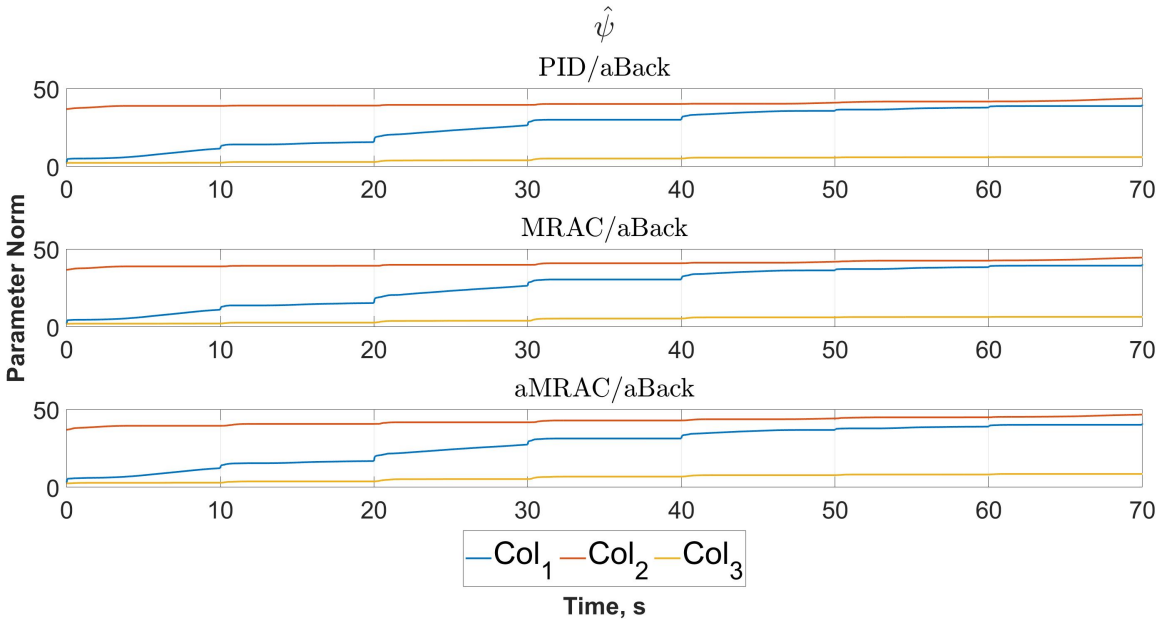


Figure 7.43: Jagged trajectory, increased mass omnicopter $\hat{\mathbf{\Psi}}$ for aBack attitude control comparison

The column-wise norms of the adaptive parameters for the adaptive attitude controllers used to execute the smooth trajectory when the omnicopter is carrying a 0.25 kg panel are shown in Figures 7.44 - 7.48.The smooth trajectory yields results that look similar to the jagged trajectory, except smoother, except for the $\hat{\Theta}$ values in both controllers. For adaptive backstepping, the norm of column 1 decreases more than it does for the jagged trajectory. For aMRAC, column 1 went from having one of the highest magnitudes to having one of the lower magnitudes.

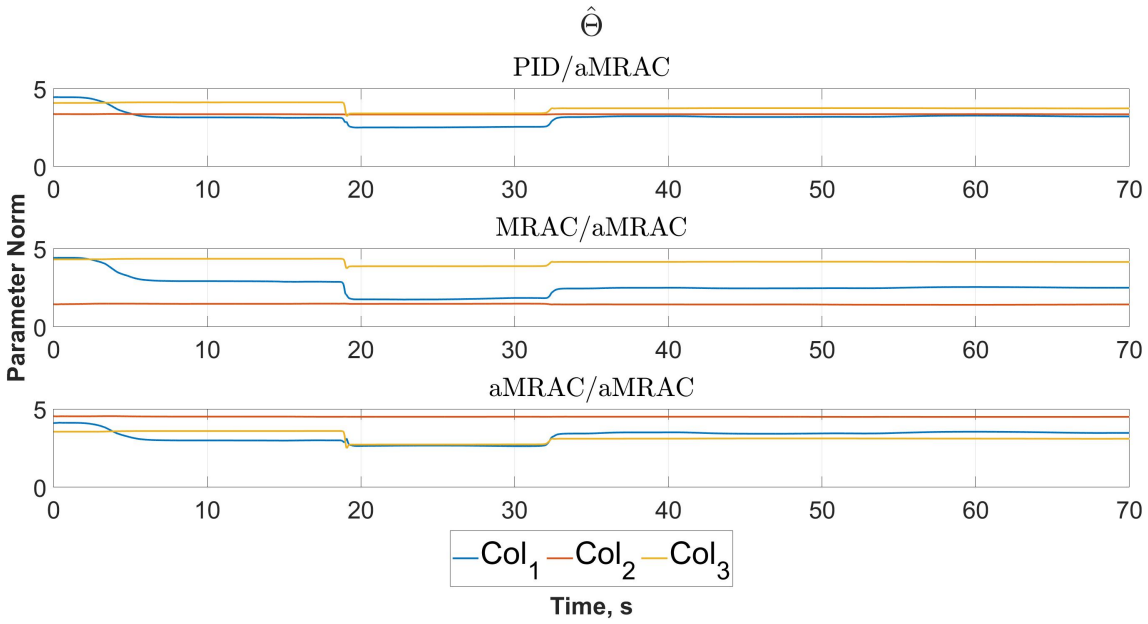


Figure 7.44: Smooth trajectory, increased mass omnicopter $\hat{\mathbf{\Theta}}$ for aMRAC attitude control comparison

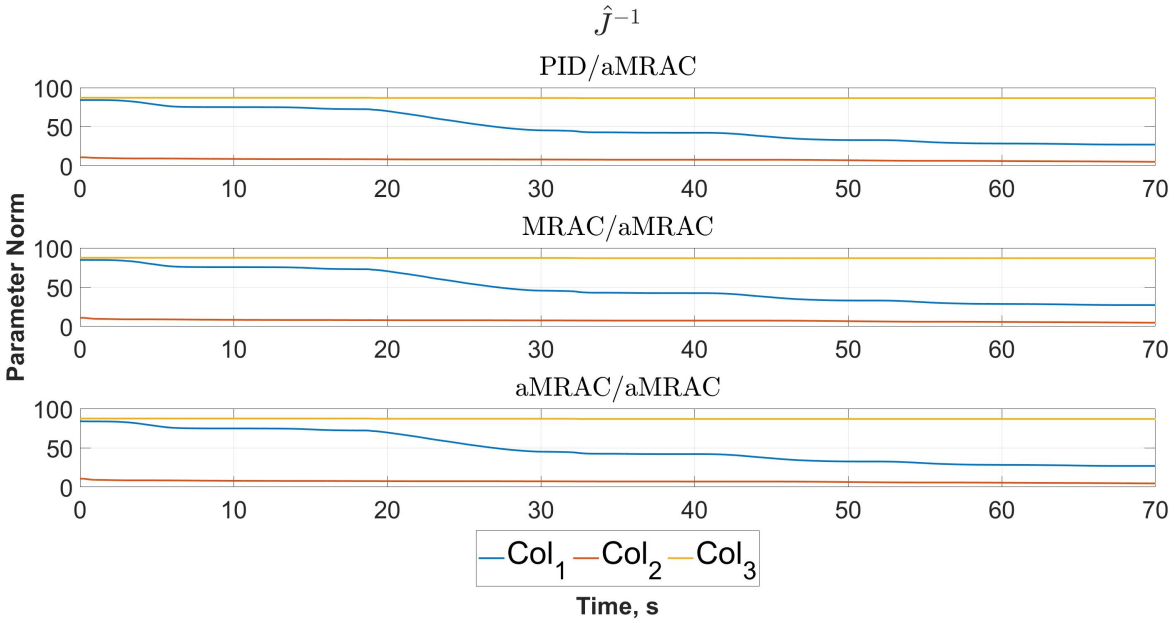


Figure 7.45: Smooth trajectory, increased mass omnicopter $\hat{\boldsymbol{J}}^{-1}$ for aMRAC attitude control comparison

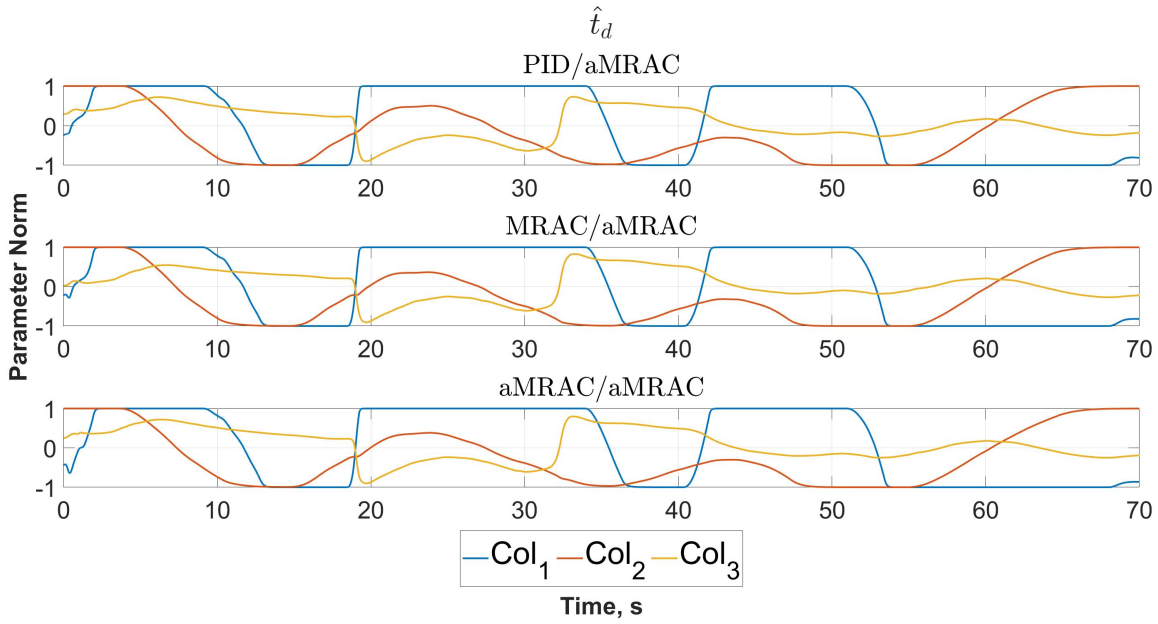


Figure 7.46: Smooth trajectory, increased mass omnicopter $\hat{\boldsymbol{t}}_d$ for aMRAC attitude control comparison

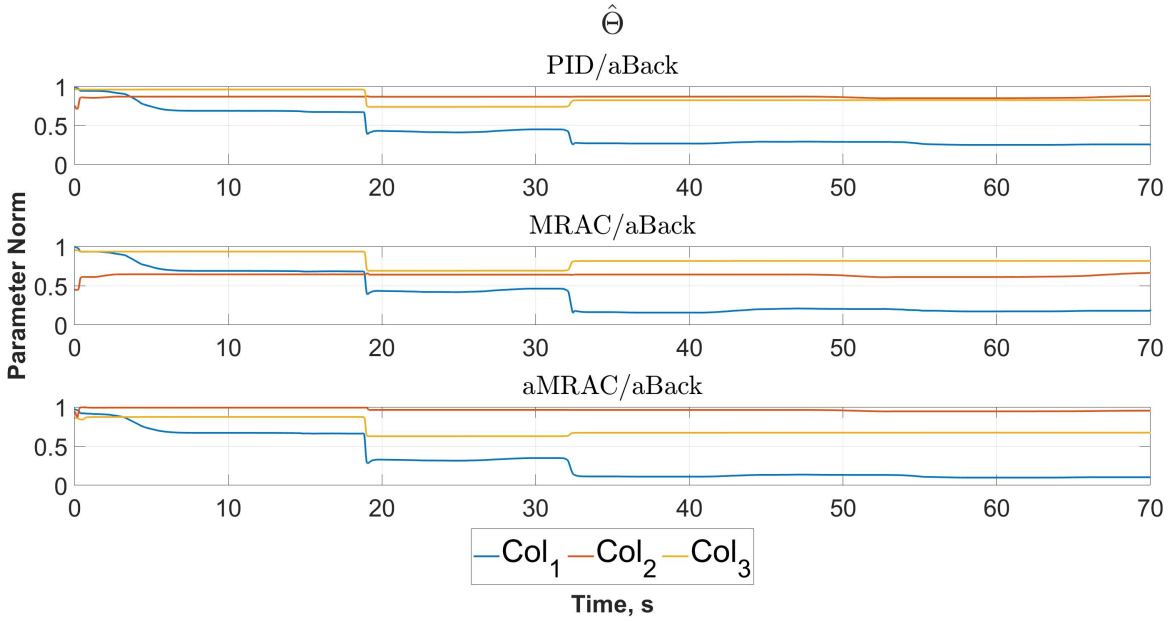


Figure 7.47: Smooth trajectory, increased mass omnicopter $\hat{\boldsymbol{\Theta}}$ for aBack attitude control comparison

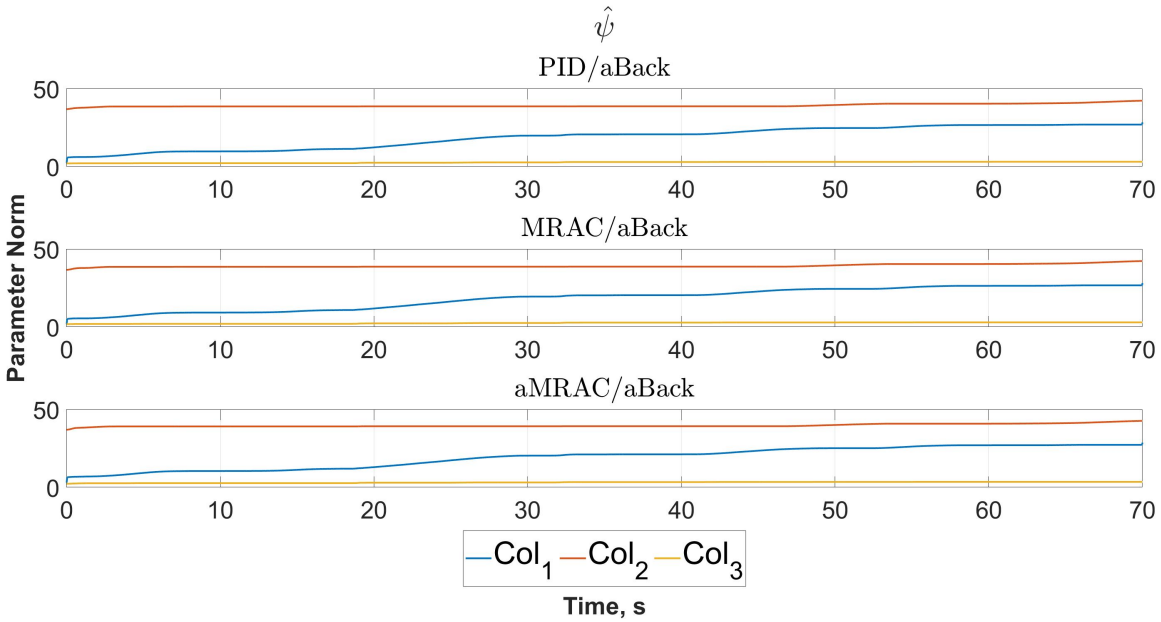


Figure 7.48: Smooth trajectory, increased mass omnicopter $\hat{\mathbf{\Psi}}$ for aBack attitude control comparison

7.4 Discussion

The aMRAC controller was shown to provide the best position control, regardless of disturbance or attitude controller choice. When paired with the smooth trajectory, the aMRAC controller could provide, on average, less than 1 cm of error for all disturbances at nominal mass and less than 2 cm of error for the increased mass. For the jagged trajectories, which include errors associated with transient responses every 10 seconds, aMRAC was able to maintain average tracking accuracy within 1.25 cm and 4 cm for nominal and increased mass, respectively.

The MRAC controller yielded similar or better tracking than the PID position controller for the nominal mass scenario. When mass increased, MRAC exhibited large transient errors and oscillations after some direction changes. This poor performance coupled with the difficulty

in designing the reference system and tuning the adaptive controller makes it the least ideal position controller. Although the PID controller had the worst tracking performance, it was the easiest to implement. It also served as a good reference model for the aMRAC controller.

For attitude tracking, the adaptive backstepping controller consistently had the smallest attitude tracking errors for both mass scenarios and trajectories. The oscillations that appeared in the torque commands also appear in the motor commands. However, the oscillations did not appear in the motor responses or omnicopter dynamics, indicating the the motor response filtered out the high frequency changes. While the oscillations do not appear to affect dynamics, they would be a concern for motor wear. Ideally, the source of the oscillations can be identified and eliminated without affecting tracking ability. The oscillations do not appear in the nonadaptive backstepping controller, so the cause is most likely due to one of the changes for adaptation, except for the adaptive parameters, which do not exhibit any oscillation in their results. As will be discussed in Ch. 8.3, further simulations revealed that eliminating the addition of the $sgn(\boldsymbol{z}_2)\hat{\boldsymbol{\psi}}$ term eliminated the oscillations without significantly affecting tracking performance.

For the nominal mass scenario, after adaptive backstepping, backstepping is the best choice for the jagged trajectory because it achieves a fast steady state response. For the smooth trajectory, the PI/FL controller maintains smaller errors than backstepping. With the omnicopter at its nominal mass, the difference between the largest and smallest attitude errors was within 1 degree. Since all the controllers were tuned for the nominal mass, it shows that with appropriate tuning, any controller is capable of achieving reasonable tracking for a "nominal"/specific configuration. The true benefit of adaptive control can be seen for large deviations from the baseline, such as the increased mass scenario. For the increased mass scenarios, the adaptive controllers consistently had smaller attitude errors than the nonadaptive controllers. This indicates that for large systemic changes, such as the addition

of a large sensor or carrying an object, aMRAC or adaptive backstepping would be the best option for attitude control.

It should be noted that a benefit of PI/FL over all the other controllers is its simplicity. It requires three tuning parameters and three relatively simple equations to implement, whereas the other controllers, especially the adaptive controllers, require several more tuning parameters, equations, and the use of a motor model. A difficulty with the PI/FL controller is its reliance on motor angular velocity measurements, which may not be directly measurable. It is not known how much the absence of this value would impact attitude tracking. Also, the use of the PI/FL controller could cause long lasting, large magnitude transient oscillations.

In conclusion, the recommended control combination for the omnicopter is aMRAC for position control and adaptive backstepping for attitude control. Adaptive control shows performance improvements for position tracking. Attitude tracking is more nuanced, although adaptive control showed benefits for large parametric uncertainties. It may be worth exploring other attitude control schemes, both adaptive and non-adaptive, that can consistently yield fast settling times and smaller average errors.

Chapter 8

Flight Tests

Flight tests of an omnicopter were performed to verify the Simulink models used to initially tune and compare the controllers. Although several controller combinations were compared using simulations in Chapter 7, due to time constraints only the best controller combination from the simulations (aMRAC/aBack) was compared to the baseline controller (PID/PIFL) during flight tests. Additionally, while the focus of this dissertation is on the evaluation of omnicopter controllers, an overview of the hardware and software used to perform these tests is provided in this chapter. Including these overviews provides context for how these flight tests were performed and highlights the laboratory and software development that were necessary to accomplish before any tests could be undertaken.

8.1 Omnicopter Hardware Overview

The omnicopter used for flight testing is designed to be a similar size and use similar motors to the original designed by Brescianini [3]. However, while the original design of the omnicopter used a custom flight computer/controller, the design of this omnicopter uses all commercial off the shelf (COTS) electronics, including, most importantly, the flight control unit (FCU). Additionally, all 3D printed parts used to build the vehicle were designed by students at Virginia Tech. 3D printed parts are shown in non-black colors in the CAD rendering in Figure 8.1.

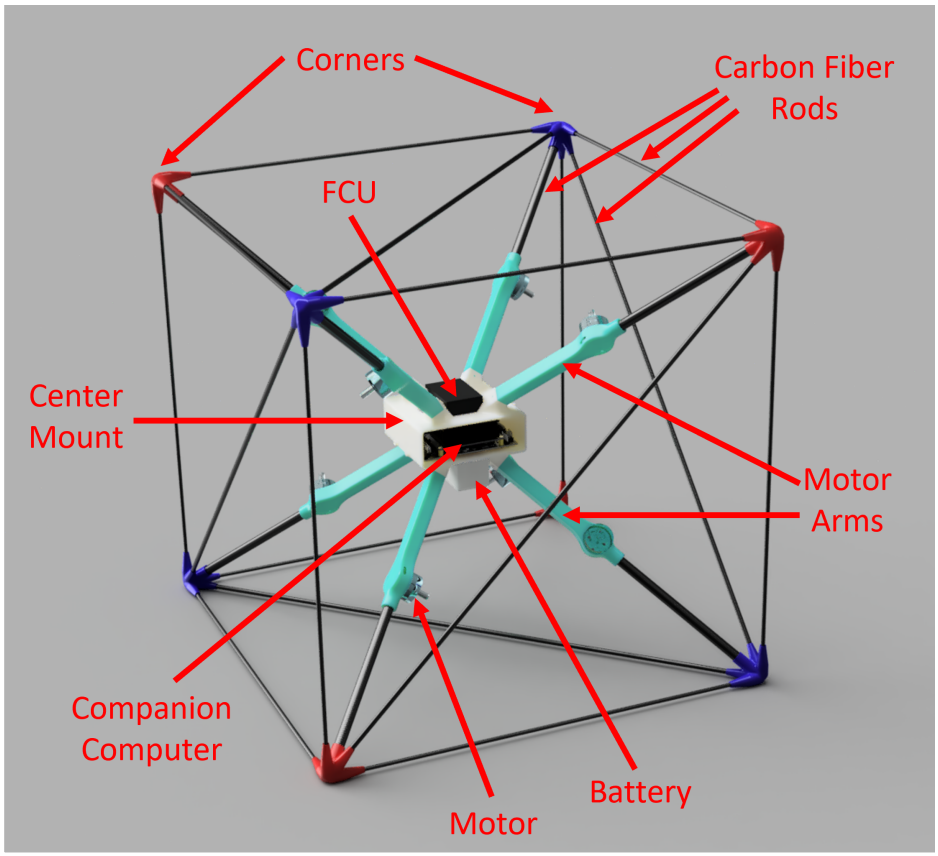


Figure 8.1: Omnicopter CAD Rendering (credit: Patrick Thomas)

Figures 8.2 shows a diagram of the omnicopter with the center mount highlighted in the inset. The cube measures 435 mm per side if using 8 inch propellers or 440 mm per side if using 9 inch propellers. The corners (blue and red (Fig. 8.1) or blue (Fig. 8.2)), motor arms (teal or black), and center mount (white or black) are 3D printed. The corners connect to each other and the motor arms with carbon fiber rods, which are the black cylinders in Figures 8.1 and 8.2. The motor arms slide into mounting holes printed on the center mount that ensure the correct motor angle is maintained. A Pixhawk 4 mini flight control unit (FCU) fits onto the top of the center mount for the omnicopter with 8 inch propellers and is mounted on a vibration isolation mount attached to the top of the center mount for the omnicopter with 9 inch propellers. A Raspberry Pi 4B or a Jeston nano can be mounted

to the interior of the center mount. For these flight tests, a Raspberry Pi 4B is used as the companion computer. A 4s battery can be velcroed to the bottom of the center mount. Two Lumenier 35A 4x1 electronic speed controllers (ESC) are mounted on either side of the center mount. The PM06 V2 power management board (PMB) is attached to the front of the center mount and connected to the battery, distributing power to the ESCs, Raspberry Pi, and FCU. To generate thrust, 8 RTF 2208-1100KV motors with 8 or 9 inch bidirectional propellers are used. The Pixhawk sends PWM commands through the ESCs to the motors. For more information on the hardware design of the omnicopter, see [85].

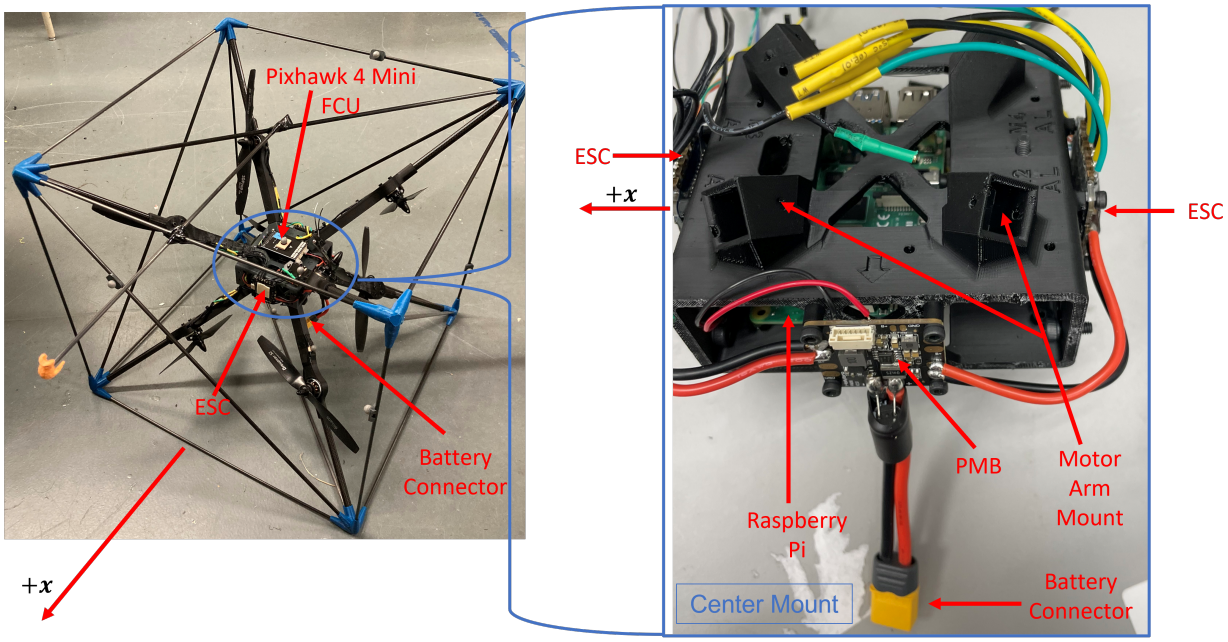


Figure 8.2: Omnicopter Hardware and Center Mount Diagram

The Pixhawk 4 mini runs the PX4 flight control firmware [86]. The flight controllers designed in the previous chapters are implemented within the firmware code. The details of how custom controllers are implemented within the firmware will be discussed in the following section.

An overview of the laboratory command and control (C2) architecture used to communicate

with the Omnicopter is shown in Figure 8.3. The Raspberry Pi 4B is installed with Ubunutu 20.04 and ROS Noetic for running MAVROS. Since the Raspberry Pi is connected to the Pixhawk via a serial connection, it is used to communicate information between the Pixhawk and the Ground Station, which is also running MAVROS. For example, desired trajectory position waypoints are published to the mavros topic **/mavros/setpoint_raw/local** , attitude setpoints are published to **/mavros/setpoint_raw/attitude**, and the Optitrack position and attitude tracking data are published to **/mavros/vision_pose/pose**. MAVROS then converts the data to MAVLink messages, which Pixhawk decodes. The Pixhawk then passes the information within the PX4 firmware as uORB topics. More information on the communication used within the PX4 firmware for the omnicopter can be found in Chapter 5 of [85], and more information on the PX4 architecture can be found in [87]. Further information on the lab set-up can be found in [2].

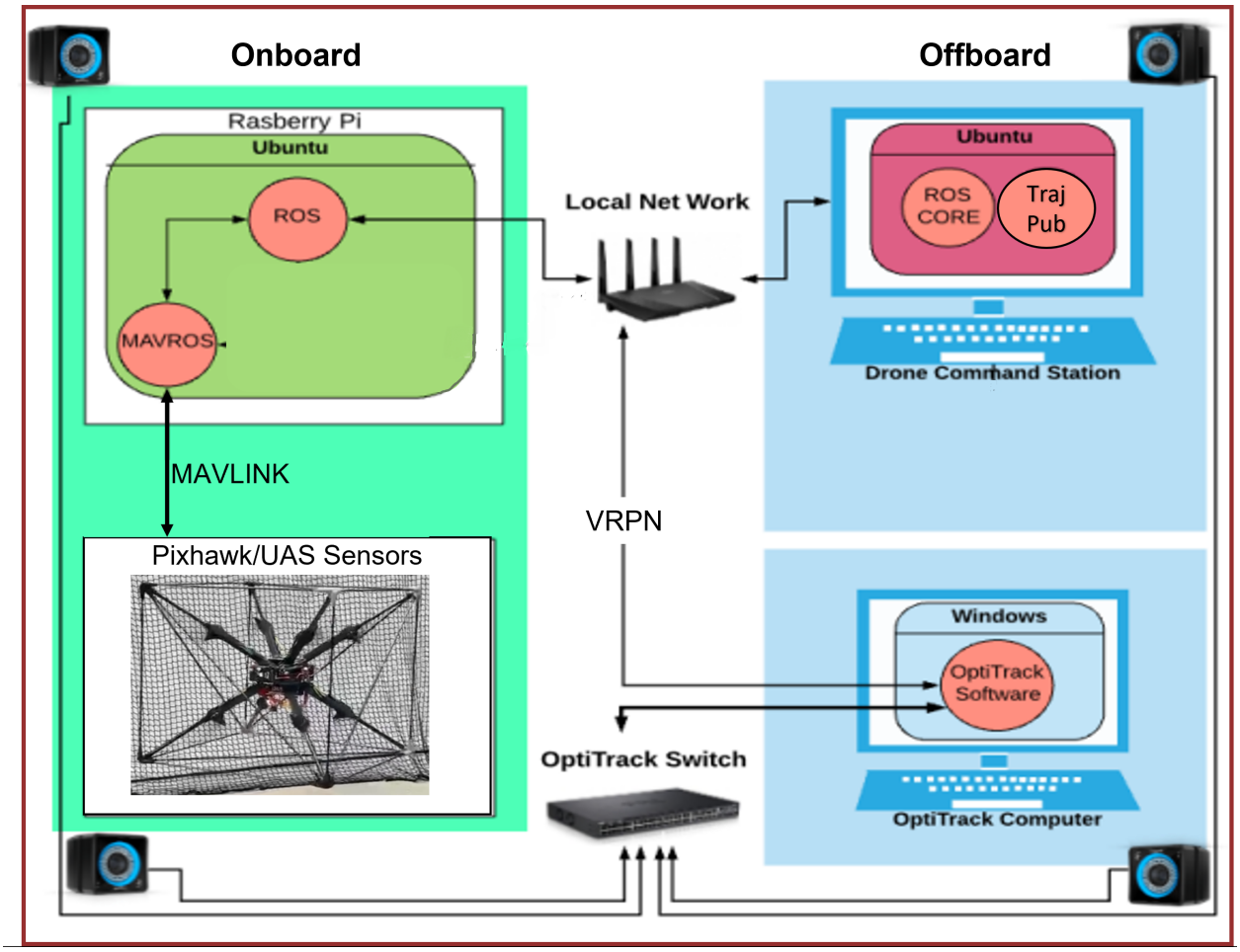


Figure 8.3: Laboratory C2 architecture, adapted from [2]

8.2 Custom Firmware Implementation

Custom controllers were implemented as modules within the **/src/modules** folder of the PX4 firmware [86]. Variable and function definition as well as uORB topic subscription and publishing occur in the .hpp file. Code for position control, attitude control, and control allocation occurs in the .cpp file. Each module consists of a unique position and attitude controller combination (aka the position and attitude controllers are not individual units that can be turned on and off as desired).

