UNIVERSITY OF OKLAHOMA

GRADUATE COLLEGE

ROBUST ADAPTIVE CONTROL LAWS FOR TILT-ROTOR QUADCOPTERS SUBJECT TO USER-DEFINED CONSTRAINTS

A THESIS

SUBMITTED TO THE GRADUATE FACULTY

in partial fulfillment of the requirements for the

Degree of

MASTER OF SCIENCE

By

ROBERT BLAKE ANDERSON

Norman, Oklahoma

2019

ROBUST ADAPTIVE CONTROL LAWS FOR TILT-ROTOR QUADCOPTERS SUBJECT TO USER-DEFINED CONSTRAINTS

A THESIS APPROVED FOR THE

SCHOOL OF AEROSPACE AND MECHANICAL ENGINEERING

BY

Acknowledgements

First and foremost, I'd like to thank my advisor, Dr. Andrea L'Afflitto. His passion for his work and dedication to his students is invaluable to all of us in his lab. He has mentored me in many academic disciplines from performing research, writing papers, giving presentations, writing proposals, and many others. Perhaps more importantly has been his mentorship outside of research. Several times, I have come to a crossroads in my career, and each time Dr. L'Afflitto has been there to act as both an advisor and a mentor. I wouldn't be where I am today if it wasn't for his guidance, and I look forward to my next journey of completing a PhD under him.

Furthermore, I'd like to extend my sincere gratitude to Dr. Wei Sun and Dr. Choon Yik Tang for taking time to be a part of my thesis committee. I greatly appreciate all of your help both in the classroom and in the revising of this thesis.

Finally, I'd like to thank those who worked with me in the lab, Tim Blackford, J.P. Burke, Cole Domann, Josh Karinshak, Julius Marshall, Karen Martinez Soto, and many others. I'll never forget those long hours completing those homework assignments we thought were too hard, finishing those projects we thought were too big, and obtaining those research results we thought were impossible. This thesis and its results are only possible with your support, and I'll be forever thankful for each and every one of you.

This thesis was partly supported by the National Science Foundation through Grant no. 1700640 , the US Army Research Lab through Grant no. 40304747, and DARPA under grant no. D18AP00069. I wish to express my gratitude to these sponsors for their continuous support

Contents

Abstract **X**

1 Introduction **1**
- 1.1 State of the Art in UAV Control . . . 1
- 1.2 Original Contribution and Organization of This Thesis . . . 3
- 1.3 Notation and Mathematical Preliminaries . . . 7

2 Model Reference Adaptive Control (MRAC) **10**
- 2.1 Classical MRAC . . . 10
- 2.2 Output Feedback Robust MRAC Based on the e-Modification . . . 15
- 2.3 Robust MRAC with User-Defined Constraints . . . 20

3 Equations of Motion of a Tilt-Rotor Quadcopter **31**
- 3.1 Problem Statement . . . 31
- 3.2 Kinematic and Dynamic Equations . . . 33
- 3.3 Analysis of Equations of Motion . . . 37

4 Control Design for a Tilt-Rotor Quadcopter **43**
- 4.1 Underactuation of Mechanical Systems . . . 43
- 4.2 Control Strategy for Tilt-Rotor Quadcopters . . . 44
- 4.3 Control Laws for Tilt-Rotor Quadcopters . . . 48
- 4.4 Realization of Control Inputs . . . 55

5 Experiment Design and Results **59**
- 5.1 Tilt-rotor Components and Parameters . . . 59
  - 5.1.1 Frame and Body . . . 59
  - 5.1.2 Pixhawk Autopilot and Odroid XU4 . . . 60
  - 5.1.3 Propulsion System . . . 63
  - 5.1.4 Battery . . . 65
  - 5.1.5 Servos for Thrust Vectoring . . . 66
  - 5.1.6 Actuation Selection – Mux and Maestro Boards . . . 67
- 5.2 Experimental Results . . . 69

6 Conclusion **74**
- 6.1 Future Directions . . . 76

A Appendix 1 **78**
- A.1 Setting Up a New Odroid . . . 78
  - A.1.1 Eclipse Settings . . . 80
  - A.1.2 Installing Flight Code . . . 81
  - A.1.3 Set Odroid to Automatically Login When Powering On . . . 82

A.1.4 Setting Odroid to Performance Mode 83
A.1.5 Set IP Address 83
A.1.6 Odroid to Pixhawk Cable 84
A.2 Control Code 85

List of Figures

1.1 Standard quadrotor platform used for testing at the Advanced Control Systems Lab. 1
1.2 Tilt-rotor quadcopter carrying a dangling payload. 2

3.1 Schematic representation of the tilt-rotor quadrotor pulling a cart. . . 32

5.1 Computer Aided Design (CAD) model of the main frame of the tilt-rotor quadcopter 59
5.2 The Pixhawk Autopilot used on the platform. 61
5.3 The current setup for Odroid, Pixhawk, and Vicon communication. Vicon cameras deduce the position and velocity of the drone and a ground station computer streams that data over WiFi to the Odroid onboard. The Odroid receives the attitude data from the Pixhawk's IMU and computes the control inputs. 62
5.4 The Castle DMR 30-40 electronic speed controllers (ESC's) used for this vehicle. 63
5.5 Tiger-Motor MN2212 920KV motors. 64
5.6 The thrust curve fitting plot showing the thrust coefficient, k, was found using bench top thrust data provided by the manufacturer of the motors. 65
5.7 The 14.8V 3250mAh battery from MaxAmps chosen for this setup. . 66
5.8 The AX-18A Dynamixel servos mounted on the tilt-rotor. 66
5.9 The usb2Dynamixel cable that allows for connecting the Dynamixel servos to a USB port on a computer. 67
5.10 The mux board used is custom built by the Army Research Lab. It is used to select whether to use the custom flight controller on the Odroid or the Pixhawk flight controller for vehicle actuation. 67
5.11 This diagram shows how the mux and maestro boards along with user inputs from the DSM radio determine which flight controller has control of the vehicle. 68
5.12 This figure shows both the vehicle and reference trajectories for the experiment. 69
5.13 This plot shows the pitch angle, $\theta(\cdot)$, for the mission. The $\theta(\cdot)$ angle oscillates around the desired point, 0, and never exceeds 6.5 degrees. . 70
5.14 Plot of the norm of the trajectory tracking error norm, $\|e\|$, which shows that the proposed control law outperforms the classical versions of Model Reference Adaptive Control. 72

5.15 A plot showing the evolution of the constraint function, $h(e^{\mathrm{T}}Me, \Delta K\Gamma^{-1}\Delta K^{\mathrm{T}})$, throughout the experiment. For the proposed control law, the constraint function stays positive for the duration of the mission, whereas both of the classical control techniques violate the constraint and go below 0. 73

A.1 Performance Mode Settings 83
A.2 Pixhawk to Odroid Connection Cables 84
A.3 FTDI Wiring Diagram 84

Abstract

This thesis focuses on Model Reference Adaptive Control (MRAC) and its application to a tilt-rotor quadcopter. After formulating two standard MRAC approaches, this thesis proposes a robust model reference adaptive control law that guarantees satisfactory trajectory following for the nonlinear dynamical system despite parametric, matched, and unmatched uncertainties. This control law is unique for its ability to exploit barrier Lyapunov functions and guarantee user-defined constraints both on the trajectory tracking error and the adaptive gains at all time. The proposed robust control law is then applied to design a control law for a tilt-rotor quadcopter with H-configuration with unknown and unsteady center of mass and matrix of inertia due to the presence of poorly modeled and dangling payloads. The tilt-rotor quadcopter equations of motion are presented and thoroughly analyzed. A novel approach is proposed to model the coupling between the translational and rotational dynamics as matched uncertainties, and a control strategy is developed to overcome the vehicle's underactuation. A tilt-rotor is designed and all of the components are presented and discussed. A challenging experiment where a tilt-rotor quadcopter pulls an unknown cart is performed and the results show the applicability of the proposed theoretical framework.

Key Words: Model reference adaptive control, robust adaptive control, tilt-rotor quadcopter, barrier Lyapunov functions, constrained control.

Chapter 1: Introduction

1.1. State of the Art in UAV Control

Quadrotors are small unmanned aerial vehicles (UAVs) which have been utilized in many diverse scenarios for the past few decades. These scenarios include typical surveillance or picture taking, crop monitoring, search and rescue missions, and many others. An important component to all of these scenarios is that the UAV is typically flying in open space, in good weather conditions, and the specifications of the vehicle such as its mass and inertia are well estimated. The small UAV of the future should not only be able to successfully perform all of the previous scenarios, which are only passive in the environment, but the new UAV should be an active part of the environment. This could include picking up and moving *unknown* objects, interacting at or near hard surfaces introducing the aerodynamic “wall effect”, or being tasked with pulling or placing cables within an environment.

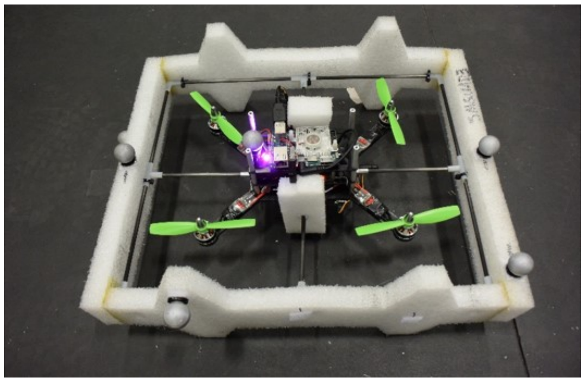


Figure 1.1: Standard quadrotor platform used for testing at the Advanced Control Systems Lab.

Current autopilots almost exclusively run proportional-integral-derivative (PID) control laws that are pre-tuned for each mission scenario. This means that the vehicle would is assumed to have a known payload, a known matrix of inertia, and a known center of gravity. This architecture allows for success in the very specific scenarios they are designed for, but these simplifying assumptions undermine the vehicle's capability to operate in off-nominal conditions. This includes scenarios wherein the UAV transports some unknown dangling payload, or the propulsion system is damaged. The future autopilot requires robust control algorithms to guarantee performance for these challenging missions.

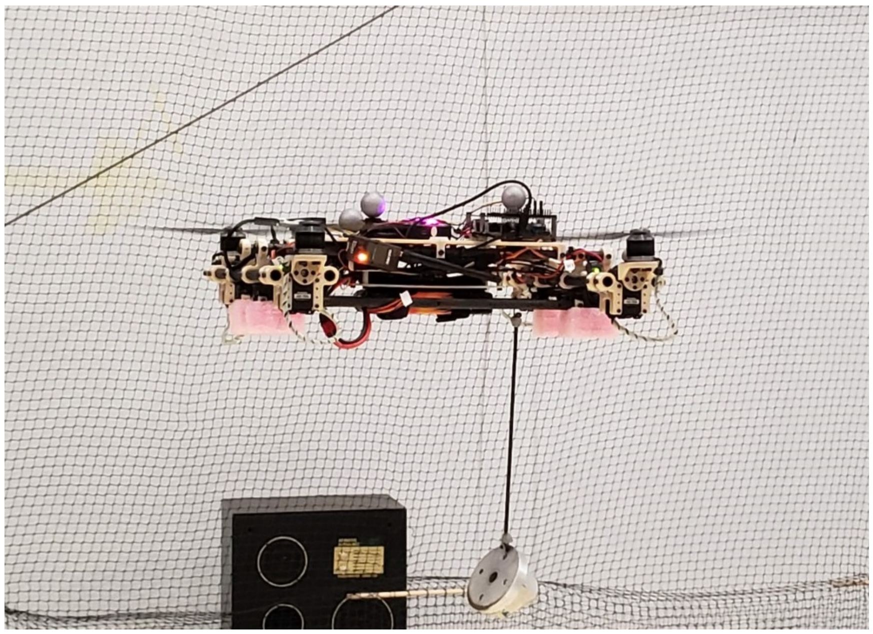


Figure 1.2: Tilt-rotor quadcopter carrying a dangling payload.

In recent years, numerous authors such as [1–4] as well as many others, have employed nonlinear robust control techniques such adaptive backstepping control, sliding mode control (SMC), and model reference adaptive control (MRAC) for quadrotor autopilots. These techniques account for inaccurate modeling assumptions and damages to the motors or propellers. Although these nonlinear control techniques undeniably provide advantages over classical autopilots with underlying PID control algorithms, it is apparent that there are still ample margins for improvements. For instance, backstepping control requires perfect knowledge of the UAV's dynamics [5]. Sliding mode control is usually affected by chattering [5, Ch. 14], and tuning autopilots based on sliding mode control laws may be daunting [6]. Lastly, MRAC is particularly effective, but is also characterized by undesired spikes in the control input during the transient period [7, Ch. 13].

An additional downside to the modern quadrotor design is its underactuation. Due to all of the propellers lying in a single plane, the vehicle is incapable of producing forces in the lateral directions without pitching or rolling. For some types of transport or manipulation missions, it may be desirable to have a vehicle that can maintain zero pitch and roll angle while moving to the destination. For this reason, tilt-rotor quadcopters have been introduced. The motors of these vehicles are typically mounted on servos for thrust vectoring to increase the number of controllable degrees of freedom. It is apparent that tilt-rotor quadcopters can be reduced to classical quadcopters by locking all of the propellers to a tilt-angle of zero.

1.2. Original Contribution and Organization of This Thesis

The original contribution of this thesis is multifold. In particular, we present a detailed analysis of the equations of motion of a tilt-rotor quadcopter with H-configuration revealing nonlinear effects unknown in the current literature. Successively, a nonlinear robust control technique is presented to overcome the aforementioned limitations of classical backstepping control, sliding mode control, and model reference adaptive control. Finally, the original control technique is applied to the autopilot design problem for tilt-rotor UAVs performing challenging tasks, such as transporting unknown dangling payloads. All of the original results are discussed in detail in [6, 8, 9]. See the following details for the structure of this thesis.

In Chapter 2, multiple MRAC control algorithms are formulated. The classical MRAC technique is first presented. In its original formulation, this technique is designed to regulate time invariant dynamical systems, and for this reason classical MRAC is insufficient for trying to account for time-varying parameters such as the inertial counter-torque. Moreover, the classical MRAC is robust to parametric and matched uncertainties, but it is not robust to unmatched uncertainties. For this reason, a robust MRAC based on the e-modification of [10] is presented. This technique is robust to unmatched uncertainties, which could include unmodeled aerodynamic disturbances. This control law, however, only guarantees that the trajectory tracking error remains bounded at all times, and this bound can only be estimated in a conservative manner [11]. For this reason, a novel robust MRAC law is proposed. This control law exploits barrier Lyapunov functions [12] to guarantee user-defined constraints on both the trajectory tracking error and the adaptive gains at all time. As previously remarked, classical MRAC and several of its robust formulations, such as the dead-zone modification [13], the σ-modification [14], and the e-modification of MRAC [10], are affected by high and rapidly varying oscillations of the control input in the transient phase.

A classical approach to bound adaptive gains is to employ the projection operator

[15, 16], which constrains the adaptive gains within some user-defined convex set. As discussed in [11], using the projection operator one can only guarantee uniform ultimate boundedness of the trajectory tracking error and the ultimate bounds can only be estimated in a conservative manner. In this thesis, we present an original robust MRAC law that allows to impose user-defined constraints on the adaptive gains and trajectory tracking error that does not require that the constraint set be convex. Moreover, the proposed framework allows to correlate the constraints on the trajectory tracking error and the adaptive gains explictly. If no constraint is imposed, then the proposed MRAC law reduces to the e-modification of MRAC [10]. Alternatively, if unmatched uncertainties are neglected, then the proposed MRAC law reduces to the one presented in [17]. Only recently, barrier Lyapunov functions have been utilized to impose user-defined constraints on the closed-loop system's trajectory tracking error, while the adaptive gains are constrained employing the projection operator [11].

In Chapter 3, the full nonlinear equations of motion of a tilt-rotor with H-configuration are derived and thoroughly discussed , including discussing a previously unnamed term, which we named the *tilt-rotor gyroscopic effect* which arises from the motion of the servos during thrust vectoring. This formulation assumes that the reference frame of the vehicle is conveniently placed at the center of the vehicle, which *does not* necessarily coincide with the location of the center of mass. This assumption implies that the generalized mass matrix is *not diagonal*, and hence the translational and rotational dynamic equations are coupled.

In Chapter 4, a control law is designed for tilt-rotor quadcopters with H-configuration under the assumption that the vehicle's mass is known and both its inertia matrix and the location of its center of mass are unknown. Specifically, considering the propellers' thrust force and moment of the thrust force as control inputs, it is proven that tilt-rotor quadcopters with H-configuration are underactuated [18, Def. 2.9], and a control strategy is proposed, whereby the vehicle's reference position, pitch angle, and yaw angle can be arbitrarily defined by the user and the reference roll angle is deduced accordingly using an "outer loop." Successively, the robust MRAC law presented in the first part of this paper is applied to the UAV's feedback-linearized equations of motion. The performance of MRAC laws, including the one proposed in this paper, is degraded by unmatched uncertainties, but not parametric and matched uncertainties [7, Ch. 9, 11]. If the vehicle's translational and rotational dynamic equations were considered as decoupled, then the control design process would be considerably simplified, but the coupling between translational and rotational dynamics would increase the detrimental effect of unmatched uncertainties. In order to maximize the robustness of the proposed control algorithm for tilt-rotor quadcopters, the regressor vector has been designed to capture the coupling of the translational and rotational dynamic equations, nonlinearities in the rotational dynamics of the vehicle, and systematic errors in the estimation of the aircraft parameters as matched uncertainties. Existing results on the control synthesis for tilt-rotor aircraft assume perfect knowledge of the location of the vehicle's center of mass and rely on feedback linearization together with proportional-derivative control [19–23], backstepping control [24, 25], sliding-mode control [26], hierarchical control [27], neural networks [28], or optimal control [29]. To the authors' best knowledge, a robust MRAC architecture has not been applied to design autopilots for tilt-rotor quadcopters and the proposed approach to reduce the effect of unmatched uncertainties is unprecedented.

In Chapter 5, our original control laws are applied to regulate a custom built tilt-rotor quadcopter with H-configuration, and the results of the flight experiments are presented. Our UAV includes a Pixhawk autopilot for estimating the attitude of the vehicle, Dynamixel servos for thrust vectoring capabilities, and T-Motor MN212 motors mounted with 9x3 propellers for actuation. This vehicle was designed to produce enough thrust for pulling a heavy cart in the experiment whose mass is $6.2kg$; more than 3 times than the UAV's mass. This cart is connected to the UAV by means of a flexible wire, and its contribution to the vehicle's inertial properties is purposefully unmodeled to test the robustness of our original MRAC law and its ability to meet user-defined constraints. Experimental results also show that our new MRAC law outperforms both the classical MRAC and the e-modification of MRAC in regards to trajectory tracking error. In addition, our new control law is able to verify the *a priori imposed user-defined constraints* on the tracking error and estimated adaptive gains while both of the classical variations of MRAC violate the constraints.

1.3. Notation and Mathematical Preliminaries

In this section, notation definitions are established, and some basic results are reviewed. Let $\mathbb{N}$ denote the set of *positive integers*, $\mathbb{R}$ denote the set of *real numbers*, $\mathbb{R}^n$ the set of $n \times 1$ *real column vectors*, and $\mathbb{R}^{n\times m}$ the set of $n \times m$ *real matrices.* The ith vector of the canonical basis in $\mathbb{R}^n$ is denoted by $\mathbf{e}_{i,n}$ or $\mathbf{e}_i$, the *identity matrix* in $\mathbb{R}^{n\times n}$ is denoted by $\mathbf{1}_n$, the *zero* $n \times m$ *matrix* in $\mathbb{R}^{n\times m}$ is denoted by $0_{n\times m}$ or 0. The *interior of the set* $\mathcal{S} \subseteq \mathbb{R}^{n\times m}$ is denoted by $\mathring{\mathcal{S}}$ and the *boundary of the set* $\mathcal{S}$ is denoted by $\partial\mathcal{S}$. We write $\|\cdot\|$ both for the *Euclidean vector norm* and the corresponding *equi-induced matrix norm*, and we define $\|B\|_{\mathrm{F},L} \triangleq \left[\mathrm{tr}\left(BLB^{\mathrm{T}}\right)\right]^{\frac{1}{2}}$ as the *weighted Frobenius norm* of $B \in \mathbb{R}^{n\times m}$, where $L \in \mathbb{R}^{m\times m}$ is symmetric and positive-definite; if $L = I$, then we write $\|B\|_{\mathrm{F}}$. The *transpose* of $B \in \mathbb{R}^{n\times m}$ is denoted by B^{T}, the *rank* of B is denoted by $\mathrm{rank}(B)$, the *spectrum* of $A \in \mathbb{R}^{n\times n}$ is denoted by $\mathrm{spec}(A)$, the *trace* of A is denoted by $\mathrm{tr}(A)$, and the smallest eigenvalue of the symmetric matrix $Q \in \mathbb{R}^{n\times n}$ is denoted by $\lambda_{\min}(Q)$. The *Kronecker product* of $A \in \mathbb{R}^{n\times m}$ and $B \in \mathbb{R}^{p\times q}$ is denoted by $A \otimes B$ [30, Def. 7.1.2]. Given $x = [x_1, x_2, x_3]^{\mathrm{T}} \in \mathbb{R}^3$, we define the *cross-product operator* $(\cdot)^{\times}$ as $x^{\times} \triangleq \begin{bmatrix} 0 & -x_3 & x_2 \\ x_3 & 0 & -x_1 \\ -x_2 & x_1 & 0 \end{bmatrix}$.

The *Fréchet derivative* of the continuously differentiable function $V : \mathcal{D} \to \mathbb{R}$ at $x \in \mathcal{D} \subseteq \mathbb{R}^n$ is denoted by $V'(x) \triangleq \frac{\partial V(x)}{\partial x}$. The *Fréchet derivative* of the continuously differentiable function $h : \mathcal{X} \to \mathbb{R}$ at $X \in \mathcal{X} \subseteq \mathbb{R}^{n\times m}$ is given by [31, Ch. 5], [32]

$$\frac{\partial h(X)}{\partial X} \triangleq \begin{bmatrix} \frac{\partial h(X)}{\partial X_{1,1}} & \cdots & \frac{\partial h(X)}{\partial X_{1,m}} \\ \vdots & \ddots & \vdots \\ \frac{\partial h(X)}{\partial X_{n,1}} & \cdots & \frac{\partial h(X)}{\partial X_{n,m}} \end{bmatrix}, \tag{1.1}$$

where $X_{i,j}$ denotes the element of X on the ith row and jth column. Moreover, given $X : [t_0, \infty) \to \mathcal{X}$, it holds that [31, Ch. 5], [32]

$$\dot{h}(X(t)) = \mathrm{tr}\left(\dot{X}(t)\frac{\partial h(X(t))}{\partial X^{\mathrm{T}}}\right), \qquad t \geq t_0. \tag{1.2}$$

Theorem 1.1 *If $A \in \mathbb{R}^{n\times m}$ and $b \in \mathbb{R}^m$, then*

$$Ab = M_W(b,n)W_M(A), \tag{1.3}$$

where

$$M_W(b,n) \triangleq \left(b^{\mathrm{T}} \otimes \mathbf{1}_n\right) \in \mathbb{R}^{n\times nm}, \tag{1.4}$$

$$W_M(A) \triangleq \sum_{i=1}^{n} \left[\mathbf{e}_{i,m} \otimes (A\mathbf{e}_{i,m})\right] \in \mathbb{R}^{nm}. \tag{1.5}$$

Proof: Let $b = [b_1, \ldots, b_m]^{\mathrm{T}}$ and $A = [A_1, \ldots A_m]$, where $A_i \in \mathbb{R}^n$ denotes the ith column of A. Then, $M_W(b,n) = [b_1\mathbf{1}_n, \ldots, b_m\mathbf{1}_n]$, $W_M(A) = [A_1^{\mathrm{T}}, \ldots, A_m^{\mathrm{T}}]^{\mathrm{T}}$, and the result can be verified by direct substitution. ■

Chapter 2: Model Reference Adaptive Control (MRAC)

2.1. Classical MRAC

In this section, the standard MRAC framework as presented by [7] is formulated. Consider the nonlinear time-varying system of the form,

$$\dot{x}(t) = Ax(t) + B\Lambda\left[u(t) + \Theta^{\mathrm{T}}\Phi(x(t))\right], \qquad x(0) = x_0, \qquad t \geq 0, \tag{2.1}$$

where $x(t) \in \mathcal{D} \subseteq \mathbb{R}^n$ denotes the *system state*, $t \geq 0$, $u(t) \in \mathbb{R}^m$ denotes the *control input*, $A \in \mathbb{R}^{n\times n}$ is *unknown*, $B \in \mathbb{R}^{n\times m}$, $\Lambda \in \mathbb{R}^{m\times m}$ is diagonal, positive-definite, and *unknown*, $\Theta \in \mathbb{R}^{N\times m}$ is *unknown*, and $\Phi(x(t)) : \mathbb{R}^n \to \mathbb{R}^N$ is the Lipschitz continuous *regressor vector*. The uncertainty of Λ is used to model inaccuracies or failures in the control system. While A is considered as unknown, its structure is usually known, and it can be assumed that the pair $(A, B\Lambda)$ is controllable.

Introduce the reference model given by,

$$\dot{x}_{\mathrm{ref}}(t) = A_{\mathrm{ref}}x_{\mathrm{ref}}(t) + B_{\mathrm{ref}}r(t), \qquad x_{\mathrm{ref}}(0) = x_{\mathrm{ref},0}, \qquad t \geq 0, \tag{2.2}$$

where $x_{\mathrm{ref}}(t) \in \mathbb{R}^n$ is the *reference model system state*, $t \geq 0$, $A_{\mathrm{ref}} \in \mathbb{R}^{n\times n}$ is Hurwitz, $B_{\mathrm{ref}} \in \mathbb{R}^{n\times m}$ is such that the pair $(A_{\mathrm{ref}}, B_{\mathrm{ref}})$ is controllable, and $r(t) \in \mathbb{R}^m$ is bounded.

The goal is to design a state feedback adaptive control law, $u(\cdot)$, such that the state tracking error, $e(t) \triangleq x(t) - x_{\mathrm{ref}}(t)$, $t \geq 0$, globally uniformly asymptotically converges to zero while all signals in the closed-loop system stay uniformly bounded. Assuming both A and Λ are perfectly known, the ideal control law could be calculated as,

$$\phi_{ideal}(\pi, K) = K\pi, \tag{2.3}$$

where $K \triangleq \left[K_x^{\mathrm{T}}, K_r^{\mathrm{T}}, -\Theta^{\mathrm{T}}\right] \in \mathbb{R}^{m\times(n+m+N)}$ denotes the control gain, $K_x \in \mathbb{R}^{n\times m}$, $K_r \in \mathbb{R}^{m\times m}$, $\Theta \in \mathbb{R}^{N\times n}$, and $\pi(t) \triangleq \left[x^{\mathrm{T}}(t), r^{\mathrm{T}}(t), \Phi^{\mathrm{T}}(x(t))\right]^{\mathrm{T}} \in \mathbb{R}^{n+m+N}$. In this case, the closed-loop system becomes,

$$\dot{x}(t) = \left(A + B\Lambda K_x^{\mathrm{T}}\right)x(t) + B\Lambda K_r^{\mathrm{T}} r(t), \qquad x(0) = x_0, \qquad t \geq 0. \tag{2.4}$$

Comparing closed-loop system, (2.4), with the reference model dynamics, (2.2), the matching conditions are generated as,

$$A_{\mathrm{ref}} = A + B\Lambda K_x^{\mathrm{T}}, \tag{2.5}$$

$$B_{\mathrm{ref}} = B\Lambda K_r^{\mathrm{T}}. \tag{2.6}$$

If the matching conditions hold, then the closed-loop system is identical to the reference model and asymptotic tracking is guaranteed.

