Formal Methods for the Design and Verification of UAV Feedback Controls

Omar A. Jasim

Department of Automatic Control and Systems Engineering

The University of Sheffield

A thesis submitted to the University of Sheffield
in partial fulfilment of the requirements
for the degree of

Doctor of Philosophy

January 2021

I would like to dedicate this thesis to my loving parents, wife and kids for their unconditional love and support.

In loving memory of my father Ahmad Jasim

Acknowledgements

First and foremost I am grateful to the Great God who gave me the strength, patience, wisdom and guides me during my PhD journey. I would like to thank my supervisor Prof Sandor Veres for introducing me to the topic of design and verification of autonomous control systems, which now forms the subject of this thesis containing my investigations and solutions to the problem of design and verification of autopilot controllers in unmanned aerial vehicles. I would like to thank Dr Owen McAree and postgraduate colleagues and all the staff at the Department of Automatic Control and Systems Engineering and Sheffield Robotics for valuable discussions, suggestions and for any advice given to me. Many thanks to my department for all the support which has been provided to me to present my work in outstanding conferences and journals in the aerospace and control field. Not the least I am grateful to my wonderful parents, my lovely wife and kids for all their support, patience and understanding during my PhD study.

Abstract

The development of control systems especially for autonomous systems leads to more complex designs and analysis. Hence, the analysis of these systems will prone to errors. Such errors may lead to disastrous consequences which may cost human life. For some applications such as unmanned aerial vehicles (UAV) which are safety-critical systems, the control system required to be robust, stable and safe to perform the given tasks. Therefore, control systems need to be tested and verified to assure their correctness and robustness to guarantee acceptable performance. This thesis is concerned with using formal methods to develop new verification schemes for UAVs control systems. Until now, control theories have been proved and verified manually by control scientists and engineers. Some computations of multivariable control systems, which include numerical bounds on modelling errors and state constraints, are difficult to check and verify manually by an engineer due to their complexity. In these cases errors made by manual derivation can be dangerous for safe especially for safety-critical system such as unmanned aerial vehicles. To mitigate these issues, this thesis presents examples of formal proofs of theoretical control theorems. These represent the first steps towards verifying control theories by software using formal methods by interactive theorem proving. The thesis presents a new nonlinear controller for unmanned aerial vehicles by jointly addressing modelling uncertainty and external disturbances. Moreover, a new verification framework is presented for verifying the control systems of UAVs. The verification framework is applied to the proposed control scheme using interactive and automated theorem proving techniques. This is promising and may encourage the use of such methods in control system verification of safety-critical systems in general. The symbolic methods are generic and potentially generalise to verification of a variety of industrial control systems, where performance loss is damaging and therefore analysis is important to be carried out formally.

Contents

Nomenclature

ATP Automated Theorem Proving

CAD Computer-Aided Design

CG Centre of Gravity

CPS Cyber-Physical Systems

DCM Direction Cosine Matrix

DIC Dynamic Inversion Control

FMI Functional Mockup Interface

FOL First-Order Logic

HOL Higher-Order Logic

ITP Interactive Theorem Proving

NRV Nichols Plot Requirements Verifier

PVS Prototype Verification System

$RNDI$ Robust Nonlinear Dynamic Inversion

RTL Register-Transfer Level

SAT Boolean Satisfiability Problem

SMT Satisfiability Modulo Theories

UAV Unmanned Aerial Vehicles

ZF Zermelo-Fraenkel Set Theory

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Control theory is an interdisciplinary domain dedicated to analysing the behaviour of dynamical systems with inputs and outputs. The correctness of controller design, implementation and system stability criteria are very important aspects in control theory especially in applications that human safety is required as in medical, aeronautical and aerospace systems which require advanced safety-critical control systems. Many modern control systems, such as adaptive control, rely on stability criteria as the major justifications for their relevance to the applications of safety-critical control systems [22].

The purpose of control design is to produce feedback, feedforward and adaptive controllers to provide robust performance in practical applications. In the aerospace field, control law for aircraft normally combines control engineering knowledge with tests of stability and smoothness of control responses under disturbances. This often takes the form of an iterative process of remodelling aerodynamics in wind tunnels and ultimately in flight tests. Given a particular open-loop dynamical model, control engineering relies on the mathematical theory that is implemented in computations of flight controllers onboard. When the flight envelop is defined, it introduces numerical values which need to be carried through derivations and proofs of stability and acceptable handling within the flight envelope. This process is normally conducted by engineers or control scientists using

manual derivations. Manual derivation, especially for complex systems, may lead to incorrect control design and analysis even if the design has been manually verified by several expert engineers. However, in some aerospace areas such as unmanned aerial vehicles (UAVs), control performance needs to be guaranteed due to safety, economic or productivity requirements. These controllers are required to be officially certified that they conform to standards. The analysis of controllers for certification has traditionally relied on symbolic computation. Such symbolic computation is not only algebraic but also uses the concepts of signal spaces and nonlinear operators.