The modules are run as work_queue tasks. For the fully adaptive controller (aMRAC/aBack), all controllers update at 100 Hz. For the baseline controller (PID/PIFL), the angular velocity controller runs at 200 Hz and the position and attitude controllers run at 50 Hz. Trajectory setpoints are published at 50 Hz. Details on the omnicopter mixer can be found in [85]. Full instructions for how to create custom controllers in the PX4 Firmware can be found in Appendix B.

The .hpp and .cpp files for the fully adaptive controller can be found in the **src/modules/omni_amrac_aback** folder in the git repository. The primary functions of interest are **update_position_control** for aMRAC position control and **update_attitude_control** for adaptive backstepping attitude control. In **update_motor_throttle**, the force and torque commands from the position and attitude controllers are combined and converted to desired motor commands.

8.3 Controller Adjustments for Flight

The physical design of the omnicopter used for flight tests was significantly different compared to the modeled omnicopter used for simulation (the original vehicle designed in [3]). The changes in physical parameters (namely mass, moment of inertia, and motor responsiveness), as well as lessons learned via simulations necessitated the implementation of several controller code adjustments compared to the simulation code before successful flight tests could be achieved. Most significantly, different tuning values were required for all the controllers. Additionally, adjustments to some of the controller equations and Pixhawk measurement parameters needed to be made to improve flight performance and compensate for the reference frames used within the code.

The omnicopter used in testing had a higher mass and moment of inertia compared to

the original design, whose parameters were used for the Simulink simulations. The change in parameters naturally necessitated different tuning gains, especially for the nonadaptive controllers. The mass of the omnicopter used for flight tests was approximately $1.4kg$ (almost 1.6 times greater than $0.892kg$ used for simulations), subject to change depending on the battery used. The moment of inertia of the flight test vehicle, calculated from a CAD model, was approximately $\boldsymbol{J} = (0.0295, 0.0294, 0.0293)kg \cdot m^2$ (almost double $\boldsymbol{J} = (0.0116, 0.0113, 0.0113)kg \cdot m^2$ used for simulations), also subject to change depending on the battery used. The tuning gains used for flight tests of the baseline controllers are shown in Table 8.1. The tuning gains used for flight tests of the adaptive controllers are shown in Table 8.2. The general trend was a decrease in magnitude of the control gains.

Table 8.1: Baseline Controller Parameters

PID		**PIFL**	
Parameter	**Value**	**Parameter**	**Value**
$k_{pos,P}$	9.28	$k_{att,P}$	3.21
$k_{pos,I}$	3.79	$k_{att,I}$	0.89
$k_{pos,D}$	5.61	τ_ω	0.085

Table 8.2: Adaptive Controller Parameters

aMRAC		**aBack**	
Parameter	**Value**	**Parameter**	**Value**
$\boldsymbol{\Gamma}$	diag([10 10 10])	$\boldsymbol{K}_1$	$5I_{3\times3}$
$\|\|f\|\|_{max}$	35	$\boldsymbol{K}_2$	$20I_{3\times3}$

Unless otherwise noted, the other parameters used for the simulations were also used for

flight tests, including the initial conditions. For aMRAC, the initial z position in $\overline{\boldsymbol{x}}_d$ is set to 0.2 to reflect the actual height of the omnicopter center of mass above the ground.

As seen in Table 8.2, the allowable norm for the commanded forces for aMRAC was increased. However, because slow ascents and descents as well as smooth trajectories were used, the motors never saturated.

Tuning of the IMU parameters also helped with the attitude controller performance. The IMU_CYRO_CUTOFF frequency was set to 30 Hz.

The PID controller uses the real mass of the omnicopter ($\sim 1.4kg$) to get the best performance. The aMRAC controller needs the addition of $m\boldsymbol{g}$, or a close approximation, to the commanded force ($f_{cmd} = R(\boldsymbol{q})(\boldsymbol{f}_{bl} + \boldsymbol{f}_{ad} + m\boldsymbol{g})$) to prevent the omnicopter from oscillating up and down while it hovers. Because the mass is currently hard-coded, it may be beneficial in the future to try including gravity within a term that can be adapted upon, for example $\boldsymbol{\Phi}$, so the term can be more robust to mass changes. For test flights, $m = 1.25kg$ was used as the mass for the offset.

The adaptive backstepping controller does not use the $\hat{\boldsymbol{\psi}}$ adaptation. During the simulations, oscillations in commanded torques were noticed. Via further simulations, it was found that removing the adaptation on the disturbance bounds (the adaptation: $sgn(\boldsymbol{z}_2)\hat{\boldsymbol{\psi}}$) also eliminated the oscillations in the torque commands without significantly decreasing attitude tracking performance.

Preliminary flight tests of roll, pitch, and yaw saw the omnicopter become unstable (gyrating) as it rotated to large angles (past 45 degrees). It was found that the error quaternion was expressed in the inertial frame instead of the body frame, requiring a rotation to the body frame. In the adaptive backstepping controller, this rotation is applied to the vector component of the error quaternion. In the PI/FL controller, the rotation is applied to the

components of ω_{cmd} that are computed using quaternion error.

8.4 Flight Test Results

During simulations, random force and torque disturbances of varying magnitude were input to the system, primarily to provide a stand-in for disturbances due to unmodeled dynamics, such as downwash interactions amongst the blades, airflow from other vehicles, or air currents within a room. For flight tests, random disturbances were not purposely input to the system. The omnicopter was flown alone and indoors with little to no airflow from the air conditioning system. Trajectories were created to keep the omnicopter away from walls, floors, and ceilings to limit crash risk as well as airflow interaction with solid surfaces. The downwash interactions amongst the blades still naturally existed, and although avoiding airflow surface interactions was attempted, these perturbations still existed within the room and would be an interesting future study.

8.4.1 Attitude Observations

Roll and pitch to large angles struggle because they require motor direction changes. For all flight tests presented here, biasing is not used in the control allocation algorithm to prevent rapid and abrupt motor direction changes. Therefore rapid attitude changes (high angular velocity/acceleration) are unattainable with the current motor control, which necessitates slow and smooth attitude trajectories. For trajectories where the omnicopter is commanded to continuously point at a specific location, passing too close to the aim point causes the omnicopter to lose control because the commanded angles and rates are too high. Yaw performance is the best because motors do not have to change direction.

Table 8.3: Flight Test Trajectory Creation Points

Time (s)	**(x,y,z) Location** (m)
0	$(0, 0, 1)$
10	$(0.75, -0.75, 1.5)$
20	$(2.5, 0, 0.8)$
30	$(1.25, 1, 1.35)$
40	$(1.25, -0.8, 1.25)$
50	$(1.85, 0, 1.65)$
60	$(0.5, 0.5, 1.25)$
70	$(0, 0, 1)$

8.4.2 Smooth Trajectory

For flight testing, a smooth trajectory, shown in Figure 8.4, was generated using waypointTrajectory.m in MATLAB, the same process that was used to generate the smooth trajectory for the previous simulations. It should be noted that the smooth trajectory used for flight testing was different than the one used for simulations due to lab space constraints, particularly in the z direction. The selected points for generating the trajectory are shown in Table 8.3 and denoted by black dots in Figure 8.4. As with the simulations, the omnicopter has 10 seconds to travel between each point. Throughout the trajectory, the omnicopter is commanded to point its +x face to $(1.25, 0, 1.25)m$, denoted by the red dot in Figure 8.4, while the +z axis is kept perpendicular to the velocity vector. As mentioned in Section 8.4.1, passing too close to the aim point commands large attitudes that cause the omnicopter to crash. Correspondingly, the turn between points 5 and 6 in the smooth trajectory in Figure 8.4 used for testing causes attitude commands that are as large as possible without the omnicopter falling out of the sky. Section 8.4.1 also mentions favorable yaw performance. A good example of yaw performance can be seen between points 4 and 5 in Figure 8.4.

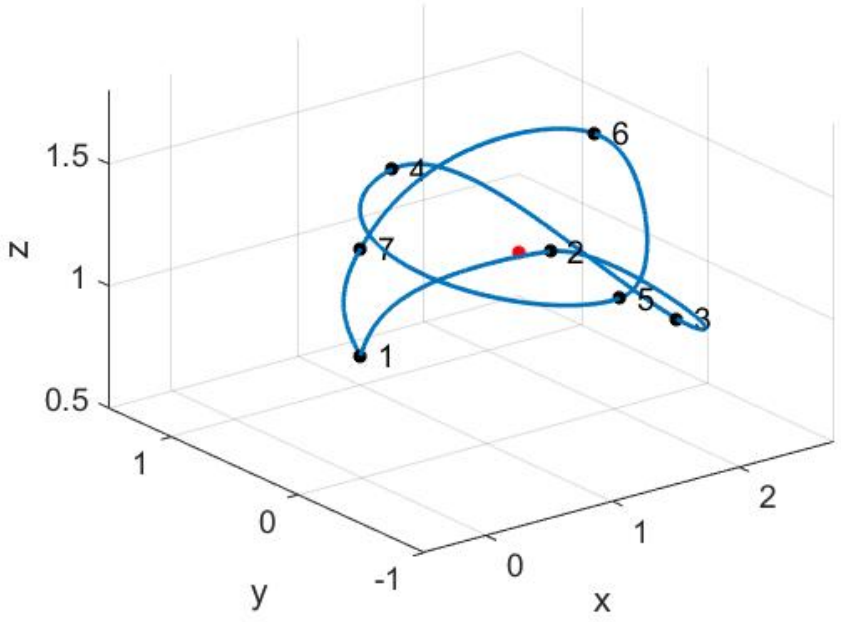


Figure 8.4: Smooth Trajectory for Flight Tests. All units are in meters

The omnicopter with 9 inch blades was flown for these flight tests. The code used the $\boldsymbol{T}$ matrix computed for the omnicopter with the 9 inch blades and the proper motor constants. Seven flight tests were accomplished for each controller, which was one flight per usable battery in the lab.

Data used for error analysis was logged at 100 Hz for aMRAC/aBack and 200Hz for PID/PIFL within the **omni_adap_debug** or **omni_debug** uORB topic. The error computations from Section 7.1 are used to calculate the average position and attitude errors for just the smooth trajectory (no takeoff and landing). The mean and maximum errors as well as the standard deviation of the errors ($\boldsymbol{\sigma}$) are shown in Tables 8.4 and 8.5.

Table 8.4: Smooth Trajectory Flight Test Position Errors

	Battery		PID/PIFL Error			aMRAC/aBack Error		
Flight	**Rating (mAh)**	**Weight (g)**	**Mean (m)**	**σ (m)**	**Max (m)**	**Mean (m)**	**σ (m)**	**Max (m)**
1	1550	157	.0509	.0353	.2571	.0321	.0264	.1259
2	1550	177	.0570	.0391	.2904	.0327	.0255	.1242
3	1550	177	.0535	.0390	.4060	.0338	.0259	.1374
4	1800	198	.0478	.0221	.1334	.0346	.0278	.1337
5	1800	198	.0530	.0257	.1859	.0340	.0271	.1424
6	1800	198	.0480	.0216	.1172	.0340	.0256	.1179
7	2200	239	.0492	.0203	.1156	.0342	.0258	.1320
	Average		.0513	.0290	.2151	.0336	.0263	.1305

Table 8.5: Smooth Trajectory Flight Test Attitude Errors

	Battery		PID/PIFL Error			aMRAC/aBack Error		
Flight	**Rating (mAh)**	**Weight (g)**	**Mean (deg)**	**σ (deg)**	**Max (deg)**	**Mean (deg)**	**σ (deg)**	**Max (deg)**
1	1550	157	8.33	4.39	34.50	8.44	2.48	16.13
2	1550	177	8.28	3.27	22.12	8.97	3.15	17.29
3	1550	177	8.21	4.33	39.51	8.21	2.97	17.49
4	1800	198	7.74	3.47	19.12	8.34	3.15	19.74
5	1800	198	8.17	3.64	26.88	9.18	3.31	19.85
6	1800	198	7.34	3.32	20.84	8.75	2.89	17.20
7	2200	239	7.15	2.93	14.47	6.78	2.42	14.17
	Average		7.89	3.62	25.35	8.38	2.91	17.41

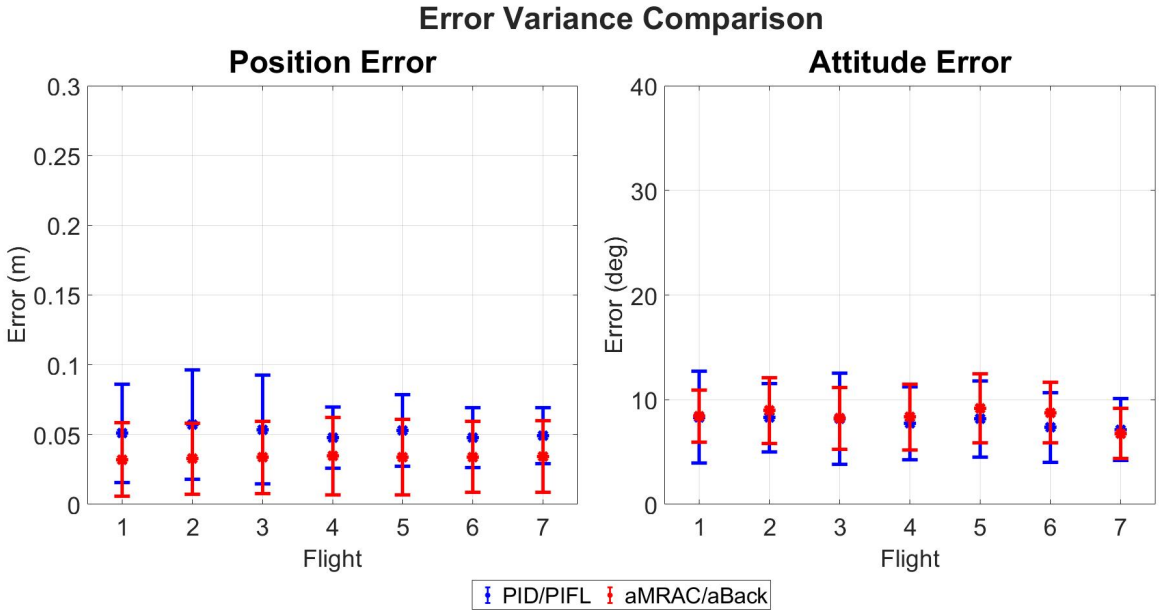


Figure 8.5: Controller error variance comparison for the smooth trajectory. Dots represent mean error, and error bars represent one standard deviation above and below the mean error.

Flight tests showed that motor switching/rapid motor switching tend to occur between points 5 and 7. If the motors were too hot, neither controller was able to command the omnicopter through the motor changes, and it would crash. Once this was learned, adequate time (about 10 minutes) was provided between flight tests to allow the motors to cool down.

The aMRAC/aBack controller was able to achieve smaller average position errors than the PID/PIFL controller by about .018 m. For attitude, the PID/PIFL controller was able to achieve smaller average attitude errors by almost 0.49 degrees. Based on average errors, the two controller combinations appear comparable, which is obvious in Figure 8.5 with the close proximity of the blue and red dots representing mean error, particularly for attitude. However, for position control, the aMRAC/aBack controller yielded smoother and more consistent tracking throughout the trajectory as well as across multiple flights. Figure 8.7 is representative of all other aMRAC/aBack flights, and Figure 8.6 is representative of the PID/PIFL flights with the 1800 and 2200 mAh batteries. PID/PIFL flights with the 1550

mAh batteries tended to twist and turn more around point 6 (approximately 50-70 seconds into flight). The aMRAC/aBack controller shows a bump at the start of the trajectory. This is due to the transition from taking off to executing the desired trajectory. The bump is not seen in the PID/PIFL position response, indicating that the aMRAC/aBack controller is more prone to overshooting a standalone set point.

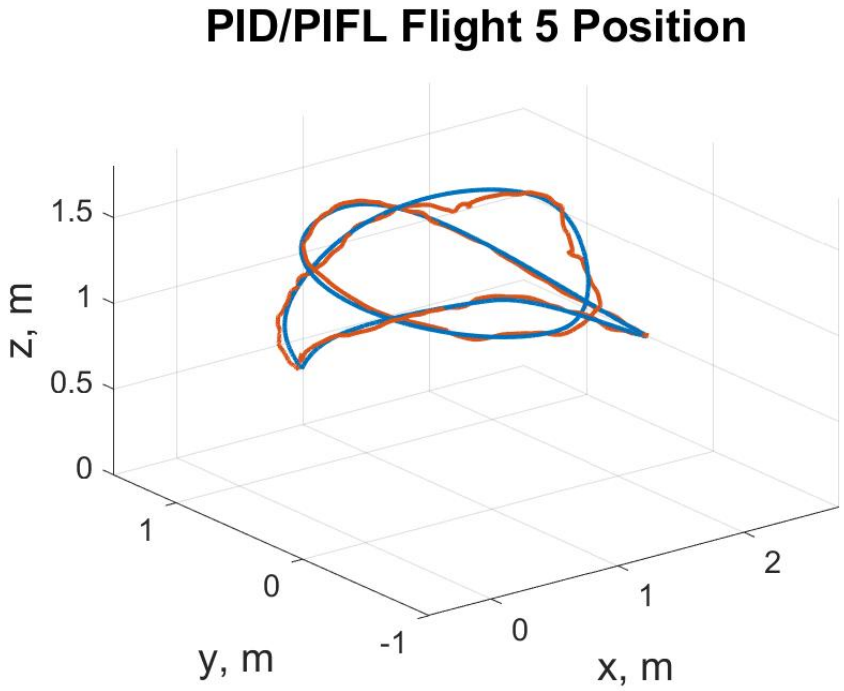


Figure 8.6: PID/PIFL 3D position tracking for smooth trajectory. The orange line is the actual trajectory flown, and the blue line is the desired trajectory.

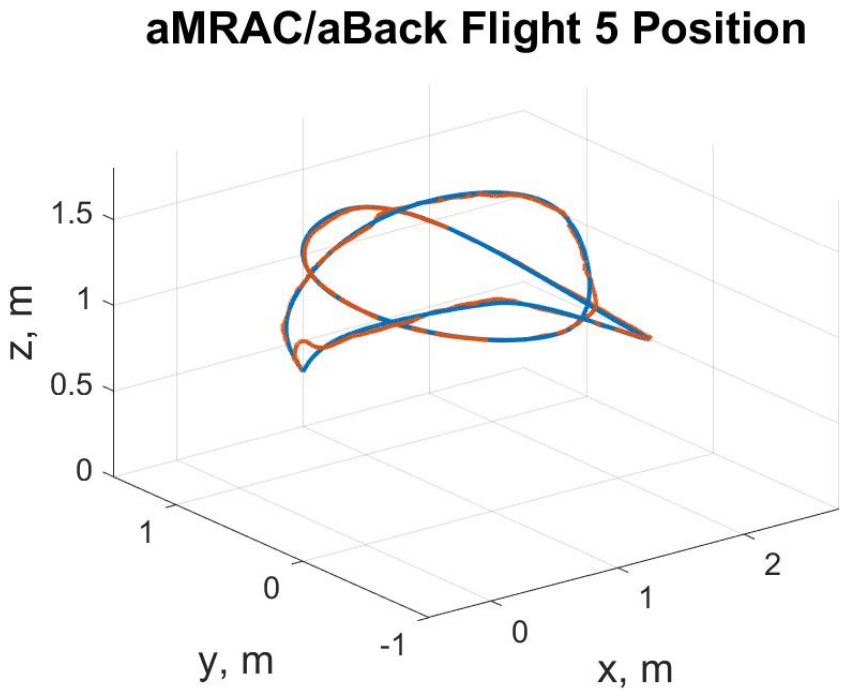


Figure 8.7: aMRAC/aBack 3D position tracking for smooth trajectory. The orange line is the actual trajectory flown, and the blue line is the desired trajectory.

The error plots for every flight can be seen in Figures 8.8 and 8.11. PID/PIFL shows a slight improvement in attitude error with increasing battery mass in the attitude table, about a 1 degree improvement between the lightest and heaviest batteries. The performance improvement can also be seen when viewing all the error plots together. Only the three light battery flights (Flights 1-3) show a large spike in position and/or attitude error around either 30 or 60 seconds, exemplified by Flight 1 in Figure 8.9. These flights also have the largest standard deviations for position error, with the largest blue error bars in Figure 8.5. The 1800 mAh batteries do exhibit increases in error around 50-60 seconds, but with a smaller magnitude, evident in Figure 8.10. The heaviest battery (2200 mAh) does not exhibit any spikes, but it could be a fluke because only one flight was accomplished with this battery, and the variation amongst other flights is apparent. This attitude error dependence on battery does not appear to extend to aMRAC/aBack. There is no significant variation in the error

plots or within the mean values for attitude error for this controller, see Figure 8.11. The heaviest battery does yield the smallest attitude error, but as with the PID/PIFL controller, one run is not enough to establish a trend. However, the improvement across both controllers is interesting to note. The omnicopters were tuned to carry the 1800 mAh battery, which is considered the standard. It is possible that the increase in battery weight applied some damping to roll and pitch as the omnicopter maneuvered through the trajectory, yielding improved tracking. The difference in controller consistency can also be seen in Tables 8.4 and 8.5. The aMRAC/aBack has smaller differences between the average max and mean position and attitude errors: 0.0969 m and 8.89 deg versus 0.1638 m and 17.46 deg. The adaptive controller also has a smaller average standard deviation than the nonadaptive controller by 0.0027 m and 0.71 deg. The similarly sized red error bars in Figure 8.5 show that the standard deviations across all flights are more consistent for the aMRAC/aBack controller.

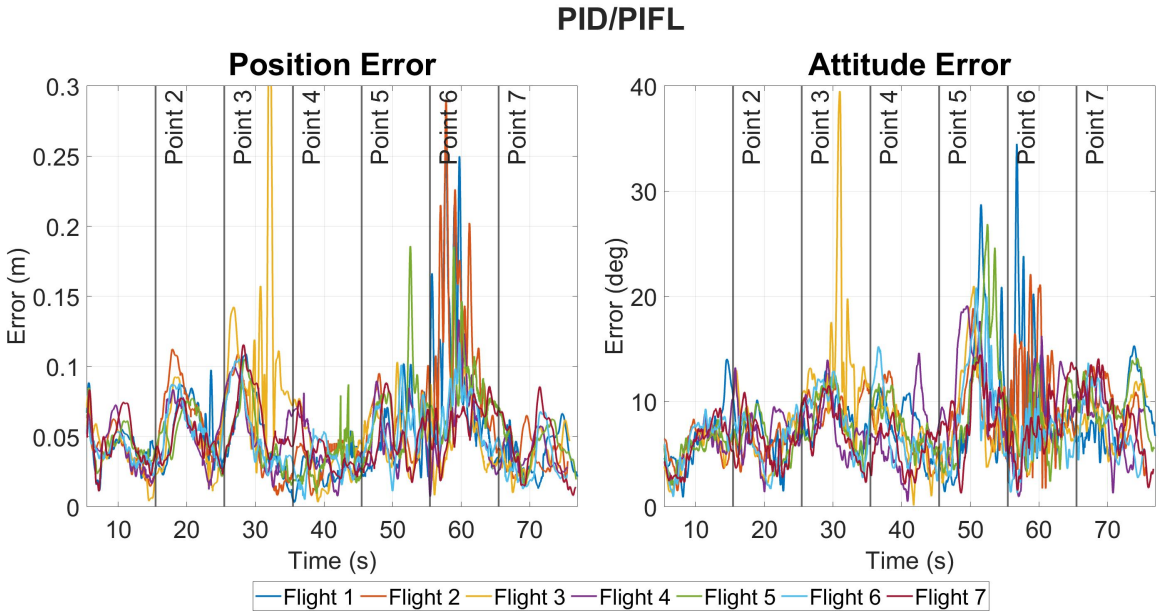


Figure 8.8: PID/PIFL position and attitude error for smooth trajectory

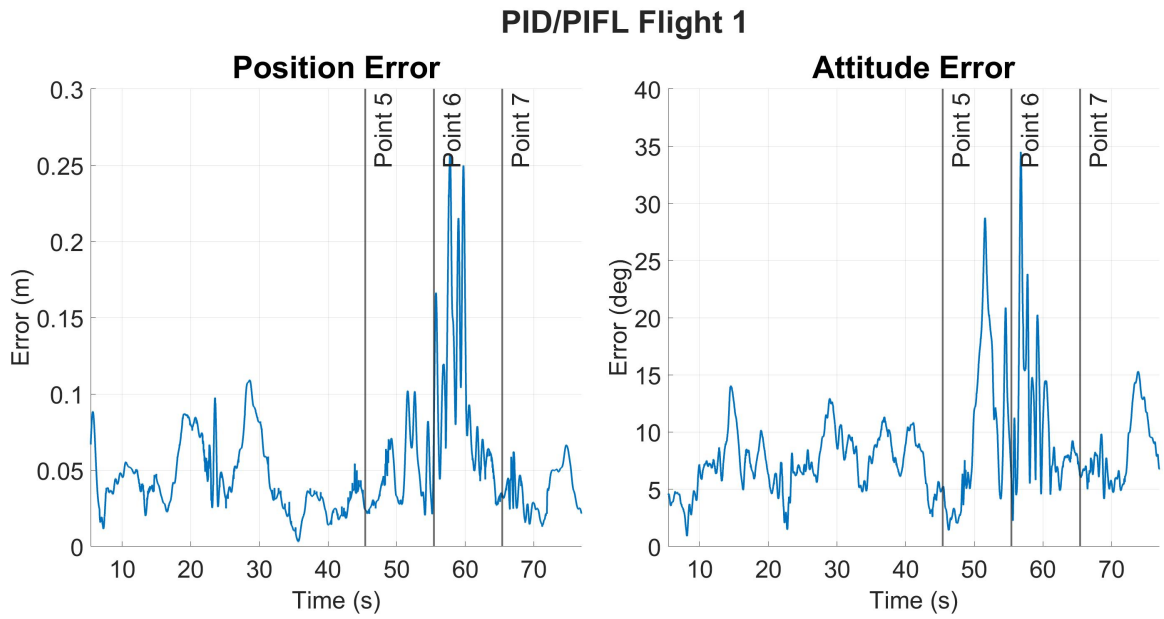


Figure 8.9: PID/PIFL position and attitude error for smooth trajectory, light battery

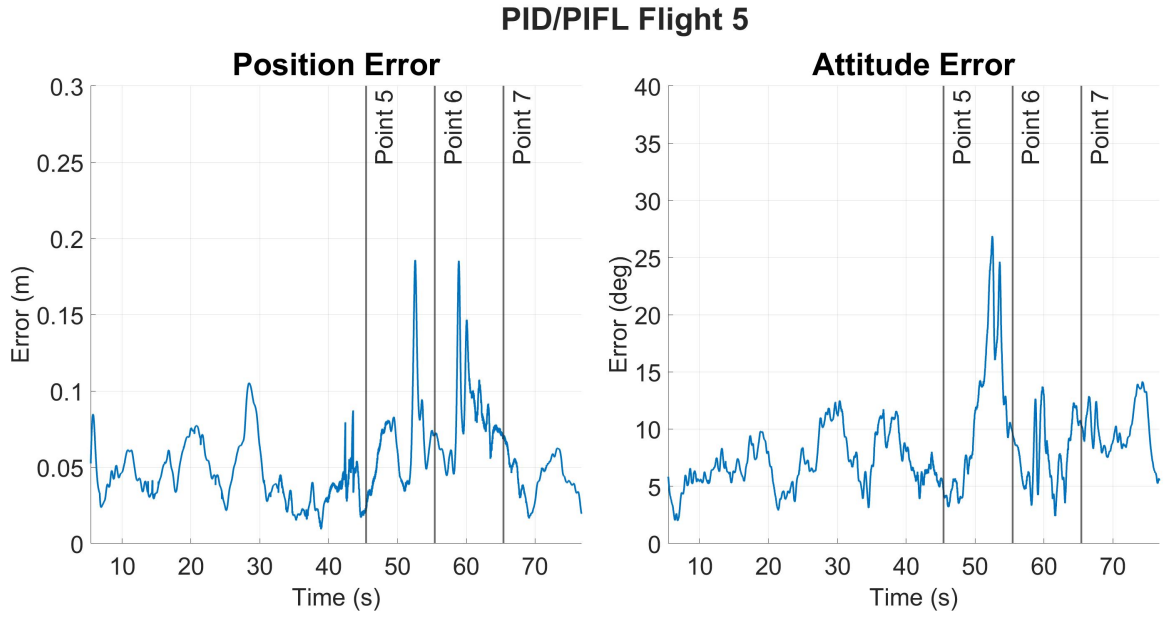


Figure 8.10: PID/PIFL position and attitude error for smooth trajectory, standard battery

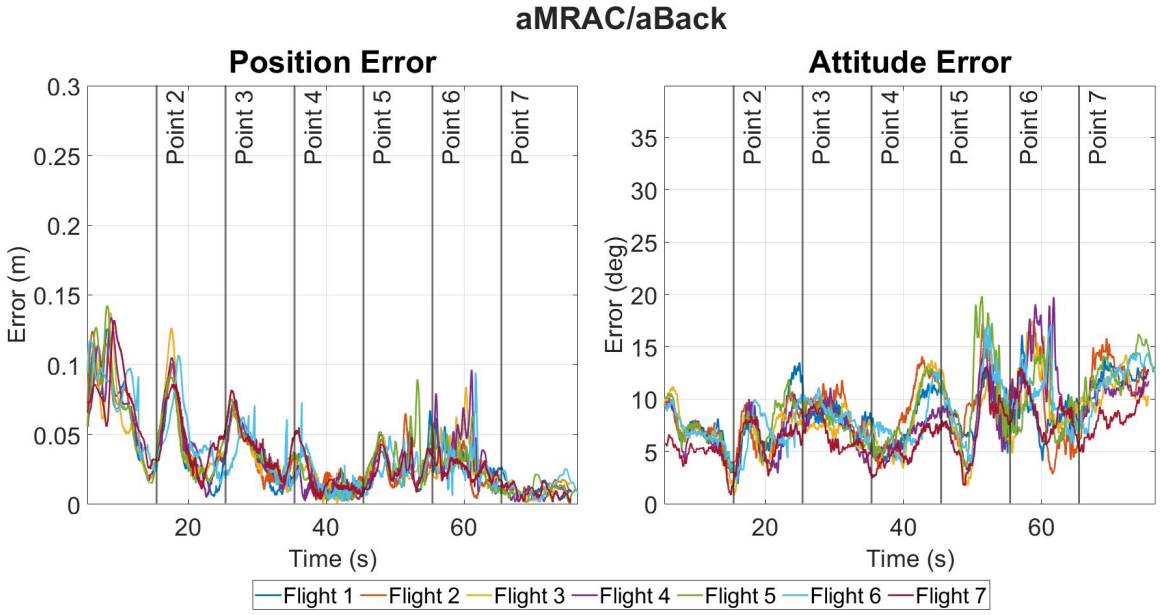


Figure 8.11: aMRAC/aBack position and attitude error for smooth trajectory, light battery

Figures 8.12 and 8.13 represent trends seen across all flights of both controllers, where the main differences were the magnitude of the largest oscillations. Both controllers appear to have similar force commands, fairly smooth and some slow oscillation around 50-70 seconds when the omnicopter is moving from point 5 to 7. The aBack controller tended to command more oscillatory torques than the PI/FL controller. As with the force commands for both controllers, torque commands become more oscillatory around 50-70 seconds. These command oscillations manifest as rapid motor direction changes. It is understandable that the omnicopter would struggle the most in this portion of the trajectory because it requires the largest attitude angles.