However, in most cases A and Λ are not fully known. In practice, the structure of A is usually known and the matrices, $(A_{\mathrm{ref}}, B_{\mathrm{ref}})$, are designed such that the matching conditions have at least one ideal solution. Assuming the ideal pair, (K_x, K_r), exists, consider the feedback control law

$$\phi(\pi, \hat{K}) = \hat{K}\pi, \tag{2.7}$$

where $\hat{K} : [t_0, \infty) \to \mathbb{R}^{m\times(n+m+N)}$ denotes the adaptive gain matrices. Defining the parameter estimation error as $\Delta K(t) \triangleq \hat{K}(t) - K$, the error tracking dynamics become,

$$\dot{e}(t) = A_{\mathrm{ref}}e(t) + B\Lambda\Delta K(t)\pi(t), \qquad e(t_0) = e_0, \qquad t \geq 0. \tag{2.8}$$

Furthermore, the control law (2.7) is used where the adaptive gains are found using the adaptive gain equation,

$$\dot{\hat{K}}^{\mathrm{T}}(t) = -\Gamma\pi(t)e^{\mathrm{T}}(t)PB, \qquad \hat{K}(t_0) = \hat{K}_0, \qquad t \geq 0, \tag{2.9}$$

where $\Gamma = \mathrm{blockdiag}\{\Gamma_x, \Gamma_r, \Gamma_\theta\} \in \mathbb{R}^{(n+m+N)\times(n+m+N)}$, where $\Gamma_x \in \mathbb{R}^{n\times n}$, $\Gamma_r \in \mathbb{R}^{m\times m}$, and $\Gamma_\theta \in \mathbb{R}^{N\times N}$ are user-defined and positive-definite, and $P \in \mathbb{R}^{n\times n}$ is the positive-definite solution to the Lyapunov equation

$$0 = A_{\mathrm{ref}}^{\mathrm{T}}P + PA_{\mathrm{ref}} + Q, \tag{2.10}$$

where $Q \in \mathbb{R}^{n\times n}$ is user-defined and positive-definite.

Theorem 2.1 *Given a nonlinear time-varying dynamical system with dynamics* (2.1) *and tracking error dynamics* (2.8), *with control law* (2.7) *and adaptive update law* (2.9), *the trajectory tracking error dynamics are globally, uniformly asymptotically* *stable.*

Proof: Consider the following Lyapunov function candidate,

$$V(e,\Delta K) = e^{\mathrm{T}}Pe + \mathrm{tr}\left([\Delta K \Gamma \Delta K^{\mathrm{T}}]\Lambda\right), \quad (e,\Delta K) \in \mathbb{R}^n \times \mathbb{R}^{m\times(n+m+N)}, \tag{2.11}$$

where tr(·) denotes the trace operator. Taking the time derivative of $V(\cdot,\cdot)$ gives,

$$\dot{V}(e,\Delta K) = \dot{e}^{\mathrm{T}}Pe + e^{\mathrm{T}}P\dot{e} + 2\mathrm{tr}\left(\left[\Delta K \Gamma^{-1}\dot{\hat{K}}^{\mathrm{T}}\right]\Lambda\right), \tag{2.12}$$

and substituting in the error dynamics (2.8), we obtain that

$$\begin{aligned}\dot{V}(e,\Delta K) = {} & \left(A_{\mathrm{ref}}e + B\Lambda\left(\Delta K\pi\right)\right)Pe + e^{\mathrm{T}}P\left(A_{\mathrm{ref}}e + B\Lambda\left(\Delta K^{\mathrm{T}}\pi\right)\right) \\ & + 2\mathrm{tr}\left(\left[\Delta \mathrm{K}\Gamma^{-1}\dot{\hat{\mathrm{K}}}^{\mathrm{T}}\right]\Lambda\right),\end{aligned} \tag{2.13}$$

$$= e^{\mathrm{T}}\left(A_{\mathrm{ref}}^{\mathrm{T}}P + PA_{\mathrm{ref}}\right)e + 2e^{\mathrm{T}}PB\Lambda\Delta K\pi + 2\mathrm{tr}\left(\left[\Delta \mathrm{K}\Gamma^{-1}\dot{\hat{\mathrm{K}}}^{\mathrm{T}}\right]\Lambda\right), \tag{2.14}$$

and applying (2.10) it holds that,

$$\dot{V}(e,\Delta K) = -e^{\mathrm{T}}Qe + 2e^{\mathrm{T}}PB\Lambda\Delta K\pi + 2\mathrm{tr}\left(\left[\Delta \mathrm{K}\Gamma^{-1}\dot{\hat{\mathrm{K}}}^{\mathrm{T}}\right]\Lambda\right). \tag{2.15}$$

Now, recall a property of the tr(·) operator where given two matrices A, B of appropriate dimensions,

$$\mathrm{tr}(AB) = \mathrm{tr}(BA). \tag{2.16}$$

Using this trace identity, the second term of (2.15) can be written as,

$$2e^{\mathrm{T}}PB\Lambda\Delta K\pi = 2\mathrm{tr}\left(\Delta K\pi e^{\mathrm{T}}PB\Lambda\right) \tag{2.17}$$

and the derivative of $V(\cdot,\cdot)$ can then be written as,

$$\dot{V}(e,\Delta K) = -e^{\mathrm{T}}Qe + 2\mathrm{tr}\left(\Delta K\left[\pi e^{\mathrm{T}}PB + \Gamma^{-1}\dot{\hat{K}}^{\mathrm{T}}\right]\Lambda\right), \tag{2.18}$$

and by applying the adaptive law (2.9), the equation becomes,

$$\dot{V}(e,\Delta K) = -e^{\mathrm{T}}Qe, \tag{2.19}$$

where it has been shown that $\dot{V}(e,\Delta K)$ is negative semi-definite in its arguments, that is,

$$\dot{V}(e,\Delta K) \leq 0, \qquad (e,\Delta K) \in \mathbb{R}^n \times \mathbb{R}^{m\times(n+m+N)}, \tag{2.20}$$

This shows that the closed-loop error dynamics are uniformly stable, so the adaptive law, $K(\cdot)$, the tracking error, $e(\cdot)$, and the estimation error, $\Delta K(\cdot)$, are uniformly bounded. Combining this statement with the pre-defined assumptions that $r(\cdot)$ is bounded and A_{ref} is Hurwitz, it can be stated that $x(\cdot)$ is bounded and the control input $u(\cdot)$ is bounded. Taking the second derivative of $V(\cdot,\cdot)$ gives,

$$\ddot{V}(e,\Delta K) = -2e^{\mathrm{T}}Q\dot{e}, \qquad (e,\Delta K) \in \mathbb{R}^n \times \mathbb{R}^{m\times(n+m+N)}, \tag{2.21}$$

which is bounded since $e(\cdot), \dot{e}(\cdot)$ are both bounded. Since it is known that $V(\cdot,\cdot)$ is bounded from below and $\dot{V}(e,\Delta K) \le 0$, applying Barbalat's Lemma, [5, Lemma 8.2] gives that

$$\lim_{t\to\infty} \dot{V}(e(t), \Delta K(t)) = 0, \tag{2.22}$$

so the trajectory tracking error, $e(\cdot)$, converges asymptotically to the origin for all $e(0) \in \mathbb{R}^n$. ■

2.2. Output Feedback Robust MRAC Based on the e-Modification

The classical MRAC law presented in Section 2.1 is state-feedback and is not robust to external disturbances. In this section, a form of robust output feedback MRAC is presented based on both the e-modification from [10] and the output feedback control law of [33]. After a finite time transient, this control law guarantees that the plant's measured output will track a given reference signal with bounded error despite uncertainties in the model and the presence of external disturbances.

Consider the nonlinear time-varying plant dynamics given by

$$\dot{x}_p(t) = A_p x_p(t) + B_p \Lambda[u(t) + \Theta^{\mathrm{T}}\Phi(x_p(t))] + \hat{\xi}(t), \qquad x_p(t_0) = x_{p,0}, \qquad t \ge t_0, \tag{2.23}$$

where $x_p(t) \in \mathcal{D}_p \subseteq \mathbb{R}^{n_p}$, $t \ge t_0$, denotes the *trajectory* of the plant, $u(t) \in \mathbb{R}^m$ denotes the *control input*, $A_p \in \mathbb{R}^{n_p \times n_p}$ is *unknown*, $B_p \in \mathbb{R}^{n_p \times m}$, $\Lambda \in \mathbb{R}^{m\times m}$ is diagonal, positive-definite, and *unknown*, $\Theta \in \mathbb{R}^{N\times m}$ is *unknown*, $\Phi : \mathbb{R}^{n_p} \to \mathbb{R}^N$ denotes the Lipschitz continuous *regressor vector*, and $\hat{\xi} : [t_0, \infty) \to \mathbb{R}^{n_p}$ is continuous, *unknown*, and is assumed to be bounded such that $\|\hat{\xi}(t)\| \le \xi_{\max}$, $t \ge t_0$. It is assumed that Λ

is such that the pair $(A_p, B_p\Lambda)$ is controllable and $\Lambda_{\min}1_m \leq \Lambda$, for some $\Lambda_{\min} > 0$. The plant's matched and parametric uncertainties are captured in Λ and $\Theta^{\mathrm{T}}\Phi(x_p)$, $x_p \in \mathcal{D}_p$, which includes possible malfunctions in the control system, and $\hat{\xi}(\cdot)$ captures the unmatched uncertainties, such as external disturbances of the plant.

Also consider the plant's sensor dynamics given by

$$\dot{y}(t) = \epsilon C_p x_p(t) - \epsilon y(t), \qquad y(t_0) = C_p x_{p,0}, \tag{2.24}$$

where $y(t) \in \mathbb{R}^m$ denotes the *system output*, $\epsilon > 0$, and $C_p \in \mathbb{R}^{m \times n_p}$. The sensor dynamics are modeled as linear dynamical systems. The uncontrolled sensor dynamics are exponentially stable and characterized by the parameter $\epsilon > 0$. Introduce the given *reference command* $r : [t_0, \infty) \to \mathbb{R}^m$, which has a continuous first derivative, define $r_2(t) \triangleq \dot{r}(t)$, $t \geq t_0$, and assume that both $r(t), r_2(t)$ are bounded, that is, $\|r(t)\| \leq r_{\max}$, $t \geq t_0$, and $\|r_2(t)\| \leq r_{\max,2}$ for some $r_{\max}, r_{\max,2} > 0$.

The goal of the proposed robust MRAC is to design a feedback control law, $u(\cdot)$, such that after a finite-time transient, the measured output $y(\cdot)$ is able to track the reference signal $r(\cdot)$ with some bounded error, that is, there exist $b > 0$ and $c > 0$, and for every $a \in (0, c)$, there exists a finite-time $T = T(a, c) \geq 0$, such that if $\|y(t_0) - r(t_0)\| \leq a$, then

$$\|y(t) - r(t)\| \leq b, \qquad t \geq t_0 + T. \tag{2.25}$$

For this, an augmented system is created where $n \triangleq n_p + m$ and

$x(t) \triangleq \begin{bmatrix} x_p^{\mathrm{T}}(t), [y(t)-r(t)]^{\mathrm{T}} \end{bmatrix}^{\mathrm{T}} \in \mathbb{R}^n$, $t \geq t_0$, which allows (2.23) and (2.24) to be expressed as

$$\dot{x}(t) = Ax(t)+B\Lambda[u(t)+\Theta^{\mathrm{T}}\Phi(x_p(t))]+\xi(t), \qquad x(t_0) = \begin{bmatrix} x_{p,0} \\ C_p x_{p,0} - r(t_0) \end{bmatrix}, \qquad t \geq 0, \tag{2.26}$$

where $x(t) \in \mathcal{D} \subseteq \mathbb{R}_n$, $\mathcal{D} \in \mathcal{D}_p \times \mathbb{R}^m$, $A \triangleq \begin{bmatrix} A_p & 0_{n_p\times m} \\ \epsilon C_p & -\epsilon 1_m \end{bmatrix}$, $B \triangleq \begin{bmatrix} B_p \\ 0_{m\times m} \end{bmatrix}$, and $\xi(t) \triangleq \begin{bmatrix} 0_{n_p\times m} \\ -1_m \end{bmatrix} \begin{bmatrix} r_2(t)+\epsilon r(t) \end{bmatrix} + \begin{bmatrix} 1_{n_p} \\ 0_{m\times n_p} \end{bmatrix} \hat{\xi}(t)$. Also consider the *reference model* whose dynamics are given by

$$\dot{x}_{\mathrm{ref}}(t) = A_{\mathrm{ref}}x_{\mathrm{ref}}(t) + B_{\mathrm{ref}}r(t), \qquad x_{\mathrm{ref}}(t_0) = \begin{bmatrix} x_{\mathrm{p},0} \\ C_p x_{\mathrm{p},0} - r(t_0) \end{bmatrix}, \qquad t \geq 0, \tag{2.27}$$

where $x_{\mathrm{ref}}(t) \in \mathbb{R}^n$ is the *reference model system state*, $t \geq 0$, $A_{\mathrm{ref}} = \begin{bmatrix} A_{\mathrm{ref},1} & 0_{n_p\times m} \\ 0_{m\times n_p} & A_{\mathrm{ref},2} \end{bmatrix}$, where $A_{\mathrm{ref},1} \in \mathbb{R}^{n_p\times n_p}$ is Hurwitz, $A_{\mathrm{ref},2} \in \mathbb{R}^{m\times m}$ is Hurwitz, and $B_{\mathrm{ref}} \in \mathbb{R}^{n\times m}$ is such that the pair $(A_{\mathrm{ref}}, B_{\mathrm{ref}})$ is controllable.

Define the error as $e \triangleq x(t) - x_{\mathrm{ref}}(t)$, and let the feedback control be of the form

$$\phi(\hat{K},\pi) = \hat{K}\pi, \qquad (x_p, x) \in [0,\infty) \times \mathcal{D}_p \times \mathcal{D}, \tag{2.28}$$

where $K = \begin{bmatrix} K_x^{\mathrm{T}}, K_r^{\mathrm{T}}, -\Theta^{\mathrm{T}} \end{bmatrix} \in \mathbb{R}^{m\times(n+m+N)}$ denotes the control gain where $K_x \in$

$\mathbb{R}^{n\times m}$, $K_r \in \mathbb{R}^{m\times m}$, $\Theta \in \mathbb{R}^{N\times n}$, and $\pi(t) = \Big[x(t), r(t), \Phi(x(t))\Big]^{\mathrm{T}} \in \mathbb{R}^{n+m+N}$. The adaptive law is given by

$$\begin{aligned}\dot{\hat{K}}^{\mathrm{T}}(t) &= -\Gamma\left(\pi(t)e^{\mathrm{T}}(t)PB + \sigma\|B^{\mathrm{T}}Pe(t)\|\hat{K}(t)\right), \\ \hat{K}(t_0) &= \hat{K}_0 \qquad t \geq 0,\end{aligned} \tag{2.29}$$

where the adaptive learning rates, $\Gamma = \text{blockdiag}\{\Gamma_x, \Gamma_r, \Gamma_\theta\} \in \mathbb{R}^{(n+m+N)\times(n+m+N)}$, $\Gamma_x \in \mathbb{R}^{n\times n}$, $\Gamma_r \in \mathbb{R}^{m\times m}$, and $\Gamma_\theta \in \mathbb{R}^{N\times N}$ are user-defined and positive-definite, $\sigma \in \mathbb{R} > 0$, and $P \in \mathbb{R}^{n\times n}$ is the positive definite solution to the Lyapunov equation given by

$$0 = A_{\text{ref}}^{\mathrm{T}}P + PA_{\text{ref}} + Q, \tag{2.30}$$

where $Q \in \mathbb{R}^{n\times n}$ is user-defined and positive definite.

Theorem 2.2 *If K_x and K_r exist such that the following matching conditions are satisfied,*

$$\begin{aligned}A_{\text{ref}} &= A + B\Lambda K_x^{\mathrm{T}}, \\ B_{\text{ref}} &= B\Lambda K_r^{\mathrm{T}},\end{aligned} \tag{2.31}$$

then the nonlinear time-varying system (2.26) *with $u(t) = \phi(\hat{K}(t), \pi(t))$, $t \geq t_0$, tracking error dynamics* (2.32)*, with control law* (2.28) *and adaptive update law* (2.29)*, the system* (2.26) *and the adaptive gains* (2.29) *remain uniformly ultimately bounded for all time and* (2.25) *is verified.*

Proof: Define $\Delta K(t) \triangleq \hat{K}(t) - K$, $t \geq t_0$, and subtract the reference model dynamics

(2.27) from the state dynamics (2.26) to obtain the error dynamics as

$$\dot{e}(t) = A_{\mathrm{ref}} e(t) + B\Lambda\Delta K(t)\pi(t) + \xi(t), \qquad e(t_0) = e_0, \qquad t \geq 0. \tag{2.32}$$

Next, consider the following Lyapunov function candidate

$$V(e, \Delta K) = e^{\mathrm{T}} P e + \mathrm{tr}\left([\Delta K \Gamma \Delta K^{\mathrm{T}}]\Lambda\right), \qquad (e, \Delta K) \in \mathbb{R}^n \times \mathbb{R}^{m\times(n+m+N)}, \tag{2.33}$$

where $\mathrm{tr}(\cdot)$ is the trace operator. Using this candidate, the error dynamics (2.32), and following the arguments of 2.1 and applying Barbalat's Lemma, it can be shown that

$$\dot{V}(e, \Delta K) < 0, \qquad (e, \Delta K) \in \mathbb{R}^n \times \mathbb{R}^{m\times(n+m+N)} \setminus \Omega, \tag{2.34}$$

for some compact set Ω containing the origin. It then follows from [5] that nonlinear dynamical system given by (2.32) with adaptive law (2.29) is uniformly ultimately bounded.

Then let $x_{\mathrm{ref}}(t) = \left[x_{\mathrm{ref},1}^{\mathrm{T}}(t), x_{\mathrm{ref},2}^{\mathrm{T}}(t)\right]^{\mathrm{T}}$, $t \geq t_0$, verify the reference model (2.27), where $x_{\mathrm{ref},1}(t) \in \mathbb{R}^{n_p}$ and $x_{\mathrm{ref},2}(t)$. It follows from the ultimate boundedness of the error dynamics that

$$\|y(t) - r(t) - x_{\mathrm{ref},2}(t)\| \leq \hat{b}, \qquad t \geq T + t_0, \tag{2.35}$$

for some $\hat{b} > 0$, $T \geq 0$, both independent of t_0. Since A_{ref} is block-diagonal and Hurwitz, $r(t)$ is bounded, it follows from (2.27) that $x_{\mathrm{ref},2}(t)$ is uniformly bounded, that is, $\|x_{\mathrm{ref},2}(t)\| \leq b_2$, $t \geq t_0$, for some $b_2 \geq 0$ independent of t_0. Finally, it follows from (2.35) that (2.25) is verified with $b = \hat{b} + b_2$. ■

If the adaptive gains verify (2.29) and $\sigma = 0$, then the control law (2.28) reduces to the standard model reference adaptive control presented in Section 2.1. The main difference is that the standard MRAC does not guarantee uniform ultimate boundedness of the system in the presence of unmatched uncertainty, $\hat{\xi}(\cdot)$, whereas the robust MRAC presented in this section does handle bounded unmatched uncertainties.

2.3. Robust MRAC with User-Defined Constraints

In this section, a robust adaptive control law is presented that steers the trajectory of an unknown dynamical system affected by matched, unmatched, and parametric uncertainties to track the trajectory of a reference dynamical model, while verifying user-defined constraints on both the trajectory tracking error and the adaptive gains at all time. Consider the unknown dynamical system

$$\dot{x}(t) = Ax(t) + B\left[u(t) + \Theta^{\mathrm{T}}\Phi(x(t))\right] + \xi(t), \qquad x(t_0) = x_0, \qquad t \geq t_0, \tag{2.36}$$

where $x(t) \in \mathcal{D} \subseteq \mathbb{R}^n$, $t \geq t_0$, denotes the *system's trajectory*, $u(t) \in U \subseteq \mathbb{R}^m$ denotes the *control input*, $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times m}$, $\Theta \in \mathbb{R}^{N\times m}$, the *regressor vector* $\Phi : \mathcal{D} \to \mathbb{R}^N$ is Lipschitz continuous, and $\xi : [t_0, \infty) \to \mathbb{R}^n$. The matrices A and Θ are unknown and capture parametric and matched uncertainties, and the continuous function $\xi(\cdot)$ is unknown, captures unmatched uncertainties, and is such that $\|\xi(t)\| \leq \xi_{\max}$, $t \geq t_0$, where $\xi_{\max} \geq 0$ is known.

Consider also the *reference dynamical model*

$$\dot{x}_{\rm ref}(t) = A_{\rm ref}x_{\rm ref}(t) + B_{\rm ref}r(t), \qquad x_{\rm ref}(t_0) = x_{{\rm ref},0}, \qquad t \geq t_0, \tag{2.37}$$

where $x_{\rm ref}(t) \in \mathbb{R}^n$, $t \geq t_0$ is the *reference model system state*, $r(t) \in \mathbb{R}^m$ is bounded and denotes the *reference command*, $A_{\rm ref} \in \mathbb{R}^{n\times n}$ is Hurwitz, $B_{\rm ref} \in \mathbb{R}^{n\times m}$, and the pair $(A_{\rm ref}, B_{\rm ref})$ is controllable. It is assumed that there exist $(K_x, K_r) \in \mathbb{R}^{n\times m}\times\mathbb{R}^{m\times m}$ such that

$$A_{\rm ref} = A + BK_x^{\rm T}, \tag{2.38}$$

$$B_{\rm ref} = BK_r^{\rm T}. \tag{2.39}$$

The *matching conditions* (2.38) and (2.39) are standard assumptions in model reference adaptive control and, as discussed in Section 2.1, guarantee that if (2.36) were not affected by matched, unmatched, and parametric uncertainties, then there would exist a control law that guarantees asymptotic convergence to zero of the *trajectory tracking error* $e(t) \triangleq x(t) - x_{\rm ref}(t)$, $t \geq t_0$.

Consider the nonlinear dynamical system (2.36) with $\xi(t) = 0$, $t \geq t_0$, and the reference dynamical model (2.37). Define $K \triangleq \left[K_x^{\rm T}, K_r^{\rm T}, -\Theta^{\rm T}\right]$, where K_x and K_r verify (2.38) and (2.39), and $\pi(t) \triangleq \left[x^{\rm T}(t), r^{\rm T}(t), \Phi^{\rm T}(x(t))\right]^{\rm T}$, $t \geq t_0$. Assume that both A and Θ are known, and there exist $K_x \in \mathbb{R}^{n\times m}$ and $K_r \in \mathbb{R}^{m\times m}$ such that

(2.38) and (2.39) are verified. If $u = \phi_{\mathrm{ideal}}(\pi, K)$, where

$$\phi_{\mathrm{ideal}}(\pi, K) = K\pi, \qquad (\pi, K) \in \mathbb{R}^{n+m+N} \times \mathbb{R}^{m\times(n+m+N)}, \tag{2.40}$$

then $\lim_{t\to\infty} e(t) = 0$.

Lastly, consider the feedback control law

$$\phi(\pi, \hat{K}) = \hat{K}\pi, \qquad (\pi, \hat{K}) \in \mathbb{R}^{n+m+N} \times \mathbb{R}^{m\times(n+m+N)}, \tag{2.41}$$

where $\hat{K} : [t_0, \infty) \to \mathbb{R}^{m\times(n+m+N)}$. Ideally, a design specification for the *adaptive gain matrix* $\hat{K}(\cdot)$ would be that the *adaptive gain's error* $\widetilde{\Delta K}(t) \triangleq \hat{K}(t) - K$, $t \geq t_0$, verifies some constraints assigned *a priori*. Since both A and Θ are unknown, K is unknown, and hence, $\widetilde{\Delta K}(\cdot)$ cannot be computed. However, in problems of practical interest it is possible to find $K_{\mathrm{e}} \in \mathbb{R}^{m\times(n+m+N)}$ that provides an estimate of K, that is, such that $\|K_{\mathrm{e}} - K\|_{\mathrm{F}} \leq \varepsilon$, for some $\varepsilon \geq 0$. In this formulation, $\hat{K}(\cdot)$ is provided such that both the trajectory tracking error $e(\cdot)$ and the *estimated adaptive gain's error* $\Delta K(t) \triangleq \hat{K}(t) - K_{\mathrm{e}}$, $t \geq t_0$, verify user-defined constraints at all time; note that $\|\widetilde{\Delta K}(t) - \Delta K(t)\|_{\mathrm{F}} \leq \varepsilon$, $t \geq t_0$. In particular, the constraints considered here are captured by the compact, connected *constraint set*

$$\mathcal{C} \triangleq \left\{(e, \Delta K) \in \mathbb{R}^n \times \mathbb{R}^{m\times(n+m+N)} : h(e^{\mathrm{T}} W e, \Delta K \Gamma^{-1} \Delta K^{\mathrm{T}}) \geq 0\right\}, \tag{2.42}$$

where $W \in \mathbb{R}^{n\times n}$ is symmetric and positive-definite, $\Gamma \in \mathbb{R}^{(n+m+N)\times(n+m+N)}$ is symmetric and positive-definite, and $h : \mathbb{R} \times \mathbb{R}^{m\times m} \to \mathbb{R}$ is continuously differentiable and such that $h(0,0) > 0$. The compactness of $\mathcal{C}$ allows to capture bounded constraint sets, and the connectedness of $\mathcal{C}$ guarantees that there exists a subset of $\mathcal{C}$ containing both $(e(t_0), \Delta K(t_0))$ and $(0_n, 0_{m\times(n+m+N)})$ that cannot be expressed as two disjoint non-empty sets. Note that the interior of $\mathcal{C}$, that is,

$$\mathring{\mathcal{C}} = \left\{(e, \Delta K) \in \mathbb{R}^n \times \mathbb{R}^{m\times(n+m+N)} : h(e^{\mathrm{T}}We, \Delta K\Gamma^{-1}\Delta K^{\mathrm{T}}) > 0\right\},$$

is not empty, since $h(0,0) > 0$, and $\mathring{\mathcal{C}}\backslash\{(0,0)\}$ is not empty, since $h(\cdot,\cdot)$ is continuous.

Define $h_e : \mathbb{R} \times \mathbb{R}^{m\times m} \to \mathbb{R}$ and $h_X : \mathbb{R} \times \mathbb{R}^{m\times m} \to \mathbb{R}^{m\times m}$ so that

$$h_e(e^{\mathrm{T}}We, \Delta K\Gamma^{-1}\Delta K^{\mathrm{T}}) \triangleq \left.\frac{\partial h(\beta, X)}{\partial \beta}\right|_{\substack{\beta=e^{\mathrm{T}}We \\ X=\Delta K\Gamma^{-1}\Delta K^{\mathrm{T}}}}, \qquad (e, \Delta K) \in \mathring{\mathcal{C}}, \tag{2.43}$$

$$h_X(e^{\mathrm{T}}We, \Delta K\Gamma^{-1}\Delta K^{\mathrm{T}}) \triangleq \left.\frac{\partial h(\beta, X)}{\partial X}\right|_{\substack{\beta=e^{\mathrm{T}}We \\ X=\Delta K\Gamma^{-1}\Delta K^{\mathrm{T}}}}, \tag{2.44}$$

and assume that

$$h_e(e^{\mathrm{T}}We, \Delta K\Gamma^{-1}\Delta K^{\mathrm{T}}) \leq 0, \tag{2.45}$$

$$h_X(e^{\mathrm{T}}We, \Delta K\Gamma^{-1}\Delta K^{\mathrm{T}}) \leq 0, \tag{2.46}$$

that is, $h_X(\cdot,\cdot)$ is symmetric and nonpositive-definite, and

$$(0_n, 0_{m\times(n+m+N)}) = \underset{(e,\Delta K)\in\mathbb{R}^n\times\mathbb{R}^{m\times(n+m+N)}}{\arg\max} h(e^{\mathrm{T}}We, \Delta K\Gamma^{-1}\Delta K^{\mathrm{T}}). \tag{2.47}$$

It follows from (2.42) and (2.45)–(2.47) that $h(\cdot,\cdot)$ must be chosen so that $h(e^{\mathrm{T}}We, \cdot)$

attains its maximum for $e = 0$ and $h(\cdot, \Delta K \Gamma^{-1} \Delta K^{\mathrm{T}})$ attains its maximum for $\Delta K = 0$. As an example, consider

$$h(e^{\mathrm{T}} W e, \Delta K \Gamma^{-1} \Delta K^{\mathrm{T}}) = h_{\max} - \|W^{\frac{1}{2}} e\|^2 - \|\Delta K\|^2_{\mathrm{F},\Gamma^{-1}}, \quad (e, \Delta K) \in \mathbb{R}^m \times \mathbb{R}^{m \times (n+m+N)}, \tag{2.48}$$

where $h_{\max} > 0$ and $M^{\frac{1}{2}}$ denotes the square root of M [30, p. 474]. In this case, (2.48) verifies (2.45)–(2.47), since $h(0, 0) = h_{\max} > 0$, $h_e(e^{\mathrm{T}} W e, \Delta K \Gamma^{-1} \Delta K^{\mathrm{T}}) = -1$, and $h_X(e^{\mathrm{T}} W e,\ \Delta K \Gamma^{-1} \Delta K^{\mathrm{T}}) = -\mathbf{1}_m$.