In order to officially certifying a designed controller, manual derivations are not an efficient way to ensure the correctness and safety of control systems and additional verification step need to be added. This is due to the human error associated with the manual derivation and verification where such error is almost possible which may lead to catastrophic consequences. To overcome this problem, in this thesis, manual derivations are verified by theorem proving methods, which are computer software that use mathematical symbols with the aid of logical techniques. The use of these methods will ensure the correctness and safety of the designed control systems where the errors produce from manual derivation can be detected at the early stage during the design. If these methods are going to be applied in the context of control theory and control systems verification, then they need to handle nonlinear causal operators and prove properties of their interconnections into a feedback system as well. The procedures presented in this thesis go beyond algebraic computation and use higher-order logic (HOL) [90], including handling of functionals, operators, concepts of convergence, stability and levels of smoothness measures. HOL is needed because it includes quantifications and type theory such as real and complex numbers that make the implementation of control properties applicable. An example of such properties is the using of high-order functions to define nonlinear operators. With the advance of automated reasoning, such formal analysis can now enter the possibilities of control system design beside traditional methods of manual derivations.

1.2 Control System Design for UAVs

There is an increasing requirement for the small multi-rotor unmanned drones, under 20kg and flying under 400ft, to be safely operated over congested, urban areas for police and security work, building inspections, fire fighting and emergency needs, etc. Drones would often carry variable payloads (cameras, measurement devices, robotic arms for picking-up objects, etc.) while they could be exposed to gusts of winds or could collide with or be attacked by other craft or birds. Other causes of instability include a temporary deterioration of actuator or processor functionality. Under such conditions, a drone's dynamical state may be easily pushed into unstable regions if controlled by off-the-shelf axis-by-axis PD/PID controllers, such as in [23, 27]. It is therefore imperative that when these drones operate semi-autonomously by an autopilot, they would need software that monitors their operational conditions and takes action if the limits of the controller performance are approached. Ultimately, semi-autonomous drones would need to decide for themselves, or they should advise the remote pilot, when to seek safety and to possibly modify or cancel flight/mission objectives. There have been many attempts to design competitive controllers of UAV but they had limited abilities to cope well with the existence of uncertainty and disturbances. Therefore, there is still a demand for such controllers. In this thesis, a novel robust nonlinear controller of a quadcopter UAV is designed and presented. The controller is then verified based on a proposed verification framework presented in this thesis using theorem proving techniques to ensure design correctness and stability validation.

1.3 Formal Proofs of Control Theories by software

Control theory can establish requirements of systems which need to be valid with all signals within the system and hence cannot be proven by simulation. One of the most basic such requirement is the stability of a control subsystem or the overall system. Other examples are statements on robust control performance in the face of dynamical uncertainties and disturbances in sensing and actuation.

Until now these theories were developed and their correctness was checked by control scientist manually using their mathematical knowledge. With the emergence of formal methods, there is now the possibility to derive and prove robust control theory by symbolic computation. There is a demand for this approach from industry for the verification of practical control systems with concrete numerical values where the applicability of a control theorem is specialised to an application with given numerical boundaries of parameter variations.

In practice, both the plant, the system to be controlled such as a UAV, and the feedback controller suffers from the variability of dynamics and disturbances. For instance, the Small-gain theorem [70] can be used to assess feedback stability for plants with variable dynamics, for which norm bounds can be measured in experiments. If for all plant and controller variations the product of the norm of the plant and the controller dynamical operators is less than 1, then the feedback loop is robustly stable. As no theory is yet widely known for automating the proofs of control systems, this thesis intends to provide an initial step to this challenge and gives an illustration on a formal proof of the Small-gain theorem using interactive theorem proving (ITP). As the mathematics of operators over signal spaces goes beyond algebra and first-order logic (FOL), the formal proofs considered rely on higher-order logic (HOL).

1.4 Control Systems Verification

Given the performance specifications for an aircraft to be built, where specifications are the mathematical representation of the desired requirements, a control system is designed [44]. Then, code and electronics are developed and chosen. The verification approach taken in this thesis fits into one of the three stages of a formally verifiable controller designs as outlined in Fig. 1.1, where robust control theory verification is followed by verification of the software used for implementation. Stage 1 is a precise mathematical definition of the plant, sensor and actuator dynamical variations and performance requirements, against which the implemented control system is to be verified. Stage 2 consists of computer-aided design (CAD) of a controller, which should be mathematically proven to meet the specifications requirements, the primary topic of this thesis. Stage 3

is the implementation of the mathematical model of the controller in computer code, while not introducing bugs or numerical errors serious enough to make the specifications violated. Finally, the code should be free from errors due to code implementations, which is ensured by code verification. Realtime code-verification is systematically checking the correctness of the encoded controller such as in [50, 67] and [125]. The scope of this thesis is formal verification algorithms for UAVs to check the correctness of control design (CAD) in Stage 2 before implementation in real-time code.