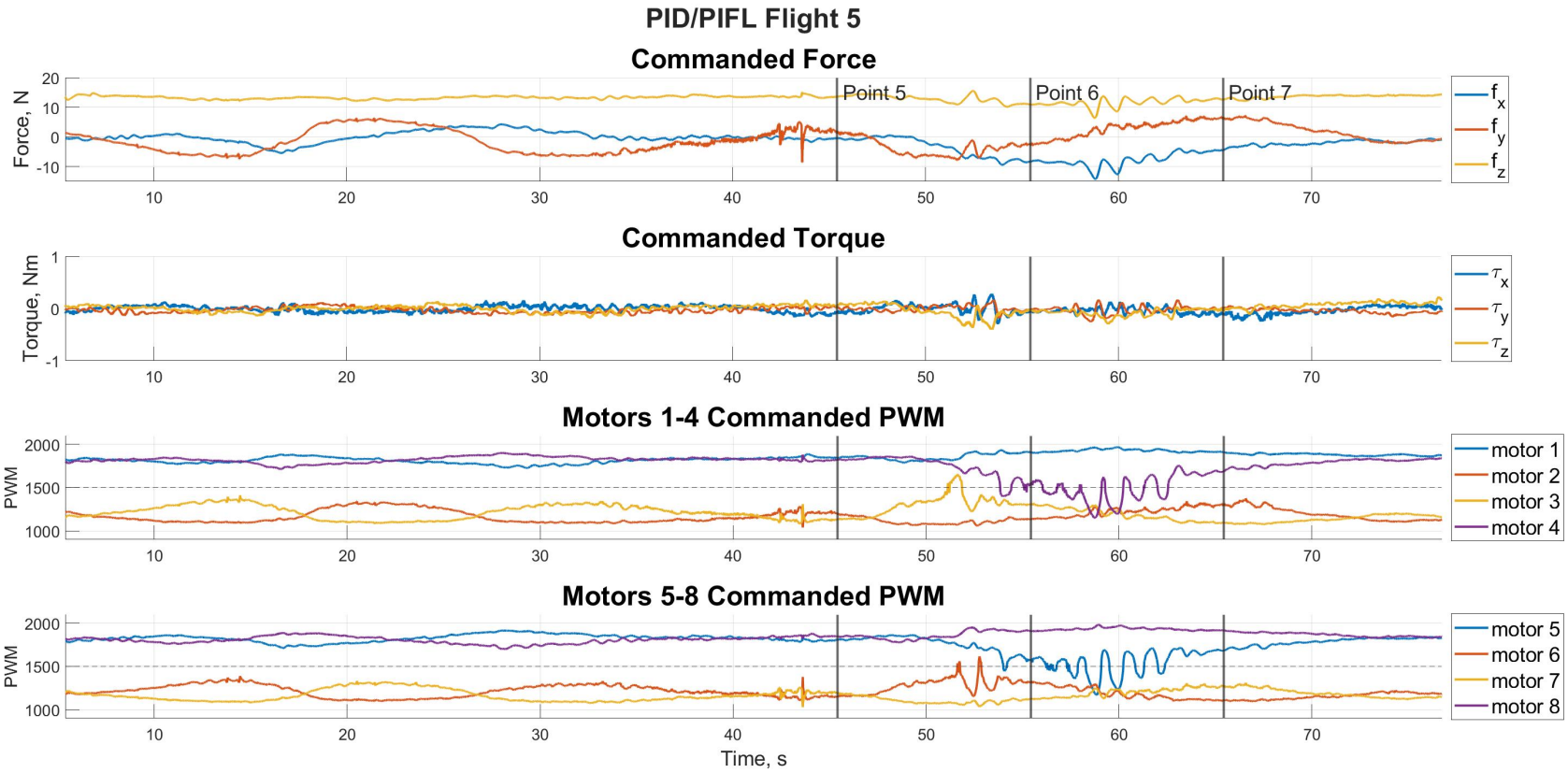


Figure 8.12: PID/PIFL force, torque, and motor commands for smooth trajectory

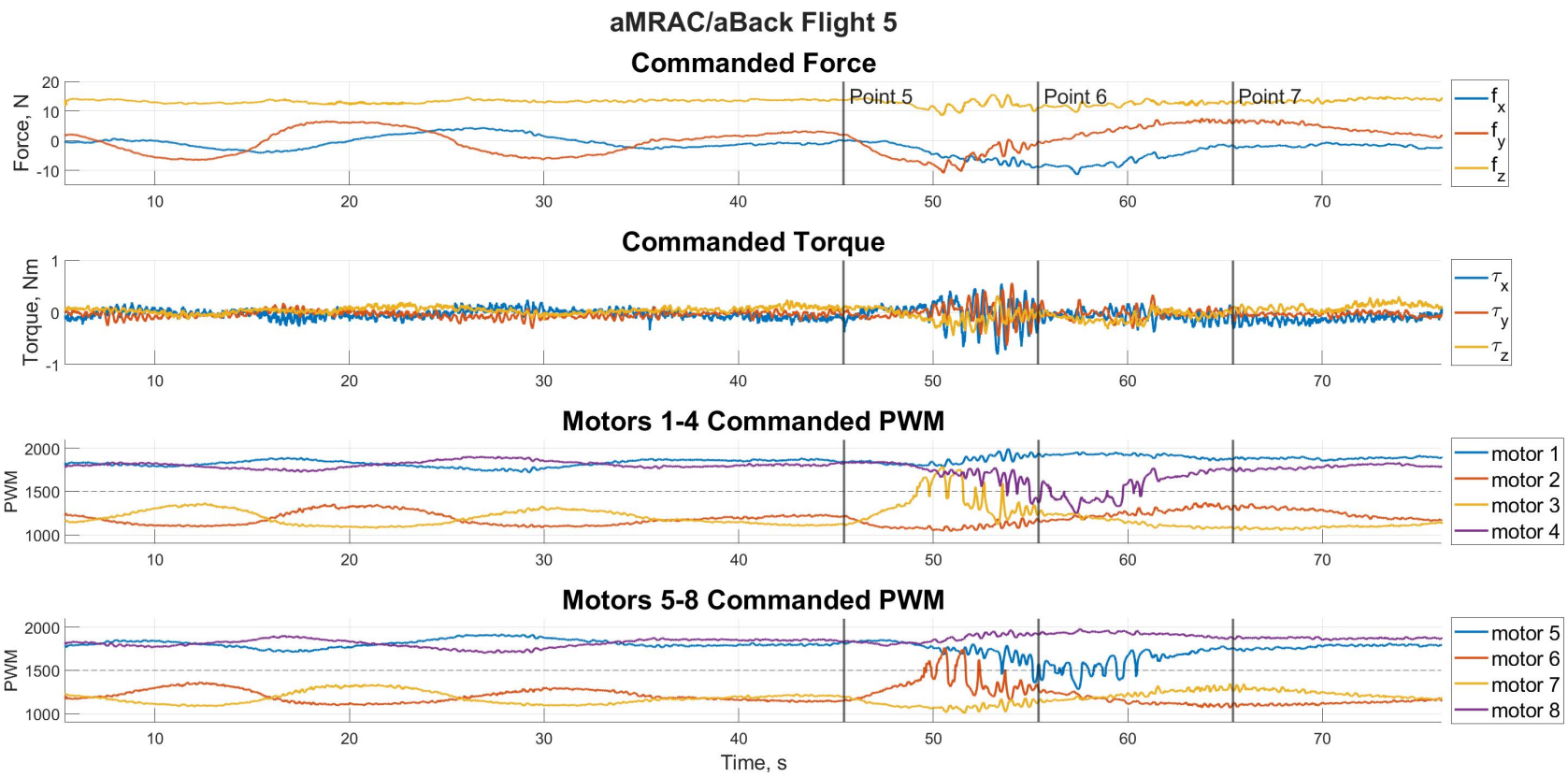


Figure 8.13: aMRAC/aBack force, torque, and motor commands for smooth trajectory

For the aMRAC and aBack controllers, the largest increases in the adaptive parameter norms, shown in Figures 8.14 and 8.15, occur for columns 1 and 2 when the motors are changing directions. This is to be expected because this is the location in the trajectory that is forcing the omnicopter to perform at its limits. For aMRAC, the column 3 norm is larger than than the other two column norms and tends to grow slowly throughout the trajectory. This trend was also witnessed in the simulation results for both the smooth and jagged trajectories.

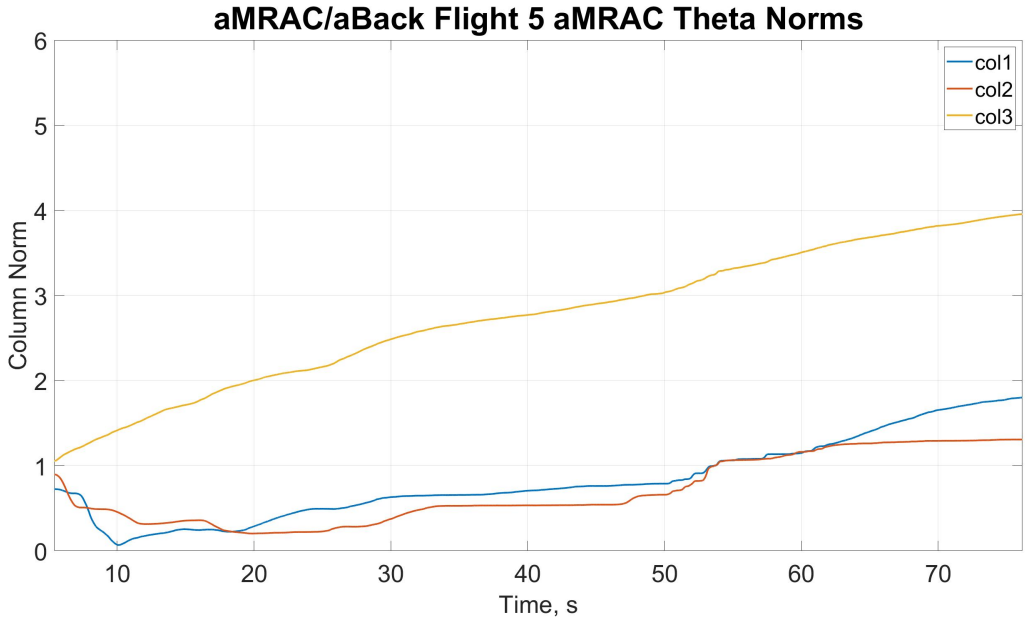


Figure 8.14: aMRAC adaptive parameter column norms for smooth trajectory

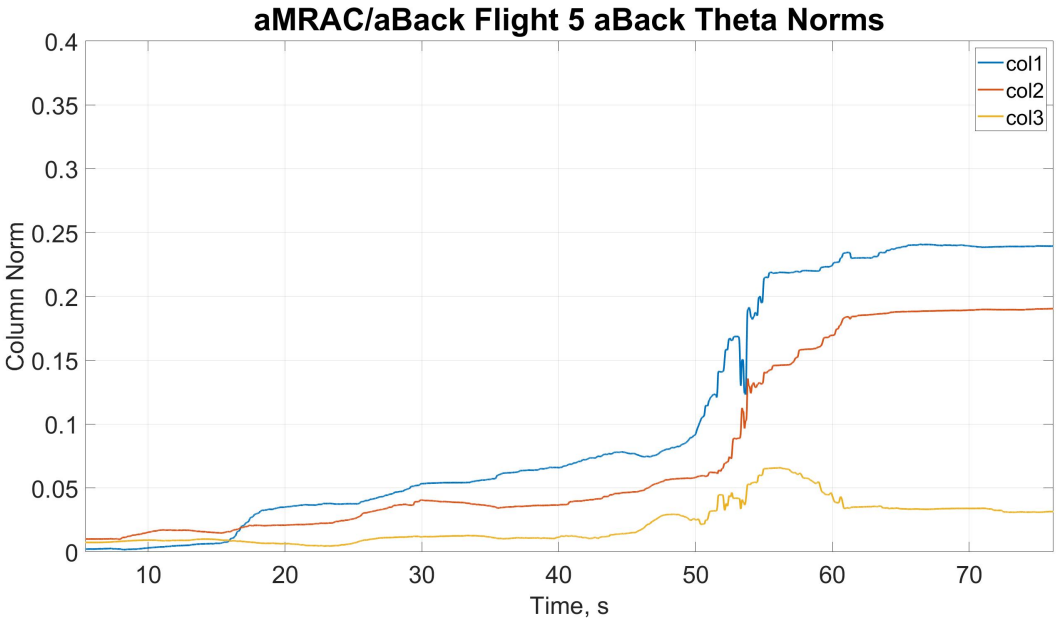


Figure 8.15: aBack adaptive parameter column norms for smooth trajectory

8.4.3 Hover Tests with Added Weight

The omnicopter with 9 inch blades was flown for hover tests using an added weight. The code used the $\boldsymbol{T}$ matrix that was calculated for the omnicopter with the 8 inch blades (Eq. (8.1)) as well as the motor coefficient of thrust for the 8 inch blades (OMNI_TAU_F = 0.0371 in the code) that was used to compute the appropriate motor command. Since the differences between the two omnicopters are slight, this was deemed to not be a significant issue, especially for hover. After the omnicopter had achieved a stable hover, a 47 gram weight was hung 38 centimeters from the center of the omnicopter, angled over the top right corner (see Figure 8.16). The weight produced approximately $0.12Nm$ of torque about each x and y body axis. The omnicopter hovered for 20 seconds with the weight attached before landing.

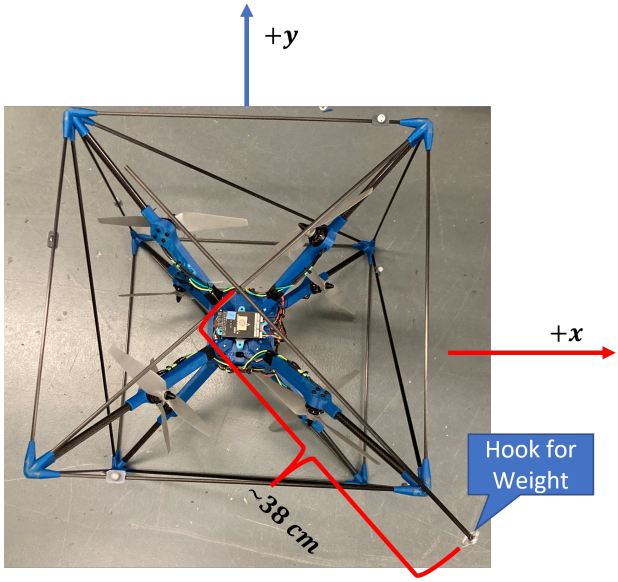


Figure 8.16: Diagram of omnicopter with 9 inch blades and attached hook

$$\boldsymbol{T}_{8"} = \begin{bmatrix} -0.7887 & 0.2113 & -0.2113 & 0.7887 & 0.7887 & -0.2113 & 0.2113 & -0.7887 \\ 0.2113 & 0.7887 & -0.7887 & -0.2113 & -0.2113 & -0.7887 & 0.7887 & 0.2113 \\ 0.5774 & -0.5774 & -0.5774 & 0.5774 & 0.5774 & -0.5774 & -0.5774 & 0.5774 \\ 0.0254 & -0.1386 & 0.1386 & -0.0254 & 0.0254 & -0.1386 & 0.1386 & -0.0254 \\ -0.1386 & -0.0254 & 0.0254 & 0.1386 & -0.1386 & -0.0254 & 0.0254 & 0.1386 \\ 0.1131 & -0.1131 & -0.1131 & 0.1131 & -0.1131 & 0.1131 & 0.1131 & -0.1131 \end{bmatrix} \tag{8.1}$$

Masses of 47 g ($\sim$ 3% omnicopter mass), 106 g ($\sim$ 7.5%), and 152 g ($\sim$ 11%) had been tried. The aMRAC/aBack controller was able to maintain steady flight with the 47 g and 106 g weights, albeit canted at an angle. The PID/PIFL controller could only maintain a hover with the 47 g weight, slowly oscillating to try to achieve a level attitude during the 20 seconds allotted. Therefore, the 47 g weight was chosen for the flight tests. Three flights were accomplished per controller.

Table 8.6: Hover Flight Test Position Errors

Flight	Battery Rating (mAh)	Battery Weight (g)	PID/PIFL Error Mean (m)	PID/PIFL Error σ (m)	PID/PIFL Error Max (m)	aMRAC/aBack Error Mean (m)	aMRAC/aBack Error σ (m)	aMRAC/aBack Error Max (m)
1	1800	198	.0516	.0257	.1077	.0261	.0297	.1528
2	1800	198	.0572	.0213	.1068	.0193	.0179	.0910
3	1800	198	.0437	.0248	.1045	.0265	.0175	.0881
	Average		.0508	.0239	.1063	.0240	.0217	.1106

Table 8.7: Hover Flight Test Attitude Errors

Flight	Battery Rating (mAh)	Battery Weight (g)	PID/PIFL Error Mean (deg)	PID/PIFL Error σ (deg)	PID/PIFL Error Max (deg)	aMRAC/aBack Error Mean (deg)	aMRAC/aBack Error σ (deg)	aMRAC/aBack Error Max (deg)
1	1800	198	9.86	5.33	24.27	19.95	2.06	24.44
2	1800	198	9.55	4.29	19.46	23.77	1.69	26.50
3	1800	198	6.69	4.61	21.29	24.16	1.74	27.13
	Average		8.70	4.74	21.71	22.63	1.83	26.03

The error computations from Section 7.1 are used to calculate the average position and attitude errors shown in Tables 8.6 and 8.7 for the 20 seconds of hover with added weight. aMRAC/aBack data was recorded at 100 Hz and PID/PIFL data was recorded at 200 Hz.

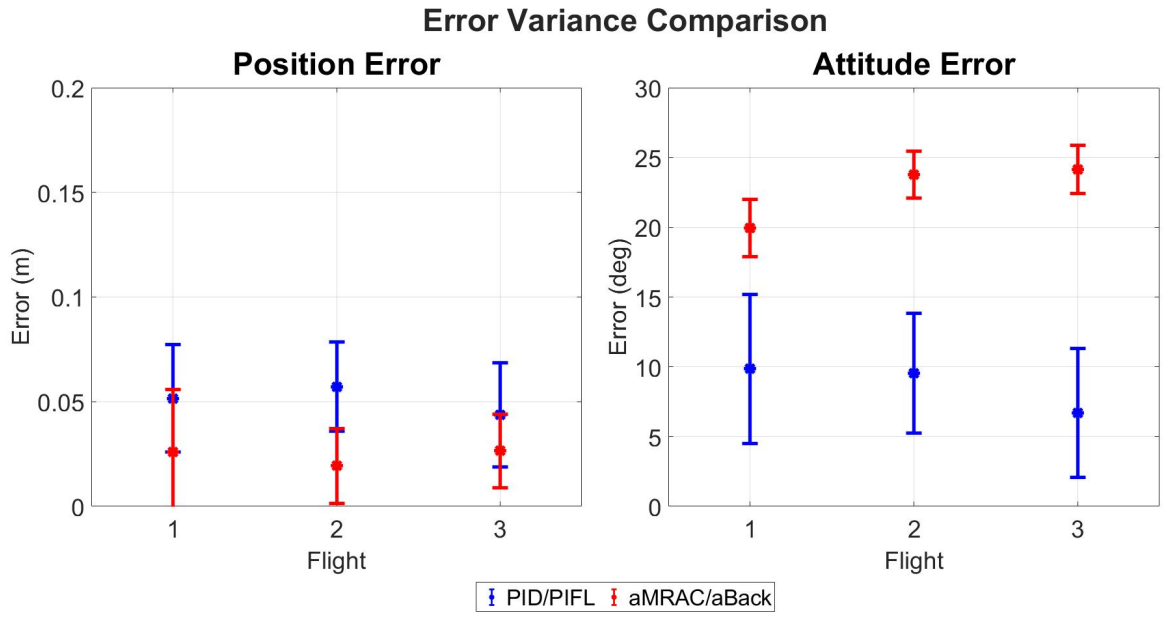


Figure 8.17: Controller error variance comparison for hover with added weight. Dots represent mean error, and error bars represent one standard deviation above and below the mean error.

The aMRAC/aBack controller was able to achieve smaller average position errors than the PID/PIFL controller by about .027 m. For attitude, the PID/PIFL controller was able to achieve smaller average attitude errors by almost 14 degrees. However, the price of small attitude errors was a constant slow oscillation as the omnicopter tried to achieve a level attitude. This oscillation can be seen in Figure 8.18 as the attitude error grows and decreases for PID/PIFL, whereas in Figure 8.19 the aMRAC/aBack attitude error is relatively constant. Correspondingly, the attitude error bars in Figure 8.17 are much smaller for the aMRAC/aBack controller than the PID/PIFL controller. Additionally, in Table 8.7 the difference between the max and mean attitude error is much larger for PID/PIFL than aMRAC/aBack (13 degrees versus 3.4 degrees). The average standard deviations for the aMRAC/aBack controller are also smaller than those for PID/PIFL by 0.0022 m and 2.91 deg, indicating that there is less variation in error throughout the flight for the adaptive

controller. Note that in Figures 8.18 and 8.19, all the errors have been shifted to start at 0 seconds to make them easier to compare. Therefore the times in these Figures do not correlate with the times shown for the full flight in the subsequent Figures for this Section.

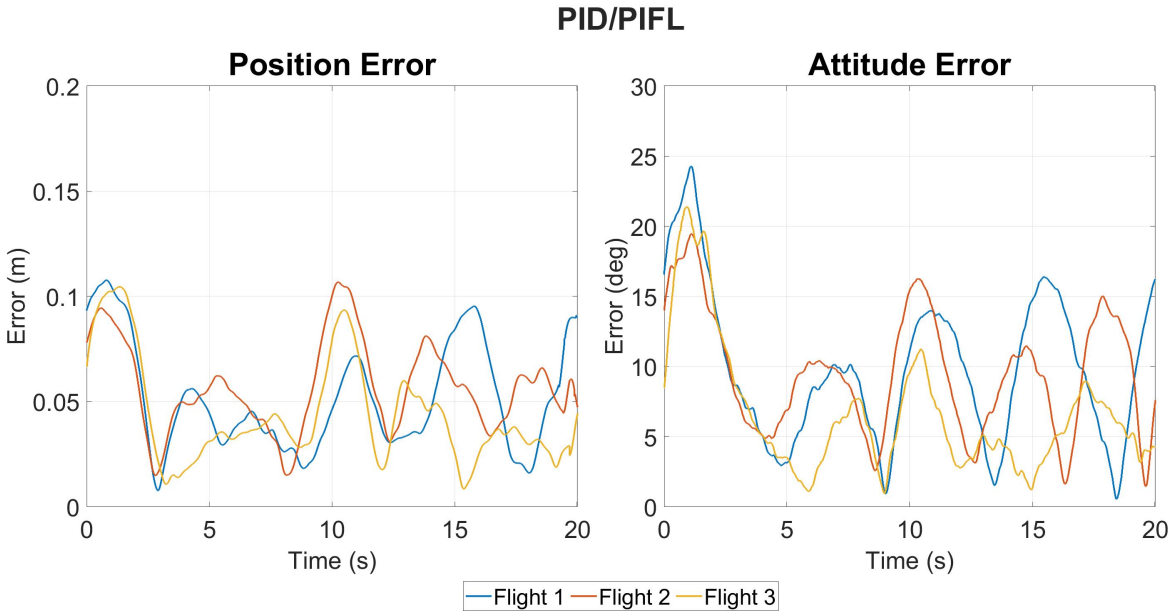


Figure 8.18: PID/PIFL position and attitude error for hover with added weight

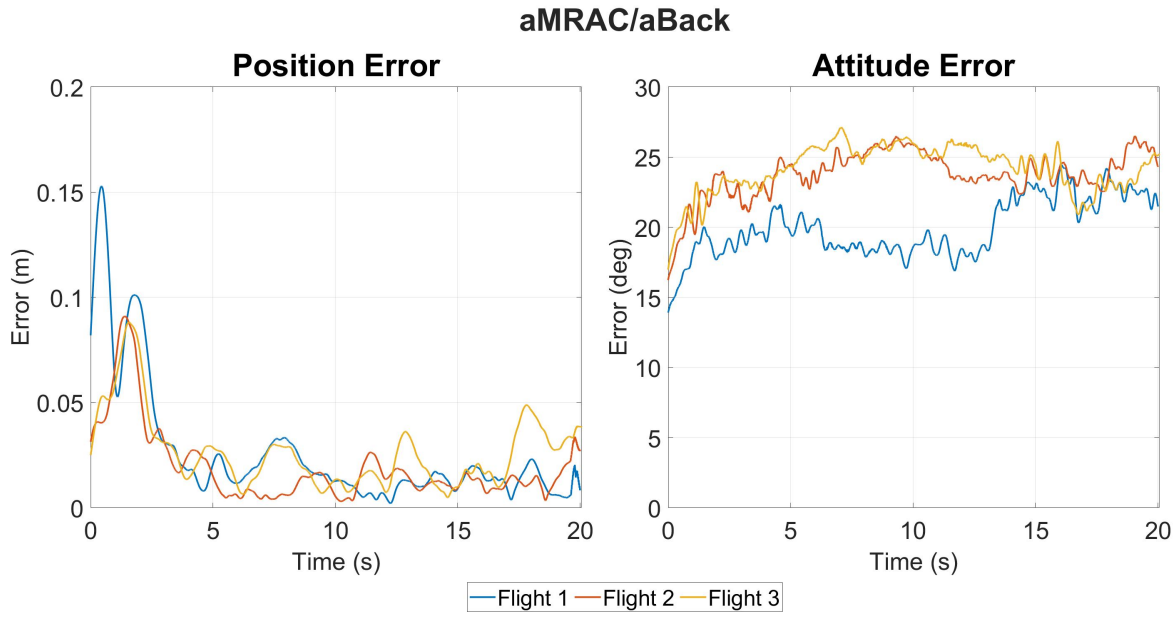


Figure 8.19: aMRAC/aBack position and attitude error for hover with added weight

Ideally, an attitude controller would allow the omnicopter to achieve level flight without prolonged oscillation. Neither controller exhibits these characteristics, however, the performance of the aBack controller is preferable since it is more stable. The aBack controller did not appear to make any significant effort to return to level flight. This may be due to tuning that emphasized angular velocity error instead of quaternion error. Changing the baseline controller gains as well as the adaptive gains may yield different performance. The adaptive gains may be particularly useful. In Figure 8.20, it is obvious when the mass has been added to the vehicle (between 10 and 15 seconds) because there is in a rapid increase in the column 1 norm and a rapid decrease of the column 2 norm of the aMRAC adaptive parameter. However, in Figure 8.21, only slight increases in the column 1 and 2 norms for the aBack adaptive parameter can be seen, representing roll and pitch. This Figure is not representative of the other two flights, which both showed a rapid decrease of the column 2 norm when the weight was added. The column two norm for those flights also started at a

values between 3-4, higher than the projection operator should have allowed. It is possible that the use of the "wrong" $\boldsymbol{T}$ matrix for the vehicle may have caused the aBack theta values to be higher than expected. Note that these figures show data for the entire flight to display the change caused by adding weight, but only the portion of hover with added mass was used for error calculations.