The next theorem is the main result of this section and provides an adaptive law $\hat{K}(\cdot)$ for the feedback control law (2.41) so that if $u(t) = \phi(\pi(t), \hat{K}(t))$, $t \geq t_0$, then both the trajectory tracking error $e(\cdot)$ and the estimated adaptive gain's error $\Delta K(\cdot)$ verify the user-defined constraints captured by (2.42) at all time, despite parametric, matched, and unmatched uncertainties. For the statement of this theorem, note that it follows from (2.36), (2.41), and (2.37) that

$$\dot{e}(t) = A_{\mathrm{ref}} e(t) + B \widetilde{\Delta K}(t) \pi(t) + \xi(t), \qquad e(t_0) = x_0 - x_{\mathrm{ref},0}, \qquad t \geq t_0. \tag{2.49}$$

In addition, define the positive-definite function

$$V(e, \Delta K) \triangleq \frac{e^{\mathrm{T}} P e + \mathrm{tr}\left(\Delta K \Gamma^{-1} \Delta K^{\mathrm{T}}\right)}{h(e^{\mathrm{T}} W e, \Delta K \Gamma^{-1} \Delta K^{\mathrm{T}})}, \qquad (e, \Delta K) \in \mathring{\mathcal{C}}, \tag{2.50}$$

where $P \in \mathbb{R}^{n \times n}$ denotes the symmetric, positive-definite solution of the algebraic

Lyapunov equation

$$0 = A_{\mathrm{ref}}^{\mathrm{T}} P + P A_{\mathrm{ref}} + Q_1, \tag{2.51}$$

and $Q_1 \in \mathbb{R}^{n\times n}$ is symmetric and positive-definite, and let

$$\mathcal{S}_\pi \triangleq \big\{(e, \Delta K) \in \mathbb{R}^n \times \mathbb{R}^{m\times(n+m+N)} : S_\pi(e, \Delta K) > 0\big\}, \tag{2.52}$$

where

$$S_\pi(e, \Delta K) \triangleq -\alpha\|e\|^2 - \sigma\|e^{\mathrm{T}}PB\|^p\|\Delta K\|_{\mathrm{F}}^2 + 2\left(\varepsilon\|\pi\| + \xi_{\max}\right)\|R(e, \Delta K)\|_{\mathrm{F}}, \tag{2.53}$$

$p \in \mathbb{N}$, $\sigma > 0$, $\alpha \triangleq \lambda_{\min}(Q_1)$, and

$$R(e, \Delta K) \triangleq e^{\mathrm{T}}\big[P - V(e, \Delta K)h_e(e^{\mathrm{T}}We, \Delta K\Gamma^{-1}\Delta K^{\mathrm{T}})W\big]B. \tag{2.54}$$

Lastly, note that the trajectory $x_{\mathrm{ref}}(t)$, $t \geq t_0$, of the reference dynamical model (2.37) is bounded, since $r(t)$ is bounded and A_{ref} is Hurwitz [34, p. 245], and if $e(t)$ is bounded, then $x(t)$ is bounded and there exists a compact set $\Pi \subset \mathbb{R}^{n+m+N}$ such that $\pi(t) = \big[x^{\mathrm{T}}(t), r^{\mathrm{T}}(t), -\Phi^{\mathrm{T}}(x(t))\big]^{\mathrm{T}} \in \Pi$ for all $t \geq t_0$. Therefore, since $S_\pi(\cdot,\cdot)$ is continuous in π, it follows from Weierstrass theorem [34, Th. 2.13] that

$$\pi^*(e, \Delta K) \triangleq \mathrm{argmax}_{\pi\in\Pi} S_\pi(e, \Delta K), \qquad (e, \Delta K) \in \mathbb{R}^n \times \mathbb{R}^{m\times(n+m+N)}, \tag{2.55}$$

exists and is finite. For simplicity, we denote $S_{\pi^*(e,\Delta K)}(e, \Delta K)$ by $S_{\pi^*}(e, \Delta K)$ and

$\mathcal{S}_{\pi^*} = \big\{(e, \Delta K) \in \mathbb{R}^n \times \mathbb{R}^{m\times(n+m+N)} : S_{\pi^*}(e, \Delta K) > 0\big\}.$

Theorem 2.3 *Consider the uncertain nonlinear dynamical system* (2.36)*, the reference model* (2.37)*, the feedback control law* (2.41)*, the trajectory tracking error dynamics* (2.49)*, the constraint set* (2.42)*, and the set* $\mathcal{S}_\pi$ *given by* (2.52)*. Let* $x_0 \in \mathbb{R}^n$*,* $x_{\mathrm{ref},0} \in \mathbb{R}^n$*, and* $\hat{K}_0 \in \mathbb{R}^{m\times(n+m+N)}$ *be such that* $(x_0 - x_{\mathrm{ref},0}, \hat{K}_0 - K_{\mathrm{e}}) \in \mathring{\mathcal{C}} \setminus \{(0,0)\}$*, and let*

$$\begin{aligned}\dot{\hat{K}}^{\mathrm{T}}(t) = -\Gamma\, &\big[\pi(t)e^{\mathrm{T}}(t)\big(P - V(e(t), \Delta K(t))h_e(e^{\mathrm{T}}(t)We(t), \Delta K(t)\Gamma^{-1}\Delta K^{\mathrm{T}}(t))W\big)B \\ &+\sigma\|e^{\mathrm{T}}(t)PB\|^p\Delta K^{\mathrm{T}}(t)\big] \\ &\cdot\big[\mathbf{1}_m - V(e(t), \Delta K(t))h_X(e^{\mathrm{T}}(t)We(t), \Delta K\Gamma^{-1}\Delta K^{\mathrm{T}}(t))\big]^{-1}, \\ &\hat{K}(t_0) = \hat{K}_0, \quad t \geq t_0, \qquad (2.56)\end{aligned}$$

where $P \in \mathbb{R}^{n\times n}$ *denotes the symmetric, positive-definite solution of* (2.51)*,* $V(\cdot,\cdot)$ *is given by* (2.50)*,* $h_e(\cdot,\cdot)$ *is given by* (2.43)*, and* $h_X(\cdot,\cdot)$ *is given by* (2.44)*. If the matching conditions* (2.38) *and* (2.39) *are verified and* $\mathcal{S}_{\pi^*} \subset \mathring{\mathcal{C}}$*, where* $\pi^*(\cdot,\cdot)$ *is given by* (2.55)*, then* (2.36) *with control law* (2.41) *and adaptive law* (2.56) *is such that* $(e(t), \Delta K(t)) \in \mathring{\mathcal{C}}$*,* $t \geq t_0$*.*

Proof: This proof is divided in two parts. First, assume that $(e(t), \Delta K(t)) \in \mathring{\mathcal{C}}$, $t \geq t_0$ and show that $\dot{V}(e, \Delta K) \leq 0$ for all $(e, \Delta K) \in \mathring{\mathcal{C}} \setminus \mathcal{S}_{\pi^*}$. Then, a contradiction argument is used to prove that if $(x_0 - x_{\mathrm{ref},0}, \hat{K}_0 - K_{\mathrm{e}}) \in \mathring{\mathcal{C}}\setminus\{(0,0)\}$, then $(e(t), \Delta K(t)) \in \mathring{\mathcal{C}}$, $t \geq t_0$.

Define

$$Q(e^{\mathrm{T}}We, \Delta K\Gamma^{-1}\Delta K^{\mathrm{T}}) \triangleq Q_1 + V(e,\Delta K)h_e(e^{\mathrm{T}}We, \Delta K\Gamma^{-1}\Delta K^{\mathrm{T}})\left(A_{\mathrm{ref}}^{\mathrm{T}}W + WA_{\mathrm{ref}}\right), \quad (e,\Delta K) \in \mathring{\mathcal{C}}, \tag{2.57}$$

and note that it follows from (2.51) and (2.57) that

$$\begin{aligned}-Q(e^{\mathrm{T}}We, \Delta K\Gamma^{-1}\Delta K^{\mathrm{T}}) = {}& A_{\mathrm{ref}}^{\mathrm{T}}\left[P - V(e,\Delta K)h_e(e^{\mathrm{T}}We, \Delta K\Gamma^{-1}\Delta K^{\mathrm{T}})W\right] \\ &+ \left[P - V(e,\Delta K)h_e(e^{\mathrm{T}}We, \Delta K\Gamma^{-1}\Delta K^{\mathrm{T}})W\right]A_{\mathrm{ref}}.\end{aligned} \tag{2.58}$$

Note also that $Q(\cdot,\cdot)$ is symmetric and $Q(e,\Delta K) \geq \alpha\mathbf{1}_n$, since $Q_1 \geq \alpha\mathbf{1}_n$, $V(e,\Delta K)$ is positive-definite, $A_{\mathrm{ref}}^{\mathrm{T}}W + WA_{\mathrm{ref}}$ is symmetric and nonpositive-definite [35], and (2.45) holds by assumption. Now, assume that $(e(t),\Delta K(t)) \in \mathring{\mathcal{C}}$, $t \geq t_0$. Since $\mathcal{C}$ is compact by assumption, both $e(\cdot)$ and $\Delta K(\cdot)$ are bounded and (2.55) exists and is finite. Additionally, it follows from (2.50), (2.49), (2.36), and (2.58) that

$$\begin{aligned}\dot{V}(e,\Delta K) = {}& -h^{-1}(e^{\mathrm{T}}We, \Delta K\Gamma^{-1}\Delta K^{\mathrm{T}})e^{\mathrm{T}}Q(e^{\mathrm{T}}We, \Delta K\Gamma^{-1}\Delta K^{\mathrm{T}})e \\ &+ 2h^{-1}(e^{\mathrm{T}}We, \Delta K\Gamma^{-1}\Delta K^{\mathrm{T}}) \\ &\quad \cdot \mathrm{tr}\Big(\widetilde{\Delta K}\pi e^{\mathrm{T}}\left[P - V(e,\Delta K)h_e(e^{\mathrm{T}}We, \Delta K\Gamma^{-1}\Delta K^{\mathrm{T}})W\right]B \\ &\qquad + \Delta K\Gamma^{-1}\dot{\hat{K}}^{\mathrm{T}}\left[\mathbf{1}_m - V(e,\Delta K)h_X(e^{\mathrm{T}}We, \Delta K\Gamma^{-1}\Delta K^{\mathrm{T}})\right]\Big) \\ &+ 2h^{-1}(e^{\mathrm{T}}We, \Delta K\Gamma^{-1}\Delta K^{\mathrm{T}})\end{aligned}$$

$$\cdot \operatorname{tr}\left(\xi(t)e^{\mathrm{T}}\left[P - V(e,\Delta K)h_e(e^{\mathrm{T}}We, \Delta K\Gamma^{-1}\Delta K^{\mathrm{T}})W\right]\right), \tag{2.59}$$

for all $(e, \Delta K) \in \mathring{\mathcal{C}}$. Since $\operatorname{tr}(X^{\mathrm{T}}Y)$ is an inner product of X and $Y \in \mathbb{R}^{n\times m}$ [30, p. 95], $\|\widetilde{\Delta K}(t) - \Delta K(t)\|_{\mathrm{F}} \leq \varepsilon$, $t \geq t_0$, by assumption, and $Q(e, \Delta K) \geq \alpha \mathbf{1}_n$, $(e, \Delta K) \in \mathring{\mathcal{C}}$, by construction, it follows from (2.56) and the Cauchy–Schwarz inequality [30, Fact 1.18.9] that

$$\dot{V}(e, \Delta K) \leq h^{-1}(e^{\mathrm{T}}We, \Delta K\Gamma^{-1}\Delta K^{\mathrm{T}})S_\pi(e, \Delta K), \qquad (e, \Delta K) \in \mathring{\mathcal{C}}, \tag{2.60}$$

where $S_\pi(\cdot,\cdot)$ is given by (2.53). Since $(e(t), \Delta K(t)) \in \mathring{\mathcal{C}}$ and $\mathcal{C}$ is compact, $e(\cdot)$ is bounded, $\pi^*(\cdot,\cdot)$ given by (2.55) is well-defined, and $\dot{V}(e, \Delta K) \leq 0$, $(e, \Delta K) \in \mathring{\mathcal{C}} \backslash \mathcal{S}_{\pi^*}$.

Next, assume that $(e(t_0), \Delta K(t_0)) \in \mathring{\mathcal{C}} \setminus \{(0,0)\}$, and suppose *ad absurdum* that there exists $T^* > 0$ such that $\lim_{t\to T^*} \operatorname{dist}((e(t), \Delta K(t)), \partial\mathcal{C}) = 0$, where $\operatorname{dist}(\cdot,\cdot)$ denotes the distance of a point from a set [36, p. 16]. In this case, $\lim_{t\to T^*} h(e^{\mathrm{T}}(t)We(t), \Delta K(t)\Gamma^{-1}\Delta K^{\mathrm{T}}(t)) = 0$ along the trajectory of (2.49) and (2.56), and it follows from the continuity of $h(\cdot,\cdot)$, $e(\cdot)$, and $\Delta K(\cdot)$ that

$$\lim_{t\to T^*} h(e^{\mathrm{T}}(t)We(t), \Delta K(t)\Gamma^{-1}\Delta K^{\mathrm{T}}(t)) = h(e^{\mathrm{T}}(T^*)We(T^*), \Delta K(T^*)\Gamma^{-1}\Delta K^{\mathrm{T}}(T^*)), \tag{2.61}$$

which implies that $(e(T^*), \Delta K(T^*)) \neq (0,0)$, since $h(e^{\mathrm{T}}We, \Delta K\Gamma^{-1}\Delta K^{\mathrm{T}}) > 0$ for all $(e, \Delta K) \in \mathring{\mathcal{C}}$ and $(0,0) \in \mathring{\mathcal{C}}$ by assumption. Moreover, since $\operatorname{tr}(\cdot)$ is continuous and $e^{\mathrm{T}}Pe + \operatorname{tr}\left(\Delta K\Gamma^{-1}\Delta K^{\mathrm{T}}\right)$ is positive-definite for all $(e, \Delta K) \in \mathbb{R}^n \times \mathbb{R}^{m\times(n+m+N)}$, it holds that $\lim_{t\to T^*}\left[e^{\mathrm{T}}(t)Pe(t)+\operatorname{tr}\left(\Delta K(t)\Gamma^{-1}\Delta K^{\mathrm{T}}(t)\right)\right]\neq 0$. Therefore,

$$\lim_{t\to T^*} V(e(t),\Delta K(t)) = \frac{e^{\mathrm{T}}(T^*)Pe(T^*)+\operatorname{tr}\left(\Delta K(T^*)\Gamma^{-1}\Delta K^{\mathrm{T}}(T^*)\right)}{h(e^{\mathrm{T}}(T^*)We(T^*),\Delta K(T^*)\Gamma^{-1}\Delta K^{\mathrm{T}}(T^*))} = \infty. \tag{2.62}$$

Now, if $(e(t_0),\Delta K(t_0))\in\mathcal{S}_{\pi^*}$, then there exists $T^{**}\geq t_0$ such that $(e(T^{**}),\Delta K(T^{**}))\in\partial\mathcal{S}_{\pi^*}$. Since $\mathcal{S}_{\pi^*}\subset\mathring{\mathcal{C}}$ by assumption, it holds that $T^{**}<T^*$ and it follows from (2.60) that for all t_1 and $t_2\in[T^{**},T^*)$ such that $t_2\geq t_1$,

$$V(e(t_2),\Delta K(t_2))\leq V(e(t_1),\Delta K(t_1))<\infty, \tag{2.63}$$

along the trajectory of (2.49) and (2.56), which contradicts (2.62). Alternatively, if $(e(t_0),\Delta K(t_0))\in\mathring{\mathcal{C}}\setminus\mathcal{S}_{\pi^*}$, then there exists $T^{**}>t_0$ such that $(e(T^{**}),\Delta K(T^{**}))\in\partial\mathcal{S}_{\pi^*}$, and (2.62) is contradicted by applying a similar argument as for the previous case. Therefore, if $(e(t_0),\Delta K(t_0))\in\mathring{\mathcal{C}}\setminus\{(0,0)\}$, then $(e(t),\Delta K(t))\in\mathring{\mathcal{C}}$, $t\geq t_0$, which concludes the proof. ■

Theorem 2.3 provides sufficient conditions for the control law (2.41) and the adaptive law (2.56) to steer the trajectory of the dynamical system (2.36) and guarantee user-defined levels of performance, which are captured by the constraint set (2.42), despite uncertainties in the dynamical model. The set $\mathcal{S}_{\pi^*}$ captures the effect of uncertainties on the performance of the control law (2.41) and the adaptive law (2.56). Indeed, if $\varepsilon>0$ and $\xi_{\max}>0$, then there exist $(e,\Delta K)\in\mathbb{R}^n\times\mathbb{R}^{m\times(n+m+N)}$ such that $S_{\pi^*}(e,\Delta K)>0$ and $\mathring{\mathcal{S}}_{\pi^*}\neq\{\emptyset\}$, and it follows from the proof of Theorem 2.3 that $\dot{V}(e,\Delta K)>0$, $(e,\Delta K)\in\mathcal{S}_{\pi^*}$. Therefore, if $\mathcal{S}_{\pi^*}$ is not a proper subset of

$\mathring{\mathcal{C}}$ and $(e(T), \Delta K(T)) \in \mathcal{S}_{\pi^*}$ for some $T \geq t_0$, then $(e(\cdot), \Delta K(\cdot))$ may violate the user-defined constraints captured by $\mathcal{C}$. The condition $\mathcal{S}_{\pi^*} \subset \mathring{\mathcal{C}}$ can be enforced by minimizing the diameter of $\mathcal{S}_{\pi^*}$ [36, p. 16], that is, choosing α and σ sufficiently large, providing accurate estimates of K so that ε is small, and designing $h(\cdot, \cdot)$ so that, using Landau's notation, $\|e^{\mathrm{T}} V(e, \Delta K) h_e(e^{\mathrm{T}} We, \Delta K \Gamma^{-1} \Delta K^{\mathrm{T}})\| = O(\|e\|^k)$, where $k = \max\{2, p\}$. The condition $\mathcal{S}_{\pi^*} \subset \mathring{\mathcal{C}}$ can be also enforced by minimizing $\xi_{\max}$. Specifically, unmatched uncertainties, which are captured by $\xi(\cdot)$ in (2.36), derive both from unknown external disturbances and those terms in the system's dynamical model that are not be captured as matched uncertainties by mean of a regressor vector. The diameter of $\mathcal{S}_{\pi^*}$ can be minimized also employing, for instance, neural networks to capture the system's nonlinearities as matched uncertainties and hence, minimizing $\xi_{\max}$ [7, Ch. 12].

Note that the adaptive law (2.56) involves the term $[\mathbf{1}_m - V(e, \Delta K) h_X(e^{\mathrm{T}} We, \Delta K \Gamma^{-1} \Delta K^{\mathrm{T}})]^{-1}$, $(e, \Delta K) \in \mathring{\mathcal{C}}$. The identity matrix $\mathbf{1}_m$ is positive-definite, $V(e, \Delta K)$, $(e, \Delta K) \in \mathring{\mathcal{C}}$, is positive-definite, and (2.46) is verified by assumption. Therefore, $[\mathbf{1}_m - V(e, \Delta K) h_X(e^{\mathrm{T}} We, \Delta K \Gamma^{-1} \Delta K^{\mathrm{T}})]$, $(e, \Delta K) \in \mathring{\mathcal{C}}$, is positive-definite, hence invertible, and the right-hand side of (2.56) has is well-defined.

Chapter 3: Equations of Motion of a Tilt-Rotor Quadcopter

3.1. Problem Statement

In this section, the equations of motion of a tilt-rotor quadcopter with H-configuration, such as the one shown in Figure 3.1 are presented and analyzed. The tilt-rotor quadcopter is considered as composed of a frame, modeled as a rigid body, and four propellers that can be tilted independently so that the vehicle can move forward or exert horizontal forces without pitching. The aircraft transports some payload of known mass that is not rigidly connected to the vehicle's frame and whose inertia matrix is unknown. The aerial vehicle and its payload are considered as a whole mechanical system, which henceforth is referred to as the quadcopter. In this paper, it is assumed that, due to the presence of a swinging payload, variations both in the position of the quadcopter's center of mass and inertia matrix are not negligible.

To uniquely identify the position and orientation of the quadcopter's frame in space, consider the orthonormal reference frame $\mathbb{I} = \{O; X, Y, Z\}$ fixed with the Earth, centered in $O \in \mathbb{R}^3$, and with axes $X, Y, Z \in \mathbb{R}^3$. The reference frame $\mathbb{I}$ is chosen so that $F_{\rm g}^{\mathbb{I}} = -mgZ$, where $m > 0$ denotes the vehicle's mass and $g > 0$ denotes the gravitational acceleration. Also, consider the orthonormal reference frame $\mathbb{J}(\cdot) = \{A(\cdot); x(\cdot), y(\cdot), z(\cdot)\}$ fixed with the vehicle's frame, centered at a point $A : [t_0, \infty) \to \mathbb{R}^3$ conveniently chosen, and with axes $x, y, z : [t_0, \infty) \to \mathbb{R}^3$; in this paper, we refer to $\mathbb{J}(\cdot)$ as the *body reference frame.* The reference frame $\mathbb{J}$ is chosen

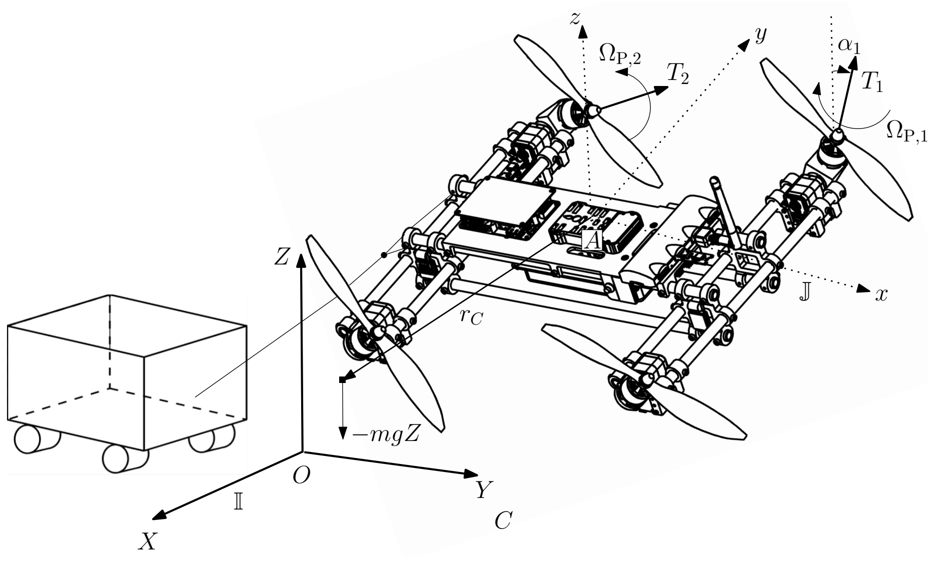


Figure 3.1: Schematic representation of the tilt-rotor quadrotor pulling a cart.

so that the propellers' arms are aligned to the $y(\cdot)$ axis; for details, see Figure 3.1. If a vector $a \in \mathbb{R}^3$ is expressed in the reference frame $\mathbb{I}$, then it is denoted by $a^{\mathbb{I}}$; if a vector is expressed in $\mathbb{J}(\cdot)$, then no superscript is used. Since quadcopters move at low speed and low altitude, we assume that the reference frame $\mathbb{I}$ is inertial.

The *position* of the vehicle's reference point $A(\cdot)$ with respect to O is denoted by $r_A^{\mathbb{I}} : [t_0, \infty) \to \mathbb{R}^3$ and the *velocity* of $A(\cdot)$ with respect to the reference frame $\mathbb{I}$ is denoted by $v_A^{\mathbb{I}} : [t_0, \infty) \to \mathbb{R}^3$. Using a 3-2-1 rotation sequence, the orientation of the reference frame $\mathbb{J}(\cdot)$ with respect to the inertial reference frame $\mathbb{I}$ is captured by the *roll angle* $\phi : [t_0, \infty) \to [0, 2\pi)$, the *pitch angle* $\theta : [t_0, \infty) \to \left(-\frac{\pi}{2}, \frac{\pi}{2}\right)$, and the *yaw angle* $\psi : [t_0, \infty) \to [0, 2\pi)$ [37, pp. 11]. The angular position of the ith propeller about its spin axis is denoted by $\Omega_i : [t_0, \infty) \to \mathbb{R}$, and the angular displacement of the ith propeller's spin axis, $i = 1, \ldots, 4$, about the $y(\cdot)$ axis and measured from the $z(\cdot)$ axis is denoted by $\alpha_i : [t_0, \infty) \to \mathbb{R}$; for purposes of this formulation, $\alpha_i(\cdot)$ is referred to as the *ith propeller's tilt angle*.

3.2. Kinematic and Dynamic Equations

The *vector of independent generalized coordinates*

$$q = \left[\left(r_A^{\mathbb{I}}\right)^{\mathrm{T}}, \phi, \theta, \psi\right]^{\mathrm{T}} \tag{3.1}$$

captures the position and orientation of the vehicle's frame. The *angular velocity* of the reference frame $\mathbb{J}(\cdot)$ with respect to $\mathbb{I}$ [37, Def. 1.9] is denoted by $\omega : \mathcal{D}\times\mathbb{R}^6 \to \mathbb{R}^3$, where $\mathcal{D} \triangleq \mathbb{R}^3 \times [0, 2\pi) \times \left(-\frac{\pi}{2}, \frac{\pi}{2}\right) \times [0, 2\pi)$, and is such that [37, Th. 1.7]

$$\omega(q(t), \dot{q}(t)) = \Gamma^{-1}(q(t)) \left[\dot{\phi}(t), \dot{\theta}(t), \dot{\psi}(t)\right]^{\mathrm{T}}, \qquad t \geq t_0, \tag{3.2}$$

where

$$\Gamma(q) \triangleq \begin{bmatrix} 1 & \sin\phi\tan\theta & \cos\phi\tan\theta \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi\sec\theta & \cos\phi\sec\theta \end{bmatrix}, \qquad q \in \mathcal{D}.$$

It is worthwhile to recall that $\Gamma(q)$ is invertible, since $\theta \in (-\frac{\pi}{2}, \frac{\pi}{2})$ [37, pp. 18-19].