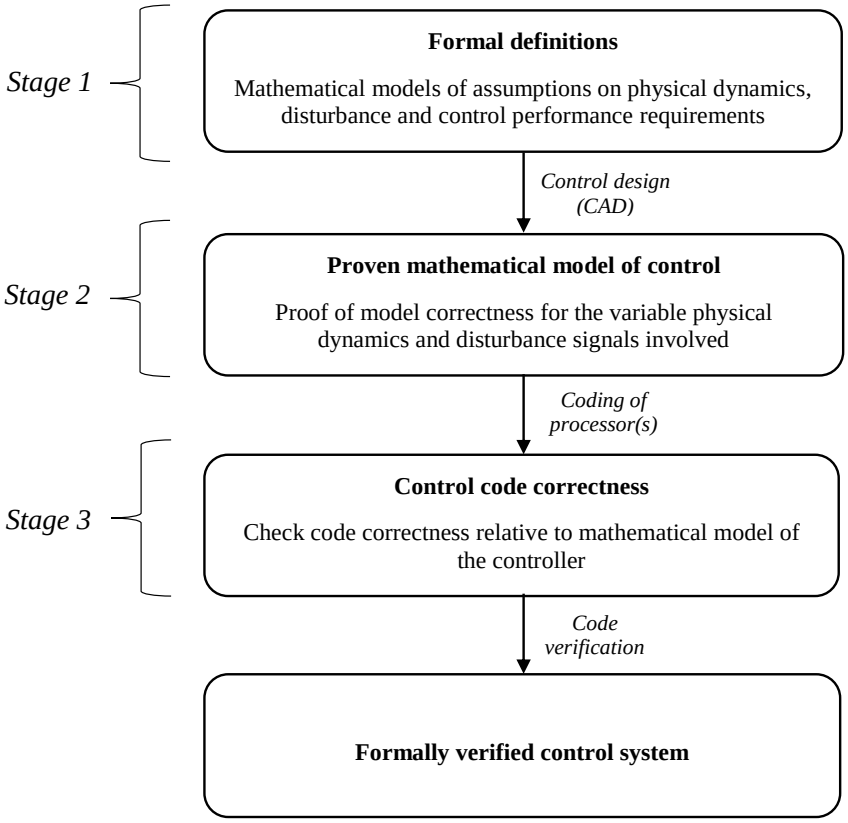


Figure 1.1: The three principle stages which lead to practical control system verification.

Often simulations are used to see whether the design is acceptable for the performance required. Simulations, however, may not uncover all signal combinations, which cause a failure in the control process. By their definitions, robust

CAD methods that rely on control theory will achieve performance requirements. Then the remaining problem is to prove that encoding does not affect the control performance due to computational errors in Stage 3. In implementations of aviation software, verification is often followed by redundancy-based safety analysis for critical sensors and actuators using voting principles, the effect of which lies outside the scope of this thesis.

The new formal verification methods proposed in this thesis need to precede software verification of controller code as they verify the correctness of control algorithms, which are implemented in software. This thesis gives an overview of past use of formal methods to verify the correctness of control system implementations to place this work in context. Efforts made to formally verify that the code used in practice correctly implements the control algorithms intended have been also reviewed. Methods of proving mathematical theorems by computers have been also reviewed. None of these past works address the verification of the control theory and algorithms by formal methods in the form of symbolic computation to prove control theory on which the control algorithms are based. For reliability and safety of practical control systems, both algorithmic verification (to be introduced in this thesis for the first time) and verification of controller implementation are needed (the latter pursued by many researchers in the past). This will provide higher standards of certification in the future.

1.5 Verification Framework of UAVs

A new functional verification framework of control system of UAVs using formal methods is developed and presented in Chapter 7 of this thesis. The aim of this framework is to demonstrate how formal methods can uncover inaccuracies in the mathematical arguments of pen and paper-based proofs and can provide the verification of the theory of robust control. The framework consists of using theorem proving methods represented by an ITP to prove the mathematically designed control system of the aircraft satisfies robustness requirements to ensure safe performance under varying environmental conditions. It also includes the use of automated theorem proving (ATP) for onboard real-time monitoring of control system stability of the aircraft during the flight to detect when its controller

reaches its flight envelop limits due to severe weather conditions. Such a detection procedure can be used to advise the remote pilot or an onboard intelligent agent to decide on alterations of the planned flight path. The proposed verification framework is applied to the control scheme of a generic quadcopter which is presented in details in Chapter 6.