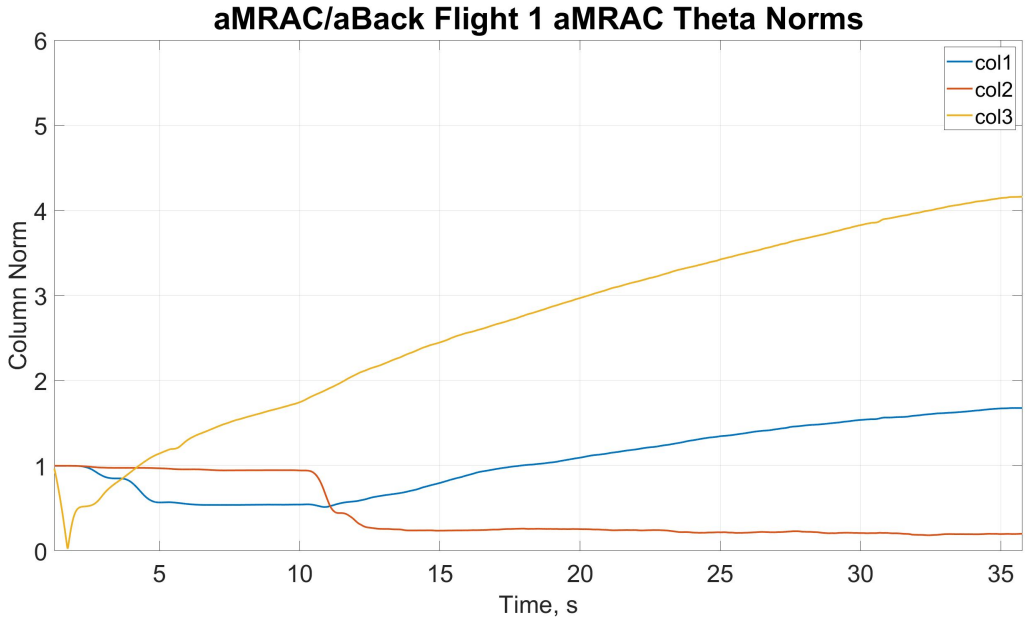


Figure 8.20: aMRAC adaptive parameter column norms for hover with added weight

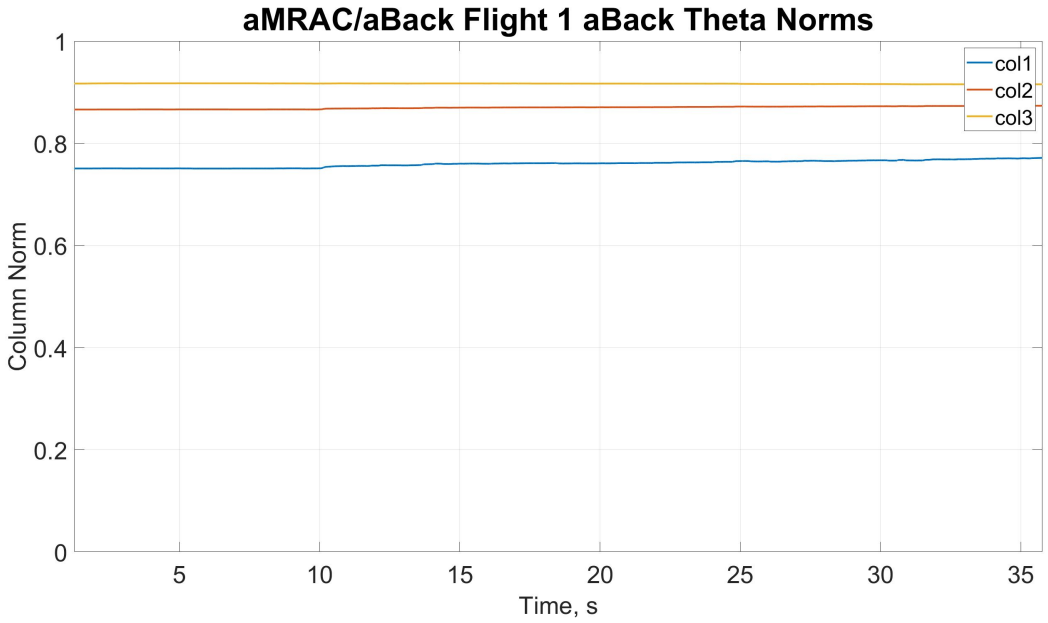


Figure 8.21: aBack adaptive parameter column norms for hover with added weight

Figures 8.22 and 8.23 show data for the entire flight to display the changes that occur in the force, torque, and motor commands when the weight is added to the omnicopter, around 10 seconds. These figures are representative of the other flights for each controller. For both controllers, the addition of the weight caused an immediate increase in magnitude for the commanded force and torque for the x and y components. The location of the weight would impart roll and pitch in the positive direction, so negative commanded torques are expected and exhibited in the figures. Little noticeable change to the z components of force and torque are expected because the weight imparts no yaw and only requires about 0.5 N of extra thrust to maintain a hover. The changes in force and torque command also cause a change in motor PWM command, as expected. While the commanded forces, torques, and motor PWM are roughly the same magnitude for both controllers, the shape of the commands reflects the shape of the attitude error for each controller. The PID/PIFL controller exhibits slow oscillations in the x and y components of the force and torque commands. These oscillations

appear in the motor PWM. These motor commands caused the omnicopter to achieve a level attitude but continue to go past it and try to correct again the other way, causing the oscillations in the attitude tracking error. The aMRAC/aBack controller, however, maintains relatively static force and torque commands in the x and y components. The force commands are steady while the torque commands oscillate rapidly about a steady value because it is attempting to maintain zero angular velocity while subject to the constant perturbation from the added weight. The steadiness in force and torque is also reflected in the motor commands and is why the attitude error maintained a constant large error with the aBack controller.

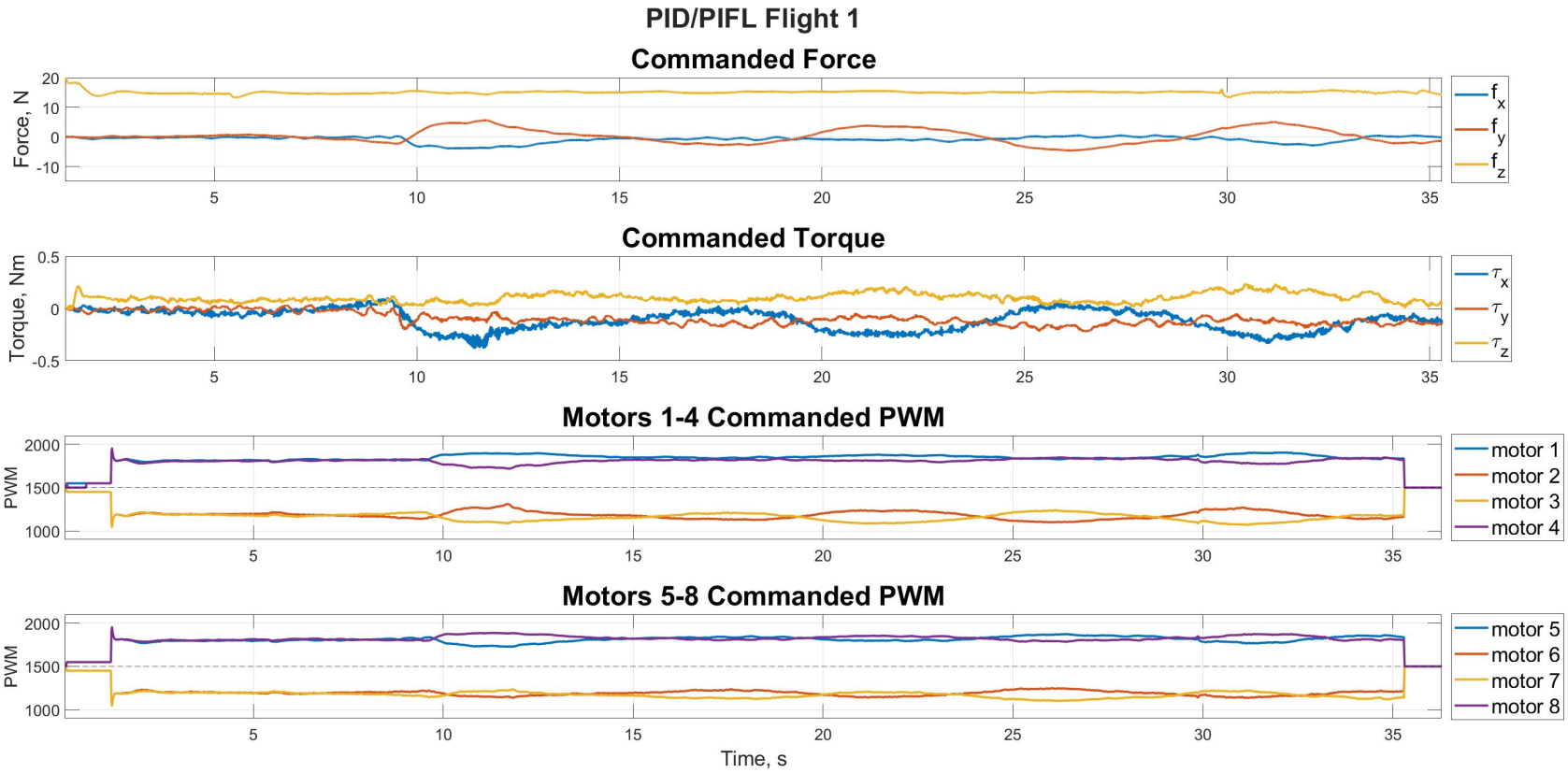


Figure 8.22: PID/PIFL force, torque, and motor commands for hovering with added weight

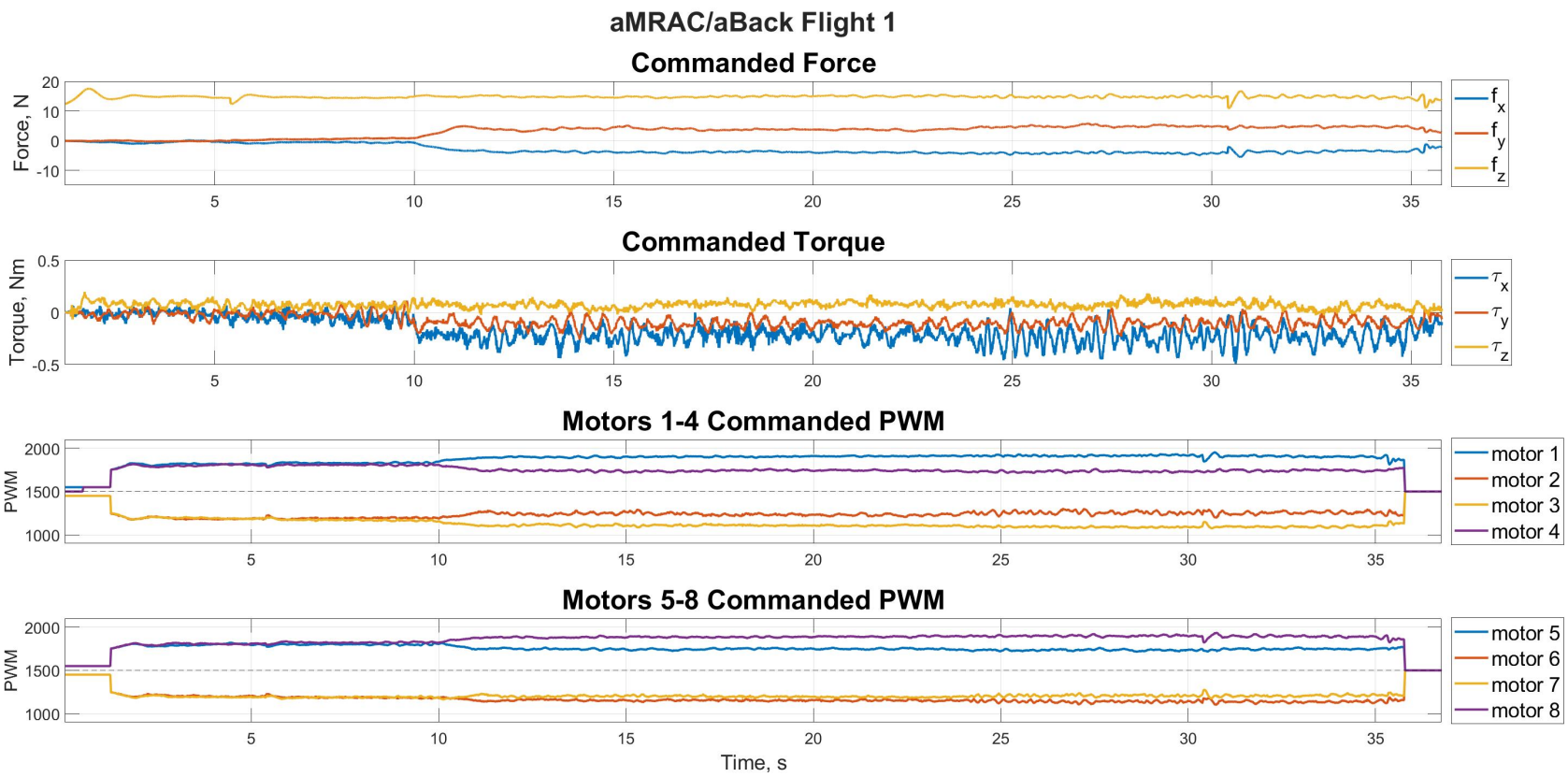


Figure 8.23: aMRAC/aBack force, torque, and motor commands for hovering with added weight

One flight was performed for each controller with the appropriate $\boldsymbol{T}$ matrix (Eq. (8.2)) and OMNI_TAU_F parameter (OMNI_TAU_F = 0.0349). The main changes to $\boldsymbol{T}$ are in the bottom 3 rows, which correlate to torque commands and are dependent on the rotor distance from the center of mass and rotor parameters. Note that the battery used was not recorded but was most likely the 1800 mAh battery, and the flights lasted about 40 seconds with the added weight instead of 20 seconds. Only analysis for the first 20 seconds of flight are presented in Figures 8.24 and 8.25 and Table 8.8 to be as consistent as possible with the previous flight data.

Although one test flight is not enough to draw definitive conclusions, it can be used to preview trends that would be expected. Both controllers saw a decrease in mean, standard deviation, and max position and attitude errors compared to the results in Tables 8.6 and 8.7. However, the improvement was greater for the PID/PIFL controller, showing that it is more sensitive to the use of parameters that accurately describe the system. The PID/PIFL mean, standard deviation, and max position errors improved by about 0.03 m, 0.01 m, and 0.02 m, respectively, whereas they only improved by about 0.005 m, 0.009 m, and 0.03 m for aMRAC/aBack. The PID/PIFL mean, standard deviation, and max attitude errors improved by about 4 degrees, 1.76 degrees, and 3 degrees, respectively, whereas they only improved by about 2 degrees, 0.49 degrees, and 2 degrees for aMRAC/aBack. Additionally, when comparing Figure 8.24 to Figure 8.18, the nonadaptive controller error traces do not exhibit the slow oscillations that were present with the "wrong" parameters. However, when comparing Figure 8.25 to Figure 8.19, the adaptive controller error traces look very similar to those from the "wrong" parameter test flights. This qualitative analysis supports the numerical analysis showing that the nonadaptive controller is more dependent on the use of accurate system parameters to achieve good performance.

$$\boldsymbol{T}_{9^{\prime\prime}} = \begin{bmatrix} -0.7887 & 0.2113 & -0.2113 & 0.7887 & 0.7887 & -0.2113 & 0.2113 & -0.7887 \\ 0.2113 & 0.7887 & -0.7887 & -0.2113 & -0.2113 & -0.7887 & 0.7887 & 0.2113 \\ 0.5774 & -0.5774 & -0.5774 & 0.5774 & 0.5774 & -0.5774 & -0.5774 & 0.5774 \\ 0.0252 & -0.1458 & 0.1458 & -0.0252 & 0.0252 & -0.1458 & 0.1458 & -0.0252 \\ -0.1458 & -0.0252 & 0.0252 & 0.1458 & -0.1458 & -0.0252 & 0.0252 & 0.1458 \\ 0.1207 & -0.1207 & -0.1207 & 0.1207 & -0.1207 & 0.1207 & 0.1207 & -0.1207 \end{bmatrix} \tag{8.2}$$

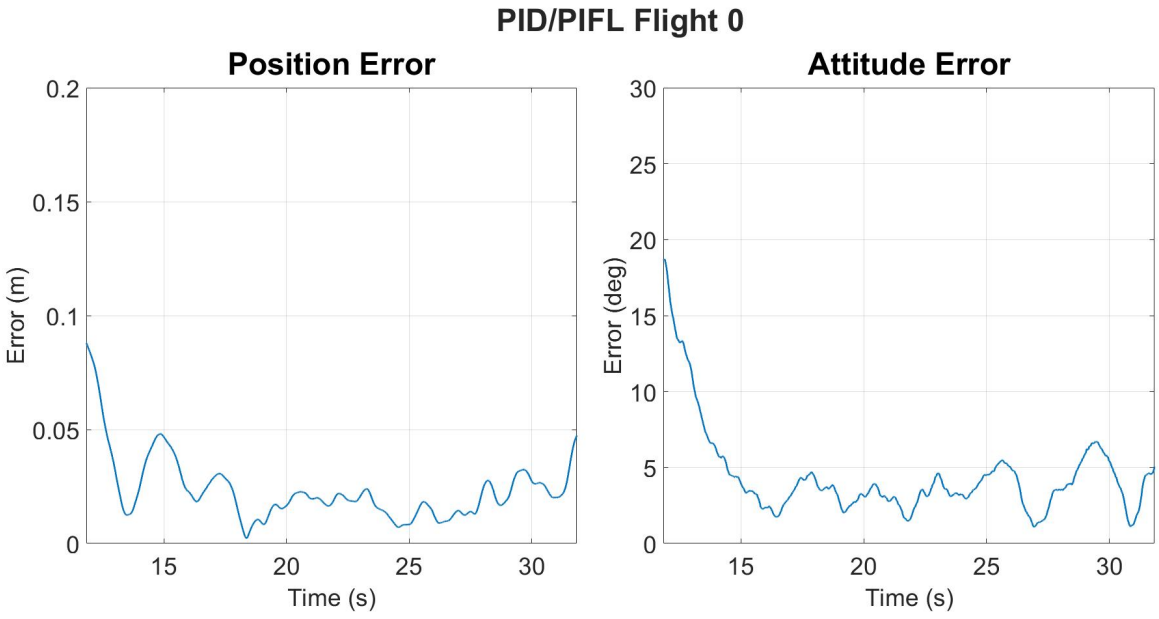


Figure 8.24: PID/PIFL position and attitude error for hover with added weight and corrected parameters

Table 8.8: Hover Flight Test Attitude Errors

	PID/PIFL Error			aMRAC/aBack Error		
Tracking	**Mean**	$\boldsymbol{\sigma}$	**Max**	**Mean**	$\boldsymbol{\sigma}$	**Max**
Position (m)	.0233	.0141	.0880	.0190	.0127	.0795
Attitude (deg)	4.44	2.98	18.69	20.67	1.34	23.96

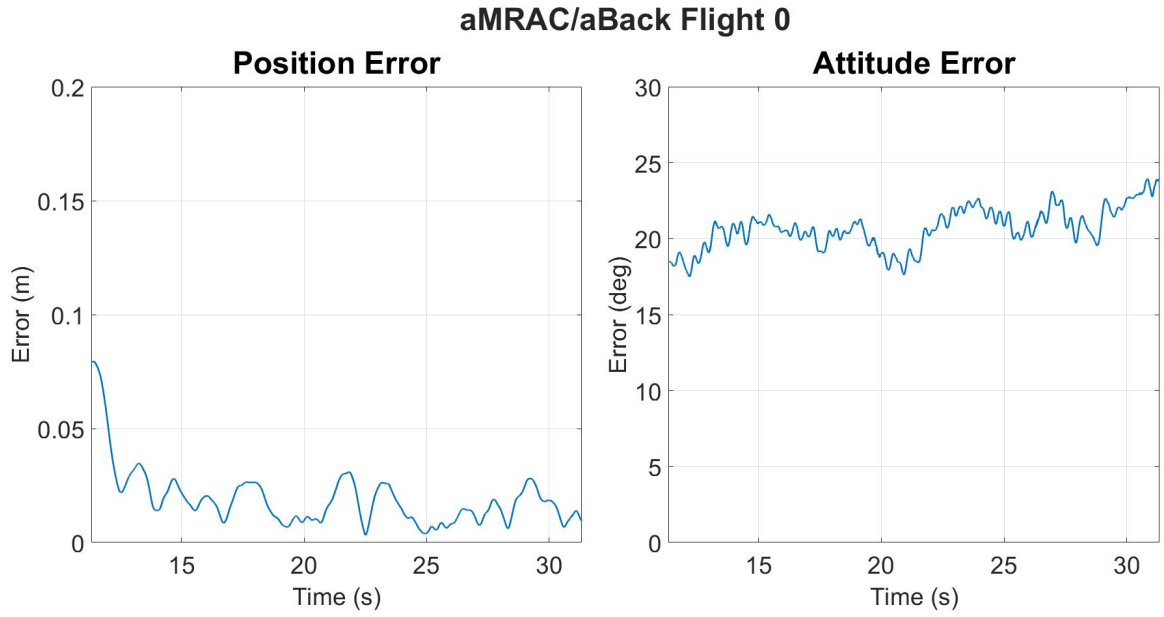


Figure 8.25: aMRAC/aBack position and attitude error for hover with added weight and corrected parameters

The aMRAC adaptive parameter norms for the flight with the correct parameters looked very similar to Figure 8.20 and are therefore not included here. The aBack parameter norms for the flight with the correct parameters, however, looked like what was expected for the flights with the wrong parameters. As seen in Figure 8.26, all the values are much smaller than one, staying below 0.1, and start from about the expected initial values. The norms stay relatively constant until the weight has been added to the system, around 10 seconds, which is when the norms exhibit a sharp increase before continuing to steadily increase. Columns one and two have the highest norms because the weight primarily affects roll and

pitch.

The entire flight is shown in Figure 8.26 instead of just the 20 seconds used for flight analysis. Interestingly, the slope of the norm increase slows around 35 seconds (about 25 seconds after adding the weight), possibly indicating that this is when steady state has fully been achieved. If the error traces for the PID/PIFL controller are plotted for the entire 40 second flight time, around 30-35 seconds into flight (20-25 seconds after adding the weight), slow oscillations begin to occur with larger amplitude. These steady state oscillations indicate the nonadaptive controller may be more unstable for long term flight than the adaptive controller, which shows a slight increase in error after 35 seconds but no prominent oscillations. While 20 seconds may have been too short a flight time to characterize true transient and steady state responses of the controller, it appears flying for longer would have made the PID/PIFL controller perform even worse than has already been shown.

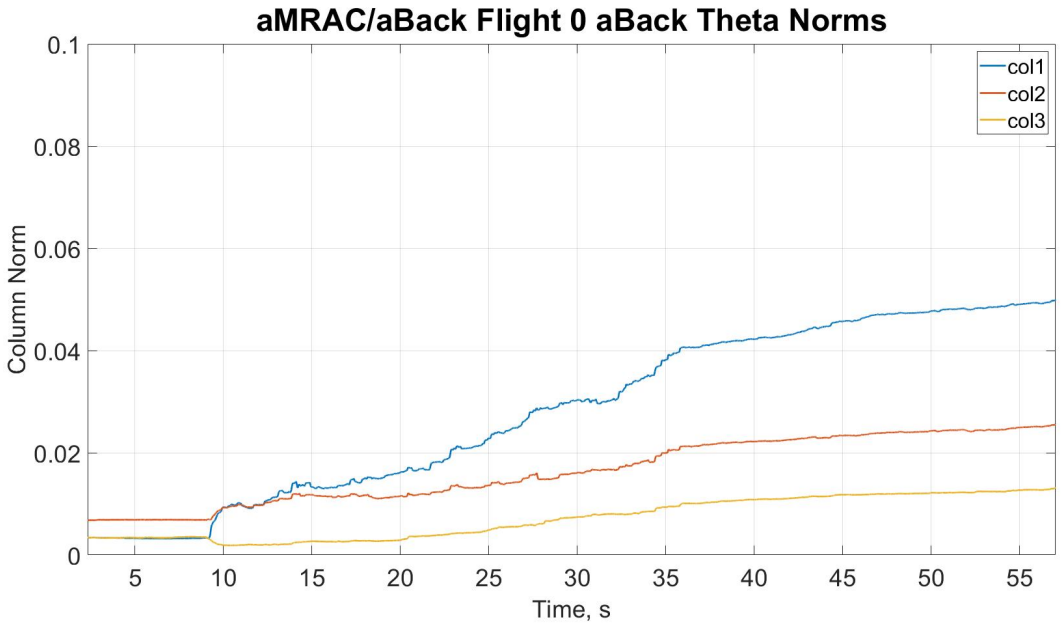


Figure 8.26: aBack adaptive parameter column norms for hover with added weight and correct parameters

8.4.4 Circular Trajectory with Broken Propeller

A broken propeller was used on motor 4 of the omnicopter that uses 8 inch blades, shown in Figure 8.27, as it flew a circular trajectory. The breaks were made to be approximately symmetrical to prevent excessive vibrations that would cause EKF failures that yielded crashes. The broken propeller produced approximately 20% less thrust than a standard propeller. The omnicopter was commanded to fly a circular trajectory at 0.5 m/s facing its +x face towards the center, denoted by the red dot in Figures 8.31 and 8.32, throughout almost the entire circle. The trajectory had an intro and exit portion to smooth attitude and position commands between hover and rapid motion. However, for consistency within the data analysis for both controllers, the tracking error during the transition periods is not included. Analysis that was done that included the transition period showed decreases in position and attitude tracking error for both controllers. This makes sense because tracking is generally better during slower and longer motion, which is what the transitions are, helping counteract the poorer tracking in the faster circular trajectory. Regardless of the inclusion of the transition, the trends that will be discussed do not change.

The error computations from Section 7.1 are used to calculate the average position and attitude errors shown in Tables 8.9 and 8.10 for the approximately 13 seconds of circling. The uORB topic data was recorded at 100 Hz for both controllers. There are no PID/PIFL results for some flights because when running this controller combination, the omnicopter was prone to crashing. No changes to the controllers or omnicopter parameters were made between flight tests, so the recurring crashes indicate that this trajectory pushes the PID/PIFL controller to the edge of stability. Seven flights were attempted for each of the controllers.