It follows from (3.2) that the *kinematic equation* of a tilt-rotor quadcopter is given by

$$\dot{q}(t) = \begin{bmatrix} v_A^{\mathbb{I}}(t) \\ \Gamma(q(t))\omega(q(t), \dot{q}(t)) \end{bmatrix}, \qquad q(t_0) = \begin{bmatrix} r_{A,0}^{\mathbb{I}} \\ \phi_0 \\ \theta_0 \\ \psi_0 \end{bmatrix}, \qquad t \geq t_0. \tag{3.3}$$

Proceeding as in [38], one can prove that the *translational dynamic equation* of a tilt-rotor quadcopter is given by

$$\begin{aligned}
m\dot{v}_A^{\mathbb{I}}(t) &- mR(q(t))r_C^{\times}(t)\dot{\omega}(q(t),\dot{q}(t)) \\
&= R(q(t))\begin{bmatrix} u_5(t) \\ 0 \\ u_1(t) \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ -mg \end{bmatrix} - mR(q(t))\left[\ddot{r}_C(t) + 2\omega^{\times}(q(t),\dot{q}(t))\dot{r}_C(t)\right. \\
&\quad \left. +\omega^{\times}(q(t),\dot{q}(t))\omega^{\times}(q(t),\dot{q}(t))r_C(t)\right], \qquad v_A^{\mathbb{I}}(t_0) = v_{A,0}^{\mathbb{I}}, \qquad t \geq t_0,
\end{aligned} \tag{3.4}$$

where

$$R(q) \triangleq \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix}\begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi & \cos\phi \end{bmatrix}, \qquad q \in \mathcal{D}, \tag{3.5}$$

$u_5, u_1 : [t_0, \infty) \to \mathbb{R}$ denote the components of the forces produced by the propellers along the $x(\cdot)$ and $z(\cdot)$ axes, respectively, and $r_C : [t_0, \infty) \to \mathbb{R}^3$ denotes the position of the vehicle's center of mass with respect to the reference point $A(\cdot)$; see Figure 3.1. Lastly, proceeding as in [38], one can also prove that the *rotational dynamic equation* of a tilt-rotor quadcopter, whose propellers are modeled as thin disks, is given by

$$
\begin{aligned}
&I(t)\dot{\omega}(q(t),\dot{q}(t)) + m r_C^{\times}(t) R^{\mathrm{T}}(q(t)) \dot{v}_A^{\mathbb{I}}(t) \\
&\quad = \begin{bmatrix} u_2(t) \\ u_3(t) \\ u_4(t) \end{bmatrix} - \omega^{\times}(q(t),\dot{q}(t)) I(t) \omega(q(t),\dot{q}(t)) - \dot{I}(t)\omega(q(t),\dot{q}(t)) \\
&\qquad - \omega^{\times}(q(t),\dot{q}(t)) \sum_{i=1}^{4} I_{P_i}(t)\omega_{P_i}(t) - \sum_{i=1}^{4} \left[I_{P_i}(t)\dot{\omega}_{P_i}(t) + \omega_{P_i}^{\times}(t) I_{P_i}(t) \omega_{P_i}(t) \right] \\
&\qquad + r_C^{\times}(t) F_{\mathrm{g}}(q(t)), \qquad \omega(t_0) = \omega_0, \qquad t \geq t_0,
\end{aligned}
\tag{3.6}
$$

where $[u_2, u_3, u_4]^{\mathrm{T}} : [t_0, \infty) \to \mathbb{R}^3$ denotes the moment of the force produced by the propellers, $F_{\mathrm{g}}(q) = R^{\mathrm{T}}(q)[0, 0, -mg]^{\mathrm{T}}$, $q \in \mathcal{D}$, the matrix function $I(t) \triangleq -\int_{\mathcal{V}} r_{mA}^{\times}(t) r_{mA}^{\times}(t) \mathrm{d}m$, $t \geq t_0$, denotes the *inertia matrix* of the vehicle with respect to the reference point $A(\cdot)$, $\mathcal{V} \subset \mathbb{R}^3$ denotes a volume containing the quadcopter, $r_{mA} : [t_0, \infty) \to \mathcal{V}$ denotes the position of an infinitesimal mass $\mathrm{d}m$ with respect to the reference point $A(\cdot)$, $I_{P_i}(t) \triangleq -\int_{\mathcal{P}_i} r_{mA}^{\times}(t) r_{mA}^{\times}(t) \mathrm{d}m$ denotes the inertia matrix of the ith propeller, $i = 1, \ldots, 4$, with respect to the reference point $A(\cdot)$, and $\mathcal{P}_i \subset \mathbb{R}^3$, $i = 1, \ldots, 4$, denotes a volume containing exclusively the ith propeller. The inertia matrix $I(\cdot)$ is considered as a function of time because the payload is not rigidly connected to the vehicle's frame. The inertia matrix $I_{P_i}(\cdot)$ is considered as a function of time, since the propellers' spin axes' tilt angles vary in time.

The translational dynamic equation (3.4) and the rotational dynamic equation

(3.6) are equivalent to

$$\mathcal{M}(t,q(t))\begin{bmatrix}\dot{v}_A^{\mathbb{I}}(t)\\ \dot{\omega}(q(t),\dot{q}(t))\end{bmatrix}=\begin{bmatrix}f_{\mathrm{dyn,tran}}(t,q(t),\dot{q}(t))\\ f_{\mathrm{dyn,rot}}(t,q(t),\dot{q}(t))\end{bmatrix}+\hat{G}(q(t))u(t),$$
$$\begin{bmatrix}v_A^{\mathbb{I}}(t_0)\\ \omega(t_0)\end{bmatrix}=\begin{bmatrix}v_{A,0}^{\mathbb{I}}\\ \omega_0\end{bmatrix},\qquad t\geq t_0, \tag{3.7}$$

where

$$u=[u_5,u_1,\ldots,u_4]^{\mathrm{T}} \tag{3.8}$$

denotes the *control input*,

$$\mathcal{M}(t,q)\triangleq\begin{bmatrix}m\mathbf{1}_3 & -mR(q)r_C^{\times}(t)\\ mr_C^{\times}(t)R^{\mathrm{T}}(q) & I(t)\end{bmatrix},\qquad (t,q)\in[t_0,\infty)\times\mathcal{D}, \tag{3.9}$$

denotes the *generalized mass matrix*,

$$f_{\mathrm{dyn,tran}}(t,q,\dot{q})\triangleq[0,0,-mg]^{\mathrm{T}}-mR(q)\left[\ddot{r}_C(t)+2\omega^{\times}(q,\dot{q})\dot{r}_C(t)+\omega^{\times}(q,\dot{q})\omega^{\times}(q,\dot{q})r_C(t)\right], \tag{3.10}$$

$$f_{\mathrm{dyn,rot}}(t,q,\dot{q})\triangleq-\omega^{\times}(q,\dot{q})I(t)\omega(q,\dot{q})-\dot{I}(t)\omega(q,\dot{q})-\sum_{i=1}^{4}\left[I_{P_i}(t)\dot{\omega}_{P_i}(t)+\omega_{P_i}^{\times}(t)I_{P_i}(t)\omega_{P_i}(t)\right]$$
$$-\omega^{\times}(q,\dot{q})\sum_{i=1}^{4}I_{P_i}(t)\omega_{P_i}(t)+r_C^{\times}(t)F_{\mathrm{g}}(q), \tag{3.11}$$

and

$$\hat{G}(q) = \begin{bmatrix} R(q)\begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix} & 0_{3\times 3} \\ 0_{3\times 2} & \mathbf{1}_3 \end{bmatrix}. \tag{3.12}$$

In this paper, we refer to (3.3) and (3.7) as the *quadcopter's equations of motion.*

3.3. Analysis of Equations of Motion

In the following section, the propellers' angular velocities $\omega_{P_i}(\cdot)$, $i = 1, \ldots, 4$, and the quadcopter's inertia matrix $I(\cdot)$ are modeled as functions of the propellers' angular position $\Omega_i(\cdot)$ and tilt angles $\alpha_i(\cdot)$. The invertibility of the generalized mass matrix $\mathcal{M}(\cdot,\cdot)$ is also analyzed. For the statement of the results in this section, define the rotation matrix

$$R_2(\alpha) \triangleq \begin{bmatrix} \cos\alpha & 0 & \sin\alpha \\ 0 & 1 & 0 \\ -\sin\alpha & 0 & \cos\alpha \end{bmatrix}, \qquad \alpha \in \mathbb{R}. \tag{3.13}$$

Consider a tilt-rotor quadcopter, let $\Omega_i : [t_0, \infty) \to \mathbb{R}$, $i = 1, \ldots 4$, denote the angular displacement of ith propeller about its spin axis, and let $\alpha_i : [t_0, \infty) \to \mathbb{R}$ denote the tilt angle of the ith spin axis. Then, the propeller's angular velocity with respect to the reference frame $\mathbb{J}(\cdot)$ is given by

$$\omega_{P_i}(t) = R_2(\alpha_i(t))\left(\begin{bmatrix}0\\0\\\dot{\Omega}_i(t)\end{bmatrix} + \begin{bmatrix}0\\\dot{\alpha}_i(t)\\0\end{bmatrix}^{\times}\begin{bmatrix}0\\0\\\Omega_i(t)\end{bmatrix}\right), \qquad i = 1,\ldots,4, \qquad t \geq t_0, \tag{3.14}$$

where $R_2(\cdot)$ is given by (3.13), and the propeller's angular acceleration is given by

$$\begin{aligned}\dot{\omega}_{P_i}(t) = R_2(\alpha_i(t))\begin{bmatrix}0\\0\\\ddot{\Omega}_i(t)\end{bmatrix} + 2R_2(\alpha_i(t))\begin{bmatrix}0\\\dot{\alpha}_i(t)\\0\end{bmatrix}^{\times}\begin{bmatrix}0\\0\\\dot{\Omega}_i(t)\end{bmatrix}\\ + R_2(\alpha_i(t))\left(\begin{bmatrix}0\\\dot{\alpha}_i(t)\\0\end{bmatrix}^{\times}\begin{bmatrix}0\\\dot{\alpha}_i(t)\\0\end{bmatrix}^{\times} + \begin{bmatrix}0\\\ddot{\alpha}_i(t)\\0\end{bmatrix}^{\times}\right)\begin{bmatrix}0\\0\\\Omega_i(t)\end{bmatrix}.\end{aligned} \tag{3.15}$$

Since the angular displacement of the ith propeller about its spin axis is captured by $[0, 0, \Omega_i(\cdot)]^{\mathrm{T}}$, $i = 1, \ldots, 4$, and the angular displacement of the ith spin axis with respect to the $z(\cdot)$ axis is captured by $[0, \alpha_i(\cdot), 0]^{\mathrm{T}}$, (3.14) directly follows from Theorem 1.3 of [37]. Since the angular acceleration is the time derivative of the angular velocity [37, Th. 1.10] and, as shown by [39, p. 366],

$$\dot{R}_2(\alpha_i(t)) = R_2(\alpha_i(t))\left([0, \dot{\alpha}_i(t), 0]^{\mathrm{T}}\right)^{\times}, \qquad i = 1, \ldots, 4, \qquad t \geq t_0, \tag{3.16}$$

equation (3.15) directly follows from (3.14). ■

The next result characterizes both the inertia matrix $I(\cdot)$ of a tilt-rotor quadcopter and its time derivative, which appear in (3.6). For the statement of this result, let $I_{\rm quad} : [t_0, \infty) \to \mathbb{R}^{3\times 3}$ denote the inertia matrix of the quadcopter, excluding its propellers, with respect to $A(\cdot)$. This matrix is modeled as a function of time to account for payloads that are not rigidly connected to the vehicle's frame. Each propeller is modeled as a thin disk of mass $m_{\rm prop} > 0$, radius $\rho_{\rm prop} > 0$, and inertia matrix [40, p. 102]

$$I_{\rm disk} = m_{\rm prop} \frac{\rho_{\rm prop}^2}{4} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 2 \end{bmatrix}. \tag{3.17}$$

The matrix $I_{\rm disk}$ is computed with respect to a principal reference frame that is fixed with the disk and rotated of an angle $\alpha_i(\cdot)$, $i = 1, \ldots, 4$, with respect to the $z(\cdot)$ axis. The matrix function $R_2(\alpha_i(t)) I_{\rm disk} R_2^{\rm T}(\alpha_i(t))$, $t \geq t_0$, captures the inertia matrix of the disk in the reference frame $\mathbb{J}(\cdot)$, where $R_2(\cdot)$ is given by (3.13) [39, p. 337]. Thus, assuming that the propellers are centered at $r_{{\rm prop},1} = [L_x, L_y, L_z]^{\rm T}$, $r_{{\rm prop},2} = [-L_x, L_y, L_z]^{\rm T}$, $r_{{\rm prop},3} = [-L_x, -L_y, L_z]^{\rm T}$, and $r_{{\rm prop},4} = [L_x, -L_y, L_z]^{\rm T}$ with respect to the reference point $A(\cdot)$, where $L_x, L_y, L_z \geq 0$ and $L_x L_y > 0$, it follows from the parallel axis theorem [41, p. 167] that the inertia matrix of ith propeller is given by

$$I_{P_i}(t) = R_2(\alpha_i(t)) I_{\rm disk} R_2^{\rm T}(\alpha_i(t)) + m_{\rm prop} r_{{\rm prop},i}^{\rm T} r_{{\rm prop},i} \mathbf{1}_3 - m_{\rm prop} r_{{\rm prop},i} r_{{\rm prop},i}^{\rm T},$$

$$i = 1, \ldots, 4, \qquad t \geq t_0. \tag{3.18}$$

The inertia matrix of a tilt-rotor quadcopter is given by

$$I(t) = I_{\text{quad}}(t) + \sum_{i=1}^{4} I_{P_i}(t), \qquad t \geq t_0, \tag{3.19}$$

where $I_{P_i}(\cdot)$ is given by (3.17), and it holds that

$$\dot{I}(t) = \dot{I}_{\text{quad}}(t) + \sum_{i=1}^{4} R_2(\alpha_i(t)) \left(\begin{bmatrix} 0 \\ \dot{\alpha}_i(t) \\ 0 \end{bmatrix}^{\times} I_{\text{disk}} - I_{\text{disk}} \begin{bmatrix} 0 \\ \dot{\alpha}_i(t) \\ 0 \end{bmatrix}^{\times} \right) R_2^{\mathrm{T}}(\alpha_i(t)), \quad t \geq t_0, \tag{3.20}$$

where I_{disk} is given by (3.17). Equation (3.19) directly follows from the fact that the inertia matrix of a compound body with respect to a given reference point is the sum of the inertia matrices of the body's components computed with respect to the same reference point, and (3.20) directly follows from (3.19), (3.18), and (3.16). ■

The effect of the propellers' motion on a conventional quadcopter's dynamics is captured in (3.6) by $\sum_{i=1}^{4} I_{P_i}(\cdot)\dot{\omega}_{P_i}(\cdot)$, which is known as *inertial counter-torque*, and $\omega^{\times}(\cdot) \sum_{i=1}^{4} I_{P_i}(\cdot)\omega_{P_i}(\cdot)$, which is known as *gyroscopic effect*. However, $\sum_{i=1}^{4} \omega_{P_i}^{\times}(t) I_{P_i}(t) \omega_{P_i}(t) = 0$, $t \geq t_0$, for conventional quadcopters [38], since $\alpha_i(t) = 0$, $i = 1, \ldots, 4$, $I_{P_i}(\cdot)$ is a diagonal matrix, and the propellers' thrust force is exerted along the $z(\cdot)$ axis of the reference frame $\mathbb{J}(\cdot)$. For the tilt-rotor quadcopter considered in this paper, it holds that $\omega_{P_i}^{\times}(t)I_{P_i}(t)\omega_{P_i}(t) \not\equiv 0$, $t \geq t_0$, $i = 1, \ldots, 4$, since $\alpha_i(t) \not\equiv 0$ and $L_x L_y > 0$ by assumption. To the authors' best knowledge, existing results on tilt-rotor quadcopters do not account for the inertial counter-torque, the gyroscopic effect, or $\sum_{i=1}^{4} \omega_{P_i}^{\times}(\cdot)I_{P_i}(\cdot)\omega_{P_i}(\cdot)$. Indeed, this last term is not identified by a specific name and henceforth, it will be referred to as the *tilt-rotor gyroscopic effect*. In the literature on conventional quadcopters, the inertial counter-torque is often neglected, whereas the gyroscopic effect is rarely ignored [42].

Next, the generalized mass matrix (3.9) is proven to be invertible; recall that generalized mass matrices are nonnegative-definite [43, p. 58] and hence, not necessarily invertible. For the next result, let $I_C(t) \triangleq -\int_{\mathcal{V}} r_{mC}^{\times}(t) r_{mC}^{\times}(t) \mathrm{d}m$ denote the inertia matrix of the vehicle with respect to the center of mass $C(\cdot)$, where $r_{mC} : [t_0, \infty) \to \mathcal{V}$ denotes the position of an infinitesimal mass $\mathrm{d}m$ with respect to $C(\cdot)$. Note that it follows from the parallel axis theorem [41, p. 167] that $I(t) = I_C(t) - m r_C^{\times}(t) r_C^{\times}(t)$, $t \geq t_0$, since $r_{mC}(t) = r_{mA}(t) - r_C(t)$.

Theorem 3.1 *The generalized mass matrix $\mathcal{M}(t, q)$, $(t, q) \in [t_0, \infty) \times \mathcal{D}$, given by (3.9) is invertible.*

Proof: The matrix $\mathcal{M}(t, q)$, $(t, q) \in [t_0, \infty) \times \mathcal{D}$, is in the same form as (3.22) with $n_1 = 3$, $n_2 = 3$, $A = m\mathbf{1}_3$, $B = -mR(q)r_C^{\times}(t)$, $C = B^{\mathrm{T}}$, $D = I(t)$. Since $m\mathbf{1}_3$ is invertible,

$$D - CA^{-1}B = I(t) + m r_C^{\times}(t) R^{\mathrm{T}}(q) R(q) r_C^{\times}(t) = I_C(t), \qquad (t, q) \in [t_0, \infty) \times \mathcal{D}, \tag{3.21}$$

and $I_C(\cdot)$ is invertible, the conditions of [30][Prop. 2.8.3] are verified and it follows from (3.23) that $\operatorname{rank}(\mathcal{M}(t,q)) = 3 + \operatorname{rank}(I_C(t)) = 6$, which proves the result. ■

The next result concerns the block matrix

$$M = \begin{bmatrix} A & B \\ C & D \end{bmatrix}, \tag{3.22}$$

where $A \in \mathbb{R}^{n_1 \times n_1}$, $B \in \mathbb{R}^{n_1 \times n_2}$, $C \in \mathbb{R}^{n_2 \times n_1}$, and $D \in \mathbb{R}^{n_2 \times n_2}$.

Theorem 3.2 ([30, Prop. 2.8.3]) *Consider the matrix M given by* (3.22)*, and assume that A is invertible. Then,*

$$\operatorname{rank}(M) = n_1 + \operatorname{rank}(D - CA^{-1}B). \tag{3.23}$$

Chapter 4: Control Design for a Tilt-Rotor Quadcopter

4.1. Underactuation of Mechanical Systems

Next, we provide a definition of underactuated mechanical system. To this goal, consider the second-order differential equation

$$\ddot{q}(t) = f(t, q(t), \dot{q}(t)) + G(t, q(t))u(t), \qquad q(t_0) = q_0, \qquad \dot{q}(t_0) = q_{\mathrm{d},0}, \qquad t \geq t_0, \tag{4.1}$$

where $q(t) \in \mathcal{D} \subset \mathbb{R}^n$ denotes the vector of *independent generalized coordinates* [41, Ch. 2], $u(t) \in \mathbb{R}^m$ denotes the *control input*, $f : [t_0, \infty) \times \mathcal{D} \times \mathbb{R}^n \to \mathbb{R}^n$, $G : [t_0, \infty) \times \mathcal{D} \to \mathbb{R}^{n \times m}$, both $f(\cdot, \cdot, \cdot)$ and $G(\cdot, \cdot)$ are continuous in their arguments, $f(t, \cdot, \cdot)$ is locally Lipschitz continuous in q and $\dot{q}$ uniformly in t in compact subsets of $[0, \infty)$, and $G(t, \cdot)$ is locally Lipschitz continuous in q uniformly in t in compact subsets of $[0, \infty)$. Recall that all mechanical systems can be expressed in the same form as (4.1); for details, see [44].

Definition 4.1 ([18, Def. 2.9]) *Consider the nonlinear dynamical system* (4.1)*. If* $\mathrm{rank}(G(t, q)) = n$*,* $(t, q) \in [t_0, \infty) \times \mathcal{D}$*, then* (4.1) *is fully actuated with respect to the the vector of independent generalized coordinates* $q(\cdot)$ *and the control input* $u(\cdot)$*. Alternatively, if* $\mathrm{rank}(G(t, q)) < n$*,* $(t, q) \in [t_0, \infty) \times \mathcal{D}$*, then* (4.1) *is underactuated with respect to the the vector of independent generalized coordinates* $q(\cdot)$ *and the control input* $u(\cdot)$*.*

4.2. Control Strategy for Tilt-Rotor Quadcopters

In this section, the robust adaptive control law presented in Section 2.3 is applied to design control laws for tilt-rotor quadcopters, and compute the thrust force each propeller must produce to realize the desired control inputs. In order to design control laws for tilt-rotor quadcopters, it is essential to determine whether the vehicle is underactuated and define a suitable control strategy accordingly. To this goal, note that the equations of motion (3.3) and (3.7) of a tilt-rotor quadcopter can be expressed in the same form as (4.1) with $n = 6$, $m = 5$, $q(\cdot)$ given by (3.1), $u(\cdot)$ given by (3.8),

$$f(t,q,\dot{q}) = \begin{bmatrix} \mathbf{1}_3 & 0 \\ 0 & \Gamma(q) \end{bmatrix} \mathcal{M}^{-1}(t,q) \begin{bmatrix} f_{\mathrm{dyn,tran}}(t,q,\dot{q}) \\ f_{\mathrm{dyn,rot}}(t,q,\dot{q}) \end{bmatrix} + \begin{bmatrix} 0_{3\times 1} \\ \dot{\Gamma}(q)\omega(q,\dot{q}) \end{bmatrix},$$

$$(t,q,\dot{q}) \in [t_0,\infty) \times \mathcal{D} \times \mathbb{R}^n, \tag{4.2}$$

$$G(t,q) = \begin{bmatrix} \mathbf{1}_3 & 0 \\ 0 & \Gamma(q) \end{bmatrix} \mathcal{M}^{-1}(t,q)\hat{G}(q), \tag{4.3}$$

$$q_0 = \left[\left(r^{\mathbb{I}}_{A,0}\right)^{\mathrm{T}}, \phi_0, \theta_0, \psi_0 \right]^{\mathrm{T}}, \tag{4.4}$$

$$q_{\mathrm{d},0} = \left[\left(v^{\mathbb{I}}_{A,0}\right)^{\mathrm{T}}, \omega_0^{\mathrm{T}}\Gamma^{\mathrm{T}}(q_0) \right]^{\mathrm{T}}, \tag{4.5}$$

and $\hat{G}(\cdot)$ is given by (3.12).

Theorem 4.1 *A mechanical system, whose equations of motion are given by* (4.1) *with* $f(\cdot,\cdot,\cdot)$, $G(\cdot,\cdot)$, q_0, *and* $q_{\mathrm{d},0}$ *given by* (4.2)–(4.5), *is underactuated with respect to the vector of independent generalized coordinates* (3.1) *and the control input* (3.8).

Proof: The result follows from Definition 4.1. Specifically, since $\Gamma(q)$, $q \in \mathcal{D}$, is invertible it holds that $\operatorname{rank}\left(\begin{bmatrix} \mathbf{1}_3 & 0 \\ 0 & \Gamma(q) \end{bmatrix}\right) = 6$. It follows from Theorem 3.1 that $\operatorname{rank}\left(\mathcal{M}^{-1}(t,q)\right) = 6$, $(t,q) \in [t_0,\infty)\times\mathcal{D}$, and, since $\left\|R(q)[1,0,0]^{\mathrm{T}}\right\| = \left\|R(q)[0,0,1]^{\mathrm{T}}\right\| = 1$, it holds that $\operatorname{rank}\big(\hat{G}(t,q)\big) = 5$. Therefore, it follows from Corollary 2.5.10 of [30] that $\operatorname{rank}\big(G(t,q)\big) = 5$, and the result follows from Definition 4.1. ■

Remark 4.1 *Some authors consider* $\left[u^{\mathrm{T}}(\cdot), \alpha_1(\cdot), \ldots, \alpha_4(\cdot)\right]^{\mathrm{T}}$ *as the control input for a tilt-rotor quadcopter and, since the number of control inputs is larger than the number of independent generalized coordinates, they consider tilt-rotor quadcopters are overactuated vehicles. However, considering* $[u^{\mathrm{T}}(\cdot), \alpha_1(\cdot), \ldots, \alpha_4(\cdot)]^{\mathrm{T}}$ *as the control input, it is impossible to reduce the equations of motion* (3.3) *and* (3.7) *to the same form as* (4.1) *and verify whether the vehicle is underactuated, fully actuated, or overactuated employing to the rank condition provided in Definition 4.1.*

The next theorem proves that the dynamical system

$$\ddot{\overline{q}}(t) = Uf(t,q(t),\dot{q}(t)) + UG(t,q(t))u(t), \quad \left[\overline{q}^{\mathrm{T}}(t_0), \dot{\overline{q}}^{\mathrm{T}}(t_0)\right]^{\mathrm{T}} = U\left[q_0^{\mathrm{T}}, q_{\mathrm{d},0}^{\mathrm{T}}\right]^{\mathrm{T}}, \quad t \geq t_0, \tag{4.6}$$

where $\overline{q}(t) \triangleq \left[\left(r_A^{\mathbb{I}}(t)\right)^{\mathrm{T}}, \theta(t), \psi(t)\right]^{\mathrm{T}}$, $U \triangleq \begin{bmatrix} \mathbf{1}_3 & 0_{3\times1} & 0_{3\times2} \\ 0_{2\times3} & 0_{2\times1} & \mathbf{1}_2 \end{bmatrix}$ $f(\cdot,\cdot,\cdot)$, $G(\cdot,\cdot)$, q_0, and $q_{\mathrm{d},0}$ are given by (4.2)–(4.5), is fully actuated with respect to the vector of independent generalized coordinates $\overline{q}(\cdot)$ and the control input $u(\cdot)$ and hence, it is possible to design control inputs $u(\cdot)$ to steer $\overline{q}(\cdot) = \left[\left(r_A^{\mathbb{I}}(\cdot)\right)^{\mathrm{T}}, \theta(\cdot), \psi(\cdot)\right]^{\mathrm{T}}$ at will.