1.6 Aims and Objectives of the Thesis

The main aim of the research conducted is to employ formal verification methods for developing a new verification framework to verify control systems especially for safety-critical applications such as UAVs. The first objective is to illustrate the possibility of proving control theories using formal methods. This is achieved by giving an example of formalising and proving a well-known control theorem using interactive theorem proving software. This will allow us to verify the control system at the design stage. The next objective is to design a robust nonlinear controller of UAVs, which take into account modelling uncertainty and external disturbances. This is followed by verifying the stability of this controller and then proving the overall controller design using formal methods. Another objective is to develop a verification framework which outlines the process of control system verification of UAVs. The verification process starts with checking the correctness of mathematical derivations of the designed control system and testing its stability at the design stage. Following this verification can go beyond the design stage to onboard monitoring of stability conditions for the UAV system during a flight. This aims to determine when the aircraft violated its flight envelop due to winds or malfunction in its electronics/mechanics that the autopilot can detect and perform an action such as emergency landing.

1.7 Contributions of the Thesis

Contributions of the thesis are summarised as follow:

Formal proof of the Small-gain theorem by formal methods: As a first attempt to prove a control theory by a computer, one of the most funda-

mental and general results of nonlinear feedback systems, the "Small-gain theorem" has been chosen. The theorem is formalised and proven using an interactive theorem proving tool. This is a fundamental theoretical result for many practical applications and plays an important role in robust control theory. Through this first example may be of limited practical applicability directly, one of the aims in this thesis is to describe the existing difficulties in the technical execution of formal proofs needed for control theory in the future using formal methods techniques.

Nonlinear Attitude Control Design and Verification for a Quadcopter: A nonlinear attitude control law is designed and simulated for a quadcopter UAV using the a dynamic inversion control technique. Controller stability is verified using ATP: MetiTarski.

Nonlinear Attitude Control Design and Verification for a Helicopter: A nonlinear attitude control law is designed and simulated for a small scale helicopter UAV using dynamic inversion control technique. An invariant set is defined with taking into account the system constraints. The controller stability is verified using MetiTarski. In addition, the system's variables are tested against the defined invariant set for further robustness and stability using the MetiTarski prover.

Multi-rotor UAV controller design and simulation: A novel robust nonlinear controller of a generic multi-rotor UAV is designed based on a dynamic inversion control technique which considers modelling uncertainty and external disturbances. The control scheme consists of attitude and position control.

A verification framework for UAVs: A new verification framework is developed for formally verifying UAVs control systems. The framework uses interactive theorem proving to verify the mathematical derivation of the controller at the design stage and using ATP for onboard monitoring of control system stability during the flight.

Multi-rotor UAV controller verification using the proposed framework: The designed multi-rotor UAV controller is verified using the proposed

framework. An interactive theorem prover is used to verify the design of a nonlinear quadcopter controller under nominal environmental conditions and an automated theorem prover is used for onboard stability and performance monitoring for excessive conditions, including some sensor or actuator failures.

1.8 Publications During Work Undertaken

During the undertaking of the work presented within this thesis, key components have been presented at international conferences and in internationally leading journals.

Refereed journal publications:

- Jasim, O. A. and Veres, S. M. (2020). A Robust Controller for Multi Rotor UAVs, In *Journal of Aerospace Science and Technology*, 22 June 2020, Elsevier.
- Jasim, O. A. and Veres, S. M. (2020). Verification Framework for Control System Functionality of Unmanned Aerial Vehicles, In *Journal of Automated Reasoning*, Springer, (under view).

Refereed conference publications:

- Jasim, O. A. and Veres, S. M. (2017). Towards Formal Proofs of Feedback Control Theory, In *Proceeding of 21st International Conference on System Theory, Control and Computing (ICSTCC)*, Sinaia, Romania, October 19-21 ,2017. IEEE.
- Jasim, O. A. and Veres, S. M. (2018). Formal Verification of Quadcopter Flight Envelop Using Theorem Prover. In *2018 IEEE Conference on Control Technology and Applications (CCTA)* (pp. 1502-1507). IEEE.
- Jasim, O. A. and Veres, S. M. (2019). Nonlinear Attitude Control Design and Verification for a Safe Flight of a Small-Scale Unmanned Helicopter, In *Proceeding of 6th International Conference on Control, Decision and Information Technologies (CoDIT)*. IEEE.