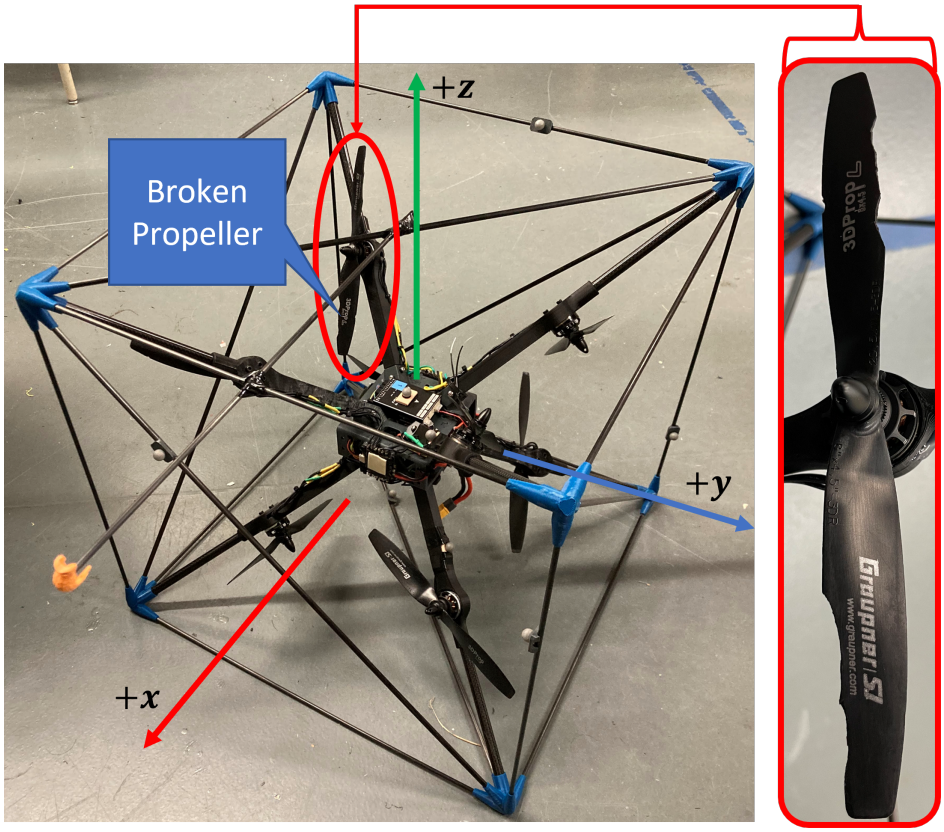


Figure 8.27: Diagram of omnicopter with 8 inch blades and broken propeller

Table 8.9: Circular Flight Test Position Errors

	Battery		PID/PIFL Error			aMRAC/aBack Error		
Flight	**Rating (mAh)**	**Weight (g)**	**Mean (m)**	**σ (m)**	**Max (m)**	**Mean (m)**	**σ (m)**	**Max (m)**
1	1550	177	.0986	.0549	.3031	.0486	.0185	.0892
2	1550	177	-	-	-	.0709	.0273	.1307
3	1550	177	-	-	-	.0741	.0344	.1602
4	1800	198	.0660	.0191	.0987	.0698	.0250	.1260
5	1800	198	.1164	.0951	.5150	.0716	.0242	.1106
6	1800	198	.1063	.0459	.2798	.0934	.0354	.1595
7	1800	198	-	-	-	.0695	.0261	.1165
	Average		.0968	.0538	.2992	.0711	.0273	.1275

Table 8.10: Circular Flight Test Attitude Errors

Flight	Battery Rating (mAh)	Battery Weight (g)	PID/PIFL Error Mean (deg)	PID/PIFL Error σ (deg)	PID/PIFL Error Max (deg)	aMRAC/aBack Error Mean (deg)	aMRAC/aBack Error σ (deg)	aMRAC/aBack Error Max (deg)
1	1550	177	18.70	10.22	49.34	15.78	1.92	19.55
2	1550	177	-	-	-	18.29	3.14	24.06
3	1550	177	-	-	-	18.01	3.78	25.42
4	1800	198	17.39	7.92	35.96	17.84	2.64	22.09
5	1800	198	18.75	9.07	38.14	18.93	4.44	25.12
6	1800	198	20.89	12.52	51.43	19.22	3.47	26.81
7	1800	198	-	-	-	17.38	2.99	22.62
	Average		18.93	9.93	43.72	17.92	3.20	23.67

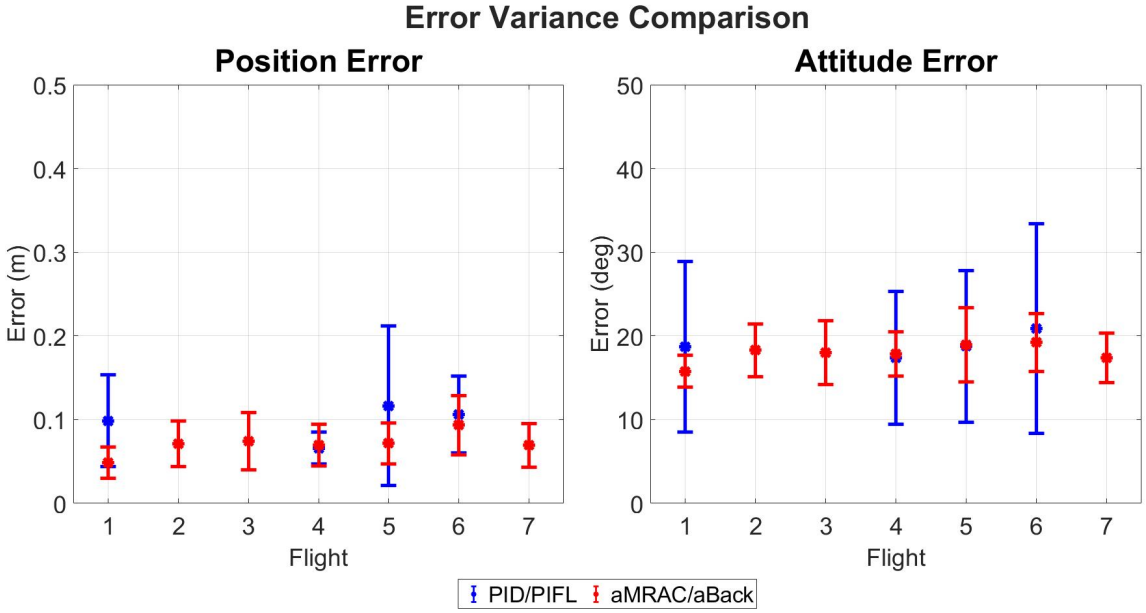


Figure 8.28: Controller error variance comparison for circular trajectory with broken propeller

The aMRAC/aBack controller combination was able to achieve smaller average position and attitude tracking errors than the PID/PIFL controller. Average position tracking was better by about 0.028 m, and average attitude tracking was better by about 1 degree. For both position and attitude tracking, the aMRAC/aBack controller was more consistent throughout

the trajectory than the PID/PIFL controller. The tracking consistency of all the flights can be seen in Figures 8.29 and 8.30. The aMRAC/aBack position and attitude errors have variation but no large changes, staying within ranges of about 0.15 m and 10 degrees. The PID/PIFL controller tends to have a large spike in position and attitude error as it passes through the halfway point of the circle. The variation in error during flights is seen in Figure 8.28 where the error bars for the PID/PIFL controller tend to be larger than the aMRAC/aBack controller for both position and attitude error. The lack of consistency across flights for the PID/PIFL controller is also exhibited by the error bars varying in size from flight to flight more than the aMRAC/aBack error bars.

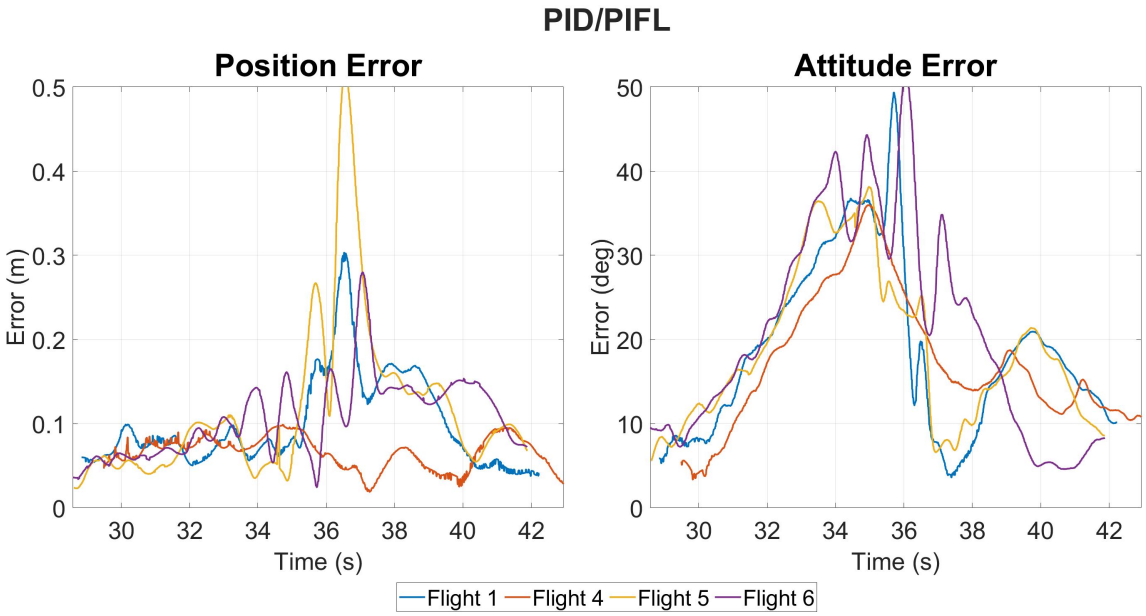


Figure 8.29: PID/PIFL position and attitude error for circular trajectory with broken propeller

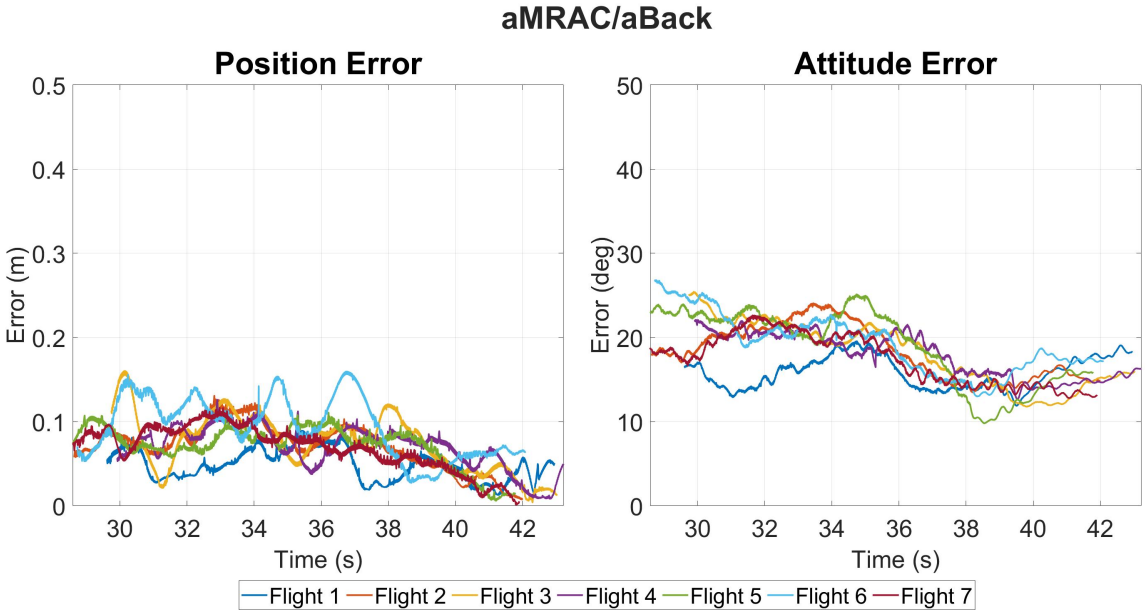


Figure 8.30: aMRAC/aBack position and attitude error for circular trajectory with broken propeller

The loss of tracking control can be seen in Figure 8.31 when the blue line starts to crinkle at the left side of the circle, whereas the aMRAC/aBack controller yields a smoother track in Figure 8.32. Additionally, the averages in Tables 8.9 and 8.10 show that the aMRAC/aBack controller is more consistent than the PID/PIFL controller because the average maximum errors are closer to the average means, 0.0564 m and 5.75 degrees compared to 0.2024 m and 24.79 degrees. Also, the average standard deviations for the aMRAC/aBack position and attitude errors are smaller than the PID/PIFL standard deviations by 0.0265 m and 6.73 degrees.

Figure 8.32 provides a good representation for the position tracking of every flight using the aMRAC/aBack controller combination. This indicates that not only is the aMRAC/aBack controller more consistent than PID/PIFL throughout a trajectory, it also has more consistent performance across different flights, even with battery weight changes. Although two different batteries were used for flight tests, the change in weight does not appear to affect

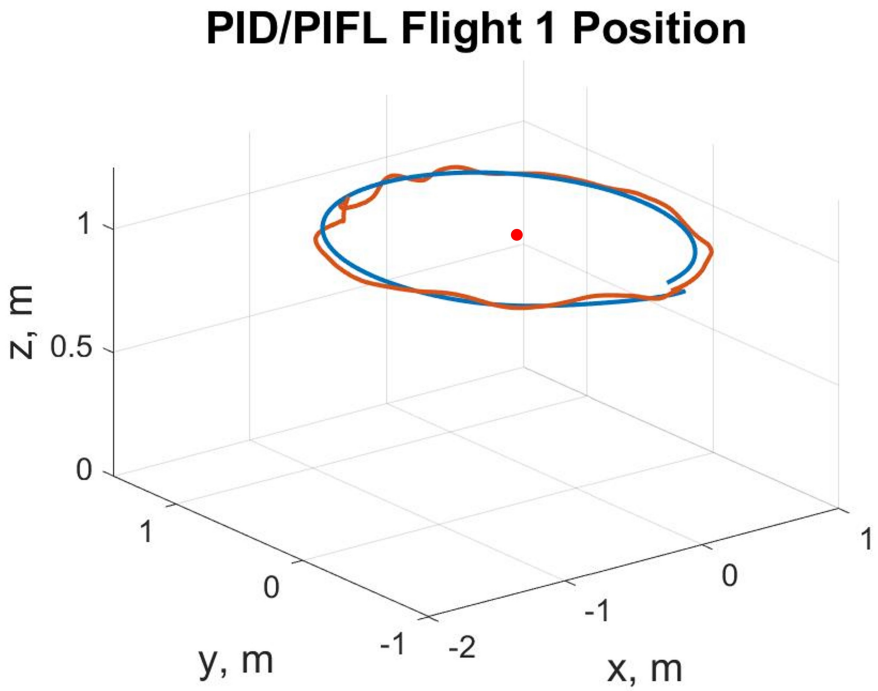


Figure 8.31: PID/PIFL 3D position tracking for circular trajectory with broken propeller

the performance of either controller. Figure 8.31 provides a good indication for where the PID/PIFL controller struggles during the circular trajectory. However, the magnitude of the struggle changes between flights. Flights 5 and 6 exhibited larger deviations/twisting, but Flight 4 had excellent position tracking that is comparable to the aMRAC/aBack controller performance.

Figure 8.34 is representative of the force, torque, and motor commands seen in the other aMRAC/aBack flights. The jump in the yaw torque command around 38 seconds represents when the omnicopter is commanded to face ahead instead of towards the center of the circle. Aside from this anomaly, the commanded torques are small in magnitude, and the commanded forces and motor PWM are fairly steady. In contrast, the PID/PIFL commands shown in Figure 8.33 exhibit large changes in the force, torque, and motor commands around 34-38 seconds into the flight. This timing corresponds to the location where the omnicopter

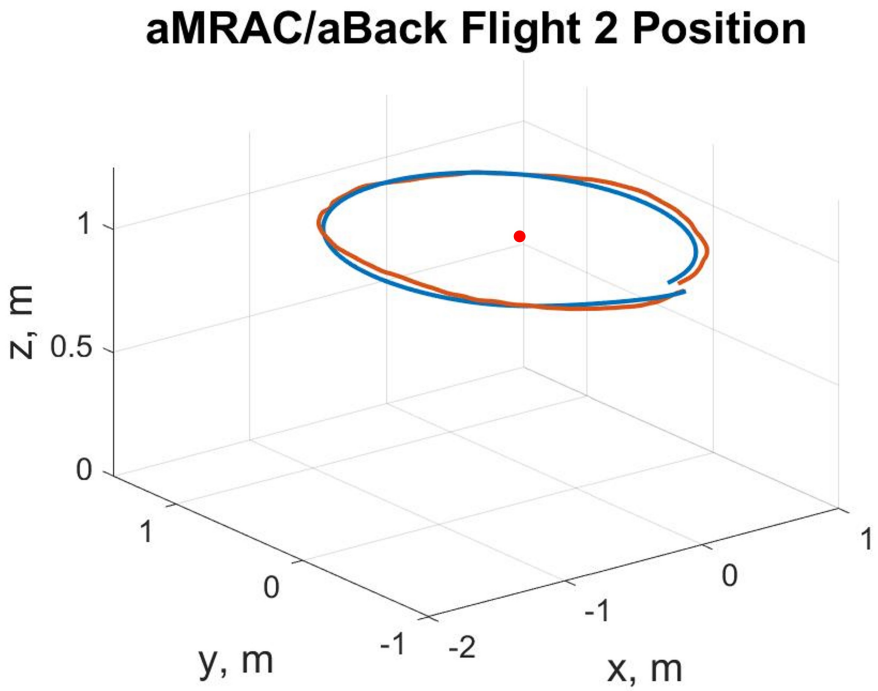


Figure 8.32: aMRAC/aBack 3D position tracking for circular trajectory with broken propeller

begins to struggle to get around the circle. The difficulty may occur because the force and torque commands change just enough to cause a couple of motors to change direction, which negatively impacts tracking accuracy that the omnicopter tries to compensate for. Flights 3 and 4 have an additional two motors that change direction, yielding the worst tracking accuracy. While Flight 1 had the best tracking accuracy, it still commanded one motor to change direction. The adaptive controller appears to have been better at compensating for a broken propeller without commanding excessively high forces and torques.

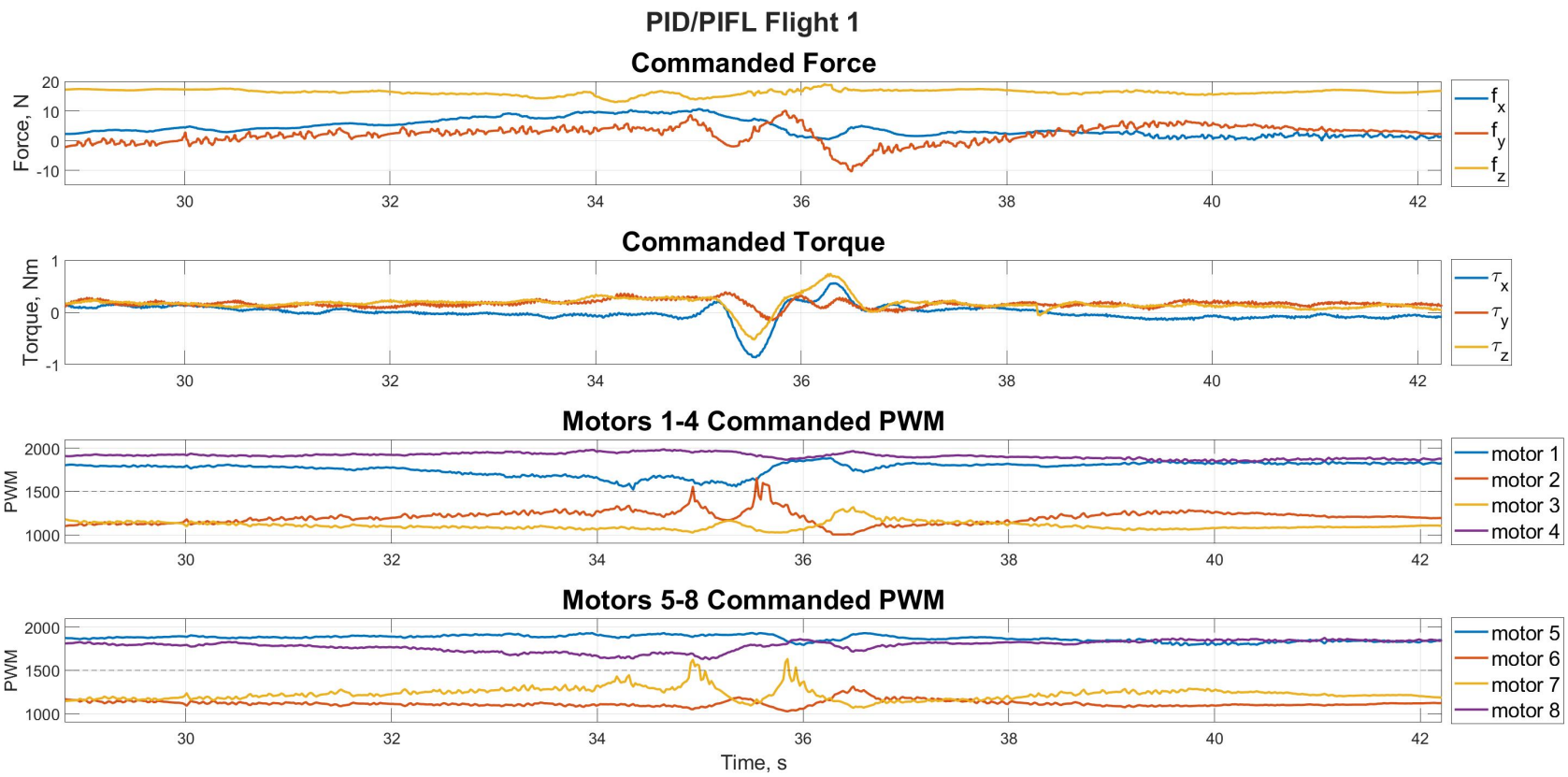


Figure 8.33: PID/PIFL force, torque, and motor commands for circling with a broken propeller

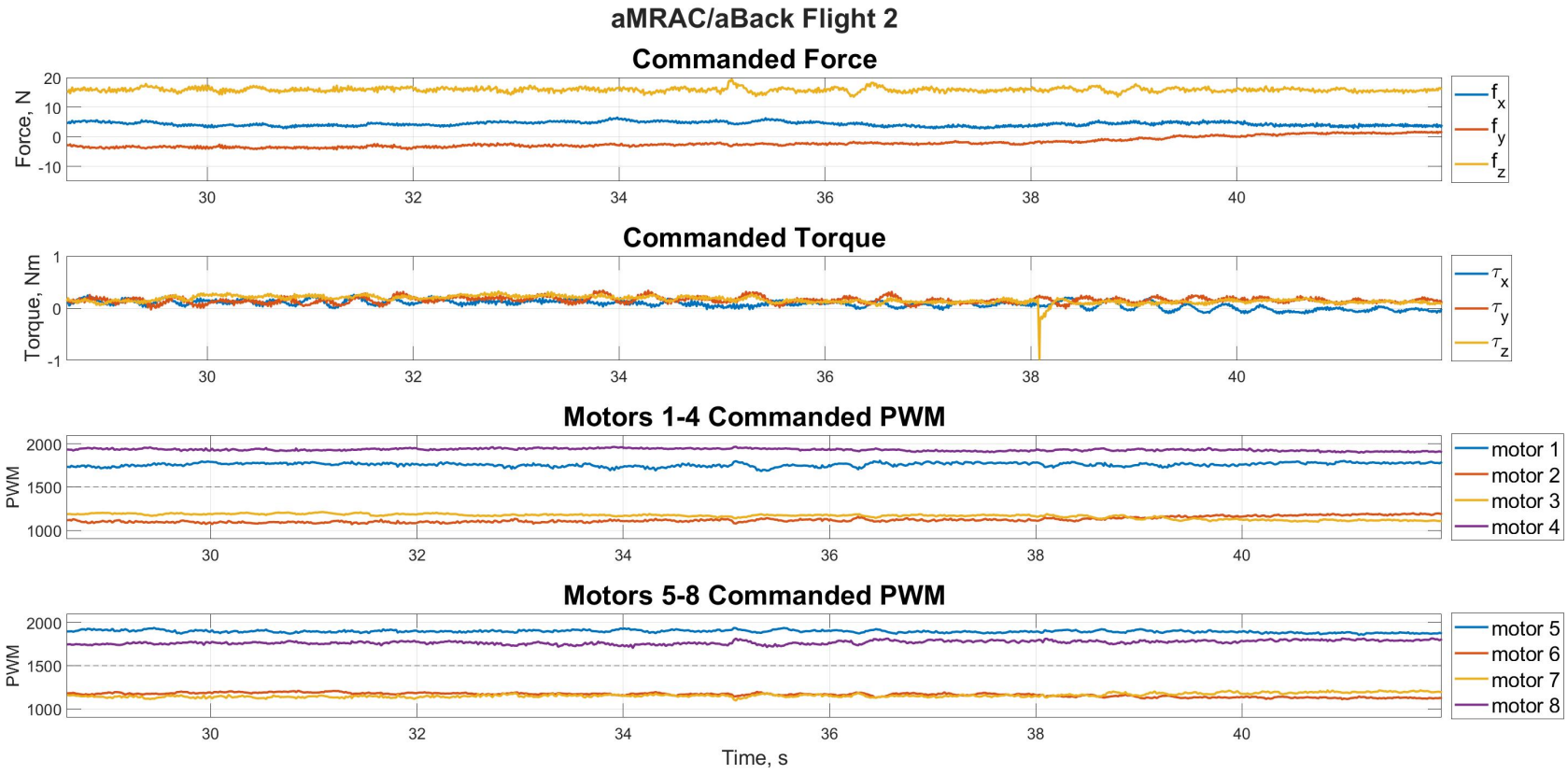


Figure 8.34: aMRAC/aBack force, torque, and motor commands for circling with a broken propeller

It is clear in Figure 8.36 that the adaptive backstepping controller was working to counteract the reduced thrust from the broken propeller, particularly its effects on roll and pitch because the column 1 and 2 norms increase the most throughout the flight. This may have been enough adaptation to maintain stable flight but was not enough to yield the desired level flight. For position control, Figure 8.35 also shows adaptation in the column 1 and 2 norms for the aMRAC adaptive parameter as the omnicopter moves around the circle. The column 3 norm shows the characteristic increase over time and larger magnitude that was seen in simulations as well as the previous flight tests.

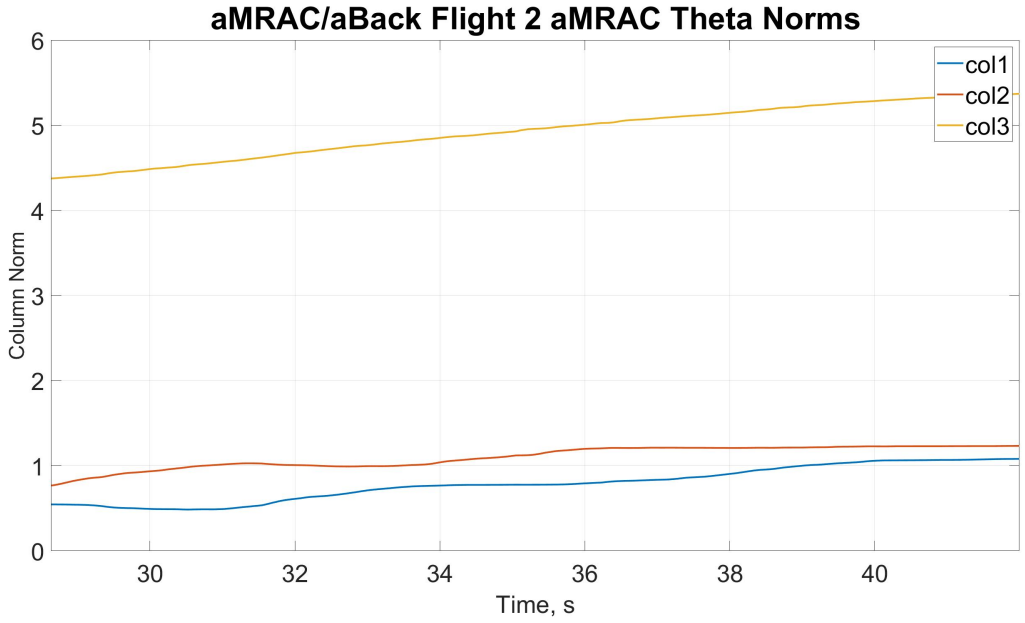


Figure 8.35: aMRAC adaptive parameter column norms for circling with a broken propeller

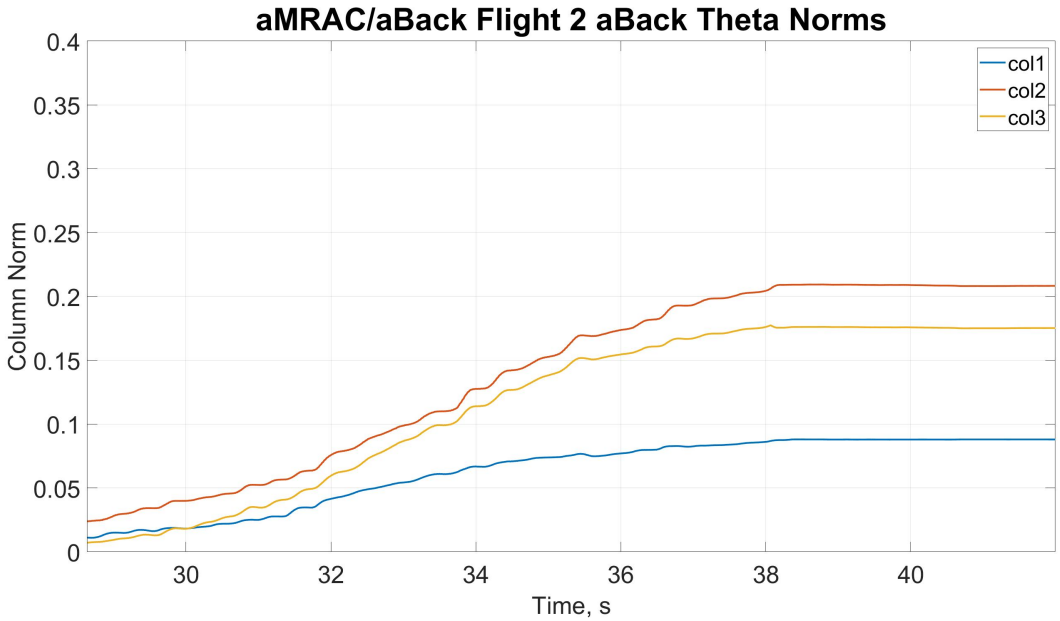


Figure 8.36: aBack adaptive parameter column norms for circling with a broken propeller

Although the trajectory was supposed to occur with zero pitch and roll (level flight, only changes to yaw), neither controller was able to accomplish level flight with the broken propeller. Therefore, the average attitude errors for both controllers are high, in the teens as opposed to being single digits or decimals. However, in the case of a broken propeller, the primary goal of a flight after a propeller break would be to safely land the vehicle. The aMRAC/aBack controller was able to maintain controlled flight much better than the PID/PIFL controller, having a higher proportion of successful flights in addition to more stable position and attitude control.