Theorem 4.2 *The dynamical system* (4.6) *is fully actuated with respect to* $\overline{q}(\cdot)$ *and* $u(\cdot)$*.*

Proof: The result follows by proceeding as in the proof of Theorem 4.1. ■

In practice, Theorem 4.2 proves that there exist control inputs that allow to steer at will the position, the pitch angle, and the yaw angle of a tilt-rotor quadcopter with H-configuration. Proceeding as in Theorems 4.1 and 4.2, one can prove that the vehicle's rotational dynamics is fully actuated with respect to the independent generalized coordinates $(\phi, \theta, \psi) \in [0, 2\pi) \times \left(-\frac{\pi}{2}, \frac{\pi}{2}\right) \times [0, 2\pi)$ and the control input $[u_2, u_3, u_4]^{\mathrm{T}} \in \mathbb{R}^3$. Therefore, the following control strategy is proposed. The continuously differentiable *reference trajectory* $r^{\mathbb{I}}_{\mathrm{ref}} : [t_0, \infty) \to \mathbb{R}^3$, the continuously differentiable *reference pitch angle* $\theta_{\mathrm{ref}} : [t_0, \infty) \to \left(-\frac{\pi}{2}, \frac{\pi}{2}\right)$, and the continuously differentiable *reference yaw angle* $\psi_{\mathrm{ref}} : [t_0, \infty) \to [0, 2\pi)$ are considered as user-defined. The *translational equivalent control input* $v_{\mathrm{tran}} : [t_0, \infty) \to \mathbb{R}^3$ and the *reference roll angle* $\phi_{\mathrm{ref}}(\cdot)$ are defined as

$$v_{\mathrm{tran}}(t) \triangleq R(q_{\mathrm{ref}}(t)) \left[u_5(t), 0, u_1(t)\right]^{\mathrm{T}}, \qquad t \geq t_0, \tag{4.7}$$

$$\phi_{\mathrm{ref}}(t) \triangleq -\tan^{-1} \frac{\tilde{v}_{\mathrm{tran},2}(t)}{\tilde{v}_{\mathrm{tran},3}(t)}, \tag{4.8}$$

where $R(\cdot)$ is given by (3.5),

$$q_{\mathrm{ref}} \triangleq \left[\left(r^{\mathbb{I}}_{\mathrm{ref}}\right)^{\mathrm{T}}, \phi_{\mathrm{ref}}, \theta_{\mathrm{ref}}, \psi_{\mathrm{ref}}\right]^{\mathrm{T}} \tag{4.9}$$

denotes the *vector of reference generalized coordinates*,

$$\tan^{-1}\frac{\alpha}{\beta} \triangleq \begin{cases} \tan^{-1}\dfrac{\alpha}{\beta}, & \beta > 0, \\ \tan^{-1}\dfrac{\alpha}{\beta} + \pi, & \alpha \geq 0, \beta < 0, \\ \tan^{-1}\dfrac{\alpha}{\beta} - \pi, & \alpha < 0, \beta < 0, \\ \pi/2, & \alpha > 0, \beta = 0, \\ -\pi/2, & \alpha < 0, \beta = 0, \\ 0, & \alpha = 0, \beta = 0, \end{cases} \tag{4.10}$$

denotes the *signed inverse tangent function*, and

$$\begin{bmatrix} \tilde{v}_{\mathrm{tran},1}(t) \\ \tilde{v}_{\mathrm{tran},2}(t) \\ \tilde{v}_{\mathrm{tran},3}(t) \end{bmatrix} = \begin{bmatrix} \cos\theta_{\mathrm{ref}}(t) & 0 & -\sin\theta_{\mathrm{ref}}(t) \\ 0 & 1 & 0 \\ \sin\theta_{\mathrm{ref}}(t) & 0 & \cos\theta_{\mathrm{ref}}(t) \end{bmatrix} \begin{bmatrix} \cos\psi_{\mathrm{ref}}(t) & \sin\psi_{\mathrm{ref}}(t) & 0 \\ -\sin\psi_{\mathrm{ref}}(t) & \cos\psi_{\mathrm{ref}}(t) & 0 \\ 0 & 0 & 1 \end{bmatrix} v_{\mathrm{tran}}(t). \tag{4.11}$$

In this case, it follows from (4.2), (4.3), and (4.7) that the equations of motion (3.3) and (3.7) are equivalent to

$$\begin{aligned} \mathcal{M}(t, q(t)) \begin{bmatrix} \mathbf{1}_3 & 0 \\ 0 & \Gamma^{-1}(q(t)) \end{bmatrix} \ddot{q}(t) &= \begin{bmatrix} f_{\mathrm{dyn,tran}}(t, q(t), \dot{q}(t)) \\ f_{\mathrm{dyn,rot}}(t, q(t), \dot{q}(t)) \end{bmatrix} + \left[\hat{G}(q(t)) - \hat{G}(q_{\mathrm{ref}}(t))\right] u(t) \\ &\quad + \mathcal{M}(t, q(t)) \begin{bmatrix} 0_{3\times 1} \\ \Gamma^{-1}(q(t))\dot{\Gamma}(q(t))\omega(q(t), \dot{q}(t)) \end{bmatrix} + v(t), \\ \left[q^{\mathrm{T}}(t_0), \dot{q}^{\mathrm{T}}(t_0)\right]^{\mathrm{T}} &= \left[q_0^{\mathrm{T}}, q_{\mathrm{d},0}^{\mathrm{T}}\right]^{\mathrm{T}}, \qquad t \geq t_0, \end{aligned} \tag{4.12}$$

and the *equivalent control input*

$$v(t) \triangleq [v_{\mathrm{tran}}^{\mathrm{T}}(t), u_2(t), u_3(t), u_4(t)]^{\mathrm{T}} \tag{4.13}$$

is designed so that $q(\cdot)$ tracks $q_{\mathrm{ref}}(\cdot)$ within user-defined bounds. In this control strategy, $q_{\mathrm{ref}}(\cdot)$ captures the state of an ideal quadcopter, and $v(\cdot)$ must be computed so that the actual vehicle, whose dynamics captured by $q(\cdot)$, mimics $q_{\mathrm{ref}}(\cdot)$. Equation (4.8) captures a constraint for the vector of reference generalized coordinates and implies that an ideal quadcopter, whose state $q_{\mathrm{ref}}(\cdot)$ must be tracked by $q(\cdot)$, is underactuated.

4.3. Control Laws for Tilt-Rotor Quadcopters

In this section, adaptive control laws are provided for the equivalent control input $v(\cdot)$ given by (4.13) and hence, for the control input $u(\cdot)$ given by (3.8), so that the vector of independent generalized coordinates $q(\cdot)$ given by (3.1) tracks the vector of reference generalized coordinates $q_{\mathrm{ref}}(\cdot)$ given by (4.9) within user-defined bounds on the trajectory tracking error, despite uncertainties in the dynamical model and external disturbances. Furthermore, the adaptive control laws employed in this section allow to enforce user-defined constraints on the adaptive gains. In particular, the equations of motion of a tilt-rotor quadcopter given by (4.12) are first feedback-linearized so that the trajectory tracking error dynamics is reduced to the same form as (2.49), and then the adaptive law presented in Theorem 2.3 is applied to a virtual control input introduced in the feedback linearization process. To feedback linearize

(4.12), the following assumption is needed.

Assumption 4.1 *Consider a tilt-rotor quadcopter, whose equations of motion are given by* (4.12). *The quadcopter's mass m, each propeller's mass m_{prop}, the position of the center of the ith propeller $r_{\text{prop},i}$, $i = 1, \ldots, 4$, the angular displacement of the ith propeller about its spin axis $\Omega_i(\cdot)$, and the tilt angle $\alpha_i(\cdot)$ are known at all time.*

Assumption 4.1 is verified in numerous problems of practical interest. Indeed, the mass of a quadcopter and its payload can be readily determined, the reference point $A(\cdot)$ can be conveniently chosen so that the position of the center of each propeller with respect to $A(\cdot)$ is known, and both $\Omega_i(\cdot)$, $i = 1, \ldots, 4$, and $\alpha_i(\cdot)$ can be either measured or deduced, as shown in Section 4.4 below. However, if the payload is not rigidly connected to the vehicle's frame, then the position of the center of mass and its inertia matrix are unknown. Assumption 4.1 involves neither $r_C(\cdot)$, that is, the position of the center of mass with respect to the reference point $A(\cdot)$, nor $I_{\text{quad}}(\cdot)$, that is, and the inertia matrix of the quadcopter, excluding its propellers. For this formulation, consider both $r_C(\cdot)$ and $I_{\text{quad}}(\cdot)$ as unknown, and define the twice continuously differentiable functions $\overline{r}_C : [t_0, \infty) \to \mathbb{R}^3$ and $\Delta r_C : [t_0, \infty) \to \mathbb{R}^3$ so that

$$r_C(t) = \overline{r}_C(t) + \Delta r_C(t), \qquad t \geq t_0. \tag{4.14}$$

The function $\overline{r}_C(\cdot)$ can be accurately estimated using analytical models of vehicle such that it is considered as known, whereas $\Delta r_C(\cdot)$ is unknown. In a similar manner, define the continuously differentiable and symmetric matrix functions $\overline{I}_{\mathrm{quad}} : [t_0, \infty) \to \mathbb{R}^{3\times 3}$ and $\Delta I : [t_0, \infty) \to \mathbb{R}^{3\times 3}$ with $\overline{I}_{\mathrm{quad}}(\cdot)$ positive-definite such that (3.19) and (3.20) can be expressed as

$$I(t) = \overline{I}(t) + \Delta I(t), \qquad t \geq t_0, \tag{4.15}$$

$$\dot{I}(t) = \dot{\overline{I}}(t) + \Delta \dot{I}(t), \tag{4.16}$$

respectively, where

$$\overline{I}(t) \triangleq \overline{I}_{\mathrm{quad}}(t) + \sum_{i=1}^{4} I_{P_i}(t), \tag{4.17}$$

$$\dot{\overline{I}}(t) = \dot{\overline{I}}_{\mathrm{quad}}(t) + \sum_{i=1}^{4} R_2(\alpha_i(t)) \left(\begin{bmatrix} 0 \\ \dot{\alpha}_i(t) \\ 0 \end{bmatrix}^{\times} I_{\mathrm{disk}} - I_{\mathrm{disk}} \begin{bmatrix} 0 \\ \dot{\alpha}_i(t) \\ 0 \end{bmatrix}^{\times} \right) R_2^{\mathrm{T}}(\alpha_i(t)). \tag{4.18}$$

The matrix function $\overline{I}_{\mathrm{quad}}(\cdot)$ denotes an estimate of the quadcopter's inertia matrix, is considered as known, and is deduced using, for example, analytical models of the vehicle's configuration. The ith propeller's inertia matrix $I_{P_i}(\cdot)$, $i = 1, \ldots, 4$, given by (3.18) is known, since each propeller's mass m_{prop}, the propeller's location $r_{\mathrm{prop},i}$, and the tilt angle $\alpha_i(\cdot)$ are known according to Assumption 4.1. The matrix $\Delta I(\cdot)$ is unknown.

To feedback linearize (4.12), define $e(t) \triangleq \left[q^{\mathrm{T}}(t) - q_{\mathrm{ref}}^{\mathrm{T}}(t), \dot{q}^{\mathrm{T}}(t) - \dot{q}_{\mathrm{ref}}^{\mathrm{T}}(t)\right]^{\mathrm{T}}$, $t \geq t_0$, where $q(\cdot)$ is given by (3.1) and $q_{\mathrm{ref}}(\cdot)$ is given by (4.9), define

$$\overline{\mathcal{M}}(t,q) \triangleq \begin{bmatrix} m\mathbf{1}_3 & -mR(q)\overline{r}_C^\times(t) \\ m\overline{r}_C^\times(t)R^{\mathrm{T}}(q) & \overline{I}(t) \end{bmatrix}, \qquad (t,q) \in [t_0,\infty) \times \mathcal{D}, \tag{4.19}$$

$$\Delta\mathcal{M}(t,q) \triangleq \begin{bmatrix} 0_{3\times 3} & -mR(q)\Delta r_C^\times(t) \\ m\Delta r_C^\times(t)R^{\mathrm{T}}(q) & \Delta I(t) \end{bmatrix}, \tag{4.20}$$

so that $\mathcal{M}(t,q) = \overline{\mathcal{M}}(t,q) + \Delta\mathcal{M}(t,q)$ and $\overline{\mathcal{M}}(\cdot,\cdot)$ is invertible, define

$$\begin{aligned} \overline{f}_{\mathrm{dyn,tran}}(t,q,\dot{q}) \triangleq{}& [0,0,-mg]^{\mathrm{T}} \\ & - mR(q)\left[\ddot{\overline{r}}_C(t) + 2\omega^\times(q,\dot{q})\dot{\overline{r}}_C(t) + \omega^\times(q,\dot{q})\omega^\times(q,\dot{q})\overline{r}_C(t)\right], \end{aligned} \tag{4.21}$$

$$\Delta f_{\mathrm{dyn,tran}}(t,q,\dot{q}) \triangleq -mR(q)\left[\Delta\ddot{r}_C(t) + 2\omega^\times(q,\dot{q})\Delta\dot{r}_C(t) + \omega^\times(q,\dot{q})\omega^\times(q,\dot{q})\Delta r_C(t)\right], \tag{4.22}$$

$$\begin{aligned} \overline{f}_{\mathrm{dyn,rot}}(t,q,\dot{q}) \triangleq{}& -\omega^\times(q,\dot{q})\overline{I}(t)\omega(q,\dot{q}) - \dot{\overline{I}}(t)\omega(q,\dot{q}) - \omega^\times(q,\dot{q})\sum_{i=1}^{4} I_{P_i}(t)\omega_{P_i}(t) \\ & - \sum_{i=1}^{4}\left[I_{P_i}(t)\dot{\omega}_{P_i}(t) + \omega_{P_i}^\times(t)I_{P_i}(t)\omega_{P_i}(t)\right] + \overline{r}_C^\times(t)F_{\mathrm{g}}(q), \end{aligned} \tag{4.23}$$

$$\Delta f_{\mathrm{dyn,rot}}(t,q,\dot{q}) \triangleq -\omega^\times(q,\dot{q})\Delta I(t)\omega(q,\dot{q}) - \Delta\dot{I}(t)\omega(q,\dot{q}) + \Delta r_C^\times F_{\mathrm{g}}(q), \tag{4.24}$$

so that

$$f_{\mathrm{dyn,tran}}(t,q,\dot{q}) = \overline{f}_{\mathrm{dyn,tran}}(t,q,\dot{q}) + \Delta f_{\mathrm{dyn,tran}}(t,q,\dot{q}), \tag{4.25}$$

$$f_{\mathrm{dyn,rot}}(t,q,\dot{q}) = \overline{f}_{\mathrm{dyn,rot}}(t,q,\dot{q}) + \Delta f_{\mathrm{dyn,rot}}(t,q,\dot{q}), \tag{4.26}$$

and define

$$\beta(t,q,q_{\rm ref},v_2) \triangleq \overline{\mathcal{M}}(t,q)\begin{bmatrix}\mathbf{1}_3 & 0_{3\times3}\\ 0_{3\times3} & \Gamma^{-1}(q)\end{bmatrix}\left(\ddot{q}_{\rm ref} - \begin{bmatrix}0_{3\times1}\\ \dot{\Gamma}(q)\omega(q,\dot{q})\end{bmatrix} - K_{\rm P}\left[q(t)-q_{\rm ref}(t)\right]\right.$$
$$\left. - K_{\rm D}\left[\dot{q}-\dot{q}_{\rm ref}\right] + w\right) - \begin{bmatrix}\overline{f}_{\rm dyn,tran}(t,q,\dot{q})\\ \overline{f}_{\rm dyn,rot}(t,q,\dot{q})\end{bmatrix}, \tag{4.27}$$

where $w \in \mathbb{R}^6$ and the matrices $K_{\rm P}$ and $K_{\rm D} \in \mathbb{R}^{6\times6}$ are symmetric and positive-definite.

If $v(t) = \beta(t, q(t), q_{\rm ref}(t), w(t))$, $t \geq t_0$, then (4.12) is feedback linearized and the trajectory tracking error dynamics is given by

$$\dot{e}(t) = \begin{bmatrix}0_{6\times6} & \mathbf{1}_6\\ -K_{\rm P} & -K_{\rm D}\end{bmatrix} e(t) + \begin{bmatrix}0_{6\times6}\\ \mathbf{1}_6\end{bmatrix} w(t) + \hat{\xi}(t),$$
$$e(t_0) = \begin{bmatrix}q_0\\ q_{\rm d,0}\end{bmatrix} - \begin{bmatrix}q_{\rm ref}(t_0)\\ \dot{q}_{\rm ref}(t_0)\end{bmatrix}, \quad t \geq t_0, \tag{4.28}$$

where $w(t) \in \mathbb{R}^6$ denotes the *virtual control input*, $\hat{\xi}(t) = \left[0_{1\times6}, \hat{\xi}_{\rm dyn}^{\rm T}(t)\right]^{\rm T} \in \mathbb{R}^{12}$, and

$$\hat{\xi}_{\rm dyn}(t) = \begin{bmatrix}\mathbf{1}_3 & 0_{3\times3}\\ 0_{3\times3} & \Gamma(q(t))\end{bmatrix}\overline{\mathcal{M}}^{-1}(t,q)\left(\begin{bmatrix}\Delta f_{\rm dyn,tran}(t,q,\dot{q})\\ \Delta f_{\rm dyn,rot}(t,q,\dot{q})\end{bmatrix} + \left[\hat{G}(q(t)) - \hat{G}(q_{\rm ref}(t))\right]u(t)\right.$$
$$\left. -\Delta\mathcal{M}(t,q)\begin{bmatrix}\mathbf{1}_3 & 0_{3\times3}\\ 0_{3\times3} & \Gamma^{-1}(q(t))\end{bmatrix}\ddot{q}(t)\right). \tag{4.29}$$

In order to account for matched uncertainties on the trajectory tracking error dynamics due to systematic errors in the location of the center of mass $r_C(\cdot)$ and the inertia matrix $I(\cdot)$, (4.28) is modified so that the trajectory tracking error dynamics is captured by

$$\dot{e}(t) = \begin{bmatrix} 0_{6\times 6} & \mathbf{1}_6 \\ -K_{\rm P} & -K_{\rm D} \end{bmatrix} e(t) + \begin{bmatrix} 0_{6\times 6} \\ \mathbf{1}_6 \end{bmatrix} \left[w(t) + \Theta^{\rm T}\Phi(q(t),\dot{q}(t)) \right] + \hat{\xi}(t),$$
$$e(t_0) = \begin{bmatrix} q_0 \\ q_{{\rm d},0} \end{bmatrix} - \begin{bmatrix} q_{\rm ref}(t_0) \\ \dot{q}_{\rm ref}(t_0) \end{bmatrix}, \qquad t \geq t_0, \tag{4.30}$$

where $\Theta^{\rm T} = \begin{bmatrix} \Theta_{\rm tran} & 0_{3\times 30} \\ 0_{3\times 9} & \Theta_{\rm rot} \end{bmatrix}$, $\Phi(q,\dot{q}) = \left[\Phi_{\rm tran}^{\rm T}(q,\dot{q}), \Phi_{\rm rot}^{\rm T}(q,\dot{q})\right]^{\rm T}$, $(q,\dot{q}) \in \mathcal{D} \times \mathbb{R}^6$,

$$\Theta_{\rm tran} = M_W(\overline{\overline{r}}_C, 3), \tag{4.31}$$

$$\Theta_{\rm rot} = \left[M_W\left(W_M(\overline{\overline{I}}), 3 \right), \overline{\overline{r}}_C^{\times} \right], \tag{4.32}$$

$$\Phi_{\rm tran}(q,\dot{q}) = -mW_M(R(q)\omega^{\times}(q,\dot{q})\omega^{\times}(q,\dot{q})), \tag{4.33}$$

$$\Phi_{\rm rot}(q,\dot{q}) = \left[-W_M^{\rm T}\left(\omega^{\times}(q,\dot{q}) M_W(\omega(q,\dot{q})) \right), F_{\rm g}^{\rm T}(q) \right]^{\rm T}, \tag{4.34}$$

$\overline{\overline{r}}_C \in \mathbb{R}^3$ and $\overline{\overline{I}} \in \mathbb{R}^{3\times 3}$ are unknown and denote systematic errors in the estimation of the aircraft position of the center of mass and inertia matrix, respectively, $M_W(\cdot,\cdot)$ is given by (1.4), and $W_M(\cdot)$ is given by (1.5) so that $\Theta_{\rm tran}\Phi_{\rm tran}(q,\dot{q}) = -mR(q)\omega^{\times}(q,\dot{q})\omega^{\times}(q,\dot{q})\overline{\overline{r}}_C$ and $\Theta_{\rm rot}\Phi_{\rm rot}(q,\dot{q}) = -\omega^{\times}(q,\dot{q})\overline{\overline{I}}\omega(q,\dot{q}) + \overline{\overline{r}}_C^{\times}F_{\rm g}(q)$.

The next theorem is the main result of this section and applies the adaptive law (2.56) to regulate the equations of motion (4.12) of a tilt-rotor quadcopter, whose inertial properties are partly unknown, and bound both the trajectory tracking error $e(\cdot)$ and the estimated adaptive gains' error $\Delta K(\cdot)$ within a user-defined constraint set. For the statement of the next result, note that if

$$w(t) = \Delta K(t)\pi(t), \qquad t \geq t_0, \tag{4.35}$$

where $\pi(t) = \left[q^{\mathrm{T}}(t), \dot{q}^{\mathrm{T}}(t), 0_{1\times 6}, -\Phi^{\mathrm{T}}(q(t), \dot{q}(t))\right]^{\mathrm{T}}$, $\Delta K(t) = \hat{K}(t) - K_{\mathrm{e}}$, $\hat{K} : [t_0, \infty) \to \mathbb{R}^{6\times 57}$, $K_{\mathrm{e}} = -\left[K_{\mathrm{P}}, K_{\mathrm{D}}, 0_{6\times 6}, \Theta_{\mathrm{e}}^{\mathrm{T}}\right]$, and $\Theta_{\mathrm{e}} \in \mathbb{R}^{39\times 6}$ denotes an estimate of Θ, that is, $\|\Theta_{\mathrm{e}} - \Theta\| \leq \varepsilon$ with $\varepsilon \geq 0$ arbitrarily small, then (4.30) is in the same form as (2.49) with $n = 12$, $m = 6$, $N = 39$, $\widetilde{\Delta K}(t) = \Delta K(t) + K_{\mathrm{e}} - K$, $K = -\left[K_{\mathrm{P}}, K_{\mathrm{D}}, 0_{6\times 6}, \Theta^{\mathrm{T}}\right]$,

$$\xi(t) = \hat{\xi}(t) + \begin{bmatrix} 0_{6\times 1} \\ (\Theta - \Theta_{\mathrm{e}})^{\mathrm{T}}\, \Phi(q(t), \dot{q}(t)) \end{bmatrix}, \tag{4.36}$$

$x_0 = \left[q_0^{\mathrm{T}}, q_{\mathrm{d},0}^{\mathrm{T}}\right]^{\mathrm{T}}$, $A_{\mathrm{ref}} = \begin{bmatrix} 0_{6\times 6} & \mathbf{1}_6 \\ -K_{\mathrm{P}} & -K_{\mathrm{D}} \end{bmatrix}$, and $B = \begin{bmatrix} 0_{6\times 6} \\ \mathbf{1}_6 \end{bmatrix}$. Furthermore, the matching conditions (2.38) and (2.39) are verified with $A = \begin{bmatrix} 0_{6\times 6} & \mathbf{1}_6 \\ 0_{6\times 6} & 0_{6\times 6} \end{bmatrix}$, $B_{\mathrm{ref}} = 0_{12\times 6}$, $K_x^{\mathrm{T}} = -\left[K_{\mathrm{P}}, K_{\mathrm{D}}\right]$, and $K_r = 0_{6\times 6}$.

Theorem 4.3 *Consider the equations of motion of a tilt-rotor quadcopter given by (4.12) with $v(t) = \beta(t, q(t), q_{\mathrm{ref}}(t), w(t))$, $t \geq t_0$, where $\beta(\cdot, \cdot, \cdot, \cdot)$ is given by (4.27) and*

$w(\cdot)$ is given by (4.35). Furthermore, consider the trajectory tracking error dynamics is given by (4.30) and the constraint set given by (2.42). If $\hat{K}(\cdot)$ verifies (2.56) for some $\sigma > 0$ and $p \in \mathbb{N}$ and the conditions of Theorem 2.3 are verified, then $(e(t), \Delta K(t)) \in \mathring{\mathcal{C}}$, $t \geq t_0$.

Proof: The result is a direct consequence of Theorem 2.3 applied to the trajectory tracking error dynamics (4.30) with $w(\cdot)$ given by (4.35). ■

4.4. Realization of Control Inputs

Once the virtual control input $v(\cdot)$ has been computed applying Theorem 4.3 and the control input $u(\cdot)$ has been computed applying (4.7) and (4.13), the forces and moments needed for $q(\cdot)$ to track $q_{\text{ref}}(\cdot)$ must be realized by generating the appropriate thrust forces $T_i(\cdot)$, $i = 1, \ldots, 4$ and tilting the propellers' axes by $\alpha_i(\cdot)$. To achieve this, the thrust force generated by the ith propeller is modeled as

$$T_i(t) = k\dot{\Omega}_i^2(t), \qquad i = 1, \ldots, 4, \qquad t \geq t_0, \tag{4.37}$$

where $k > 0$ [45], [46, Ch. 2]. The moment of the aerodynamic drag induced by the ith propeller is modeled as

$$D_i(t) = k_{\text{T}} T_i(t), \qquad i = 1, \ldots, 4, \tag{4.38}$$

where $k_{\text{T}} > 0$ [45]. Hence, assuming that adjacent propellers spin in opposite directions, it holds that

$$u(t) = MT(t), \qquad t \geq t_0, \tag{4.39}$$

where $T(t) \triangleq [T_{1,\mathrm{c}}(t), T_{1,\mathrm{s}}(t), T_{2,\mathrm{c}}(t), T_{2,\mathrm{s}}(t), T_{3,\mathrm{c}}(t),\ T_{3,\mathrm{s}}(t), T_{4,\mathrm{c}}(t), T_{4,\mathrm{s}}(t)]^{\mathrm{T}}$ denotes the *vector of thrust forces*, $T_{i,\mathrm{c}}(t) \triangleq T_i(t)\cos\alpha_i(t)$, $i = 1, \ldots, 4$, denotes the component of the ith propeller's thrust force along the $z(\cdot)$ axis of the reference frame $\mathbb{J}$, $T_{i,\mathrm{s}}(t) \triangleq T_i(t)\sin\alpha_i(t)$ denotes the component of the ith propeller's thrust force along the $x(\cdot)$ axis, and

$$M \triangleq \begin{bmatrix} 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ L_y & k_{\mathrm{T}} & L_y & -k_{\mathrm{T}} & -L_y & k_{\mathrm{T}} & -L_y & -k_{\mathrm{T}} \\ -L_x & L_z & L_x & L_z & L_x & L_z & -L_x & L_z \\ k_{\mathrm{T}} & -L_y & -k_{\mathrm{T}} & -L_y & k_{\mathrm{T}} & L_y & -k_{\mathrm{T}} & L_y \end{bmatrix}. \tag{4.40}$$

Since $L_xL_y > 0$ and $k_{\mathrm{T}} > 0$ by assumption, the matrix M given by (4.40) is full-rank and the Moore-Penrose inverse of M is given by $M^+ = M^{\mathrm{T}}\left[MM^{\mathrm{T}}\right]^{-1}$ [30, Prop. 6.1.5] which is computed as

$$M^+ = \frac{1}{4}\begin{bmatrix} \frac{L_z}{L_x} & 1 & \frac{L_y}{L_y^2+k_T^2} & \frac{-1}{L_x} & \frac{k_T}{L_y^2+k_T^2} \\ \frac{L_x^2+L_z^2}{L_x^2} - \frac{L_z^2}{L_x} & 0 & \frac{k_T}{L_y^2+k_T^2} & 0 & \frac{-L_y}{L_y^2+k_T^2} \\ \frac{-L_z}{L_x} & 1 & \frac{L_y}{L_y^2+k_T^2} & \frac{1}{L_x} & \frac{-k_T}{L_y^2+k_T^2} \\ \frac{L_x^2+L_z^2}{L_x^2} - \frac{L_z^2}{L_x} & 0 & \frac{-k_T}{L_y^2+k_T^2} & 0 & \frac{-L_y}{L_y^2+k_T^2} \\ \frac{-L_z}{L_x} & 1 & \frac{-L_y}{L_y^2+k_T^2} & \frac{1}{L_x} & \frac{k_T}{L_y^2+k_T^2} \\ \frac{L_x^2+L_z^2}{L_x^2} - \frac{L_z^2}{L_x} & 0 & \frac{k_T}{L_y^2+k_T^2} & 0 & \frac{L_y}{L_y^2+k_T^2} \\ \frac{L_z}{L_x} & 1 & \frac{-L_y}{L_y^2+k_T^2} & \frac{-1}{L_x} & \frac{-k_T}{L_y^2+k_T^2} \\ \frac{L_x^2+L_z^2}{L_x^2} - \frac{L_z^2}{L_x} & 0 & \frac{-k_T}{L_y^2+k_T^2} & 0 & \frac{L_y}{L_y^2+k_T^2}. \end{bmatrix} \tag{4.41}$$

Thus, given $u : [t_0, \infty) \to \mathbb{R}^5$, the vector of thrust forces are computed as

$$T^*(t) = M^+ u(t), \qquad t \geq t_0, \tag{4.42}$$

and the propellers' tilt angles are computed as

$$\alpha_i(t) \triangleq \tan^{-1} \frac{T^*_{i,\mathrm{s}}(t)}{T^*_{i,\mathrm{c}}(t)}, \qquad i = 1, \ldots, 4, \tag{4.43}$$

where the signed inverse tangent function $\tan^{-1}(\cdot)$ is given by (4.10). It is worthwhile to recall that, given the control input $u(\cdot)$,

$$T^*(t) = \arg\min \|MT(t) - u(t)\|^2, \qquad t \geq t_0, \tag{4.44}$$

and hence, (4.42) captures the vector of thrust forces that most closely realizes some $u(\cdot)$ [47, p. 153], [30, Prop. 6.1.5].