The first conference paper consists of the content of Chapter 3. The second conference paper consists of the content of Chapter 4. The content of Chapter 5 comes from the third conference paper. The contents of Chapter 6 is the first journal paper which is submitted. The contents of Chapter 7 is the second journal paper which is also submitted.

1.9 Thesis Structure

The thesis is organized into the following chapters:

Chapter 1 : introduces the motivation of the research conducted and the proposed solutions to tackle the challenges.

Chapter 2 : presents a brief overview and general background of the topics related to the Small-gain theorem, UAVs dynamics and control and formal methods. The literature review of related works is presented.

Chapter 3 : presents the formal proof of the Small-gain theorem using formal methods represented by an interactive theorem proving.

Chapter 4 : presents the design and simulation of a nonlinear attitude controller of a quadcopter UAV. The controller stability is verified using formal methods represented by an ATP.

Chapter 5 : presents the design and simulation of a nonlinear attitude controller of a small-scale helicopter UAV. The controller is verified using formal methods via ATP. The verification includes checking if the produced torques are within the required limits, the system is stable and all system states are varying and staying within the defined invariant control-enabled-set.

Chapter 6 : presents the design and simulation of a robust nonlinear controller of multi-rotor UAVs. The control scheme consists of position and attitude control.

Chapter 7 : presents a new verification framework of UAVs. The proposed framework is illustrated by verifying the control design of a quadcopter presented in Chapter 6.

Chapter 8 : presents and outlines the conclusions and future works.

Chapter 2

Background and Literature Review

2.1 UAV Dynamics

This section presents the dynamics of UAVs including different referencing frames and flight equations of motion, which are essential to understand before reading the rest of the thesis.

2.1.1 Reference Frames

It is relevant to briefly overview and understand the reference frames of an aircraft before studying its dynamics and control. The reference frames represented by the right-hand system are [32]:

- *World/Earth frame (W-frame)*: The origin point of this frame is located at any point on the surface of the Earth. The x-axis (X_W) points towards the North, y-axis (Y_W) points towards the West, and z-axis (Z_W) points to the opposite direction of the centre of the Earth.
- *Rigid body frame (B-frame)*: The origin point of this frame is usually fixed at the aircraft's centre of gravity (CG). The x-axis (X_B) is located on the aircraft's symmetric plane and points towards the nose of the aircraft. The y-axis (Y_B) points towards the left-side of the aircraft. The z-axis

(Z_B) lies on the aircraft's symmetric plane and points upwards since it is perpendicular on the X_B and Y_B axes.

2.1.2 Flight Equations of Motion

2.1.2.1 Euler-Newton Representation

An aircraft's orientation is represented by a set of three consecutive and ordered rotations called *Euler angles*. These angles consist of: *roll* (ϕ) rotation about the X_B-axis which represents the lateral rotation of the aircraft, *pitch* (θ) rotation about the Y_B-axis which represents the forward rotation in the aircraft's nose direction, and *yaw* (ψ) rotation about the Z_B-axis which represents turning the aircraft around the vertical axis; since Euler vector is represented as $\boldsymbol{\eta} = [\phi\ \theta\ \psi]^T$. The sequence of the rotations is important, e.g. ZYX sequence, common in aerospace, indicates first the rotation around the z-axis, $R_Z(\psi)$, then the rotation around the y-axis, $R_Y(\theta)$, followed by the rotation around the x-axis, $R_X(\phi)$. The three principal rotation matrices $R_Z(\psi), R_Y(\theta), R_X(\phi)$ are [110]:

$$R_Z(\psi) = \begin{bmatrix} \cos(\psi) & -\sin(\psi) & 0 \\ \sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{2.1}$$

$$R_Y(\theta) = \begin{bmatrix} \cos(\theta) & 0 & \sin(\theta) \\ 0 & 1 & 0 \\ -\sin(\theta) & 0 & \cos(\theta) \end{bmatrix}, \tag{2.2}$$

$$R_X(\phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\phi) & -\sin(\phi) \\ 0 & \sin(\phi) & \cos(\phi) \end{bmatrix}. \tag{2.3}$$