8.5 Discussion

Simulations showed that adaptive controllers were more robust to mass changes and disturbances than the nonadaptive baseline controllers. The aMRAC/aBack controller always

yielded better position and attitude tracking than the PID/PIFL controller in simulation. For every flight test scenario, the adaptive controllers always yielded better position tracking than the baseline controllers, supporting the simulation results. The attitude tracking results were more nuanced. For the smooth trajectory and the hover with added weight, the baseline controller had better attitude tracking than the adaptive controller. For the damaged propeller flight tests, the adaptive controller had better attitude tracking than the baseline controller. However, the difference in mean attitude error for the damaged propeller and smooth trajectory flight tests was small (within a degree). While the baseline controller was able to achieve lower mean errors more often, it also exhibited greater variation in its tracking performance throughout the trajectory and across flights for all flight testing scenarios. The adaptive controllers provided smaller variations in tracking error and did not fail as often as the baseline controllers. While the attitude tracking was not the best, the adaptive controller was shown to be the best choice for attitude tracking for its stability, consistency, and the lack of significant tracking difference compared to the baseline controller. This selection supports the conclusions derived from simulations.

Chapter 9

Conclusions and Future Work

Controller development for 6DOF controllable multirotors, such as the omnicopter, has been limited, especially with regards to adaptive control. Specific to the omnicopter, a PID controller was developed for position control, and a PI/Feedback linearization controller was designed for attitude control. Successful flight tests were accomplished using this controller combination in [3]. This dissertation additionally developed MRAC and aMRAC controllers for omnicopter position control and backstepping, adaptive backstepping, and aMRAC controllers for omnicopter attitude control. Simulations that input parametric uncertainty and disturbances into the system showed that the aMRAC controller was the best choice for position control and adaptive backstepping performed better than the other controllers tested for attitude control, particularly when the omnicopter was subject to disturbances and uncertainty. Error results showed that position controller tracking performance was not significantly affected by the choice of attitude controller, and vice versa. While the aMRAC position controller clearly yielded position tracking with the least error in all circumstances, the choice for attitude control was less obvious because controller performance was mass dependent. Based on the simulation results, the aMRAC/aBack controller was chosen to be compared to the baseline PID/PIFL controller during flight tests, becoming the first fully adaptive controller successfully flown on the omnicopter. Flight test results corroborated the simulation conclusion that adaptive control was the best choice for omnicopter control. The adaptive controller had better position trajectory tracking, more stable attitude tracking,

and was less prone to crashing when subject to unknown mass increases, parameter uncertainty, and damaged propellers. The adaptive backstepping controller, while stable, tended to maintain large attitude errors in some flight scenarios, which corroborates the simulation finding that further attitude controller development is needed. While the aMRAC position controller yielded the best position tracking, the tuned gains found during simulations could not be directly applied to flight tests because of modelling limitations. The BLSA/MM method for estimating MIMO adaptive controller time delay margins was found to estimate realistic time delay margins for an aMRAC controller that could be compared to desired transient responses and used to tune the adaptive parameter gain matrix for a quadrotor. Although this method was not used to tune the omnicopter MRAC and aMRAC position controllers, it is possible the analysis would have helped select $\mathbf{\Gamma}$ values that were applicable in simulation and flights. In conclusion, this dissertation provides a framework for tuning future aMRAC controllers and shows that adaptive control can be implemented within the PX4 firmware architecture to successfully control an omnicopter.

9.1 Proposed Attitude and Motor Control Improvements

As was concluded with the simulations in Ch. 7, flight tests showed that there is opportunity to explore other attitude controllers that can provide fast transients and accurate tracking in the face of many types of parametric uncertainty. For the smooth trajectory, target attitude error would be under five degrees, although under one degree would be the best. Neither controller was able to accomplish this. For the hover with added weight and damaged propeller flight tests, the adaptive backstepping controller never seemed to attempt to level out the omnicopter. For the hover, the PI/FL controller attempted to level out the omnicopter but

failed, and it was prone to crashing during damaged propeller flights. Ideally, the attitude controller would be able to achieve level flight in these scenarios without crashing.

There are several modifications to the preexisting controllers that may yield improved attitude tracking. Making either the PI or the feedback linearization portion of the PI/FL controller adaptive is worth investigating because the aid of adaptation may have prevented the omnicopter from going unstable as it tried to achieve level flight. The PI controller is particularly interesting to augment because it contains the integral of quaternion error that is required to achieve level flight (0 attitude error tracking) during hover. However, the feedback linearization controller uses the moment of inertia matrix and omnicopter dynamics, which are subject to uncertainties, making it well set-up for adaptation. Turning the PI controller into a PID controller may also help prevent large oscillations by providing some damping in the control command. The inclusion of the derivative gain may also provide a better reference model for adaptive control. The aMRAC attitude controller explored in simulations may also provide this desired performance. Initial flight tests of the aMRAC attitude controller combined with aMRAC position control have been promising, and further testing and analysis is recommended. The benefit of the PI/FL controller was the inclusion of the integral error that caused the omnicopter to attempt to level out during hover, whereas adaptive backstepping and aMRAC do not include this term and appeared to put more emphasis on tracking desired angular velocity. Incorporating integral error into these controllers may yield the desired stable, level flight. Integral backstepping and adaptive integral backstepping have been accomplished in [88] and [89], and the necessary adjustment to the virtual control input appears to be minor. Additionally, further tuning of the controllers flown for flight tests may improve performance beyond the values used for the flight tests in Ch. 8 that were found using trial and error until reasonably steady flight was achieved. It should be noted that the ideal tuning for the omnicopter may change depending on the

desired trajectory. While tuning optimized for a stable hover can allow the omnicopter to track a slow or fast trajectory, it would probably yield larger errors than tuning specific for the fast or slow trajectory. Therefore, it is important to determine the desired use for the omnicopter to yield the best tuning.

A significant portion of the error was driven by motor direction changes during the most aggressive portion of the smooth trajectory. The severity of the effects of motor direction changes was not anticipated or modeled in the simulations. Implementing the motor optimization presented in [3] to prevent rapid motor switching may help smooth performance and decrease error. Another potential solution to the challenges associated with motor performance is the use of DSHOT, a faster digital protocol than the analog PWM signals currently being used to communicate between the flight controller and ESCs. DSHOT is also supposed to be more robust to electrical noise. Although the ESCs used for flight tests are capable of DSHOT, the Pixhawk 4 mini is only capable of simultaneously supporting DSHOT on 4 channels instead of the 8 required for the omnicopter. In the future, the use of a FCU capable of supporting DSHOT simultaneously on 8 or motor motors may improve tracking and allow for larger attitude angles and higher angular velocities.

9.2 Modeling and Tuning

While the simulations were good for determining general controller behavior, unfortunately, the gains used for Simulink simulations could not be used for hardware flight tests. The models used in Simulink allowed for the use of higher gains for more aggressive tracking. A better motor model, especially for motor direction flips, and better modeling of aerodynamics from rotor-rotor downwash interactions and downwash-solid surface interactions may help reduce the performance difference between simulations and reality. Improved modeling could

also help set better projection bounds. As was seen in the results of Ch. 8, the adaptive parameter norms rarely reached the projection bounds during flights, even with the large uncertainties of unknown mass and a damaged propeller. Keeping the bounds too high may allow the parameters to grow to a value that eventually causes the system to go unstable. Choosing tight bounds that still allow for adequate tracking of the desired trajectories is the ideal scenario, and is difficult to do without having a good idea of how the system performs.

However, Simulink was beneficial for determining a starting point for tuning values as well as for identifying general trends that occur when changing tuning parameters (e.g. effects on rise time, overshoot). By paying attention to the transient step response upon takeoff and the effect on the motors, the general order of magnitude and ratio for tuning values could be found that provided a reasonable starting point for tuning the values for actual flight. While the BLSA/MM method from Ch. 3 was not used to tune the MRAC and aMRAC controllers that were used on the omnicopter, it is likely that its implementation would have helped select gains for use in simulation that were more likely to be applicable to the physical omnicopter for flight tests. The changes in gain magnitude made comparing the adaptive parameter values between simulations and flight tests difficult. However, general trends for aMRAC could be identified, namely the high value and steady rise of the column 3 norm, most likely due to needing to compensate for gravity. Simulations also showed performance improvements associated with smooth trajectories, which was why smooth trajectories were chosen for flight tests.

9.3 Position Control Comments

The aMRAC position controller was shown to yield the best position trajectory tracking. Additionally, it was intuitive to design and implement because the baseline controller pro-

vided an easy selection for the reference model. The challenge with the MRAC controller was designing the reference model. In this case, an LQR controller was designed to serve as the reference. The issue with this design was the need to compute a matrix inverse. This work used the pseudoinverse, which was a quick but not necessarily the most computationally sound solution. This inverse computation issue is also present in the input saturation computation. While an appropriate choice of matrix inversion may have yielded better MRAC performance, it still would not be the preferred controller. If the work has already been done to develop a controller on which to base the reference model, it makes more sense to simply augment the previously developed controller. A PID controller served as the baseline for the aMRAC position controller. A common comparison in literature is with an LQR controller with integral feedback for trajectory tracking [31, 32, 38]. Future work could explore if tracking could be improved or tuning made easier by using the LQR with integral feedback as a baseline controller. A drawback to these controllers, however, is the use of a linear time invariant plant model to develop the reference model. The linear model inherently places limits on the maneuvers the omnicopter is able to execute. If more aggressive position trajectories were commanded, tracking error may increase. Interestingly, the aMRAC attitude controller uses a nonlinear, time-varying reference model based on its nonlinear baseline controller, allowing for aggressive maneuvers. This framework may provide a roadmap for turning the PI or FL portions of the PI/FL controller into their own aMRAC controllers or for developing aMRAC position control that does not rely on a linear baseline controller and reference model.

9.4 Research Question Conclusions

This research sought to determine what is the best control scheme for omnicopter trajectory tracking. The aMRAC controller was shown to be the best choice for position tracking in simulations and flight tests. The adaptive backstepping controller was shown to be the best choice of the considered attitude controllers. However, flight tests uncovered performance issues that were not obvious in simulation, leaving open the opportunity for more exploration.

Another goal of this research was to determine if the Simulink simulations used to initially tune and gauge controller performance were effective. The Simulink motor models did not accurately reflect the true motor and dynamics performance in flight tests, so tuning could not be directly translated from Simulink to the hardware. However, the use of Simulink ensured that the controllers were coded properly and provided reasonable numbers from which to begin tuning the flight vehicle. Additionally, it was able to show that adaptive controllers can outperform nonadaptive controllers, which was corroborated in the flight tests.

Finally, this research sought to determine if there is an intuitive, methodical way to tune an aMRAC (or standard MRAC) controller. Using a quadrotor as an example MIMO system, it was shown that estimating time delay margins using bounded linear stability analysis and the matrix measure produced values that could be correlated with transient responses to select an effective tuning gain ($\mathbf{\Gamma}$).

9.5 Broader Applications

While the controllers and controller comparisons presented in this dissertation are specific to the omnicopter, there is no reason other vehicles using similar states to define position

and attitude cannot use these controller frameworks, modified for the appropriate plant model and control allocation. It is important to remember that the backstepping controller was based on satellite attitude control, and the aMRAC attitude controller was taken from quadrotor attitude control. The key factor in choosing these controllers as a basis was their use of quaternions to define attitude. Other 6DOF controllable vehicles that may benefit from these controllers/controller combinations are unmanned underwater vehicles and satellites. It would be interesting to know if similar comparison trends are seen across the different 6DOF controllable vehicles.

Appendices

Appendix A

Projection Operator

The projection operator is implemented to counteract parameter drift, which prevents uniform boundedness of the adaptive parameter. The projection operator is mathematically noted as Eq. (A.1), and is developed and discussed in [82].

$$\dot{\hat{\boldsymbol{\Theta}}} = Proj(\hat{\boldsymbol{\Theta}}, \boldsymbol{\Gamma}\boldsymbol{\Phi}\boldsymbol{e}_{\boldsymbol{a}}^{\boldsymbol{T}}\boldsymbol{P}\boldsymbol{B}) \tag{A.1}$$

In practice, Eq. (A.1) translates to the adaptive law in Eq. (A.2), which uses Eq. (A.3) and (A.4) where j represents the j^{th} column of a matrix, Θ_j^{max} is the maximum bound for the j^{th} column of $\hat{\boldsymbol{\Theta}}$ and $\varepsilon_i > 0$ is the projection tolerance.

$$\dot{\hat{\boldsymbol{\Theta}}} = \Gamma \begin{cases} (\boldsymbol{\Phi}\boldsymbol{e}_{\boldsymbol{a}}^{\boldsymbol{T}}\boldsymbol{P}\boldsymbol{B})_j - \dfrac{\boldsymbol{\nabla f}_j \boldsymbol{\nabla f}_j^T}{\boldsymbol{\nabla f}_j^T \boldsymbol{\Gamma} \boldsymbol{\nabla f}_j} \boldsymbol{\Gamma}(\boldsymbol{\Phi}\boldsymbol{e}_{\boldsymbol{a}}^{\boldsymbol{T}}\boldsymbol{P}\boldsymbol{B})_j f_j \text{ , if } f_j > 0 \text{ and } (\boldsymbol{\Phi}\boldsymbol{e}_{\boldsymbol{a}}^{\boldsymbol{T}}\boldsymbol{P}\boldsymbol{B})_j^T \boldsymbol{\Gamma}\boldsymbol{\nabla f}_j > 0 \\ (\boldsymbol{\Phi}\boldsymbol{e}_{\boldsymbol{a}}^{\boldsymbol{T}}\boldsymbol{P}\boldsymbol{B})_j \text{ , else} \end{cases} \tag{A.2}$$

$$\boldsymbol{\nabla f}_j = \frac{2(1+\varepsilon_j)}{\varepsilon_j (\Theta_j^{max})^2} \hat{\boldsymbol{\Theta}}_j \tag{A.3}$$

$$f_j = \frac{(1+\varepsilon_j)\|\hat{\boldsymbol{\Theta}}_j\|^2 - (\Theta_j^{max})^2}{\varepsilon_j (\Theta_j^{max})^2} \tag{A.4}$$

Appendix B

Instructions for Implementing Custom Controllers in PX4 Firmware

All these edits will occur in the **src/PX4-Autopilot** folder. These instructions assume the PX4 Firmware was installed via the standard instructions for Ubuntu [90]. The instructions in this appendix provide a high level overview of file additions/edits. For more information on the specific workings of various aspects of the firmware code, refer to the online documentation [90].

1. Create controller code: **src/modules**
   (a) CREATE new module folder(s) and name appropriately, will contain the following files
      i. CREATE a .cpp file to implement custom controls
      ii. CREATE a .hpp to be the header to the .cpp file created (e.g. instantiate all functions, declare global variables, publishing/subscribing instantiation)
      iii. CREATE a CMake file
      iv. CREATE a Kconfig file to implement custom controllers
      v. CREATE a **<module_name>_params.yaml** for any modules that use custom parameters
         A. Make sure the yaml file appears in the CMake file

B. MODIFY .hpp and .cpp to appropriately use the parameters

2. Create the Mixer and Vehicle Setup: **ROMFS/px4fmu_common**

   (a) **/mixers**

   <vehicle name>.main.mix mixer file for motor passthroughADD <vehicle name>.main.mix to CMakeLists.txt

   (bi) **/init.d**

   rc.<vehicle name>_apps (start the correct controllers for your vehicle)CREATE **rc.<vehicle name>_defaults** (set default PWM, Disarm, etc values, set VEHICLE_TYPE name to <vehicle name>) ADD **rc.<vehicle name>_apps** and **rc.<vehicle name>_defaults** to **CMakeLists.txt** MODIFY **rc.vehicle_setup** by copying and pasting for another vehicle, modify to fit <vehicle_name >type, and make sure the correct mixer is specified: **$MIXER=<vehicle name>**, also make sure you set the correct **rc.<vehicle name>_apps** in the last line of the copied block

   (ii) **/init.d-posix/airframes** or **/init.d/airframes**

      i. CREATE file: **<number>_<vehicle_name>** (this is the name that will be used when compiling): Choose an unused number (we like 90001 or so).

      Values to set:

      default file to pull from: **. $R/etc/init.d/rc.<vehicle_name>_defaults**

      set desired mixer: **set MIXER <vehicle_name>**

      set desired PWM

      ii. ADD <number>_<vehicle_name> to **CMakeLists.txt**

3. Create launch/trajectory publishing files:

(a) CREATE a new folder: e.g. **<vehicle name>_traj**

i. CREATE folder **scripts**

A. ADD python script for publishing desired trajectory (e.g. **traj_pub.py**)

B. ADD trajectory files as .txt or .csv (however the python script wants them)

4. Create uORB messages if desired: **/msg**

(a) CREATE a new .msg file for recording any additional data (e.g. debugging)

(b) ADD .msg filename to **CMakeLists.txt**

(c) For debugging

i. It's best practice to put **debug** in the name of the topic

ii. ADD the topic name to the list **void LoggedTopics::add_debug_topics()** in **/src/ modules/logger/logged_topics.cpp**

iii. SET **SDLOG_PROFILE** in **rc.<vehicle name>_defaults** to 33 (or any value that has the 5th bit turned on for logging debugging topics)

(d) If you are having issues with names/MAVLINK, look in **/src/ modules/ mavlink/mavlink_receiver**, either the .h or .cpp file

5. Compile Controllers: **boards/px4/fmu-v5/default.px4board**

(a) Add the line **'CONFIG_MODULES_<name of module created above>=y'** for each module created (the 'y' indicates you wish this module to be compiled, the other option is 'n')

OR, if you want to be able to specify a controller at compile time

(a) Create **boards/px4/fmu-v5/<name>.px4board**

(b) Within the newly created file add the lines for the module you want to compile, e.g. **'CONFIG_MODULES_<name of module created above>=y'**

(c) When it comes time to compile the code, plug in the Pixhawk and type **make px4_fmu-v5_<name> upload** in the command line

Appendix C

Matrix Measure Computation

The notation in this Appendix does not necessarily exactly match the notation used in Chapter 3, specifically looking at the subscripts. The notation between Appendix C.1 and Appendix C.2 is purposely kept similar so parallels can be drawn between the derivations. It is important to pay attention to how each of the plants and errors are derived in the beginnings of each section in this Appendix because they influence the rest of the math in that section.

C.1 Standard MRAC [1]

C.1.1 Set-Up

Plant

$$\dot{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}[\boldsymbol{u} + \boldsymbol{\Theta}^{*T}\boldsymbol{\Phi}] \tag{C.1}$$

Plant with Input Time-Delay

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{A}\boldsymbol{x}(t) + \boldsymbol{B}[\boldsymbol{u}(t - t_d) + \boldsymbol{\Theta}^{*T}\boldsymbol{\Phi}] \tag{C.2}$$

Reference System

$$\dot{\boldsymbol{x}}_{ref} = \boldsymbol{A}_{ref}\boldsymbol{x}_{ref} + \boldsymbol{B}_{ref}\boldsymbol{r} \tag{C.3}$$

Note:

$$\boldsymbol{A}_{ref} = \boldsymbol{A} - \boldsymbol{B}\boldsymbol{K}_x, \boldsymbol{B}_{ref} = \boldsymbol{B}\boldsymbol{K}_r \tag{C.4}$$

Total Control

$$\boldsymbol{u}(t) = -\boldsymbol{K}_x\boldsymbol{x}(t) + \boldsymbol{K}_r\boldsymbol{r}(t) - \boldsymbol{u}_{ad}(t) \tag{C.5}$$

Adaptive Control

$$\boldsymbol{u}_{ad}(t) = \hat{\boldsymbol{\Theta}}^T\boldsymbol{\Phi} \tag{C.6}$$

Adaptive Parameter Update Law

$$\dot{\hat{\boldsymbol{\Theta}}} = -\boldsymbol{\Gamma}\boldsymbol{\Phi}\boldsymbol{e}^T\boldsymbol{P}\boldsymbol{B} \tag{C.7}$$

Local Bound on Adaptation

$$\dot{\hat{\boldsymbol{\Theta}}}\boldsymbol{\Phi} \leq -\gamma\boldsymbol{B}^T\boldsymbol{P}\boldsymbol{e} \tag{C.8}$$

Computation of γ: This can be computed using the trapezoidal rule (or another numerical integration method of choice). This analysis can only be performed after a simulation has been run/data has been collected because the values of $\boldsymbol{\Phi}$ through time are required to compute γ.

$$\gamma = \frac{1}{T_0}\int_{t_k - T_0}^{t_k} \boldsymbol{\Phi}^T\boldsymbol{\Gamma}\boldsymbol{\Phi} d\tau \tag{C.9}$$

C.1.2 Error Equation Development

This is a summary of the relevant theory developed in [1] with the interval math steps shown.

Error Computation:

$$\boldsymbol{e} = \boldsymbol{x}_{ref} - \boldsymbol{x} \tag{C.10}$$

Derivative of Error using Time-Delayed Input

$$\begin{aligned}
\dot{\boldsymbol{e}}(t) &= \dot{\boldsymbol{x}}_{ref} - \dot{\boldsymbol{x}} \\
&= \boldsymbol{A}_{ref}\boldsymbol{x}_{ref} + \boldsymbol{B}_{ref}\boldsymbol{r} - \boldsymbol{A}\boldsymbol{x} - \boldsymbol{B}[-\boldsymbol{K}_x\boldsymbol{x}(t-t_d) + \boldsymbol{K}_r\boldsymbol{r}(t-t_d) - \boldsymbol{u}_{ad}(t-t_d) + \boldsymbol{\Theta}^{*T}\boldsymbol{\Phi}] \\
&= \boldsymbol{A}_{ref}\boldsymbol{x}_{ref} + \boldsymbol{B}_{ref}\boldsymbol{r} - \cancel{\boldsymbol{A}\boldsymbol{x}} + \cancel{\boldsymbol{B}\boldsymbol{K}_x\boldsymbol{x}(t-t_d)} - \boldsymbol{B}\boldsymbol{K}_r\boldsymbol{r}(t-t_d) + \boldsymbol{B}\boldsymbol{u}_{ad}(t-t_d) - \boldsymbol{B}\boldsymbol{\Theta}^{*T}\boldsymbol{\Phi} \\
&+ \cancel{\boldsymbol{A}\boldsymbol{x}_{ref}} - \boldsymbol{A}\boldsymbol{x}_{ref} + \boldsymbol{B}\boldsymbol{K}_x\boldsymbol{x}_{ref}(t-t_d) - \cancel{\boldsymbol{B}\boldsymbol{K}_x\boldsymbol{x}_m(t-t_d)} \\
&= \boldsymbol{A}\boldsymbol{e}(t) - \boldsymbol{B}\boldsymbol{K}_x\boldsymbol{e}(t-t_d) + (\cancel{\boldsymbol{A}} - \boldsymbol{B}\boldsymbol{K}_x)\boldsymbol{x}_m - \cancel{\boldsymbol{A}\boldsymbol{x}_m} + \boldsymbol{B}\boldsymbol{K}_x\boldsymbol{x}_{ref}(t-t_d) + \boldsymbol{B}\boldsymbol{K}_r r \\
&- \boldsymbol{B}\boldsymbol{K}_r\boldsymbol{r}(t-t_d) + \boldsymbol{B}\boldsymbol{u}_{ad}(t-t_d) - \boldsymbol{B}\boldsymbol{\Theta}^{*T}\boldsymbol{\Phi} \\
&= \boldsymbol{A}\boldsymbol{e}(t) - \boldsymbol{B}\boldsymbol{K}_x\boldsymbol{e}(t-t_d) - \boldsymbol{B}\boldsymbol{K}_x(\boldsymbol{x}_{ref}(t) - \boldsymbol{x}_{ref}(t-t_d)) + \boldsymbol{B}\boldsymbol{K}_r(\boldsymbol{r}(t) - \boldsymbol{r}(t-t_d)) \\
&+ \boldsymbol{B}\boldsymbol{u}_{ad}(t-t_d) - \boldsymbol{B}\boldsymbol{\Theta}^{*T}\boldsymbol{\Phi}
\end{aligned} \tag{C.11}$$

Second Derivative of Error using Time-Delayed Input

$$\begin{aligned}
\ddot{\boldsymbol{e}}(t) = \boldsymbol{A}\dot{\boldsymbol{e}}(t) - \boldsymbol{B}\boldsymbol{K}_x\dot{e}(t-t_d) - \boldsymbol{B}\boldsymbol{K}_x(\dot{\boldsymbol{x}}_{ref}(t) - \dot{\boldsymbol{x}}_{ref}(t-t_d)) + \boldsymbol{B}\boldsymbol{K}_r(\dot{\boldsymbol{r}}(t) - \dot{\boldsymbol{r}}(t-t_d)) \\
+ \boldsymbol{B}\dot{\boldsymbol{u}}_{ad}(t-t_d) - \boldsymbol{B}\boldsymbol{\Theta}^{*T}\dot{\boldsymbol{\Phi}}
\end{aligned} \tag{C.12}$$

Derivative of Adaptive Control

$$\dot{\boldsymbol{u}}_{ad}(t) = \dot{\hat{\boldsymbol{\Theta}}}^T \boldsymbol{\Phi} + \hat{\boldsymbol{\Theta}}^T \dot{\boldsymbol{\Phi}} \tag{C.13}$$

Approximation of $\dot{\hat{\boldsymbol{\Theta}}}^T \boldsymbol{\Phi}$ (locally bounded with linear approximation γ - see Eq. (C.9)):

$$\dot{\hat{\boldsymbol{\Theta}}}^T \boldsymbol{\Phi} = -\boldsymbol{B}^T \boldsymbol{P} \boldsymbol{e} \boldsymbol{\Phi}^T \boldsymbol{\Gamma} \boldsymbol{\Phi} \leq -\gamma \boldsymbol{B}^T \boldsymbol{P} \boldsymbol{e}(t) \tag{C.14}$$

Approximation of $\dot{\boldsymbol{\Phi}}$ using Taylor Series Expansion:

$$\begin{aligned} \dot{\boldsymbol{\Phi}}(\boldsymbol{x}(t)) = &\sum_{l=1}^{L} \frac{\partial \boldsymbol{\Phi}(\boldsymbol{x}(t_k))}{\partial \boldsymbol{x}_l} \dot{\boldsymbol{x}}_l(t_k) + \sum_{l=1}^{L} \sum_{m=1}^{L} \frac{\partial^2 \boldsymbol{\Phi}(\boldsymbol{x}(t_k))}{\partial \boldsymbol{x}_l \partial \boldsymbol{x}_m} \dot{\boldsymbol{x}}_l(t_k) \dot{\boldsymbol{x}}_m(t_k) \Delta t \\ &+ \sum_{l=1}^{L} \frac{\partial \boldsymbol{\Phi}(\boldsymbol{x}(t_k))}{\partial \boldsymbol{x}_l} \ddot{\boldsymbol{x}}_l(t_k) \Delta t + H.O.T. \end{aligned} \tag{C.15}$$

$$\dot{\boldsymbol{x}}_m(t_k) = \frac{\boldsymbol{x}_m(t) - \boldsymbol{x}_m(t_k)}{\Delta t}, \text{similarly for } \ddot{\boldsymbol{x}}_l(t_k) \tag{C.16}$$

$$\boldsymbol{x}_m(t) = \boldsymbol{x}_{ref}(t) - \boldsymbol{e}(t) \tag{C.17}$$

$$\dot{\boldsymbol{\Phi}}(\boldsymbol{x}(t)) = \boldsymbol{\Phi}'_k[\dot{\boldsymbol{x}}_{ref}(t) - \dot{\boldsymbol{e}}(t)] + \boldsymbol{\Phi}''_k \dot{\boldsymbol{x}}_k[\boldsymbol{x}_{ref}(t) - \boldsymbol{e}(t) - \boldsymbol{x}_k] \tag{C.18}$$

$$(\boldsymbol{\Phi}'_k)_l = \frac{\partial \boldsymbol{\Phi}(\boldsymbol{x}(t_k))}{\partial \boldsymbol{x}_l}, \#\text{ eqns in } \boldsymbol{\Phi} \times \#\text{ states} \tag{C.19}$$

$$(\boldsymbol{\Phi}''_k)\dot{\boldsymbol{x}}_k = \sum_{m=1}^{L} \frac{\partial^2 \boldsymbol{\Phi}(\boldsymbol{x}(t_k))}{\partial \boldsymbol{x}_l \partial \boldsymbol{x}_m} \dot{\boldsymbol{x}}_m(t_k), \#\text{ eqns in } \boldsymbol{\Phi} \times \#\text{ states} \tag{C.20}$$

Substituting Eq. (C.18) into Eq. (C.12) and ignoring terms that don't include some version of $\boldsymbol{e}$ because they can be lumped into disturbances that don't affect the stability of the error

dynamics yields Eq. (C.21) and Eq. (C.22).

$$\begin{aligned}\ddot{\boldsymbol{e}}(t) &= \boldsymbol{B}\boldsymbol{\Theta}^{*T}\boldsymbol{\Phi}_k''\dot{x}_k\boldsymbol{e}(t) + [\boldsymbol{A} + \boldsymbol{B}\boldsymbol{\Theta}^{*T}\boldsymbol{\Phi}_k']\dot{\boldsymbol{e}}(t) \\ &\quad - [\gamma\boldsymbol{B}\boldsymbol{B}^T\boldsymbol{P} + \boldsymbol{B}\hat{\boldsymbol{\Theta}}^T\boldsymbol{\Phi}_k''\dot{x}_k]\boldsymbol{e}(t-t_d) - [\boldsymbol{B}\boldsymbol{K} + \boldsymbol{B}\hat{\boldsymbol{\Theta}}_k^T\boldsymbol{\Phi}_k']\dot{\boldsymbol{e}}(t-t_d)\end{aligned} \tag{C.21}$$

$$\begin{bmatrix}\dot{\boldsymbol{e}}(t)\\ \ddot{\boldsymbol{e}}(t)\end{bmatrix} = \begin{bmatrix}\boldsymbol{0} & \boldsymbol{I}\\ \boldsymbol{0} & [\boldsymbol{A} + \boldsymbol{B}\boldsymbol{\Theta}^{*T}\boldsymbol{\Phi}_k']\end{bmatrix}\begin{bmatrix}\boldsymbol{e}(t)\\ \dot{\boldsymbol{e}}(t)\end{bmatrix} - \begin{bmatrix}\boldsymbol{0} & \boldsymbol{0}\\ \gamma\boldsymbol{B}\boldsymbol{B}^T\boldsymbol{P} & [\boldsymbol{B}\boldsymbol{K} + \boldsymbol{B}\hat{\boldsymbol{\Theta}}_k^T\boldsymbol{\Phi}_k']\end{bmatrix}\begin{bmatrix}\boldsymbol{e}(t-t_d)\\ \dot{\boldsymbol{e}}(t-t_d)\end{bmatrix} \tag{C.22}$$