For conventional quadcopters, it holds that $\alpha_i = 0$, $i = 1, \ldots, 4$, and $T_i(t) = T_{i,\mathrm{c}}(t)$, $i = 1, \ldots, 4$, $t \geq t_0$, $T_{i,\mathrm{s}}(t) = 0$. Hence, it follows from (4.39) that $u_5(t) = 0$, (4.40) reduces to

$$M = \begin{bmatrix} 1 & 1 & 1 & 1 \\ L_y & L_y & -L_y & -L_y \\ -L_x & L_x & L_x & -L_x \\ k_\mathrm{T} & -k_\mathrm{T} & k_\mathrm{T} & -k_\mathrm{T} \end{bmatrix}, \tag{4.45}$$

whose determinant is $\det(M) = 16 L_x L_y k_T$, which is full-rank since both $L_x L_y > 0$

and $k_T > 0$ by assumption, and (4.42) reduces to

$$\begin{bmatrix} T_1^*(t) \\ T_2^*(t) \\ T_3^*(t) \\ T_4^*(t) \end{bmatrix} = \frac{1}{4} \begin{bmatrix} 1 & L_y^{-1} & -L_x^{-1} & k_{\mathrm{T}}^{-1} \\ 1 & L_y^{-1} & L_x^{-1} & -k_{\mathrm{T}}^{-1} \\ 1 & -L_y^{-1} & L_x^{-1} & k_{\mathrm{T}}^{-1} \\ 1 & -L_y^{-1} & -L_x^{-1} & -k_{\mathrm{T}}^{-1} \end{bmatrix} \begin{bmatrix} u_1(t) \\ u_2(t) \\ u_3(t) \\ u_4(t) \end{bmatrix}; \tag{4.46}$$

for details, see [48].

Chapter 5: Experiment Design and Results

This chapter starts by describing the components on the tilt-rotor shown in Figure 1.2 and used to test the robust adaptive control technique in Chapter 2 and the control strategy presented in Chapter 4. Next, a thorough description of the laboratory and the setup for the experiment is given including descriptions of all of the communications to and from the vehicle and the motion capture system. Finally, the results of flight experiments are provided and discussed.

5.1. Tilt-rotor Components and Parameters

5.1.1 Frame and Body

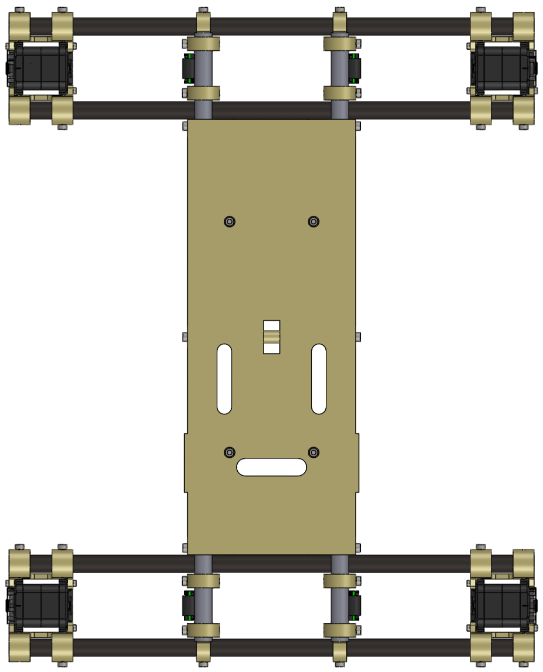


Figure 5.1: Computer Aided Design (CAD) model of the main frame of the tilt-rotor quadcopter

The main frame of the tilt-rotor, depicted as a SolidWorks rendering in Figure 5.1, was entirely custom designed by the Army Research Lab (ARL). The main supports are $10mm$ diameter braided carbon fiber rods, which are very strong and lightweight. All of the connector pieces and the main plates, which hold all of the onboard electronics, are 3D printed plastic parts. While these parts are not as sturdy as the carbon fiber, they can be rapidly replaced due to being 3D printable. This entire frame is $0.50m$ by $0.35m$ long and weighs in at approximately $0.65kg$.

5.1.2 Pixhawk Autopilot and Odroid XU4

For the experiments performed on this research, it was desirable to have an autopilot which could act as both the inertial measurement unit (IMU) and as a backup flight controller as a fail safe while tuning the new adaptive law. The Pixhawk autopilot [49], developed by 3D Robotics, is the solution to this issue. The main chip is a $180MHz$ single core chip which contains a real-time operating system. There are several fully integrated sensors including a 16 bit gyroscope, a 14 bit accelerometer and magnetometer, and a barometer. The Pixhawk also contains 5 serial port connections for communication and a Spektrum DSM compatible radio input. Moreover, it contains 8 pulse-width modulation (PWM) out ports for doing motor control as well as 4 additional auxiliary ports for controlling servos.

The flight stack code that is currently used on this Pixhawk is the PX4 stack, version 1.6.5. The PX4 stack has the capability of blending all of the sensor data using an Extended Kalman Filter (EKF), and it allows for switching the position

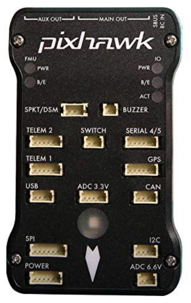


Figure 5.2: The Pixhawk Autopilot used on the platform.

inputs from a global positioning system (GPS) to inputs from the Vicon Motion Capture System located in the lab. This flight stack also has its own control code, running proportional-integral-derivative (PID) control.

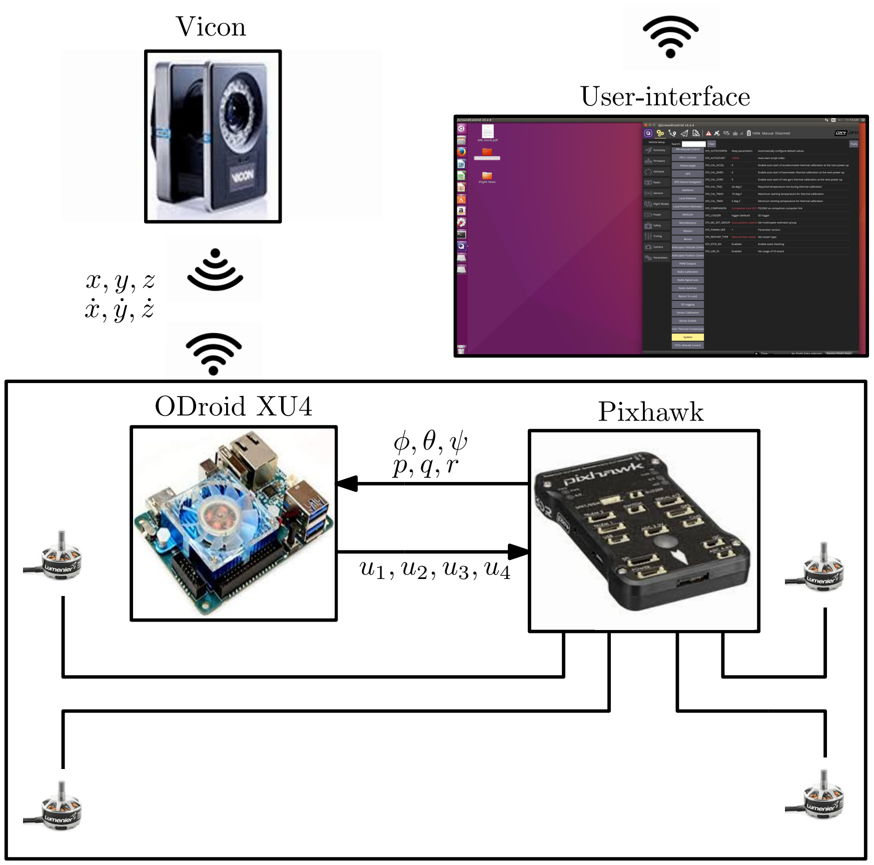


Figure 5.3: The current setup for Odroid, Pixhawk, and Vicon communication. Vicon cameras deduce the position and velocity of the drone and a ground station computer streams that data over WiFi to the Odroid onboard. The Odroid receives the attitude data from the Pixhawk's IMU and computes the control inputs.

In addition to the Pixhawk, an Odroid XU4 is also used as part of the autopilot structure. The Odroid is a single board computer with a $2Ghz$ processor and $64GB$ embedded multi-media chip (eMMc) flash storage. For this autopilot, the Odroid receives the attitude of the vehicle courtesy of the Pixhawk's EKF filter which is transmitted to the Odroid via serial communication. The global position and velocity of the drone is deduced using a Vicon Motion Capture System and is transmitted from a ground station to the Odroid via a dedicated UDP stream over WIFI. With the full state data, the Odroid then runs the control algorithm as described in 4 and transmits the final control signals to both the mux board and Pixhawk for actuation. For details on how to set up the Odroid for the current flight architecture, see A.1.

5.1.3 Propulsion System

Figure 5.4: The Castle DMR 30-40 electronic speed controllers (ESC's) used for this vehicle.

The propulsion system for small multi-rotor vehicles usually consists of three main components, namely electronic speed controllers (ESCs), brushless DC motors, and propellers. The ESC's chosen for this vehicle are the Castle DMR 30-40 [50] as shown in Figure 5.4. These specific ESCs can handle up to $40A$ of current and up to 25.2 Volts, or the equivalent of a $6s$ Lithium Polymer (LiPo) battery. Both are well above what is run on this vehicle as the current configuration only includes a 4 cell $14.8V$ battery, and the system rarely pulls above $30A$. This means that if more thrust is needed on this vehicle, the ESCs won't be the limiting factor.

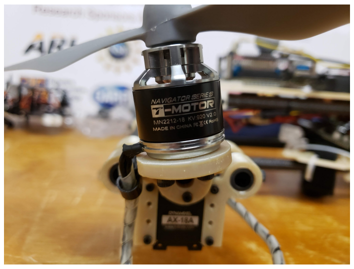


Figure 5.5: Tiger-Motor MN2212 920KV motors.

Motors and propellers are almost always chosen in tandem for small UAVs. In this case, the vehicle was estimated to weigh approximately $2.0kg$, and a typical design specification for small UAVs is to require at least a 1.5-1 thrust to weight ratio for the vehicle. This would require at least $3.5kg$ of thrust, and for additional safety margin, there should be a small buffer. For this reason, the chosen motors were the Tiger Motor MN2212 940KV motors [51], see fig. 5.5, paired with the T-Motor 9×3 carbon fiber propellers [52]. The thrust of this motor/propeller combination was estimated using manufacturer provided experimental data. The results of plotting the provided data for 1 motor/propeller shows a maximum thrust of just under $9N$. This gives an estimated total thrust of the vehicle of $35N$ or $3.6kg$ for a thrust to weight ratio of 1.8. In addition, for computing the desired angular velocity of the propeller as in eqn. (4.37), the thrust coefficient should be known. Using a linear fit as shown in fig. 5.6, the thrust coefficient of this combination was found to be $k = 1.73e^{-7}N/Hz^2$.

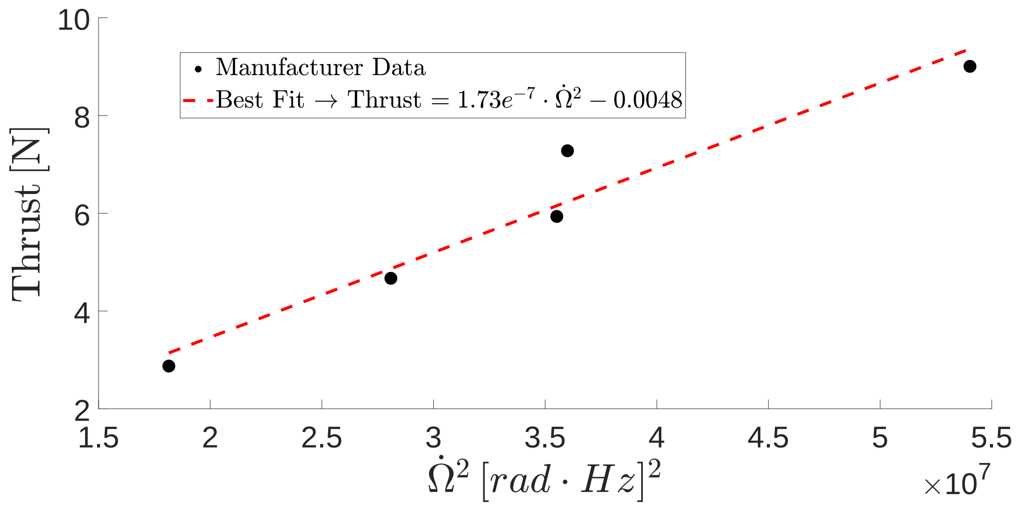


Figure 5.6: The thrust curve fitting plot showing the thrust coefficient, k, was found using bench top thrust data provided by the manufacturer of the motors.

5.1.4 Battery

To power the vehicle, the chosen battery was the MaxAmps $14.8V$ $3250mAh$ version. It is common nowadays for small UAVs to run anything from 3-6 cell LiPo batteries, but for this case the 4 cell was chosen. The reason for this is mutlifold. Firstly, the motors chosen are not capable of running on the higher voltage of $6s$ batteries, so they would have to be upgraded to do so. In addition, the chosen battery weighs approximately $0.327kg$ whereas a comparable $6s$ battery weighs twice as much at $0.692kg$, and while cost is not a limiting factor, the $6s$ battery costs three times more and would provide very little additional flight time.

Figure 5.7: The $14.8V$ $3250mAh$ battery from MaxAmps chosen for this setup.

5.1.5 Servos for Thrust Vectoring

To actuate the motors and propellers, the AX-18A Dynamixel servos are used [53]. According to the servo's manual, they have a resolution of less than 1/3 of a degree, and they have a stall torque of $18.3kg \cdot cm$ which should be more than enough for UAV applications. In addition to having a high resolution, these servos also have a large range having more than 300 degrees of total freedom. This is desired since one of the future desired experiments is to fly the vehicle at a 90 degree pitch angle.

Figure 5.8: The AX-18A Dynamixel servos mounted on the tilt-rotor.

There is very little setup required for the Dynamixels. They are pre-calibrated by the manufacturer, and the user is only thing required to number them correctly according to Figure 3.1. This operation is performed by using a usb2Dynamixel cable as shown in Figure 5.9 and downloading the Dynamixel Wizard (link). In particular, after having plugged the cable into the Dynamixels and a USB port on a computer, load the Dynamixel Wizard, and then change the “ID Number” of the Dynamixel to the same number as the motor it is connected to in the vehicle diagram.

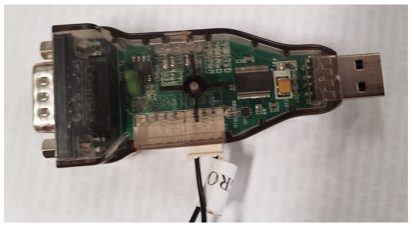


Figure 5.9: The usb2Dynamixel cable that allows for connecting the Dynamixel servos to a USB port on a computer.

5.1.6 Actuation Selection – Mux and Maestro Boards

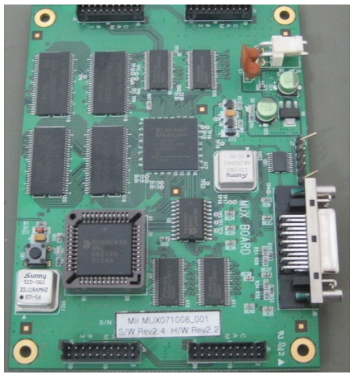


Figure 5.10: The mux board used is custom built by the Army Research Lab. It is used to select whether to use the custom flight controller on the Odroid or the Pixhawk flight controller for vehicle actuation.

To actuate the vehicle from the custom flight code, a mux and Maestro board are both used. The control inputs are first sent from the Odroid to the mux board. This mux board is custom built at ARL, and it takes in desired PWM motor commands from the Pixhawk's PID controller and the custom MRAC controller on the Odroid and selects which signals to pass on depending on the state of the auxilliary switch on the DX9 radio. This way, the user is given the option to choose what control algorithm to use. Furthermore, the user is given manual control of the vehicle in case of failures in the autopilot. After the signal passes the mux board, it goes into the Maestro board, a Mini Maestro 12-Channel USB Servo Controller, which converts the desired PWM commands into actual PWM signals to be sent to the ESCs. A diagram depicting this architecture is shown in Figure 5.11.

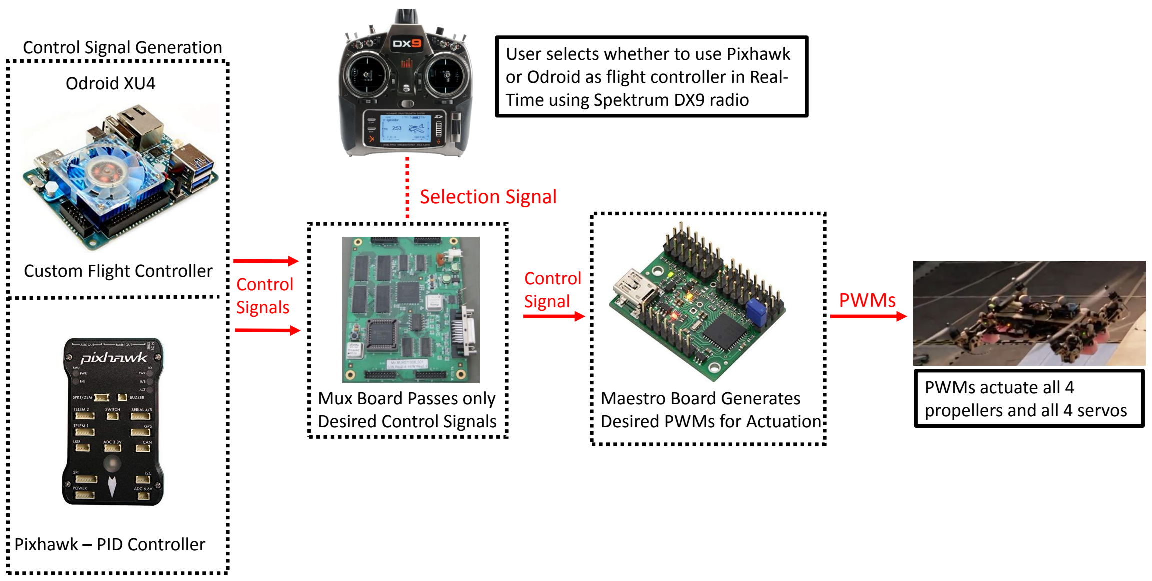

Control Signal Generation
Odroid XU4
Custom Flight Controller
Pixhawk – PID Controller
Control Signals
User selects whether to use Pixhawk or Odroid as flight controller in Real-Time using Spektrum DX9 radio
Selection Signal
Mux Board Passes only Desired Control Signals
Control Signal
Maestro Board Generates Desired PWMs for Actuation
PWMs
PWMs actuate all 4 propellers and all 4 servos

Figure 5.11: This diagram shows how the mux and maestro boards along with user inputs from the DSM radio determine which flight controller has control of the vehicle.

5.2. Experimental Results

In this section, an experiment is performed to show the ability of the robust MRAC control algorithm from Section 2.3 using the vehicle described in Section 5.1 to accomplish a challenging mission scenario, while imposing user-defined constraints on both the trajectory tracking error and the estimated adaptive gain's error at all time. This experiment involves the custom-made tilt-rotor quadcopter with H-configuration connected to a cart by a thin rope (a fishing wire); see Figure 3.1 for a schematic representation of the test configuration. The aircraft's mass is $2.0\,kg$ and the cart's mass is $6.2\,kg$ so that $m = 8.2\,kg$.

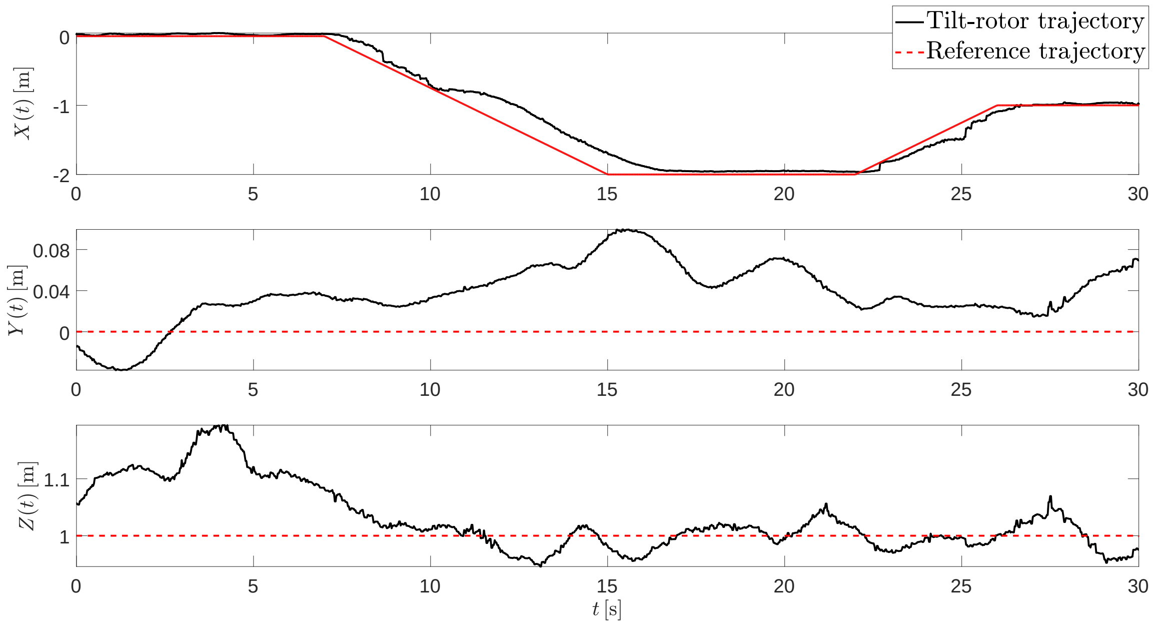


Figure 5.12: This figure shows both the vehicle and reference trajectories for the experiment.

The inertia matrix of the cart is unknown and the aircraft inertia matrix is esti-

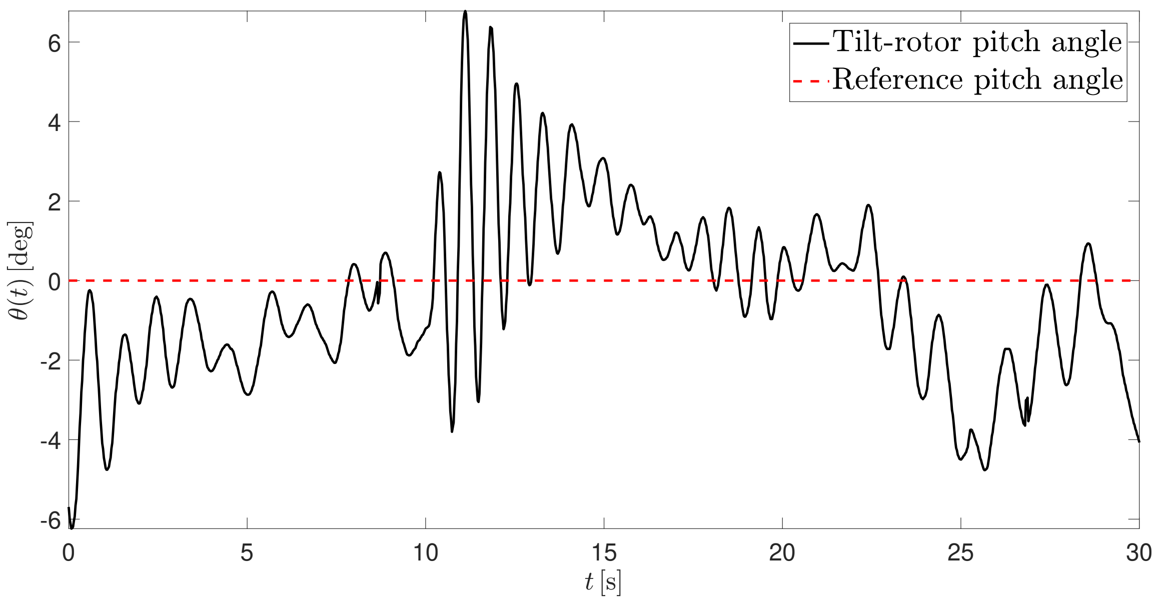


Figure 5.13: This plot shows the pitch angle, $\theta(\cdot)$, for the mission. The $\theta(\cdot)$ angle oscillates around the desired point, 0, and never exceeds 6.5 degrees.

mated using a CAD (computer aided design) model so that

$\overline{I}_{\text{quad}}(t) = \begin{bmatrix} 0.0208 & 0.0000 & 0.0000 \\ 0.0000 & 0.0468 & 0.0000 \\ 0.0000 & 0.0000 & 0.0303 \end{bmatrix}$ $\text{kg}\cdot\text{m}^2$, $t \geq t_0$. The quadcopter's propellers are modeled as thin disks of mass $m_{\text{prop}} = 0.0057\,\text{kg}$ and radius $\rho_{\text{prop}} = 0.1145\,\text{m}$. The body reference frame is centered at a conveniently located point $A(\cdot)$ so that the propellers are located at $r_{\text{prop},1} = [-0.1625, -0.1958, 0.0183]^{\text{T}}\,m$, $r_{\text{prop},2} = [0.1625, -0.1958, 0.0183]^{\text{T}}\,m$, $r_{\text{prop},3} = [0.1625, 0.1958, 0.0183]^{\text{T}}\,m$, and $r_{\text{prop},4} = [-0.1625, 0.1958, 0.0183]^{\text{T}}\,m$, respectively.

The quadcopter's mission is to hover for $t \in [0, 7]\,s$, travel two meters in the negative X axis direction for $t \in [7, 15]\,s$, hover for $t \in [15, 22]\,s$, travel one meter in the positive X axis direction for $t \in [22, 26]\,s$, and hover for $t \in [26, 30]\,\text{s}$, while maintaining a constant altitude of one meter and zero pitch angle at all times. The challenges for this mission are multifold. Firstly, for $t \in [0, 10.1]\, s$, the rope is not taut and hence, initially the controlled mechanical system consists of the aircraft only. At $t = 10.1\, s$, the controlled mechanical system's inertia properties vary almost instantly, and the control algorithm must produce satisfactory results despite substantial uncertainties in the location of the center of mass and the overall system's inertia matrix introduced by the cart. Moreover, the friction between the cart and the floor is not modeled. Lastly, the rope is not connected to the tilt-rotor's center of mass and hence, when the rope is taut, the cart induces a pitching moment on the vehicle.

Firstly, one can see in Figure 5.12 that the vehicle was successfully able to track the desired trajectory despite the unknown and unexpected disturbance of the attached payload. The vehicle is able to adjust to the new mass and track the moving desired x position while maintaining altitude within $0.15m$ and y position within $0.09m$. At the end of the simulation, after $22s$, the rope is no longer taut, and it can be seen that the vehicle still maintains accurate tracking. As seen in Figure 5.13, the vehicle was able to maintain a very small pitch angle while pulling this very heavy cart along the floor. The absolute value of the pitch angle never grows larger than 6.5 degrees , which is something a standard quadrotor could not do while pulling a cart.

In addition to accurate tracking, the novel control law proposed in this thesis was able to verify the user-defined constraints at all time. To validate the necessity of having the constrained version of MRAC, both the classical MRAC presented in Section 2.1 and the e-modification of MRAC presented in Section 2.2 were used to perform the identical experiment. Figure 5.15 shows how the constraint function

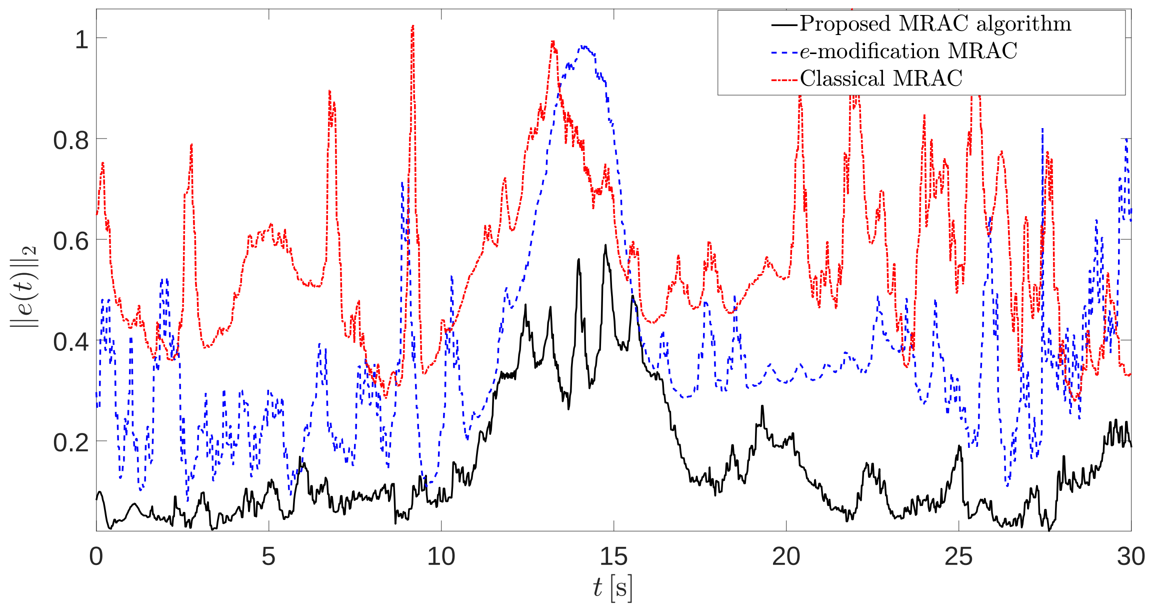


Figure 5.14: Plot of the norm of the trajectory tracking error norm, $\|e\|$, which shows that the proposed control law outperforms the classical versions of Model Reference Adaptive Control.

evolves throughout the mission. It can be seen that only the proposed constrained version of MRAC is able to verify the constraints for the duration of the mission. Both the classical and e-modification versions significantly violate the constraints within seconds of takeoff. In addition, while meeting the constraints, the norm of the trajectory tracking error of the new control law also proved to be consistently lower than the classical versions. Figure 5.14 shows how the error evolves throughout the test, and one can see that the biggest disturbance in the experiment is when the rope gets taut around $12 - 15s$. While the constrained MRAC has the best performance, it is worthwhile to not that the e-modification of MRAC does seem to outperform the classical version when it comes to maintaining a lower state tracking error.