The direct cosine matrix (DCM), R_N, which represents the three principal rotation matrices for ZYX sequence and transfer from the B-frame to W-frame

is given by

$$R_N = R_Z(\psi).R_Y(\theta).R_X(\phi) = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}, \tag{2.4}$$

where

$$\begin{aligned} r_{11} &= \cos(\psi)\cos(\theta) \\ r_{12} &= \cos(\psi)\sin(\theta)\sin(\phi) - \sin(\psi)\cos(\phi) \\ r_{13} &= \cos(\psi)\sin(\theta)\cos(\phi) + \sin(\psi)\sin(\phi) \\ r_{21} &= \sin(\psi)\cos(\theta) \\ r_{22} &= \sin(\psi)\sin(\theta)\sin(\phi) + \cos(\psi)\cos(\phi) \\ r_{23} &= \sin(\psi)\sin(\theta)\cos(\phi) - \cos(\psi)\sin(\phi) \\ r_{31} &= -\sin(\theta) \\ r_{32} &= \cos(\theta)\sin(\phi) \\ r_{33} &= \cos(\theta)\cos(\phi). \end{aligned} \tag{2.5}$$

Note that the matrix R_N is orthogonal, which means $R_N^{-1} = R_N^T$ and the later matrix transfers from the W-frame to B-frame.

The translational dynamics in the B-frame using a Newton's equation of motion is given by

$$m\dot{\boldsymbol{v}} + \Gamma(\boldsymbol{\omega})m\boldsymbol{v} = \boldsymbol{f}_t, \tag{2.6}$$

where $m \in \Re$ is the total mass of the aircraft, $\boldsymbol{v} = [v_x(t)\ v_y(t)\ v_z(t)]^T \in \Re^3$ is the velocity vector of mass centre, $\dot{\boldsymbol{v}} = [\dot{v}_x(t)\ \dot{v}_y(t)\ \dot{v}_z(t)]^T \in \Re^3$ is the acceleration vector, $\boldsymbol{f}_t \in \Re^3$ is the total forces vector, and $\Gamma(\boldsymbol{\omega}) \in \Re^{3\times3}$ is the cross-product matrix for the Coriolis forces such that $\boldsymbol{\omega} \times m\boldsymbol{v} = \Gamma(\boldsymbol{\omega})m\boldsymbol{v}$, and is given by

$$\Gamma(\boldsymbol{\omega}) = \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix}, \tag{2.7}$$

where $\boldsymbol{\omega} = [\omega_x\ \omega_y\ \omega_z]^T \in \Re^3$ is the angular velocities vector. For the W-frame

$W = [X_W\ Y_W\ Z_W]^T$, the translational dynamics equation is described as

$$m\ddot{\boldsymbol{r}} = \boldsymbol{f}_t, \tag{2.8}$$

where $\boldsymbol{r} = [x(t)\ y(t)\ z(t)]^T \in \Re^3$ is the position vector in W-frame, since $\ddot{\boldsymbol{r}} = R_N\dot{\boldsymbol{v}}$. The centrifugal term, $\Gamma(\boldsymbol{\omega})m\boldsymbol{v}$, is omitted due to the fact that the W-frame does not rotate.

The rotational dynamics in the B-frame using a Newton-Euler equation is given by

$$I\dot{\boldsymbol{\omega}} + \Gamma(\boldsymbol{\omega})I\boldsymbol{\omega} = \boldsymbol{\tau}, \tag{2.9}$$

where $\boldsymbol{\tau} = [\tau_\phi(t)\ \tau_\theta(t)\ \tau_\psi(t)]^T \in \Re^3$ is the torque vector in the B-frame and $I \in \Re^{3\times3}$ is the symmetric and positive-definite inertia matrix of the craft about its mass centre

$$I = \begin{bmatrix} I_{11} & I_{12} & I_{13} \\ I_{21} & I_{22} & I_{23} \\ I_{31} & I_{32} & I_{33} \end{bmatrix}. \tag{2.10}$$

2.1.2.2 Euler-Lagrange Representation

Another representation of the aircraft's dynamical model is using the Euler-Lagrange equations of motion. The Lagrangian L definition consist of three energies: translational (L_T), rotational (L_R), and potential (L_P) [85]:

$$L = T_T + L_R - L_P, \tag{2.11}$$

where

$$L_T = (1/2)\ m\ \dot{\boldsymbol{r}}^T\dot{\boldsymbol{r}}, \tag{2.12}$$

$$L_R = (1/2)\ \boldsymbol{\omega}^T I\boldsymbol{\omega}, \tag{2.13}$$

$$L_P = mgz, \tag{2.14}$$

since $\dot{\boldsymbol{r}} \in \Re^3$ is the linear velocity vector, $I \in \Re^{3\times3}$ is the inertia matrix as in (2.10), and g is the gravitational constant.