As a higher order term, Φ_k'' can be ignored. Note that if $\Phi = x$, then $\Phi_k' = I$ and $\Phi_k'' = 0$. Eqs. C.23 and C.24 correspond with Eqs. 3.16 and 3.17 in Chapter 3.

$$\ddot{\boldsymbol{e}}(t) = [\boldsymbol{A} + \boldsymbol{B}\boldsymbol{\Theta}^{*T}\boldsymbol{\Phi}_k']\dot{\boldsymbol{e}}(t) - [\gamma\boldsymbol{B}\boldsymbol{B}^T\boldsymbol{P}]\boldsymbol{e}(t-t_d) - [\boldsymbol{B}\boldsymbol{K}_x + \boldsymbol{B}\hat{\boldsymbol{\Theta}}_k^T\boldsymbol{\Phi}_k']\dot{\boldsymbol{e}}(t-t_d) \tag{C.23}$$

$$\begin{bmatrix}\dot{\boldsymbol{e}}(t)\\ \ddot{\boldsymbol{e}}(t)\end{bmatrix} = \begin{bmatrix}\boldsymbol{0} & \boldsymbol{I}\\ \boldsymbol{0} & [\boldsymbol{A} + \boldsymbol{B}\boldsymbol{\Theta}^{*T}\boldsymbol{\Phi}_k']\end{bmatrix}\begin{bmatrix}e(t)\\ \dot{e}(t)\end{bmatrix} - \begin{bmatrix}\boldsymbol{0} & \boldsymbol{0}\\ \gamma\boldsymbol{B}\boldsymbol{B}^T\boldsymbol{P} & [\boldsymbol{B}\boldsymbol{K}_x + \boldsymbol{B}\hat{\boldsymbol{\Theta}}_k^T\boldsymbol{\Phi}_k']\end{bmatrix}\begin{bmatrix}\boldsymbol{e}(t-t_d)\\ \dot{\boldsymbol{e}}(t-t_d)\end{bmatrix} \tag{C.24}$$

C.1.3 Matrix Measure Method

The resulting MM equations for computing time delay margin using the characteristic equation are Eqs. C.25 and C.26 and correspond with Eqs. 3.30 and 3.31 in Chapter 3.

$$t_{dk} \leq \frac{1}{\omega_k} cos^{-1}\left[\frac{\omega_k\overline{\mu}(\boldsymbol{A}+\boldsymbol{B}\boldsymbol{\Theta}^{*T}\boldsymbol{\Phi}_k')[\omega_k||\boldsymbol{B}\boldsymbol{K}_x+\boldsymbol{B}\hat{\boldsymbol{\Theta}}_k^T\boldsymbol{\Phi}_k'||-\overline{\mu}(j\gamma\boldsymbol{B}\boldsymbol{B}^T\boldsymbol{P})]}{\omega_k^2||\boldsymbol{B}\boldsymbol{K}_x+\boldsymbol{B}\hat{\boldsymbol{\Theta}}_k^T\boldsymbol{\Phi}_k'||^2+||\gamma\boldsymbol{B}\boldsymbol{B}^T\boldsymbol{P}||^2}+\frac{[\omega_k\overline{\mu}[j(\boldsymbol{B}\boldsymbol{K}_x+\boldsymbol{B}\hat{\boldsymbol{\Theta}}_k^T\boldsymbol{\Phi}_k')]+||\gamma\boldsymbol{B}\boldsymbol{B}^T\boldsymbol{P}||]}{\sqrt{\omega_k^2||\boldsymbol{B}\boldsymbol{K}_x+\boldsymbol{B}\hat{\boldsymbol{\Theta}}_k^T\boldsymbol{\Phi}_k'||^2+||\gamma\boldsymbol{B}\boldsymbol{B}^T\boldsymbol{P}||^2}}\right] \tag{C.25}$$

$$\omega_k = \frac{\overline{\mu}(-j(\boldsymbol{A}+\boldsymbol{B}\boldsymbol{\Theta}^{*T}\boldsymbol{\Phi}_k'))+||\boldsymbol{B}\boldsymbol{K}_x+\boldsymbol{B}\hat{\boldsymbol{\Theta}}_k^T\boldsymbol{\Phi}_k'||}{2}\left(1+\sqrt{1+\frac{4||\gamma\boldsymbol{B}\boldsymbol{B}^T\boldsymbol{P}||}{\left(\overline{\mu}(-j(\boldsymbol{A}+\boldsymbol{B}\boldsymbol{\Theta}^{*T}\boldsymbol{\Phi}_k'))+||\boldsymbol{B}\boldsymbol{K}_x+\boldsymbol{B}\hat{\boldsymbol{\Theta}}_k^T\boldsymbol{\Phi}_i'||\right)^2}}\right) \tag{C.26}$$

These equations are derived from the Laplace Transform of Eq. C.23. Note that $s = j\omega$ and $e^{-t_d j\omega} = cos(t_d\omega) - jsin(t_d\omega)$

$$s^2e(s) = [\boldsymbol{A}+\boldsymbol{B}\boldsymbol{\Theta}^{*T}\boldsymbol{\Phi}_k']s\boldsymbol{e}(s) - [\gamma\boldsymbol{B}\boldsymbol{B}^T\boldsymbol{P}]\boldsymbol{e}(s)e^{-t_ds} - [\boldsymbol{B}\boldsymbol{K}_x+\boldsymbol{B}\hat{\boldsymbol{\Theta}}_k^T\boldsymbol{\Phi}_k']s\boldsymbol{e}(s)e^{-t_ds} \tag{C.27}$$

Subtracting $s^2\boldsymbol{e}(s)$, dividing by s, and factoring out $\boldsymbol{e}(s)$ yields Eq. C.30.

$$0 = s\cancel{\boldsymbol{e}(s)} - [\boldsymbol{A}+\boldsymbol{B}\boldsymbol{\Theta}^{*T}\boldsymbol{\Phi}_k']\cancel{\boldsymbol{e}(s)} - [\gamma\boldsymbol{B}\boldsymbol{B}^T\boldsymbol{P}]\frac{\cancel{\boldsymbol{e}(s)}}{s}e^{-t_ds} - [\boldsymbol{B}\boldsymbol{K}_x+\boldsymbol{B}\hat{\boldsymbol{\Theta}}_k^T\boldsymbol{\Phi}_k']\cancel{\boldsymbol{e}(s)}e^{-t_ds} \tag{C.28}$$

$$0 = j\omega\boldsymbol{I} - [\boldsymbol{A}+\boldsymbol{B}\boldsymbol{\Theta}^{*T}\boldsymbol{\Phi}_k'] + [\frac{\gamma\boldsymbol{B}\boldsymbol{B}^T\boldsymbol{P}}{j\omega} + (\boldsymbol{B}\boldsymbol{K}_x+\boldsymbol{B}\hat{\boldsymbol{\Theta}}_k^T\boldsymbol{\Phi}_k')]e^{-t_dj\omega} \tag{C.29}$$

$$0 = j\omega\boldsymbol{I} - [\boldsymbol{A}+\boldsymbol{B}\boldsymbol{\Theta}^{*T}\boldsymbol{\Phi}_k'] + [\frac{-\gamma\boldsymbol{B}\boldsymbol{B}^T\boldsymbol{P}}{j\omega} + (\boldsymbol{B}\boldsymbol{K}_x+\boldsymbol{B}\hat{\boldsymbol{\Theta}}_k^T\boldsymbol{\Phi}_k')][cos(t_d\omega) - jsin(t_d\omega)] \tag{C.30}$$

Side Note: In the case of a system where the time delay margin can be computed exactly via Eq. (C.31), the time delay margin estimate using the matrix measure method is based on the eigenvalues (poles) of the closed loop system.

$$det(j\omega \boldsymbol{I} - \boldsymbol{A} + \boldsymbol{BK}e^{-t_d j\omega}) = 0 \tag{C.31}$$

The Real Parts of the poles are bounded from above by Eq. (C.32), assuming $0 \leq \omega t_d \leq \frac{\pi}{2}$:

$$\sigma = Re\lambda(\boldsymbol{A} - \boldsymbol{BK}e^{t_d j\omega}) \leq \overline{\mu}(\boldsymbol{A}) + \overline{\mu}(-\boldsymbol{BK})|cos(\omega t_d)| + \overline{\mu}(j\boldsymbol{BK})|sin(\omega t_d)| \leq 0 \tag{C.32}$$

$$\begin{aligned} &\overline{\mu}(\boldsymbol{A}) + \overline{\mu}(-\boldsymbol{BK})|cos(\omega t_d)| \leq +(\overline{\mu}(j\boldsymbol{BK})|sin(\omega t_d)|)^2 \\ &\rightarrow [\overline{\mu}^2(\boldsymbol{BK}) + \overline{\mu}^2(j\boldsymbol{BK})]cos^2(\omega t_d) - 2\overline{\mu}(\boldsymbol{A})\overline{\mu}(\boldsymbol{BK})cos(\omega t_d) + \overline{\mu}^2(\boldsymbol{A}) - \overline{\mu}^2(j\boldsymbol{BK}) \geq 0 \end{aligned} \tag{C.33}$$

Solve for ωt_d using the quadratic formula and simplifications, including $\overline{\mu}^2(\boldsymbol{BK}) + \overline{\mu}^2(j\boldsymbol{BK}) \leq \|\boldsymbol{BK}\|^2$. Solve for t_d, ignoring $-\overline{\mu}^2(\boldsymbol{A})$ in the square root as negligible.

$$t_d \leq \frac{1}{\omega} cos^{-1}[\frac{\overline{\mu}(\boldsymbol{A}) + \overline{\mu}(j\boldsymbol{BK})}{\|\boldsymbol{BK}\|}] \tag{C.34}$$

The Imaginary Parts of the poles are bounded above by Eq. (C.35).

$$\begin{aligned} \omega = Im\lambda(-j\boldsymbol{A} + j\boldsymbol{BK}e^{-t_d j\omega}) &\leq \overline{\mu}(-j\boldsymbol{A}) + \overline{\mu}(j\boldsymbol{BK})|cos(\omega t_d)| + \overline{\mu}(\boldsymbol{BK})|sin(\omega t_d)| \\ &\leq \overline{\mu}(-j\boldsymbol{A}) + ||\boldsymbol{BK}|| \end{aligned} \tag{C.35}$$

In a similar fashion, using the real and imaginary values of the characteristic equation in Eq. (C.30), the MM equations in Eqs. (C.25) and (C.26) can be calculated.

For the portion of the TDM estimation using the real component of the system poles:

$$\begin{aligned}
\sigma &= Re\lambda([\boldsymbol{A}+\boldsymbol{B}\boldsymbol{\Theta}^{*T}\boldsymbol{\Phi}_k'] - [\frac{\gamma \boldsymbol{B}\boldsymbol{B}^T\boldsymbol{P}}{j\omega} + (\boldsymbol{B}\boldsymbol{K}_x + \boldsymbol{B}\hat{\boldsymbol{\Theta}}_k^T\boldsymbol{\Phi}_k')]e^{-t_d j\omega}) \\
&\leq \overline{\mu}(\boldsymbol{A}+\boldsymbol{B}\boldsymbol{\Theta}^{*T}\boldsymbol{\Phi}_k') + \overline{\mu}(-(\boldsymbol{B}\boldsymbol{K}_x + \boldsymbol{B}\hat{\boldsymbol{\Theta}}_k^T\boldsymbol{\Phi}_k'))e^{-t_d j\omega} + \overline{\mu}(\frac{-\gamma \boldsymbol{B}\boldsymbol{B}^T\boldsymbol{P}}{j\omega})e^{-t_d j\omega} \\
&\leq \overline{\mu}(\boldsymbol{A}+\boldsymbol{B}\boldsymbol{\Theta}^{*T}\boldsymbol{\Phi}_k') + \overline{\mu}(-(\boldsymbol{B}\boldsymbol{K}_x + \boldsymbol{B}\hat{\boldsymbol{\Theta}}_k^T\boldsymbol{\Phi}_k'))(cos(t_d\omega) - jsin(t_d\omega)) \\
&+ \frac{1}{\omega}\overline{\mu}(j\gamma \boldsymbol{B}\boldsymbol{B}^T\boldsymbol{P})(cos(t_d\omega) - jsin(t_d\omega)) \\
&\leq \overline{\mu}(\boldsymbol{A}+\boldsymbol{B}\boldsymbol{\Theta}^{*T}\boldsymbol{\Phi}_k') + [\overline{\mu}(-(\boldsymbol{B}\boldsymbol{K}_x + \boldsymbol{B}\hat{\boldsymbol{\Theta}}_k^T\boldsymbol{\Phi}_k')) + \frac{1}{\omega}\overline{\mu}(j\gamma \boldsymbol{B}\boldsymbol{B}^T\boldsymbol{P})]cos(t_d\omega) \\
&+ [\overline{\mu}(j(\boldsymbol{B}\boldsymbol{K}_x + \boldsymbol{B}\hat{\boldsymbol{\Theta}}_k^T\boldsymbol{\Phi}_k')) + \frac{1}{\omega}\overline{\mu}(\gamma \boldsymbol{B}\boldsymbol{B}^T\boldsymbol{P})]sin(t_d\omega) \leq 0
\end{aligned} \tag{C.36}$$

The final version of Eq. (C.36) takes the form $0 = M_1 + M_2 cos(\omega t_d) + M_3 sin(\omega t_d)$. Subtracting $M_3 sin(\omega t_d)$ from both sides, squaring both sides, and recognizing that $(1 - cos^2(\omega t_d)) = sin^2(\omega t_d)$ yields $0 = (M_2^2 + M_3^2)cos^2(\omega t_d) + 2M_1M_2cos(\omega t_d) + (M_1^2 - M_3^2)$. The quadratic formula yields Eq. C.37.

$$\begin{aligned}
cos(\omega t_d) &\leq \frac{-2M_1M_2 + \sqrt{(2M_1M_2)^2 - 4(M_2^2+M_3^2)(M_1^2-M_3^2)}}{2(M_2^2+M_3^2)} \\
&\leq \frac{-\cancel{2}M_1M_2 + \sqrt{\cancel{4}(M_1M_2)^2 - \cancel{4}(M_1M_2)^2 + \cancel{4}(M_2M_3)^2 - \cancel{4}(M_1M_3)^2 + \cancel{4}M_3^4}}{\cancel{2}(M_2^2+M_3^2)} \\
&\leq \frac{-M_1M_2 + \sqrt{M_3^2(M_2^2+M_3^2) - M_3^2M_1^2}}{(M_2^2+M_3^2)} \\
&\leq \frac{-M_1M_2 + M_3\sqrt{M_2^2+M_3^2}}{(M_2^2+M_3^2)}
\end{aligned} \tag{C.37}$$

The subtraction of M_1^2 is ignored to maximize the TDM estimate and prevent the possibility of imaginary values.

Let

$$M_2^2 + M_3^2 = ||\boldsymbol{BK}_x + \boldsymbol{B}\hat{\boldsymbol{\Theta}}_k^T\boldsymbol{\Phi}_k'||^2 + \frac{1}{\omega^2}||\gamma\boldsymbol{BB}^T\boldsymbol{P})||^2 \tag{C.38}$$

$$-M_1M_2 = \overline{\mu}(\boldsymbol{A} + \boldsymbol{B}\boldsymbol{\Theta}^{*T}\boldsymbol{\Phi}')[||\boldsymbol{BK}_x + \boldsymbol{B}\hat{\boldsymbol{\Theta}}_k^T\boldsymbol{\Phi}_k'|| - \frac{1}{\omega}\overline{\mu}(j\gamma\boldsymbol{BB}^T\boldsymbol{P}] \tag{C.39}$$

$$M_3 = [\overline{\mu}(j(\boldsymbol{BK}_x + \boldsymbol{B}\hat{\boldsymbol{\Theta}}_k^T\boldsymbol{\Phi}_k')) + \frac{1}{\omega}||\gamma\boldsymbol{BB}^T\boldsymbol{P}||] \tag{C.40}$$

When the imaginary number (j) is not present, the norm is used because $\overline{\mu}(M) \leq ||M||$. For M_2^2 and M_3^2 each, the inequality $\overline{\mu}^2(M) + \overline{\mu}^2(jM) \leq ||M||^2$ is used.

For the portion of the TDM estimation using the imaginary part of the poles:

$$\begin{aligned}
\omega &= Im\lambda(-j(\boldsymbol{A} + \boldsymbol{B}\boldsymbol{\Theta}^{*T}\boldsymbol{\Phi}') + j(\boldsymbol{BK}_x + \boldsymbol{B}\hat{\boldsymbol{\Theta}}_k^T\boldsymbol{\Phi}_k')e^{-t_d j\omega} + \frac{1}{\omega}(\gamma\boldsymbol{BB}^T\boldsymbol{P}e^{-t_d j\omega})) \\
&\leq \overline{\mu}(-j(\boldsymbol{A} + \boldsymbol{B}\boldsymbol{\Theta}^{*T}\boldsymbol{\Phi}')) + \overline{\mu}(j(\boldsymbol{BK}_x + \boldsymbol{B}\hat{\boldsymbol{\Theta}}_k^T\boldsymbol{\Phi}_k')e^{-t_d j\omega}) + \frac{1}{\omega}\overline{\mu}(\gamma\boldsymbol{BB}^T\boldsymbol{P}e^{-t_d j\omega}) \\
&\leq \overline{\mu}(-j(\boldsymbol{A} + \boldsymbol{B}\boldsymbol{\Theta}^{*T}\boldsymbol{\Phi}')) + ||\boldsymbol{BK}_x + \boldsymbol{B}\hat{\boldsymbol{\Theta}}_k^T\boldsymbol{\Phi}_k'|| + \frac{1}{\omega}||\gamma\boldsymbol{BB}^T\boldsymbol{P}e^{-t_d j\omega}||
\end{aligned} \tag{C.41}$$

Multiplying both sides of the inequality by ω allows for the use of the quadratic formula to solve for ω that minimizes the TDM:

$$\begin{aligned}
\omega = &\frac{\cancel{-}(\overline{\mu}(-j(\boldsymbol{A} + \boldsymbol{B}\boldsymbol{\Theta}^{*T}\boldsymbol{\Phi}')) + ||\boldsymbol{BK}_x + \boldsymbol{B}\hat{\boldsymbol{\Theta}}_k^T\boldsymbol{\Phi}_k'||)}{2\cancel{(-1)}} \\
&+ \frac{\sqrt{(\overline{\mu}(-j(\boldsymbol{A} + \boldsymbol{B}\boldsymbol{\Theta}^{*T}\boldsymbol{\Phi}')) + ||\boldsymbol{BK}_x + \boldsymbol{B}\hat{\boldsymbol{\Theta}}_k^T\boldsymbol{\Phi}_k'||)^2 - 4(-1)(||\gamma\boldsymbol{BB}^T\boldsymbol{P}||)}}{2\cancel{(-1)}}
\end{aligned} \tag{C.42}$$

C.2 MRAC Augmentation of an Optimal Baseline Controller (aMRAC)

C.2.1 Set-Up

Plant: The augmented plant, where $\boldsymbol{x} = (\boldsymbol{x}_p, \int \boldsymbol{e})^T$

$$\dot{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{\Lambda}[\boldsymbol{u} + \boldsymbol{\Theta}^{*T}\boldsymbol{\Phi}] + \boldsymbol{B}_{ref}\boldsymbol{r} \tag{C.43}$$

Plant with Input Time-Delay

$$\begin{aligned}
\dot{\boldsymbol{x}}(t) &= \boldsymbol{A}\boldsymbol{x}(t) + \boldsymbol{B}\boldsymbol{\Lambda}[\boldsymbol{u}(t-t_d) + \boldsymbol{\Theta}^{*T}\boldsymbol{\Phi}] + \boldsymbol{B}_{ref}\boldsymbol{r}(t) \\
&= \boldsymbol{A}\boldsymbol{x}(t) + \boldsymbol{B}\boldsymbol{\Lambda}[-\boldsymbol{K}\boldsymbol{x}(t-t_d) + \boldsymbol{u}_{ad}(t-t_d) + \boldsymbol{\Theta}^{*T}\boldsymbol{\Phi}] + \boldsymbol{B}_{ref}\boldsymbol{r}(t) \\
&= \boldsymbol{A}\boldsymbol{x}(t) - \boldsymbol{B}\boldsymbol{K}\boldsymbol{x}(t-t_d) + \boldsymbol{B}\boldsymbol{\Lambda}[(\boldsymbol{I} - \boldsymbol{\Lambda}^{-1})\boldsymbol{u}_{bl}(t-t_d) + \boldsymbol{u}_{ad}(t-t_d) + \boldsymbol{\Theta}^{*T}\boldsymbol{\Phi}] + \boldsymbol{B}_{ref}\boldsymbol{r}(t) \\
&= \boldsymbol{A}\boldsymbol{x}(t) - \boldsymbol{B}\boldsymbol{K}\boldsymbol{x}(t-t_d) + \boldsymbol{B}\boldsymbol{\Lambda}[\boldsymbol{u}_{ad}(t-t_d) + \overline{\boldsymbol{\Theta}}^{*T}\overline{\boldsymbol{\Phi}}] + \boldsymbol{B}_{ref}\boldsymbol{r}(t)
\end{aligned} \tag{C.44}$$

$$\overline{\boldsymbol{\Theta}}^{*T} = \begin{bmatrix} \boldsymbol{\Theta}^* & (\boldsymbol{I} - \boldsymbol{\Lambda}^{-1}) \end{bmatrix}^T \tag{C.45}$$

$$\overline{\boldsymbol{\Phi}} = \begin{bmatrix} \boldsymbol{\Phi} \\ \boldsymbol{u}_{bl}(t) \end{bmatrix} \tag{C.46}$$

Reference System

$$\dot{\boldsymbol{x}}_{ref} = \boldsymbol{A}_{ref}\boldsymbol{x}_{ref} + \boldsymbol{B}_{ref}\boldsymbol{r} \tag{C.47}$$

Note:

$$\boldsymbol{A}_{ref} = \boldsymbol{A} - \boldsymbol{B}\boldsymbol{K} \tag{C.48}$$

Total Control

$$\boldsymbol{u}(t) = \boldsymbol{u}_{bl}(t) + \boldsymbol{u}_{ad}(t) \tag{C.49}$$

Baseline Control

$$\boldsymbol{u}(t) = -\boldsymbol{K}\boldsymbol{x}(t) \tag{C.50}$$

Adaptive Control

$$\boldsymbol{u}_{ad}(t) = -\hat{\overline{\boldsymbol{\Theta}}}^{T}\overline{\boldsymbol{\Phi}} \tag{C.51}$$

Adaptive Parameter Update Law

$$\dot{\hat{\overline{\boldsymbol{\Theta}}}} = \boldsymbol{\Gamma}\overline{\boldsymbol{\Phi}}\boldsymbol{e}^{T}\boldsymbol{P}\boldsymbol{B} \tag{C.52}$$

Local Bound on Adaptation

$$\dot{\hat{\overline{\boldsymbol{\Theta}}}}\overline{\boldsymbol{\Phi}} \leq -\gamma\boldsymbol{B}^{T}\boldsymbol{P}\boldsymbol{e} \tag{C.53}$$

Computation of γ: This can be computed using the trapezoidal rule (or another numerical integration method of choice). This analysis can only be performed after a simulation has been run/data has been collected because the values of $\overline{\boldsymbol{\Phi}}$ through time are required to compute γ.

$$\gamma = \frac{1}{T_0}\int_{t_k - T_0}^{t_k} \overline{\boldsymbol{\Phi}}^{T}\boldsymbol{\Gamma}\overline{\boldsymbol{\Phi}}d\tau \tag{C.54}$$

C.2.2 Error Equation Development

This is a summary of the relevant theory developed in [1] with the interval math steps shown.

Error Computation:

$$\boldsymbol{e} = \boldsymbol{x} - \boldsymbol{x}_{ref} \tag{C.55}$$

Derivative of Error using Time-Delayed Input

$$\begin{aligned}
\dot{\boldsymbol{e}}(t) &= \dot{\boldsymbol{x}} - \dot{\boldsymbol{x}}_{ref} \\
&= \boldsymbol{A}\boldsymbol{x}(t) - \boldsymbol{B}\boldsymbol{K}\boldsymbol{x}(t-t_d) + \boldsymbol{B}\boldsymbol{\Lambda}[\boldsymbol{u}_{ad}(t-t_d) + \overline{\boldsymbol{\Theta}}^{*T}\overline{\boldsymbol{\Phi}}] + \cancel{\boldsymbol{B}_r\boldsymbol{r}(t)} - \boldsymbol{A}_{ref}\boldsymbol{x}_{ref}(t) - \cancel{\boldsymbol{B}_{ref}\boldsymbol{r}(t)} \\
&= \cancel{\boldsymbol{A}\boldsymbol{x}(t)} - \cancel{\boldsymbol{B}\boldsymbol{K}\boldsymbol{x}(t-t_d)} + \boldsymbol{B}\boldsymbol{\Lambda}[\boldsymbol{u}_{ad}(t-t_d) + \overline{\boldsymbol{\Theta}}^{*T}\overline{\boldsymbol{\Phi}}] - \boldsymbol{A}_{ref}\boldsymbol{x}_{ref}(t) \\
&- \cancel{\boldsymbol{A}\boldsymbol{x}_{ref}} + \boldsymbol{A}\boldsymbol{x}_{ref}(t) - \boldsymbol{B}\boldsymbol{K}\boldsymbol{x}_{ref}(t)(t-t_d) + \cancel{\boldsymbol{B}\boldsymbol{K}\boldsymbol{x}_{ref}(t-t_d)} \\
&= \boldsymbol{A}\boldsymbol{e}(t) - \boldsymbol{B}\boldsymbol{K}\boldsymbol{e}(t-t_d) - (\cancel{\boldsymbol{A}} - \boldsymbol{B}\boldsymbol{K})\boldsymbol{x}_{ref}(t) + \cancel{\boldsymbol{A}\boldsymbol{x}_{ref}(t)} - \boldsymbol{B}\boldsymbol{K}\boldsymbol{x}_{ref}(t-t_d) \\
&+ \boldsymbol{B}\boldsymbol{\Lambda}\boldsymbol{u}_{ad}(t-t_d) - \boldsymbol{B}\boldsymbol{\Lambda}\overline{\boldsymbol{\Theta}}^{*T}\overline{\boldsymbol{\Phi}} \\
&= \boldsymbol{A}\boldsymbol{e}(t) - \boldsymbol{B}\boldsymbol{K}\boldsymbol{e}(t-t_d) + \boldsymbol{B}\boldsymbol{K}(\boldsymbol{x}_{ref}(t)) - \boldsymbol{x}_{ref}(t-t_d)) + \boldsymbol{B}\boldsymbol{\Lambda}\boldsymbol{u}_{ad}(t-t_d) + \boldsymbol{B}\boldsymbol{\Lambda}\overline{\boldsymbol{\Theta}}^{*T}\overline{\boldsymbol{\Phi}}
\end{aligned} \tag{C.56}$$

Second Derivative of Error using Time-Delayed Input

$$\ddot{\boldsymbol{e}}(t) = \boldsymbol{A}\dot{\boldsymbol{e}}(t) - \boldsymbol{B}\boldsymbol{K}\dot{\boldsymbol{e}}(t-t_d) - \boldsymbol{B}\boldsymbol{K}(\dot{\boldsymbol{x}}_{ref}(t) - \dot{\boldsymbol{x}}_{ref}(t-t_d)) + \boldsymbol{B}\boldsymbol{\Lambda}\dot{\boldsymbol{u}}_{ad}(t-t_d) + \boldsymbol{B}\boldsymbol{\Lambda}\overline{\boldsymbol{\Theta}}^{*T}\dot{\overline{\boldsymbol{\Phi}}} \tag{C.57}$$

Derivative of Adaptive Control

$$\begin{aligned}
\dot{\boldsymbol{u}}_{ad}(t) &= \dot{\hat{\overline{\boldsymbol{\Theta}}}}^{T}\overline{\boldsymbol{\Phi}} + \hat{\overline{\boldsymbol{\Theta}}}^{T}\dot{\overline{\boldsymbol{\Phi}}} \\
&= -\gamma\boldsymbol{B}^{T}\boldsymbol{P}\boldsymbol{e}(t) - \hat{\overline{\boldsymbol{\Theta}}}^{T}[\overline{\boldsymbol{\Phi}}'_k[\dot{\boldsymbol{x}}_{ref}(t) + \dot{\boldsymbol{e}}(t)] + \overline{\boldsymbol{\Phi}}''_k\dot{\boldsymbol{x}}_k[\boldsymbol{x}_{ref}(t) + \boldsymbol{e}(t) - \boldsymbol{x}_k]]
\end{aligned} \tag{C.58}$$

Approximation of $\dot{\hat{\overline{\boldsymbol{\Theta}}}}^{T}\overline{\boldsymbol{\Phi}}$ (locally bounded with linear approximation γ - see Eq.