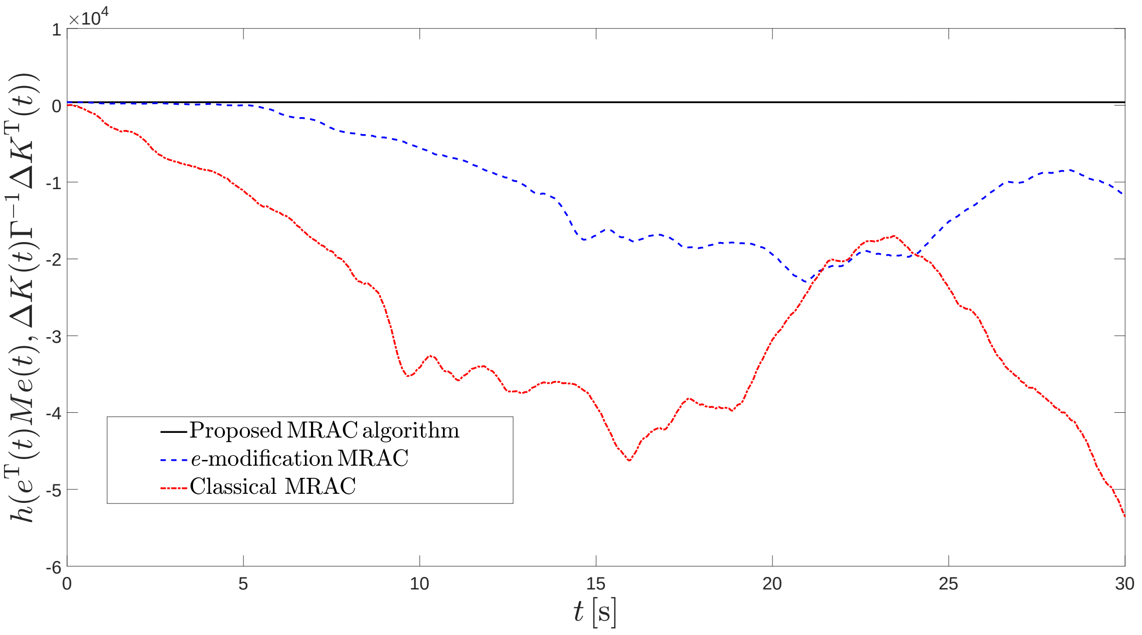


Figure 5.15: A plot showing the evolution of the constraint function, $h(e^{\mathrm{T}}Me, \Delta K\Gamma^{-1}\Delta K^{\mathrm{T}})$, throughout the experiment. For the proposed control law, the constraint function stays positive for the duration of the mission, whereas both of the classical control techniques violate the constraint and go below 0.

Chapter 6: Conclusion

As small unmanned aerial systems start encountering more challenging mission scenarios, the more important having robust control algorithms becomes. In addition, it may be even more important to have vehicles that can retain guaranteed margins on the trajectory tracking error as these systems start performing interactions in close quarters to humans.

In Chapter 2, a form of adaptive control known as Model Reference Adaptive Control (MRAC) was introduced starting with classical MRAC. This form of control has adaptive gains and can acheive good performance when there are small uncertainties in the dynamic model or control input. The second form of MRAC introduced was Robust MRAC which utilized the σ-modification to retain stability even in the presence of unmatched uncertainty. In the final section, a novel form of MRAC was formulated which allowed for user-defined constraints to be imposed on both the trajectory tracking error and the error estimate of the adaptive gains. More importantly, these constraints can be imposed a priori, or before the system is even active. This control algorithm was formulated and asymptotic convergence was proven.

In Chapter 3, the equations of motion of a tilt-rotor quadcopter with H-configuration were developed. This type of vehicle is different from the standard quadcopter in that not all of the thrust force is in 1 plane. This led to a direct coupling between the kinematic and rotational equations of motion. In addition, due to the thrust vectoring capability of the vehicle, this platform can actuate along one of the position coordinates without having to rotate the body frame along that axis. One of the key assumptions in this formulation involved the idea that the position of the center of mass and the vehicle's matrix of inertia could be considered as unknown. This allowed for a very challenging test case for the new proposed adaptive control law.

In Chapter 4, the constrained MRAC control law is applied to a tilt-rotor quadcoptor. After writing the equations of motion in a standard form, it is shown that this vehicle is underactuated. To this end, an outer loop is designed to regulate the one uncontrollable degree of freedom for this system, the $y(\cdot)$ position, through manipulation of the vehicle's roll angle, $\phi(\cdot)$. For the main adaptive law, the regressor vector is chosen such that it contains the uncertainty in the vehicle's center of mass.

In Chapter 5, all of the components making up the tilt-rotor quadcopter are discussed. The autopilot architecture includes having a Pixhawk IMU as well as an Odroid XU4 for calculating the control outputs. These outputs are then used to actuate all of the motors as well as the Dynamixel servos which allow for thrust vectoring. A description of the experimental setup is given where it is shown how the Vicon Motion Capture space is used to deduce the position and velocity of the UAV, and the Odroid on the vehicle takes in all of the state data and computes the desired thrusts and tilt angles. To test the control law and the vehicle, an experiment was performed where the $2kg$ vehicle is tasked with pulling a $6.2kg$ cart along the x axis to introduce a significant uncertainty in the center of mass of the vehicle as the control algorithm knows nothing of the attached payload. The results of these experiments show that even with this large uncertainty, the control law still stayed within a priori defined bounds on the trajectory tracking error and the estimated adaptive gain error. Similar experiments with both classical MRAC and classical Robust MRAC were performed, and in each case the constraints were violated and the error tracking norms were much higher indicating worse performance.

6.1. Future Directions

A small vehicle being able to withstand an unsteady center of mass is a major step towards doing aerial manipulation. The last decade of drone research has largely been spent focusing on the guidance and navigation part of spectrum. Drones have mostly been tasked with either surveying or mapping an environment. In the near future, these UAVs should interact with the enviornment. The goal is to have a small vehicle that can pick up objects, manipulate objects such as doors or windows, or even place objects within an environment. Specifically, this research group is looking to have a vehicle that can autonomously seek and install a surveillance package high up on a wall. This scenario would include the unsteady center of mass in addition to the challenges of flying close to hard surfaces, namely the "wall effect." It is believed that robust adaptive control laws such as the one presented in this thesis will be necessary to complete such missions.

This lab is not only interested in the completion of challenging missions, but the consistency of the performance is equally important. We are interested in looking at performing the same mission scenario dozens of times, and doing statistical analysis on these trajectories to help optimize the control gains. As seen in 2.3, there are dozens of gains from Γ to K_p, K_d, which are all picked by the user. To the author's best knowledge, there is currently no scientific method to selecting or tuning these parameters. We are looking at possibly using industrial engineering techniques such as the Taguchi Method to optimize these gains for the best trajectory tracking error, which is deduced using a large amount of actual flight experiment data. In addition to giving the best control gains, this method should also allow us to study the repeatability of the experiments as the standard deviations in the tracking error are one of the key parameters in Taguchi's method.

Chapter A: Appendix 1

This Appendix has been written as a user manual to setup a new Odroid XU4 microcomputer similar to those used to perform flight experiments discussed in this thesis. The final section of the appendix is the C++ code of the control algorithm of Section 2.3.

A.1. Setting Up a New Odroid

The Odroid we are using came with Ubuntu 16.04 already preloaded, so this assumes that you already have Ubuntu installed. The first thing we will do is update, upgrade, and finally start installing the necessary packages and applications that we wish to use. For those new to Linux/Ubuntu *sudo* means 'superuser' (similar to 'admin' for Windows). The following is a list of commands to run at the terminal window which can be accessed by pressing ctrl+alt+t. Note when you first get an Odroid the password is odroid (all lowercase).

- `sudo apt-get update`

  This command updates the Odroid. Windows does this type of thing in the background each time you login to your computer. We recommend doing this command almost daily.

- `sudo apt-get upgrade`

  This command that should be done often

- `sudo apt-get install build-essential`

  This command installs build-essential which is compilers for c, c++, etc

- `sudo apt-get install wireshark`

  Wireshark is a tool that can be used to help debug and/or set up communications

- `sudo apt-get install default-jdk default-jre`

  This command installs Java and the Java SDK

- `sudo apt-get install eclipse-cdt`

  This command installs the development environment and code editor that we use for the flight code

- `sudo apt-get install libboost-all-dev`

  Get the Boost libraries that we use for both communication protocol as well as numerical integration in the control

- `sudo apt-get install git`

  This will allow for easy access to the github repositories

Troubleshooting

If you get an error of "Unable to lock the administration directory," then run the following commands in the terminal:

```
- sudo rm /var/lib/apt/lists/lock
- sudo rm /var/cache/apt/archives/lock
- sudo rm /var/lib/dpkg/lock
```

If you get an error of "dpkg status database is locked by another process," then run the following commands in the terminal:

```
- lsof /var/lib/dpkg/lock
- kill PID
- #wait
- kill -9 PID
- sudo rm /var/lib/dpkg/lock
- sudo dpkg --configure -a
```

A.1.1 Eclipse Settings

One of the things we noticed is that not enough memory was allocated for Eclipse, and it was crashing during code builds. A way to fix the problem is to give Eclipse more memory as follows:

- Go to the file explorer and navigate to /usr/lib/eclipse.
- Right-click on eclipse.ini and edit as administrator.

- Change the Xmx384m line to Xmx1024m (This gives the program significantly more RAM access).
- Save and close the file.

A.1.2 Installing Flight Code

After these initial updates and installs, copy the flight code onto the Odroid and save in /home/odroid. Now open eclipse by typing its name at the command line. Open the workspace in which the flight code is saved, in our case the folder is /workspace3_BASELINEBUILD. The first time we open this we may have to tell Eclipse to include a few things before we can compile. Specifically,

- On the left side of the screen under Project Explorer, right-click PixhawkControl_4 and select properties.
- Hit the drop-down arrow next to C/C++ Build.
- In the center under Tool Settings, look at GCC C++ Compiler, includes. Click on "include paths" on the right make sure it says /home/odroid/workspace3_BASELINEBUILD/PixhawkControl_4/src.
- Under the same Tool Settings, look at GCC C++ Linker. On the right under libraries you need to add separately boost_filesystem, boost_system, and pthread. (insert image to show what result looks like).

The provided C++ flight code is almost ready-to-use. One necessary change is that the Mavlink libraries which are currently located in the same folder as the workspace3 folder need to be moved by following these instructions:

Placing MAVLink Libraries

- Open a file explorer window.
- Navigate to filesystemusr and open the "include" folder as administrator.
- When the new window with the include folder opens, copy the MAVLink folder from the odroid flight software location to include.
- Now the files are in the correct location, but we need to correct the permissions access to them.
- Open a terminal and navigate to usrinclude, then type the following
- `sudo chmod -R 755 MAVLink/`

A.1.3 Set Odroid to Automatically Login When Powering On

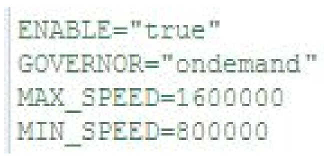
Since the Odroid is run onboard the vehicle, it is beneficial to have it login automatically when the battery is plugged into the drone. To accomplish this we simply need to edit one file. Navigate to the folder /usr/share/lightdm/lightdm.conf.d/ and right click on the file named 60-lightdm-gtk-greeter.conf. Open this file as administrator and after the line,

```
greeter-session=lightdm-gtk-greeter
```

A.1.4 Setting Odroid to Performance Mode

Perfomance mode allows the Odroid to run at closer to its full power which removes an issue where either the data stream or control applications don't run fast enough. First install cpufrequtils from the terminal with,

```
sudo apt-get install cpufrequtils.
```



```
ENABLE="true"
GOVERNOR="ondemand"
MAX_SPEED=1600000
MIN_SPEED=800000
```

Figure A.1: Performance Mode Settings

A.1.5 Set IP Address

To make sure the data is streamed to the Odroid, it must have a known static IP address on the network. Changing this is fairly straightforward and can be done by first connecting to the desired network. After connecting, select edit connections in the same place as you would select a wireless network to join. From the list of Wi-Fi connections, select the one we wish to change and click edit. From here, go to the IPv4 settings and add an address that matches your desired IP address as well as the Netmask and Gateway. Click save and exit.

A.1.6 Odroid to Pixhawk Cable

(a) (b)

Figure A.2: Pixhawk to Odroid Connection Cables

This short section covers how to make the cable allowing for communication between our two main components, the Odroid and the Pixhawk. The first thing needed is a 3.3 Volt FTDI to USB cable as shown above. Then you need one of the 6 pin connectors used on the Pixhawk for telemetry and serial port connections. To combine the two, the FTDI cable is cut near the USB side which will remain as the method for connecting to the computer. solder the wires together as shown in the figure below. Note only pins 1,4,5 have connections, and the rest of the wires can be cut.