The kinematic relationship between the Euler rates vector $\dot{\boldsymbol{\eta}} = [\dot{\phi}\ \dot{\theta}\ \dot{\psi}]^T \in \Re^3$ and the body angular velocities vector $\boldsymbol{\omega}$ (for the rotation sequence ZYX) is given

by [47]:

$$\boldsymbol{\omega} = W\dot{\boldsymbol{\eta}}\ , \quad \begin{bmatrix} \boldsymbol{\omega}_x \\ \boldsymbol{\omega}_y \\ \boldsymbol{\omega}_z \end{bmatrix} = \begin{bmatrix} 1 & 0 & -\sin(\theta) \\ 0 & \cos(\phi) & \cos(\theta)\sin(\phi) \\ 0 & -\sin(\phi) & \cos(\theta)\cos(\phi) \end{bmatrix} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix}, \tag{2.15}$$

and $\dot{\boldsymbol{\eta}} = W^{-1}\boldsymbol{\omega}$. From (2.13) and (2.15), we have the rotational energy

$$L_R = (1/2)(\dot{\boldsymbol{\eta}})^T J(\boldsymbol{\eta})\dot{\boldsymbol{\eta}}, \tag{2.16}$$

since the matrix $J(\boldsymbol{\eta})$,

$$J(\boldsymbol{\eta}) = W^T I W = \begin{bmatrix} j_{11} & j_{12} & j_{13} \\ j_{21} & j_{22} & j_{23} \\ j_{31} & j_{32} & j_{33} \end{bmatrix} \tag{2.17}$$

where

$$\begin{aligned} j_{11} &= I_{11} \\ j_{12} &= 0 \\ j_{13} &= -I_{11}\sin(\theta) \\ j_{21} &= 0 \\ j_{22} &= I_{22}\cos^2(\phi) + I_{33}\sin^2(\phi) \\ j_{23} &= (I_{22} - I_{33})\cos(\phi)\sin(\phi)\cos(\theta) \\ j_{31} &= -I_{11}\sin(\theta) \\ j_{32} &= (I_{22} - I_{33})\cos(\phi)\sin(\phi)\cos(\theta) \\ j_{33} &= I_{11}\sin^2(\theta) + I_{22}\sin^2(\phi)\cos^2(\theta) + I_{33}\cos^2(\phi)\cos^2(\theta) \end{aligned} \tag{2.18}$$

is the Jacobian symmetric positive definite matrix (is invertible) which transfers the angular velocities $\boldsymbol{\omega}$ in (2.13) to their corresponding Euler rates $\dot{\boldsymbol{\eta}}$. The rotational dynamics in B-frame using Euler-Lagrange equation becomes

$$J(\boldsymbol{\eta})\ddot{\boldsymbol{\eta}} + C(\boldsymbol{\eta}, \dot{\boldsymbol{\eta}})\dot{\boldsymbol{\eta}} = \boldsymbol{\tau}, \tag{2.19}$$

where $\ddot{\boldsymbol{\eta}}$ and $\boldsymbol{\tau}$ are Euler acceleration of the vehicle and the control torque vector in

B-frame, respectively. $C(\boldsymbol{\eta}, \dot{\boldsymbol{\eta}})$ is the Coriolis matrix which contains the gyroscopic and centripetal terms

$$C(\boldsymbol{\eta}, \dot{\boldsymbol{\eta}}) = \dot{J}(\boldsymbol{\eta}) - \frac{1}{2}\frac{\partial}{\partial \boldsymbol{\eta}}(\dot{\boldsymbol{\eta}}^T J(\boldsymbol{\eta})) = \begin{bmatrix} c_{11} & c_{12} & c_{13} \\ c_{21} & c_{22} & c_{23} \\ c_{31} & c_{32} & c_{33} \end{bmatrix}, \tag{2.20}$$