(C.54))

$$\dot{\hat{\overline{\boldsymbol{\Theta}}}}^T \overline{\boldsymbol{\Phi}} = -\boldsymbol{B}^T \boldsymbol{P} \boldsymbol{e} \overline{\boldsymbol{\Phi}}^T \boldsymbol{\Gamma} \overline{\boldsymbol{\Phi}} \leq -\gamma \boldsymbol{B}^T \boldsymbol{P} \boldsymbol{e}(t) \tag{C.59}$$

$\overline{\boldsymbol{\Phi}}'_k$ and $\overline{\boldsymbol{\Phi}}''_k$ are approximates of $\dot{\overline{\boldsymbol{\Phi}}}$ using the Taylor Series Expansion in the same manner as in the previous section (Eqs. C.15 - C.20).

Substituting Eq. (C.58) and the augmented version of Eq. (C.18) into Eq. (C.57) and ignoring terms that don't include some version of $\boldsymbol{e}$ because they can be lumped into disturbances that don't affect the stability of the error dynamics yields Eq. (C.60) and Eq. (C.61).

$$\begin{aligned} \ddot{\boldsymbol{e}}(t) = {} & \boldsymbol{B}\boldsymbol{\Lambda}\overline{\boldsymbol{\Theta}}^{*T}\overline{\boldsymbol{\Phi}}''_k \dot{\boldsymbol{x}}_k \boldsymbol{e}(t) + [\boldsymbol{A} + \boldsymbol{B}\boldsymbol{\Lambda}\overline{\boldsymbol{\Theta}}^{*T}\overline{\boldsymbol{\Phi}}'_k]\dot{\boldsymbol{e}}(t) \\ & - [\gamma \boldsymbol{B}\boldsymbol{\Lambda}\boldsymbol{B}^T\boldsymbol{P} + \boldsymbol{B}\boldsymbol{\Lambda}\hat{\overline{\boldsymbol{\Theta}}}^T\overline{\boldsymbol{\Phi}}''_k \dot{\boldsymbol{x}}_k]\boldsymbol{e}(t - t_d) - [\boldsymbol{B}\boldsymbol{K} + \boldsymbol{B}\Lambda\hat{\overline{\boldsymbol{\Theta}}}_i^T\overline{\boldsymbol{\Phi}}'_k]\dot{\boldsymbol{e}}(t - t_d) \end{aligned} \tag{C.60}$$

$$\begin{aligned} \begin{bmatrix} \dot{\boldsymbol{e}}(t) \\ \ddot{\boldsymbol{e}}(t) \end{bmatrix} = {} & \begin{bmatrix} \boldsymbol{0} & \boldsymbol{I} \\ \boldsymbol{B}\boldsymbol{\Lambda}\overline{\boldsymbol{\Theta}}^{*T}\overline{\boldsymbol{\Phi}}''_k \dot{x}_k & [\boldsymbol{A} + \boldsymbol{B}\boldsymbol{\Lambda}\overline{\boldsymbol{\Theta}}^{*T}\overline{\boldsymbol{\Phi}}'_k] \end{bmatrix} \begin{bmatrix} \boldsymbol{e}(t) \\ \dot{\boldsymbol{e}}(t) \end{bmatrix} \\ & - \begin{bmatrix} \boldsymbol{0} & \boldsymbol{0} \\ [\gamma \boldsymbol{B}\boldsymbol{\Lambda}\boldsymbol{B}^T\boldsymbol{P} + \boldsymbol{B}\boldsymbol{\Lambda}\overline{\boldsymbol{\Theta}}^{*T}\overline{\boldsymbol{\Phi}}''_k \dot{x}_k] & [\boldsymbol{B}\boldsymbol{K} + \boldsymbol{B}\Lambda\hat{\overline{\boldsymbol{\Theta}}}_k^T\overline{\boldsymbol{\Phi}}'_k] \end{bmatrix} \begin{bmatrix} \boldsymbol{e}(t - t_d) \\ \dot{\boldsymbol{e}}(t - t_d) \end{bmatrix} \end{aligned} \tag{C.61}$$

As a higher order term, Φ''_k can be ignored. Note that if $\Phi = x$, then $\Phi'_k = I$ and $\Phi''_k = 0$. Eqs. C.62 and C.63 correspond with Eqs. 3.19 and 3.20 in Chapter 3.

$$\ddot{\boldsymbol{e}}(t) = [\boldsymbol{A} + \boldsymbol{B}\boldsymbol{\Lambda}\boldsymbol{\Theta}^{*T}\boldsymbol{\Phi}'_k]\dot{\boldsymbol{e}}(t) - [\gamma \boldsymbol{B}\boldsymbol{\Lambda}\boldsymbol{B}^T\boldsymbol{P}]\boldsymbol{e}(t - t_d) - [\boldsymbol{B}\boldsymbol{K}_x + \boldsymbol{B}\boldsymbol{\Lambda}\hat{\boldsymbol{\Theta}}_k^T\boldsymbol{\Phi}'_k]\dot{\boldsymbol{e}}(t - t_d) \tag{C.62}$$

$$\begin{bmatrix} \dot{\boldsymbol{e}}(t) \\ \ddot{\boldsymbol{e}}(t) \end{bmatrix} = \begin{bmatrix} \mathbf{0} & \boldsymbol{I} \\ \mathbf{0} & [\boldsymbol{A} + \boldsymbol{B}\boldsymbol{\Lambda}\boldsymbol{\Theta}^{*T}\boldsymbol{\Phi}'_k] \end{bmatrix} \begin{bmatrix} e(t) \\ \dot{e}(t) \end{bmatrix} - \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \gamma \boldsymbol{B}\boldsymbol{\Lambda}\boldsymbol{B}^T\boldsymbol{P} & [\boldsymbol{B}\boldsymbol{K}_x + \boldsymbol{B}\boldsymbol{\Lambda}\hat{\boldsymbol{\Theta}}_k^T\boldsymbol{\Phi}'_k] \end{bmatrix} \begin{bmatrix} \boldsymbol{e}(t-t_d) \\ \dot{\boldsymbol{e}}(t-t_d) \end{bmatrix} \tag{C.63}$$

C.2.3 Matrix Measure Method

The resulting MM equations for computing time delay margin using the characteristic equation are Eqs. C.64 and C.65 and correspond with Eqs. 3.32 and 3.33 in Chapter 3.

$$t_{dk} \leq \frac{1}{\omega_k} cos^{-1} \left[\frac{\omega_k \overline{\mu}(\boldsymbol{A} + \boldsymbol{B}\boldsymbol{\Lambda}\overline{\boldsymbol{\Theta}}^{*T}\overline{\boldsymbol{\Phi}}'_k)[\omega_k||\boldsymbol{B}\boldsymbol{K} + \boldsymbol{B}\boldsymbol{\Lambda}\hat{\overline{\boldsymbol{\Theta}}}_k^T\overline{\boldsymbol{\Phi}}'_k|| - \overline{\mu}(j\gamma\boldsymbol{B}\boldsymbol{\Lambda}\boldsymbol{B}^T\boldsymbol{P})]}{\omega_k^2||\boldsymbol{B}\boldsymbol{K} + \boldsymbol{B}\boldsymbol{\Lambda}\hat{\overline{\boldsymbol{\Theta}}}_i^T\overline{\boldsymbol{\Phi}}'_k||^2 + ||\gamma\boldsymbol{B}\boldsymbol{\Lambda}\boldsymbol{B}^T\boldsymbol{P}||^2} + \frac{[\omega_k\overline{\mu}[j(\boldsymbol{B}\boldsymbol{K} + \boldsymbol{B}\boldsymbol{\Lambda}\hat{\overline{\boldsymbol{\Theta}}}_k^T\overline{\boldsymbol{\Phi}}'_k)] + ||\gamma\boldsymbol{\Lambda}\boldsymbol{B}\boldsymbol{B}^T\boldsymbol{P}||]}{\sqrt{\omega_k^2||\boldsymbol{B}\boldsymbol{K}_x + \boldsymbol{B}\boldsymbol{\Lambda}\hat{\overline{\boldsymbol{\Theta}}}_k^T\overline{\boldsymbol{\Phi}}'_k||^2 + ||\gamma\boldsymbol{B}\boldsymbol{\Lambda}\boldsymbol{B}^T\boldsymbol{P}||^2}} \right] \tag{C.64}$$

$$\omega_k = \frac{\overline{\mu}[-j(\boldsymbol{A} + \boldsymbol{B}\boldsymbol{\Lambda}\overline{\boldsymbol{\Theta}}^{*T}\overline{\boldsymbol{\Phi}}'_k)] + ||\boldsymbol{B}\boldsymbol{K} + \boldsymbol{B}\boldsymbol{\Lambda}\hat{\overline{\boldsymbol{\Theta}}}_k^T\overline{\boldsymbol{\Phi}}'_k||}{2} \left(1 + \sqrt{1 + \frac{4||\gamma\boldsymbol{B}\boldsymbol{\Lambda}\boldsymbol{B}^T\boldsymbol{P}||}{(\overline{\mu}(-j(\boldsymbol{A} + \boldsymbol{B}\boldsymbol{\Lambda}\overline{\boldsymbol{\Theta}}^{*T}\overline{\boldsymbol{\Phi}}'_k)) + ||\boldsymbol{B}\boldsymbol{K} + \boldsymbol{B}\boldsymbol{\Lambda}\hat{\overline{\boldsymbol{\Theta}}}_k^T\overline{\boldsymbol{\Phi}}'_k||)^2}} \right) \tag{C.65}$$

These equations match the format for BLAS for standard MRAC. Therefore, the matrix measure equations and their derivations for standard MRAC (Eq. (C.25) and Eq. (C.26)) apply to aMRAC. The key differences are the inclusion of Λ in some places, and the use of $\overline{\boldsymbol{\Phi}}, \overline{\boldsymbol{\Theta}}$ in the MM equations and for computing γ. Because of these similarities, derivations using the Laplace Transform are not provided for aMRAC as was done for standard MRAC.

Bibliography

[1] N. Nguyen and E. Summers, "On time delay margin estimation for adaptive control and robust modification adaptive laws," in *AIAA Guidance, navigation, and control conference*, p. 6438, 2011.

[2] G. Gargioni, M. Peterson, J. Persons, K. Schroeder, and J. Black, "A full distributed multipurpose autonomous flight system using 3d position tracking and ros," in *2019 International Conference on Unmanned Aircraft Systems (ICUAS)*, pp. 1458–1466, IEEE, 2019.

[3] D. Brescianini and R. D'Andrea, "An omni-directional multirotor vehicle," *Mechatronics*, vol. 55, pp. 76–93, 2018.

[4] H. Lee and H. J. Kim, "Trajectory tracking control of multirotors from modelling to experiments: A survey," *International Journal of Control, Automation and Systems*, vol. 15, no. 1, pp. 281–292, 2017.

[5] J. Kim, S. A. Gadsden, and S. A. Wilkerson, "A comprehensive survey of control strategies for autonomous quadrotors," *Canadian Journal of Electrical and Computer Engineering*, vol. 43, no. 1, pp. 3–16, 2019.

[6] H. Nguyen, M. Kamel, K. Alexis, and R. Siegwart, "Model predictive control for micro aerial vehicles: A survey," in *2021 European Control Conference (ECC)*, pp. 1556–1563, 2021.

[7] S. I. Abdelmaksoud, M. Mailah, and A. M. Abdallah, "Control strategies and novel tech-

niques for autonomous rotorcraft unmanned aerial vehicles: A review," *IEEE Access*, vol. 8, pp. 195142–195169, 2020.

[8] S. Bouabdallah, A. Noth, and R. Siegwart, "Pid vs lq control techniques applied to an indoor micro quadrotor," in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566)*, vol. 3, pp. 2451–2456, IEEE, 2004.

[9] O. Araar and N. Aouf, "Full linear control of a quadrotor uav, lq vs h∞," in *2014 UKACC International Conference on Control (CONTROL)*, pp. 133–138, IEEE, 2014.

[10] W. Giernacki, D. Horla, T. Sadalla, and J. P. Coelho, "Robust cdm and pole placement pid based thrust controllers for multirotor motor-rotor simplified model: The comparison in a context of using anti-windup compensation," in *2016 International Siberian Conference on Control and Communications (SIBCON)*, pp. 1–5, 2016.

[11] A. Benaddy, M. Bouzi, and M. Labbadi, "Comparison of the different control strategies for quadrotor unmanned aerial vehicle," in *2020 International Conference on Intelligent Systems and Computer Vision (ISCV)*, pp. 1–6, 2020.

[12] G. Jiang, R. Voyles, K. Sebesta, and H. Greiner, "Estimation and optimization of fully-actuated multirotor platform with nonparallel actuation mechanism," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 6843–6848, IEEE, 2017.

[13] P. Segui-Gasco, Y. Al-Rihani, H.-S. Shin, and A. Savvaris, "A novel actuation concept for a multi rotor uav," *Journal of Intelligent & Robotic Systems*, vol. 74, no. 1-2, pp. 173–191, 2014.

[14] Y. Long and D. J. Cappelleri, "Linear control design, allocation, and implementation

for the omnicopter mav," in *2013 IEEE International Conference on Robotics and Automation*, pp. 289–294, IEEE, 2013.

[15] S. Park, J. Her, J. Kim, and D. Lee, "Design, modeling and control of omni-directional aerial robot," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1570–1575, IEEE, 2016.

[16] P. Cao, J. Strawson, T. Bewley, and F. Kuester, "Decoupled translational and rotational flight control designs of canted-rotor hexacopters," in *AIAA Scitech 2021 Forum*, p. 1058, 2021.

[17] T. Shimizu, S. Suzuki, T. Kawamura, H. Ueno, and H. Murakami, "Proposal of 6dof multi-copter and verification of its controllability," in *2015 54th Annual Conference of the Society of Instrument and Control Engineers of Japan (SICE)*, pp. 810–815, IEEE, 2015.

[18] M. Kamel, S. Verling, O. Elkhatib, C. Sprecher, P. Wulkop, Z. Taylor, R. Siegwart, and I. Gilitschenski, "The voliro omniorientational hexacopter: An agile and maneuverable tiltable-rotor aerial vehicle," *IEEE Robotics Automation Magazine*, vol. 25, no. 4, pp. 34–44, 2018.

[19] D. Invernizzi, M. Giurato, P. Gattazzo, and M. Lovera, "Full pose tracking for a tilt-arm quadrotor uav," in *2018 IEEE Conference on Control Technology and Applications (CCTA)*, pp. 159–164, 2018.

[20] A. M. Ahmed and J. Katupitiya, "Modeling and control of a novel vectored-thrust quadcopter," *Journal of Guidance, Control, and Dynamics*, vol. 44, no. 7, pp. 1399–1409, 2021.

[21] B. Crowther, A. Lanzon, M. Maya-Gonzalez, and D. Langkamp, "Kinematic analysis

and control design for a nonplanar multirotor vehicle," *Journal of Guidance, Control, and Dynamics*, vol. 34, no. 4, pp. 1157–1171, 2011.

[22] S. Rajappa, M. Ryll, H. H. Bülthoff, and A. Franchi, "Modeling, control and design optimization for a fully-actuated hexarotor aerial vehicle with tilted propellers," in *2015 IEEE international conference on robotics and automation (ICRA)*, pp. 4006–4013, IEEE, 2015.

[23] M. Ryll, H. H. Bülthoff, and P. R. Giordano, "Modeling and control of a quadrotor uav with tilting propellers," in *2012 IEEE International Conference on Robotics and Automation*, pp. 4606–4613, IEEE, 2012.

[24] E. Kaufman, K. Caldwell, D. Lee, and T. Lee, "Design and development of a free-floating hexrotor uav for 6-dof maneuvers," in *2014 IEEE Aerospace Conference*, pp. 1–10, IEEE, 2014.

[25] M. Ryll, D. Bicego, and A. Franchi, "Modeling and control of fast-hex: A fully-actuated by synchronized-tilting hexarotor," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1689–1694, IEEE, 2016.

[26] A. Nikou, G. C. Gavridis, and K. J. Kyriakopoulos, "Mechanical design, modelling and control of a novel aerial manipulator," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4698–4703, IEEE, 2015.

[27] J. J. Wang, *Modeling and Control of a 6-DOF Multi-rotorcraft: Air-Ball.* PhD thesis, 2018.

[28] M. Davoudi Dehkordi and M. Danesh, "Positionable rotor quadrotor: Dynamic modeling and adaptive finite-time sliding-mode controller design," *Journal of Guidance, Control, and Dynamics*, vol. 45, no. 3, pp. 424–433, 2022.

[29] D. Invernizzi, M. Giurato, P. Gattazzo, and M. Lovera, "Comparison of control methods for trajectory tracking in fully actuated unmanned aerial vehicles," *IEEE Transactions on Control Systems Technology*, vol. 29, no. 3, pp. 1147–1160, 2021.

[30] C. Diao, B. Xian, Q. Yin, W. Zeng, H. Li, and Y. Yang, "A nonlinear adaptive control approach for quadrotor uavs," in *2011 8th Asian Control Conference (ASCC)*, pp. 223–228, IEEE, 2011.

[31] Z. T. Dydek, *Adaptive control of unmanned aerial systems*. PhD thesis, Massachusetts institute of Technology, 2010.

[32] A. Chamseddine, Y. Zhang, C.-A. Rabbath, J. Apkarian, and C. Fulford, "Model reference adaptive fault tolerant control of a quadrotor uav," in *Infotech@ Aerospace 2011*, p. 1606, 2011.

[33] A. Lanzon, A. Freddi, and S. Longhi, "Flight control of a quadrotor vehicle subsequent to a rotor failure," *Journal of Guidance, Control, and Dynamics*, vol. 37, no. 2, pp. 580–591, 2014.

[34] J. Stephan, L. Schmitt, and W. Fichter, "Linear parameter-varying control for quadrotors in case of complete actuator loss," *Journal of Guidance, Control, and Dynamics*, vol. 41, no. 10, pp. 2232–2246, 2018.

[35] B. Whitehead and S. Bieniawski, "Model reference adaptive control of a quadrotor uav," in *AIAA Guidance, Navigation, and Control Conference*, p. 8148, 2010.

[36] S. Ristevski, A. T. Koru, T. Yucelen, K. M. Dogan, and J. A. Muse, "Experimental results of a quadrotor uav with a model reference adaptive controller in the presence of unmodeled dynamic," in *AIAA SCITECH 2022 Forum*, p. 1381, 2022.

[37] A. F. Abdul Ghaffar and T. S. Richardson, "Position tracking of an underactuated quadrotor using model reference adaptive control," in *AIAA Guidance, Navigation, and Control Conference*, p. 1388, 2016.

[38] Z. Dydek, A. Annaswamy, and E. Lavretsky, "Combined/composite adaptive control of a quadrotor uav in the presence of actuator uncertainty," in *AIAA Guidance, Navigation, and Control Conference*, p. 7575, 2010.

[39] M. Huang, B. Xian, C. Diao, K. Yang, and Y. Feng, "Adaptive tracking control of underactuated quadrotor unmanned aerial vehicles via backstepping," in *Proceedings of the 2010 American Control Conference*, pp. 2076–2081, IEEE, 2010.

[40] T. X. Dinh and K. K. Ahn, "Adaptive tracking control of a quadrotor unmanned vehicle," *International Journal of Precision Engineering and Manufacturing*, vol. 18, no. 2, pp. 163–173, 2017.

[41] Y. Fan, Y. Cao, and T. Li, "Adaptive integral backstepping control for trajectory tracking of a quadrotor," in *2017 4th International Conference on Information, Cybernetics and Computational Social Systems (ICCSS)*, pp. 619–624, IEEE, 2017.

[42] H. Bouadi and F. Mora-Camino, "Direct adaptive backstepping flight control for quadcopter trajectory tracking," in *2018 IEEE/AIAA 37th Digital Avionics Systems Conference (DASC)*, pp. 1–8, IEEE, 2018.

[43] S. Islam, M. Faraz, R. Ashour, G. Cai, J. Dias, and L. Seneviratne, "Adaptive sliding mode control design for quadrotor unmanned aerial vehicle," in *2015 international conference on unmanned aircraft systems (ICUAS)*, pp. 34–39, IEEE, 2015.

[44] H. Bouadi, A. Aoudjif, and M. Guenifi, "Adaptive flight control for quadrotor uav in the

presence of external disturbances," in *2015 6th International Conference on Modeling, Simulation, and Applied Optimization (ICMSAO)*, pp. 1–6, IEEE, 2015.

[45] K. Eliker and W. Zhang, "Finite-time adaptive integral backstepping fast terminal sliding mode control application on quadrotor uav," *International Journal of Control, Automation and Systems*, vol. 18, no. 2, pp. 415–430, 2020.

[46] J. Zhang, Z. Ren, C. Deng, and B. Wen, "Adaptive fuzzy global sliding mode control for trajectory tracking of quadrotor uavs," *Nonlinear Dynamics*, vol. 97, no. 1, pp. 609–627, 2019.

[47] D. W. Kun and I. Hwang, "Linear matrix inequality-based nonlinear adaptive robust control of quadrotor," *Journal of Guidance, Control, and Dynamics*, vol. 39, no. 5, pp. 996–1008, 2016.

[48] H. Jafarnejadsani, D. Sun, H. Lee, and N. Hovakimyan, "Optimized l 1 adaptive controller for trajectory tracking of an indoor quadrotor," *Journal of Guidance, Control, and Dynamics*, vol. 40, no. 6, pp. 1415–1427, 2017.

[49] S. Mallikarjunan, B. Nesbitt, E. Kharisov, E. Xargay, N. Hovakimyan, and C. Cao, "L1 adaptive controller for attitude control of multirotors," in *AIAA guidance, navigation, and control conference*, p. 4831, 2012.

[50] P. Mukherjee and S. Waslander, "Direct adaptive feedback linearization for quadrotor control," in *AIAA Guidance, Navigation, and Control Conference*, p. 4917, 2012.

[51] F. A. Goodarzi, D. Lee, and T. Lee, "Geometric adaptive tracking control of a quadrotor unmanned aerial vehicle on se (3) for agile maneuvers," *Journal of Dynamic Systems, Measurement, and Control*, vol. 137, no. 9, p. 091007, 2015.

[52] S. Rajappa, C. Masone, H. H. Bülthoff, and P. Stegagno, "Adaptive super twisting controller for a quadrotor uav," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2971–2977, IEEE, 2016.

[53] D. Mihailescu-Stoica, R. Acuna, and J. Adamy, "High performance adaptive attitude control of a quadrotor," in *2019 18th European Control Conference (ECC)*, pp. 3462–3469, IEEE, 2019.

[54] B.-Y. Lee, H.-I. Lee, and M.-J. Tahk, "Analysis of adaptive control using on-line neural networks for a quadrotor uav," in *2013 13th International Conference on Control, Automation and Systems (ICCAS 2013)*, pp. 1840–1844, IEEE, 2013.

[55] Q. Lin, Z. Cai, Y. Wang, J. Yang, and L. Chen, "Adaptive flight control design for quadrotor uav based on dynamic inversion and neural networks," in *2013 Third International Conference on Instrumentation, Measurement, Computer, Communication and Control*, pp. 1461–1466, IEEE, 2013.

[56] T. Furgiuele, A. Leonessa, D. Doyle, and J. Black, "Position control of an omnidirectional aerial vehicle for simulating free-flyer in-space assembly operations," in *ASCEND 2021*, p. 4100, 2021.

[57] T. Furgiuele, K. Schroeder, D. Doyle, and J. Black, "Attitude control of an omnidirectional aerial vehicle for simulating free-flyer in-space assembly operations," in *AIAA SCITECH 2022 Forum*, p. 0358, 2022.

[58] K. A. Wise, E. Lavretsky, and N. Hovakimyan, "Adaptive control of flight: theory, applications, and open problems," in *2006 American Control Conference*, pp. 6–pp, IEEE, 2006.

[59] M. Steinberg, "Historical overview of research in reconfigurable flight control," *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering*, vol. 219, no. 4, pp. 263–275, 2005.

[60] C. D. Heise, M. Leitão, and F. Holzapfel, "Performance and robustness metrics for adaptive flight control-available approaches," in *AIAA Guidance, Navigation, and Control (GNC) Conference*, p. 5090, 2013.

[61] S. Jacklin, "Closing the certification gaps in adaptive flight control software," in *AIAA Guidance, Navigation and Control Conference and Exhibit*, p. 6988, 2008.

[62] C. Belcastro and C. Belcastro, "On the validation of safety critical aircraft systems, part i: An overview of analytical & simulation methods," in *AIAA Guidance, Navigation, and Control Conference and Exhibit*, p. 5559, 2003.

[63] E. Lavretsky, "Adaptive control: Introduction, overview, and applications," in *Lecture notes from IEEE Robust and Adaptive Control Workshop*, 2008.

[64] Q. Sang and G. Tao, "Gain margins of model reference adaptive control systems," in *2008 7th World Congress on Intelligent Control and Automation*, pp. 2946–2951, IEEE, 2008.

[65] Q. Sang and G. Tao, "Gain margins of adaptive control systems," *IEEE transactions on automatic control*, vol. 55, no. 1, pp. 104–115, 2009.

[66] Z. Dydek, H. Jain, J. Jang, A. Annaswamy, and E. Lavretsky, "Theoretically verifiable stability margins for an adaptive controller," in *AIAA Guidance, Navigation, and Control Conference and Exhibit*, p. 6416, 2006.

[67] J. E. Stellet, "Influence of adaptation gain and reference model parameters on system

performance for model reference adaptive control," *World Academy of science, engineering and technology*, vol. 60, pp. 1768–1773, 2011.

[68] V. Stepanyan and K. Krishnakumar, "Mrac revisited: guaranteed performance with reference model modification," in *Proceedings of the 2010 American Control Conference*, pp. 93–98, IEEE, 2010.

[69] T. E. Gibson, A. M. Annaswamy, and E. Lavretsky, "Adaptive systems with closed-loop reference models: Stability, robustness and transient performance," *arXiv preprint arXiv:1201.4897*, 2012.

[70] T. E. Gibson, A. M. Annaswamy, and E. Lavretsky, "On adaptive control with closed-loop reference models: transients, oscillations, and peaking," *IEEE Access*, vol. 1, pp. 703–717, 2013.

[71] N. Nguyen, A. Ishihara, K. Krishnakumar, and M. Bakhtiari-Nejad, "Bounded linear stability analysis-a time delay margin estimation approach for adaptive control," in *AIAA guidance, navigation, and control conference*, p. 5968, 2009.

[72] V. Stepanyan and K. Krishnakumar, "On the robustness properties of m-mrac," in *Infotech@ Aerospace 2012*, p. 2407, 2012.

[73] A. Melo Pacheco, "Time delay margin analysis for model reference adaptive flight control laws: A bounded linear stability approach and application to aeroservoelasticity models," 2018.

[74] A. Dorobantu, "Time delay margin analysis for adaptive flight control laws.," 2010.

[75] A. Hodel, M. Whorton, and J. Zhu, "Stability metrics for simulation and flight-software assessment and monitoring of adaptive control assist compensators," in *AIAA Guidance, Navigation and Control Conference and Exhibit*, p. 7005, 2008.

[76] A. Dorobantu, P. Seiler, and G. J. Balas, "Time-delay margin analysis for an adaptive controller," *Journal of Guidance, Control, and Dynamics*, vol. 35, no. 5, pp. 1418–1425, 2012.

[77] A. Annaswamy, J. Jang, and E. Lavretsky, "Stability margins for adaptive controllers in the presence of time-delay," in *AIAA Guidance, Navigation and Control Conference and Exhibit*, p. 6659, 2008.

[78] J. Jang *et al.*, *Adaptive control design with guaranteed margins for nonlinear plants*. PhD thesis, Massachusetts Institute of Technology, 2009.

[79] H. S. Hussain, Y. Yildiz, M. Matsutani, A. M. Annaswamy, and E. Lavretsky, "Computable delay margins for adaptive systems with state variables accessible," *IEEE Transactions on Automatic Control*, vol. 62, no. 10, pp. 5039–5054, 2017.

[80] M. S. Mahmoud, *Robust control and filtering for time-delay systems*. Marcel-Dekker,Inc., 2000.

[81] E. Lavretsky and K. A. Wise, "Robust adaptive control," in *Robust and adaptive control*, pp. 317–353, Springer, 2013.

[82] L. Eugene, W. Kevin, and D. Howe, *Robust and adaptive control with aerospace applications*. Springer London, 2013.

[83] A. Leonessa, W. M. Haddad, T. Hayakawa, and Y. Morel, "Adaptive control for nonlinear uncertain systems with actuator amplitude and rate saturation constraints," *International Journal of Adaptive Control and Signal Processing*, vol. 23, no. 1, pp. 73–96, 2009.

[84] R. Kristiansen and P. J. Nicklasson, "Satellite attitude control by quaternion-based

backstepping," in *Proceedings of the 2005, American Control Conference, 2005.*, pp. 907–912, IEEE, 2005.

[85] C. A. Deans, *Simulation and Integration of a 6-DOF Controllable Multirotor Vehicle*. PhD thesis, Virginia Tech, 2020.

[86] Dronecode, "Px4 open source autopilot." https://px4.io/. [Online; accessed July 2022].

[87] "Px4 architectural overview." https://docs.px4.io/main/en/concept/architecture.html. [Online; accessed July 2022].

[88] Y. Tan, J. Hu, J. Chang, and H. Tan, "Adaptive integral backstepping motion control and experiment implementation," in *Conference Record of the 2000 IEEE Industry Applications Conference. Thirty-Fifth IAS Annual Meeting and World Conference on Industrial Applications of Electrical Energy (Cat. No. 00CH37129)*, vol. 2, pp. 1081–1088, IEEE, 2000.

[89] Z. Fang and W. Gao, "Adaptive integral backstepping control of a micro-quadrotor," in *2011 2nd International conference on intelligent control and information processing*, vol. 2, pp. 910–915, IEEE, 2011.

[90] "Px4 development." https://docs.px4.io/main/en/development/development.html. [Online; accessed July 2022].