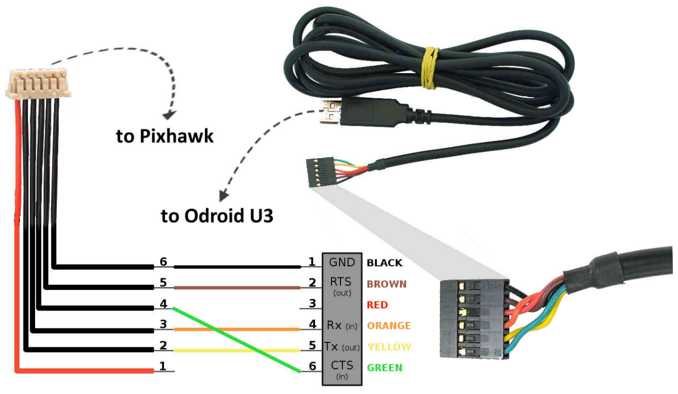


Figure A.3: FTDI Wiring Diagram

A.2. Control Code

```
void Control_code()

{

CMRAC_params.t = CMRAC_params.t / 1000; //scale time from ms to s



//This initialization only runs once, sets up all matrices for the controller

    if(CMRAC_params.initialization == 0){

    // Input tunable gains into Kp, Kd

                Kp(0,0) = Control_Gains_ouCMRAC.kx;

                Kp(1,1) = Control_Gains_ouCMRAC.ky;

    Kp(2,2) = Control_Gains_ouCMRAC.kz;

    Kp(3,3) = Control_Gains_ouCMRAC.kphi;

    Kp(4,4) = Control_Gains_ouCMRAC.ktheta;

    Kp(5,5) = Control_Gains_ouCMRAC.kpsi;

    Kd(0,0) = Control_Gains_ouCMRAC.kx_dot;

    Kd(1,1) = Control_Gains_ouCMRAC.ky_dot;

    Kd(2,2) = Control_Gains_ouCMRAC.kz_dot;

    Kd(3,3) = Control_Gains_ouCMRAC.kphi_dot;

    Kd(4,4) = Control_Gains_ouCMRAC.ktheta_dot;

    Kd(5,5) = Control_Gains_ouCMRAC.kpsi_dot;


    // Input Kp, Kd into Aref Matrix

    Aref << MatrixXf::Zero(6,6), MatrixXf::Identity(6,6),

         -1*Kp, -1*Kd;


    // Bref, B From Matrices on Pg 12
```

```
 Bref << MatrixXf::Zero(6,6), MatrixXf::Identity(6,6);

 B << MatrixXf::Zero(6,6), MatrixXf::Identity(6,6);


// Initial guess for Theta_e

Theta_e << MatrixXf::Zero(39,6);


// User-defined, positive-definite, best to go with diagonal matrix

Q << MatrixXf::Identity(12,12);


// Identity

Im << MatrixXf::Identity(6,6);


// Hx, He from pg 4, right column

Hx << -1*MatrixXf::Identity(6,6);//-1*MatrixXf::Identity(6,6);

CMRAC_params.he = 0;


// Weights for trajectory in Constraint function,

            //only requirement is positive definite

M << MatrixXf::Identity(12,12);


// Combine Kp, Kd, Theta_e into Ke matrix

Ke << Kp, Kd, MatrixXf::Zero(6,6), Theta_e.transpose();


// Adaptive Gains -- Come from text file

Gamma_x << MatrixXf::Identity(57,57);

Gamma_x(0,0) = Control_Gains_ouCMRAC.gammaxx;

Gamma_x(1,1) = Control_Gains_ouCMRAC.gammaxy;
Gamma_x(2,2) = Control_Gains_ouCMRAC.gammaxz;

Gamma_x(3,3) = Control_Gains_ouCMRAC.gammaxphi;

Gamma_x(4,4) = Control_Gains_ouCMRAC.gammaxtheta;

Gamma_x(5,5) = Control_Gains_ouCMRAC.gammaxpsi;

Gamma_x(6,6) = Control_Gains_ouCMRAC.gammaxxdot;

Gamma_x(7,7) = Control_Gains_ouCMRAC.gammaxydot;

Gamma_x(8,8) = Control_Gains_ouCMRAC.gammaxzdot;

Gamma_x(9,9) = Control_Gains_ouCMRAC.gammaxphidot;

Gamma_x(10,10) = Control_Gains_ouCMRAC.gammaxthetadot;

Gamma_x(11,11) = Control_Gains_ouCMRAC.gammaxpsidot;


for (int i = 12; i < 57; i++)

   {

      Gamma_x(i,i) = Control_Gains_ouCMRAC.gammatheta;

   }


// Prepare Gammax_inverse for later

Gamma_x_inv = Gamma_x.inverse();


// Initialize qdesdot and Alpha

qdesddot << MatrixXf::Zero(6,1);

Alpha << MatrixXf::Zero(6,1);


// P: Solution to Lyapunov Eqn, found in Matlab

P << MatrixXf::Identity(12,12);

P(0,0) = 2.633;

P(0,6) = 0.1667;
```

```
P(1,1) = 2.633;

P(1,7) = 0.1667;

P(2,2) = 1.2083;

P(2,8) = 0.125;

P(3,3) = 2.6833;

P(3,9) = 0.0167;

P(4,4) = 2.4136;

P(4,10) = 0.0227;

P(5,5) = 1.9971;

P(5,11) = 0.02;

P(6,6) = 0.8333;

P(6,0) = 0.1667;

P(7,7) = 0.8333;

P(7,1) = 0.1667;

P(8,8) = 0.2083;

P(8,2) = 0.125;

P(9,9) = 0.0861;

P(9,3) = 0.0167;

P(10,10) = 0.1045;

P(10,4) = 0.0227;

P(11,11) = 0.0743;

P(11,5) = -0.02;


// Input values that don't change in rotation matrices

R_theta(1,0) = 0;

R_theta(0,1) = 0;

R_theta(1,1) = 1;
R_theta(2,1) = 0;

R_theta(1,2) = 0;

R_psi(2,1) = 0;

R_psi(2,0) = 0;

R_psi(0,2) = 0;

R_psi(1,2) = 0;

R_psi(2,2) = 1;


// Inertia guess for Tiltrotor: off diagonal terms seem to cause no difference

Inertia(0,0) = 0.02078;

Inertia(1,0) = 0.0;//-.00004;

Inertia(2,0) = 0.00004;

Inertia(0,1) = 0.0;//-0.00003;

Inertia(1,1) = 0.04682;

Inertia(2,1) = 0.0;//-0.00001;

Inertia(0,2) = 0.00004;

Inertia(1,2) = 0.0;//-0.00001;

Inertia(2,2) = 0.03032;


// Propeller inertia

Inertia_prop = MatrixXf::Zero(3,3);

Inertia_prop(0,0) = 0.0002391;

Inertia_prop(1,1) = 0.0001711;

Inertia_prop(2,2) = 0.0004064;


// Mbar from eqn 47: off diagonal terms are zero

            // since we assume rc = 0, ie cg at center of coordinate system
```

```
Mbar << CMRAC_params.mass*MatrixXf::Identity(3,3), MatrixXf::Zero(3,3),
        MatrixXf::Zero(3,3), Inertia;

Omega_cross(0,0) = 0;
Omega_cross(1,1) = 0;
Omega_cross(2,2) = 0;

Gamma_matrix(0,0) = 1;
Gamma_matrix(1,0) = 0;
Gamma_matrix(2,0) = 0;

// Initialize regressor vector
Phi_tran = MatrixXf::Zero(9,1);
Phi_rot = MatrixXf::Zero(30,1);



CMRAC_params.dt_derivative = 0.0;
CMRAC_params.phi_des_prev = 0.0;
CMRAC_params.t_prev_derivatives = 0.0;
CMRAC_params.derivative_counter = 0;

CMRAC_params.xdesdot = 0.0;
CMRAC_params.ydesdot = 0.0;
CMRAC_params.zdesdot = 0.0;
CMRAC_params.phi_desdot = 0.0;
CMRAC_params.theta_desdot = 0.0;
CMRAC_params.psi_desdot = 0.0;
```

```
    CMRAC_params.x_des_prev = 0.0;

    CMRAC_params.y_des_prev = 0.0;

    CMRAC_params.z_des_prev = 0.0;


    CMRAC_params.a1_prev = 0.0;

    CMRAC_params.a2_prev = 0.0;

    CMRAC_params.a3_prev = 0.0;

    CMRAC_params.a4_prev = 0.0;

    CMRAC_params.a5_prev = 0.0;

    CMRAC_params.a6_prev = 0.0;


    CMRAC_params.initialization = 1;

}




   // Open File to save variables

   if ((CMRAC_params.myfile.is_open() != 1) && (CMRAC_params.t > 45))

      {

        CMRAC_params.myfile.open("Flight_Data_CMRAC2.txt");

        CMRAC_params.counter2 = 30;

        CMRAC_params.initial_x = CMRAC_params.x_in;

        CMRAC_params.initial_y = CMRAC_params.y_in;

      }


        CMRAC_params.Lambda_switch = 1;

        CMRAC_params.Lambda_switch_tilt = 1;

        CMRAC_params.Lambda_integration = 1;
```

```
// Note the trajectory for now is hard-coded.
// We have had issues with the get trajectory function
// Mission now takes off at whatever location,
// holds for a few seconds, then moves back 2.5m in x,
// moves forward 1m in x, then lands
/**********Initialization***********************/
if ((CMRAC_params.t < 50) && (CMRAC_params.t > 0.0))
{
     CMRAC_params.refs_temp[0][0] = 0;
     CMRAC_params.refs_temp[1][0] = 0;
     CMRAC_params.refs_temp[2][0] = 0;
     CMRAC_params.refs_temp[3][0] = 0;
     CMRAC_params.Lambda_switch = 0;
     CMRAC_params.Lambda_integration = 0;
}

/**********Takeoff Phase***********************/
if ((CMRAC_params.t > 50) && (CMRAC_params.t < 53))
{
   CMRAC_params.refs_temp[0][0] = 0;
   CMRAC_params.refs_temp[1][0] = 0;
   CMRAC_params.refs_temp[2][0] = (CMRAC_params.t-50)/3.0;
   CMRAC_params.refs_temp[3][0] = 0;
   //CMRAC_params.Lambda_integration = 0;
}
```

```
if ((CMRAC_params.t >= 53) && (CMRAC_params.t < 55))
{
                CMRAC_params.refs_temp[0][0] = 0.0;
    CMRAC_params.refs_temp[1][0] = 0.0;
    CMRAC_params.refs_temp[2][0] = 1.0;
    CMRAC_params.refs_temp[3][0] = 0;
}
if ((CMRAC_params.t >= 55) && (CMRAC_params.t < 59))
{
    CMRAC_params.refs_temp[0][0] = 0.0;//(CMRAC_params.t - 55)/4;
    CMRAC_params.refs_temp[1][0] = 0.0;
    CMRAC_params.refs_temp[2][0] = 1.0;
    CMRAC_params.refs_temp[3][0] = 0;
}
if ((CMRAC_params.t >= 59) && (CMRAC_params.t < 62))
{
    CMRAC_params.refs_temp[0][0] = 0.0;
    CMRAC_params.refs_temp[1][0] = 0.0;
    CMRAC_params.refs_temp[2][0] = 1.0;
    CMRAC_params.refs_temp[3][0] = 0;
}
if ((CMRAC_params.t >= 62) && (CMRAC_params.t < 72))
{
    CMRAC_params.refs_temp[0][0] = -1*(CMRAC_params.t - 62)/4;
    CMRAC_params.refs_temp[1][0] = 0.0;
    CMRAC_params.refs_temp[2][0] = 1.0;
    CMRAC_params.refs_temp[3][0] = 0;
}
if ((CMRAC_params.t >= 72) && (CMRAC_params.t < 77))
{
    CMRAC_params.refs_temp[0][0] = -2.5;
                CMRAC_params.refs_temp[1][0] = 0;
    CMRAC_params.refs_temp[2][0] = 1.0;
    CMRAC_params.refs_temp[3][0] = 0;
}
if ((CMRAC_params.t >= 77) && (CMRAC_params.t < 81))
{
    CMRAC_params.refs_temp[0][0] = -2.5 + (CMRAC_params.t - 77)/4;
    CMRAC_params.refs_temp[1][0] = 0.0;
    CMRAC_params.refs_temp[2][0] = 1.0;
    CMRAC_params.refs_temp[3][0] = 0;
}
if ((CMRAC_params.t >= 81) && (CMRAC_params.t < 85))
{
    CMRAC_params.refs_temp[0][0] = -1.5;
    CMRAC_params.refs_temp[1][0] = 0;
    CMRAC_params.refs_temp[2][0] = 1.0;
    CMRAC_params.refs_temp[3][0] = 0;
}
if ((CMRAC_params.t > 85) && (CMRAC_params.t < 93))
{
    CMRAC_params.refs_temp[0][0] = -1.5;
    CMRAC_params.refs_temp[1][0] = 0;
    CMRAC_params.refs_temp[2][0] = 1.0 - (CMRAC_params.t-85)/4;;

    CMRAC_params.refs_temp[3][0] = 0;

}

if ((CMRAC_params.t > 106) && (CMRAC_params.t < 1000))

{

                CMRAC_params.refs_temp[0][0] = 0;

    CMRAC_params.refs_temp[1][0] = 0;

    CMRAC_params.refs_temp[2][0] = 0;

    CMRAC_params.refs_temp[3][0] = 0;

    CMRAC_params.Lambda_switch = 0;

}




CMRAC_params.x_des = CMRAC_params.refs_temp[0][0] + CMRAC_params.initial_x;

CMRAC_params.y_des = CMRAC_params.refs_temp[1][0] + CMRAC_params.initial_y;

CMRAC_params.r[0][0] = CMRAC_params.refs_temp[2][0] -0.2;   //z/

CMRAC_params.theta_des = 0.0;

if (CMRAC_params.t > 53){

                CMRAC_params.theta_des = 0.09;//0.08;

}

CMRAC_params.r[3][0] = CMRAC_params.refs_temp[3][0];   //yaw (rad)


CMRAC_params.Value_Check(CMRAC_params.x_in, CMRAC_params.xprev);

CMRAC_params.Value_Check(CMRAC_params.y_in, CMRAC_params.yprev);

CMRAC_params.Value_Check(CMRAC_params.z_in, CMRAC_params.zprev);

CMRAC_params.Value_Check(CMRAC_params.phi_in, CMRAC_params.phi_prev);

CMRAC_params.Value_Check(CMRAC_params.theta_in, CMRAC_params.theta_prev);
CMRAC_params.Value_Check(CMRAC_params.psi_in, CMRAC_params.psi_prev);


/********** Input the State Vector***********/

CMRAC_params.x[0][0] = CMRAC_params.x_in;

CMRAC_params.x[1][0] = CMRAC_params.xdot_in;

CMRAC_params.x[2][0] = CMRAC_params.y_in;

CMRAC_params.x[3][0] = CMRAC_params.ydot_in;

CMRAC_params.x[4][0] = CMRAC_params.z_in;

CMRAC_params.x[5][0] = CMRAC_params.zdot_in;

CMRAC_params.x[6][0] = CMRAC_params.phi_in;

CMRAC_params.Angles[0][0] = CMRAC_params.phi_in;

CMRAC_params.PQR[0][0] = CMRAC_params.P_in;

CMRAC_params.x[8][0] = CMRAC_params.theta_in;

CMRAC_params.Angles[1][0] = CMRAC_params.theta_in;

CMRAC_params.PQR[1][0] = CMRAC_params.Q_in;

CMRAC_params.x[10][0] = CMRAC_params.psi_in;

CMRAC_params.Angles[2][0] = CMRAC_params.psi_in;

CMRAC_params.PQR[2][0] = CMRAC_params.R_in;


Omega(0) = CMRAC_params.P_in;

Omega(1) = CMRAC_params.Q_in;

Omega(2) = CMRAC_params.R_in;

Omega_cross(0,1) = -Omega(2);

Omega_cross(1,0) = Omega(2);

Omega_cross(0,2) = Omega(1);

Omega_cross(2,0) = -Omega(1);

Omega_cross(1,2) = -Omega(0);
Omega_cross(2,1) = Omega(0);

CMRAC_params.Angular_velocities_from_PQR(CMRAC_params.PQR,
 CMRAC_params.Angles, CMRAC_params.Angular_rates);

CMRAC_params.x[7][0] = CMRAC_params.Angular_rates[0][0];
CMRAC_params.x[9][0] = CMRAC_params.Angular_rates[1][0];
CMRAC_params.x[11][0] = CMRAC_params.Angular_rates[2][0];

for (int i = 0; i < 12; i++)
{
 X_states(i,0) = CMRAC_params.x[i][0];
}

//Baseline outer loop
CMRAC_params.Fz = (9.81)*CMRAC_params.mass;
CMRAC_params.xddot = -1.9 * CMRAC_params.x[1][0]
 - 1.9 * (CMRAC_params.x[0][0] - CMRAC_params.x_des);
CMRAC_params.yddot = -1.9 * CMRAC_params.x[3][0]
 - 1.9 * (CMRAC_params.x[2][0] - CMRAC_params.y_des);
if (CMRAC_params.Fz == 0)
{
                CMRAC_params.Fz = 9.81*CMRAC_params.mass;
}
CMRAC_params.ref_angles[0][0] = (CMRAC_params.xddot*sin(CMRAC_params.x[10][0])
 - CMRAC_params.yddot*cos(CMRAC_params.x[10][0]))*CMRAC_params.mass / CMRAC_params.Fz;
CMRAC_params.ref_angles[1][0] = (CMRAC_params.xddot*cos(CMRAC_params.x[10][0])
 + CMRAC_params.yddot*sin(CMRAC_params.x[10][0]))*CMRAC_params.mass / CMRAC_params.Fz;


// Limit the angle reference values -- raise these values as tuning improves
for (int i = 0; i < 2; i++)
{
                if (CMRAC_params.ref_angles[i][0] > 0.12) {
                                CMRAC_params.ref_angles[i][0] = 0.12;
    }
                else if (CMRAC_params.ref_angles[i][0] < -0.12) {
                                CMRAC_params.ref_angles[i][0] = -0.12;
    }
}// end for loop




/****************** Adaptive Outer loop ****************************************/
// Generate pie vector
pie << X_states,
MatrixXf::Zero(6,1),
Phi_tot;

// Eqn 93 -- V2 is w
V2 = Delta_k*pie;

for (int i = 0; i < 3; i++)
```

```
{
                Vtrans(i,0) = Alpha(i,0);
}


// Build Rtheta and Rpsi matrices
R_theta(0,0) = cos(CMRAC_params.theta_des);
R_theta(2,0) = sin(CMRAC_params.theta_des);
R_theta(0,2) = -1*sin(CMRAC_params.theta_des);
R_theta(2,2) = cos(CMRAC_params.theta_des);


R_psi(0,0) = cos(CMRAC_params.r[3][0]);
R_psi(1,0) = -sin(CMRAC_params.r[3][0]);
R_psi(0,1) = sin(CMRAC_params.r[3][0]);
R_psi(1,1) = cos(CMRAC_params.r[3][0]);


//Eqn 69
Vtrans_bar = R_theta*R_psi*Vtrans;


CMRAC_params.phi_des = -atan2(Vtrans_bar(1,0),Vtrans_bar(2,0));


CMRAC_params.Value_Check(CMRAC_params.phi_des, CMRAC_params.phi_des_prev);


// -------------------------------------------------------------------------
/*  Set q and qdot, qref and qdotref    */
q(0,0) = CMRAC_params.x[0][0];
q(1,0) = CMRAC_params.x[2][0];
q(2,0) = CMRAC_params.x[4][0];
```

```
q(3,0) = CMRAC_params.x[6][0];

q(4,0) = CMRAC_params.x[8][0];

q(5,0) = CMRAC_params.x[10][0];


qdot(0,0) = CMRAC_params.x[1][0];

qdot(1,0) = CMRAC_params.x[3][0];

qdot(2,0) = CMRAC_params.x[5][0];

qdot(3,0) = CMRAC_params.x[7][0];

qdot(4,0) = CMRAC_params.x[9][0];

qdot(5,0) = CMRAC_params.x[11][0];


qdes(0,0) = CMRAC_params.x_des;

qdes(1,0) = CMRAC_params.y_des;

qdes(2,0) = CMRAC_params.r[0][0];

qdes(3,0) = CMRAC_params.phi_des;

qdes(4,0) = CMRAC_params.theta_des;

qdes(5,0) = CMRAC_params.r[3][0];






if (CMRAC_params.phi_desdot > 1.0)

{

                CMRAC_params.phi_desdot = 1.0;

}

else if (CMRAC_params.phi_desdot < -1.0)

{
```

```
                CMRAC_params.phi_desdot = -1.0;

}

if (CMRAC_params.xdesdot > 2.0)

{

                CMRAC_params.xdesdot = 2.0;

}

else if (CMRAC_params.xdesdot < -2.0)

{

                CMRAC_params.xdesdot = -2.0;

}

if (CMRAC_params.ydesdot > 2.0)

{

                CMRAC_params.ydesdot = 2.0;

}

else if (CMRAC_params.ydesdot < -2.0)

{

                CMRAC_params.ydesdot = -2.0;

}


qdesdot(0,0) = CMRAC_params.xdesdot;

qdesdot(1,0) = CMRAC_params.ydesdot;

qdesdot(2,0) = CMRAC_params.zdesdot;

qdesdot(3,0) = CMRAC_params.phi_desdot;

qdesdot(4,0) = CMRAC_params.theta_desdot;

qdesdot(5,0) = CMRAC_params.psi_desdot;


// Concatenate position and derivate, Concatenate desired with desired derivative
```

```
qtotal << q,
                                        qdot;
qdestotal << qdes,
             qdesdot;

// Define error:
e = qtotal - qdestotal;

// Introduce small deadband for x,y
for (int i = 0; i < 2; i++){
                if (abs(e(2*i,0)) < 0.005){
                                e(2*i,0) = 0.0;
    }
}

// ---------------------------- Find h, v, Kdot -------------------------------

// Difference between adaptive gain and initial estimate: pg. 4
Delta_k = K - Ke;

// Next two lines calculate Frobenius norm of Delta_K*Gamma_inverse*Delta_K
Frobenius_mat_gdk = Delta_k*Gamma_x_inv*Delta_k.transpose();
CMRAC_params.trace_dk_gx = Frobenius_mat_gdk.trace();

// (Sqrt of M )*e, M is identity so I dont actually do the square root for now
Root_me = M * e;
```

```
// Find Norm of (Sqrt(M)*e)

CMRAC_params.root_me_dot = Root_me.dot(Root_me);

CMRAC_params.norm_root_me = sqrt(CMRAC_params.root_me_dot);


// Eqn 21 for constraint function h

CMRAC_params.h = Control_Gains_ouCMRAC.hmax

 - CMRAC_params.norm_root_me

 - CMRAC_params.trace_dk_gx;


// Dummy value for h, used when testing conventional sigma modification

CMRAC_params.h_sigma = Control_Gains_ouCMRAC.hmax

 - CMRAC_params.norm_root_me

 - CMRAC_params.trace_dk_gx;


// e^T * P * e

CMRAC_params.epe = (e.transpose()*P*e)(0);


// Calculate Lyapunov function v(): Eqn 23

CMRAC_params.v = (CMRAC_params.epe + CMRAC_params.trace_dk_gx)/CMRAC_params.h;


// Find ||e^TPB||

epb_trans = (e.transpose()*P*B).transpose();

CMRAC_params.epb_dot = epb_trans.dot(epb_trans);

CMRAC_params.norm_epb = sqrt(CMRAC_params.epb_dot);


// Im - v*Hx in Eqn: 37

I_minus_hxv = (Im - CMRAC_params.v*Hx);
```

```
// Rotational Part of Regressor Vector: Eqn 92

Phi_rot(1,0) = -Omega(2)*Omega(0);

Phi_rot(2,0) = Omega(1)*Omega(0);

Phi_rot(3,0) = Omega(2)*Omega(0);


Phi_rot(5,0) = -Omega(0)*Omega(0);

Phi_rot(6,0) = -Omega(1)*Omega(0);

Phi_rot(7,0) = Omega(0)*Omega(0);


Phi_rot(10,0) = -Omega(2)*Omega(1);

Phi_rot(11,0) = Omega(1)*Omega(1);

Phi_rot(12,0) = Omega(2)*Omega(1);


Phi_rot(14,0) = -Omega(0)*Omega(1);

Phi_rot(15,0) = -Omega(1)*Omega(1);

Phi_rot(16,0) = Omega(0)*Omega(1);


Phi_rot(19,0) = -Omega(2)*Omega(2);

Phi_rot(20,0) = Omega(1)*Omega(2);

Phi_rot(22,0) = Omega(2)*Omega(2);


Phi_rot(23,0) = -Omega(2)*Omega(0);

Phi_rot(24,0) = -Omega(1)*Omega(2);

Phi_rot(25,0) = Omega(2)*Omega(0);


Phi_rot(27,0) = -CMRAC_params.mass*CMRAC_params.g*(sin(CMRAC_params.phi_in)
        *sin(CMRAC_params.psi_in) + cos(CMRAC_params.phi_in)

        *cos(CMRAC_params.psi_in)*sin(CMRAC_params.theta_in));

Phi_rot(28,0) = CMRAC_params.mass*CMRAC_params.g*(sin(CMRAC_params.phi_in)

        *cos(CMRAC_params.psi_in) - cos(CMRAC_params.phi_in)

        *sin(CMRAC_params.psi_in)*sin(CMRAC_params.theta_in));

Phi_rot(29,0) = -CMRAC_params.mass*CMRAC_params.g*

        cos(CMRAC_params.phi_in)*cos(CMRAC_params.theta_in);


// Augment Translational and Rotational part of Regressor Vector

Phi_tot << Phi_tran,

                Phi_rot;


// Regenerate PIE

pie << X_states,

        MatrixXf::Zero(6,1),

        Phi_tot;


/* Adaptive Law  -- Eqn 37 */

Kdot_transpose = -Gamma_x * (pie * e.transpose()*(P-(CMRAC_params.v*CMRAC_params.he*M))

        *B+ Control_Gains_ouCMRAC.sigma*CMRAC_params.norm_epb*Delta_k.transpose())

        *I_minus_hxv.inverse();


Kdot = Kdot_transpose.transpose();




// Numerically calculate the derivatives

CMRAC_params.dt_derivative = CMRAC_params.t - CMRAC_params.t_prev_derivatives;
```

```
if (CMRAC_params.dt_derivative > 0.003)
{
                CMRAC_params.phi_desdot = (CMRAC_params.phi_des -
                        CMRAC_params.phi_des_prev)/CMRAC_params.dt_derivative;
                CMRAC_params.xdesdot = (CMRAC_params.x_des -
                        CMRAC_params.x_des_prev)/CMRAC_params.dt_derivative;
                CMRAC_params.ydesdot = (CMRAC_params.y_des -
                        CMRAC_params.y_des_prev)/CMRAC_params.dt_derivative;
                CMRAC_params.zdesdot = (CMRAC_params.r[0][0] -
                        CMRAC_params.z_des_prev)/CMRAC_params.dt_derivative;

                CMRAC_params.phi_des_prev = CMRAC_params.phi_des;
                CMRAC_params.x_des_prev = CMRAC_params.x_des;
                CMRAC_params.y_des_prev = CMRAC_params.y_des;
                CMRAC_params.z_des_prev = CMRAC_params.r[0][0];

                CMRAC_params.t_prev_derivatives = CMRAC_params.t;
}
//-------------------- Calculate Control Law ------------------------------
// Angular velocities of propellers,
Omega_prop(0) = sqrt(prop_angular_vel.w1);
Omega_prop(1) = sqrt(Mixer_functions.get_w2_sq);
Omega_prop(2) = sqrt(Mixer_functions.get_w3_sq);
Omega_prop(3) = sqrt(Mixer_functions.get_w4_sq);

for (int i = 0; i<4; i++){
  Gyro_term(2,0) = Gyro_term(2,0) + Inertia_prop(2,2)*Omega_prop(i);
}

// Eqn 79, rc is 0 for now, so not all terms are seen

f_tran(0,0) = 0.0;

f_tran(1,0) = 0.0;

f_tran(2,0) = -CMRAC_params.mass*CMRAC_params.g;




// Eqn 81

f_rot = -Omega_cross*(Inertia)*Omega - Gyro_term;


f_total << f_tran,

                                         f_rot;


// From eqn 40

Gamma_matrix(0,1) = sin(CMRAC_params.x[6][0])*tan(CMRAC_params.x[8][0]);

Gamma_matrix(1,1) = cos(CMRAC_params.x[6][0]);

Gamma_matrix(2,1) = sin(CMRAC_params.x[6][0])/cos(CMRAC_params.x[8][0]);

Gamma_matrix(0,2) = cos(CMRAC_params.x[6][0])*tan(CMRAC_params.x[8][0]);

Gamma_matrix(1,2) = -sin(CMRAC_params.x[6][0]);

Gamma_matrix(2,2) = cos(CMRAC_params.x[6][0])/cos(CMRAC_params.x[8][0]);




Inv_Gamma = Gamma_matrix.inverse();


// [I3, 03,03, I3]

Mat1 << MatrixXf::Identity(3,3), MatrixXf::Zero(3,3),
                                MatrixXf::Zero(3,3), MatrixXf::Identity(3,3);


// Computationally better to set at zero than calculate lots of sin and cos values

Beta = MatrixXf::Zero(6,1);


// Control Law, Eqn 85

Alpha = Mbar*Mat1*(qdesddot - Beta - Kp*(q - qdes) - Kd*(qdot-qdesdot) + V2)

                                        - f_total;


// Checks for bad control data

if (abs(Alpha(0)) < 0.01)

{

                Alpha(0) = CMRAC_params.a0_prev;

}

else if (abs(Alpha(0)) > 0.01)

{

                CMRAC_params.a0_prev = Alpha(0);

}

if (abs(Alpha(3)) < 0.0000001)

{

                Alpha(3) = CMRAC_params.a3_prev;

}

else if (abs(Alpha(3)) > 0.0000001)

{

                CMRAC_params.a3_prev = Alpha(3);

}

if (abs(Alpha(4)) < 0.000001)
```

```
{
                Alpha(4) = CMRAC_params.a4_prev;
}
else if (abs(Alpha(4)) > 0.000001)
{
                CMRAC_params.a4_prev = Alpha(4);
}
if (abs(Alpha(5)) < 0.000001)
{
                Alpha(5) = CMRAC_params.a5_prev;
}
else if (abs(Alpha(5)) > 0.000001)
{
                CMRAC_params.a5_prev = Alpha(5);
}


// Input to Controls, U
// Lambda_switch allows me to turn motors off
CMRAC_params.U1 = CMRAC_params.Lambda_switch*Alpha(2);


CMRAC_params.U2 = CMRAC_params.Lambda_switch*Alpha(3);


CMRAC_params.U3 = CMRAC_params.Lambda_switch*Alpha(4);


CMRAC_params.U4 = CMRAC_params.Lambda_switch*Alpha(5);
```

```
// Check values before storing into previous value

CMRAC_params.Prev_value_check(CMRAC_params.x_in, CMRAC_params.xprev);

CMRAC_params.Prev_value_check(CMRAC_params.y_in, CMRAC_params.yprev);

CMRAC_params.Prev_value_check(CMRAC_params.z_in, CMRAC_params.zprev);

CMRAC_params.Prev_value_check(CMRAC_params.phi_in, CMRAC_params.phi_prev);

CMRAC_params.Prev_value_check(CMRAC_params.theta_in, CMRAC_params.theta_prev);

CMRAC_params.Prev_value_check(CMRAC_params.psi_in, CMRAC_params.psi_prev);

CMRAC_params.Prev_value_check(CMRAC_params.phi_des, CMRAC_params.phi_des_prev);


//cout << CMRAC_params.U3 << endl;

 //remap to correct channels for Pixhawk

CMRAC_params.remap_U(CMRAC_params.U1,CMRAC_params.U2,CMRAC_params.U3,CMRAC_params.U4);

CMRAC_params.dt = CMRAC_params.t - CMRAC_params.t_prev;





//cout << CMRAC_params.dt << endl;

CMRAC_params.t_prev = CMRAC_params.t;

CMRAC_params.t=CMRAC_params.t*1000; //rescale time to ms

} // end of Dynamics
```

Bibliography

[1] H. Bouadi, S. S. Cunha, A. Drouin, and F. Mora-Camino, "Adaptive sliding mode control for quadrotor attitude stabilization and altitude tracking," in *2011 IEEE 12th International Symposium on Computational Intelligence and Informatics (CINTI).* IEEE, 2011, pp. 449–455.

[2] Z. T. Dydek, A. M. Annaswamy, and E. Lavretsky, "Adaptive control of quadrotor uavs: A design trade study with flight evaluations," *IEEE Transactions on control systems technology*, vol. 21, no. 4, pp. 1400–1406, 2013.

[3] A. G. Loukianov, "Robust block decomposition sliding mode control design," *Mathematical Problems in Engineering*, vol. 8, no. 4-5, pp. 349–365, 2002.

[4] E.-H. Zheng, J.-J. Xiong, and J.-L. Luo, "Second order sliding mode control for a quadrotor uav," *ISA transactions*, vol. 53, no. 4, pp. 1350–1356, 2014.

[5] H. K. Khalil, *Nonlinear Systems*, ser. Pearson Education. Princeton, NJ: Prentice Hall, 2002.

[6] A. L'Afflitto, R. B. Anderson, and K. Mohammadi, "An introduction to nonlinear robust control for unmanned quadrotor aircraft: How to design control algorithms for quadrotors using sliding mode control and adaptive control techniques," *IEEE Control Systems Magazine*, vol. 38, no. 3, pp. 102–121, 2018.

[7] E. Lavretsky and K. Wise, *Robust and Adaptive Control: With Aerospace Applications.* London, UK: Springer, 2012.

[8] R. B. Anderson, J. Marshall, J. Burke, and A. L'Afflitto, "Robust adaptive control of a tilt-rotor quadcopter with unknown inertial properties," in *American Control Conference – To Appear*, 2019.

[9] R. B. Anderson, J. Marshall, J.-P. Burke, and A. L'Afflitto, "Robust adaptive control of a tilt-rotor quadcopter with unknown inertial properties," *IEEE Transactions on Control Systems Technology – Submitted*, 2019.

[10] K. Narendra and A. Annaswamy, "A new adaptive law for robust adaptation without persistent excitation," *IEEE Transactions on Automatic Control*, vol. 32, no. 2, pp. 134–145, 1987.

[11] E. Arabi, B. C. Gruenwald, T. Yucelen, and N. T. Nguyen, "A set-theoretic model reference adaptive control architecture for disturbance rejection and uncertainty suppression with strict performance guarantees," *International Journal of Control*, vol. 91, no. 5, pp. 1195–1208, 2018.

[12] Y.-J. Liu, S. Lu, D. Li, and S. Tong, "Adaptive controller design-based ablf for a class of nonlinear time-varying state constraint systems," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 47, no. 7, pp. 1546–1553, 2017.

[13] B. Peterson and K. Narendra, "Bounded error adaptive control," *IEEE Transactions on Automatic Control*, vol. 27, no. 6, pp. 1161–1168, December 1982.

[14] P. Ioannou and P. Kokotovic, *Adaptive Systems with Reduced Models.* New York, NY: Springer, 1983.

[15] G. Kreisselmeier and K. Narendra, “Stable model reference adaptive control in the presence of bounded disturbances,” *IEEE Transactions on Automatic Control*, vol. 27, no. 6, pp. 1169–1175, 1982.

[16] J. B. Pomet and L. Praly, “Adaptive nonlinear regulation: estimation from the Lyapunov equation,” *IEEE Transactions on Automatic Control*, vol. 37, no. 6, pp. 729–740, 1992.

[17] A. L’Afflitto, “Barrier Lyapunov functions and constrained model reference adaptive control,” *IEEE Control Systems Letters*, vol. 2, no. 3, pp. 441–446, 2018.

[18] I. Fantoni and R. Lozano, *Non-linear Control for Underactuated Mechanical Systems.* Berlin, Germany: Springer, 2002.

[19] R. Falconi and C. Melchiorri, “Dynamic model and control of an over-actuated quadrotor UAV,” in *IFAC Symposium on Robot Control*, vol. 45, no. 22, 2012, pp. 192–197.

[20] D. Invernizzi and M. Lovera, “Geometric tracking control of a quadcopter tiltrotor UAV,” in *IFAC World Congress*, vol. 50, no. 1, 2017, pp. 11 565–11 570.

[21] A. Nemati and M. Kumar, “Modeling and control of a single axis tilting quadcopter,” in *American Control Conference*, 2014, pp. 3077–3082.

[22] D. Yoo, H. Oh, D. Won, and M. Tahk, “Dynamic modeling and control system

design for tri-rotor UAV," in *International Symposium on Systems and Control in Aeronautics and Astronautics*, 2010, pp. 762–767.

[23] 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, pp. 173–191, 2014.

[24] A. Sanchez, J. Escareño, O. Garcia, and R. Lozano, "Autonomous hovering of a noncyclic tiltrotor UAV: Modeling, control and implementation," in *IFAC World Congress*, vol. 41, no. 2, 2008, pp. 803 – 808.

[25] G. Scholz, M. Popp, J. Ruppelt, and G. F. Trommer, "Model independent control of a quadrotor with tiltable rotors," in *IEEE/ION Position, Location and Navigation Symposium*, 2016, pp. 747–756.

[26] C. Yih, "Flight control of a tilt-rotor quadcopter via sliding mode," in *International Automatic Control Conference*, 2016, pp. 65–70.

[27] Y. Yildiz, M. Unel, and A. E. Demirel, "Nonlinear hierarchical control of a quad tilt-wing UAV: An adaptive control approach," *International Journal of Adaptive Control and Signal Processing*, vol. 31, no. 9, pp. 1245–1264.

[28] E. D'Amato, G. D. Francesco, I. Notaro, G. Tartaglione, and M. Mattei, "Nonlinear dynamic inversion and neural networks for a tilt tri-rotor UAV," in *IFAC Workshop on Advanced Control and Navigation for Autonomous Aerospace Vehicles*, vol. 48, no. 9, 2015, pp. 162 – 167.

[29] M.-D. Hua, T. Hamel, and C. Samson, "Control of VTOL vehicles with thrust-tilting augmentation," in *IFAC World Congress*, vol. 47, no. 3, 2014, pp. 2237 – 2244.

[30] D. S. Bernstein, *Matrix Mathematics,* 2nd ed. Princeton, NJ: Princeton University Press, 2009.

[31] J. R. Magnus and H. Neudecker, *Matrix Differential Calculus with Applications in Statistics and Econometrics.* New York, NY: Wiley, 1999.

[32] J. R. Magnus, "On the concept of matrix derivative," *Journal of Multivariate Analysis*, vol. 101, no. 9, pp. 2200 – 2206, 2010.

[33] K. Mohammadi and A. L'Afflitto, "Robust adaptive output tracking for quadrotor helicopters," in *Adaptive Robust Control Systems.* IntechOpen, 2017.

[34] W. M. Haddad and V. Chellaboina, *Nonlinear Dynamical Systems and Control: A Lyapunov-Based Approach.* Princeton, NJ: Princeton Univ. Press, 2008.

[35] C. R. Johnson, "Positive definite matrices," *The American Mathematical Monthly*, vol. 77, no. 3, pp. 259–264, 1970.

[36] E. Kreyszig, *Introductory Functional Analysis with Applications*, ser. Wiley. New York, NY: Wiley, 1989.

[37] A. L'Afflitto, *A Mathematical Perspective on Flight Dynamics and Control.* London, UK: Springer, 2017, doi:10.1007/978-3-319-47467-0.

[38] A. L'Afflitto and K. Mohammadi, "Equations of motion of rotary-wing UAS with time-varying inertial properties," *AIAA Journal of Guidance, Control, and Dynamics*, vol. 41, no. 2, pp. 559–564, 2018.

[39] H. Baruh, *Analytical Dynamics.* New York, NY: McGraw-Hill, 1999.

[40] E. DiBenedetto, *Classical mechanics: theory and mathematical modeling.* Berlin, Germany: Springer, 2010.

[41] D. Greenwood, *Advanced Dynamics.* New York, NY: Cambridge University Press, 2006.

[42] S. Bouabdallah, P. Murrieri, and R. Siegwart, "Design and control of an indoor micro quadrotor," in *International Conference on Robotics and Automation*, vol. 5, 2004, pp. 4393–4398.

[43] A. Jain, *Robot and multibody dynamics: analysis and algorithms.* Berlin, Germany: Springer, 2010.

[44] F. E. Udwadia and R. E. Kalaba, "What is the general form of the explicit equations of motion for constrained mechanical systems?" *Journal of Applied Mechanics*, vol. 69, no. 3, pp. 335–339, 2002.

[45] L. R. García Carrillo, A. E. Dzul López, R. Lozano, and C. Pégard, *Vision-Based Control of a Quad-Rotor UAV.* London, UK: Springer, 2013, pp. 103–137.

[46] S. Bouabdallah, "Design and control of quadrotors with applications to au-

tonomous flying," Ph.D. dissertation, École Polytechnique Fédérale de Lausanne, Lausanne, Switzerland, 2007.

[47] S. P. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge, UK: Cambridge University Press, 2004.

[48] I. H. B. Pizetta, A. S. Brandão, and M. Sarcinelli-Filho, "Modelling and control of a quadrotor carrying a suspended load," in *Workshop on Research, Education and Development of Unmanned Aerial Systems*, 2015, pp. 249–257.

[49] "Pixhawk Autopilot," http://pixhawk.org/, 2019, [Online; accessed 12-April-2019].

[50] "Castle DMR 30/40 Dedicated Multi-Rotor ESC," http://www.castlecreations.com/en/dmr-3040-4pack-010-0156-00, 2019, [Online; accessed 12-April-2019].

[51] "T-Motor MN2212," http://store-en.tmotor.com/goods.php?id=389, 2019, [Online; accessed 12-April-2019].

[52] "T-Motor 9x3 Carbon Fiber Propellers," https://www.getfpv.com/antigravity-9x3-carbon-fiber-prop-pair-black.html, 2019, [Online; accessed 12-April-2019].

[53] "Dynamixel AX-18A," https://www.trossenrobotics.com/dynamixel-ax-18A-robot-actuator.aspx, 2019, [Online; accessed 12-April-2019].