where

$$\begin{aligned}
c_{11} &= 0 \\
c_{12} &= (I_{22} - I_{33})(\dot{\theta}\cos(\phi)\sin(\phi) + \dot{\psi}\sin^2(\phi)\cos(\theta)) + (I_{33} - I_{22})\dot{\psi}\cos^2(\phi)\cos(\theta) \\
&\quad - I_{11}\dot{\psi}\cos(\theta) \\
c_{13} &= (I_{33} - I_{22})\dot{\psi}\cos(\phi)\sin(\phi)\cos^2(\theta) \\
c_{21} &= (I_{33} - I_{22})(\dot{\theta}\cos(\phi)\sin(\phi) + \dot{\psi}\sin^2(\phi)\cos(\theta)) + (I_{22} - I_{33})\dot{\psi}\cos^2(\phi)\cos(\theta) \\
&\quad + I_{11}\dot{\psi}\cos(\theta) \\
c_{22} &= (I_{33} - I_{22})\dot{\phi}\cos(\phi)\sin(\phi) \\
c_{23} &= -I_{11}\dot{\psi}\sin(\theta)\cos(\theta) + I_{22}\dot{\psi}\sin^2(\phi)\sin(\theta)\cos(\theta) + I_{33}\dot{\psi}\cos^2(\phi)\sin(\theta)\cos(\theta) \\
c_{31} &= (I_{22} - I_{33})\dot{\psi}\cos^2(\theta)\sin(\phi)\cos(\phi) - I_{11}\dot{\theta}\cos(\theta) \\
c_{32} &= (I_{33} - I_{22})(\dot{\theta}\cos(\phi)\sin(\phi)\sin(\theta) + \dot{\phi}sin^2(\phi)\cos(\theta)) + (I_{22} - I_{33})\dot{\phi}\cos^2(\phi) \\
&\quad \cos(\theta) + I_{11}\dot{\psi}\sin(\theta)\cos(\theta) - I_{22}\dot{\psi}\sin^2(\phi)\sin(\theta)\cos(\theta) \\
&\quad - I_{33}\dot{\psi}\cos^2(\phi)\sin(\theta)\cos(\theta) \\
c_{33} &= (I_{22} - I_{33})\dot{\phi}\cos(\phi)\sin(\phi)\cos^2(\theta) - I_{22}\dot{\theta}\sin^2(\phi)\cos(\theta)\sin(\theta) - I_{33}\dot{\theta}\cos^2(\phi) \\
&\quad \cos(\theta)\sin(\theta) + I_{11}\dot{\theta}\cos(\theta)\sin(\theta)
\end{aligned} \tag{2.21}$$

2.1.2.3 Quaternions Representation

The quaternions representation is another way of describing the dynamical model of the aircraft. It is an alternative method which is use to avoid the singularity associated with the gimbal lock [121] that occurs in the classical 3D Euler representation [29]. Gimbal lock occurs due to the possible singularity of the direction cosine matrix (DCM) in terms of Euler angles. To avoid gimbal lock, the quater-

nions representation [40, 116] can be used to define rigid body attitude. The quaternion is suitable to describe any attitude of a rigid body by Euler's theorem, which states that two geometrically identical bodies can be transformed into each other by a parallel shift of one of the bodies and a single rotation around some axis in 3D space.

The quaternions representation is a hyper complex of four elements $q = q_0 + q_1 i + q_2 j + q_3 k$ where $i^2 = j^2 = k^2 = -1$ and $ij = k, jk = i, ki = j$. The unit quaternion is defined by an angle rotates about a three-dimension rotational axis, such that

$$\boldsymbol{q} = \begin{bmatrix} \cos(\frac{\theta}{2}) \\ \boldsymbol{a}.\sin(\frac{\theta}{2}) \end{bmatrix} = \begin{bmatrix} q_0 \\ \boldsymbol{q}_v \end{bmatrix} = \begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{bmatrix}, \tag{2.22}$$

where $\boldsymbol{q} \in \Re^4$ is the quaternion, $q_0 \in \Re$ is its scalar element (cosine of a rotation angle), and $\boldsymbol{q}_v = [q_1 \ q_2 \ q_3]^T \in \Re^3$ is its vector element (aligned with the axis of rotation), $\boldsymbol{a} = [l \ m \ n]^T \in \Re^3$ is a unit vector where $\|\boldsymbol{a}\| = 1$, and $\theta = 2 \arccos q_0$. The quaternion is suitable to describe any attitude of a rigid body by Euler's theorem, which states that two geometrically identical bodies can be transformed into each other by a parallel shift of one of the bodies and a single rotation around some axis in 3D space .There is the convention that for attitude, unit quaternions are used such that

$$\|\boldsymbol{q}\| = \sqrt{q_0^2 + q_1^2 + q_2^2 + q_3^2} = 1. \tag{2.23}$$

Note that no-rotation (no attitude change) is not the zero quaternion but [1 0 0 0].

The transformation from Euler angles sequence (yaw ψ, pitch θ, roll ϕ) to quaternion can be described as [116]

$$\begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{bmatrix} = \begin{bmatrix} \cos(\frac{\phi}{2})\cos(\frac{\theta}{2})\cos(\frac{\psi}{2}) + \sin(\frac{\phi}{2})\sin(\frac{\theta}{2})\sin(\frac{\psi}{2}) \\ -\cos(\frac{\phi}{2})\sin(\frac{\theta}{2})\sin(\frac{\psi}{2}) + \cos(\frac{\theta}{2})\cos(\frac{\psi}{2})\sin(\frac{\phi}{2}) \\ \cos(\frac{\phi}{2})\cos(\frac{\psi}{2})\sin(\frac{\theta}{2}) + \sin(\frac{\phi}{2})\cos(\frac{\theta}{2})\sin(\frac{\psi}{2}) \\ \cos(\frac{\phi}{2})\cos(\frac{\theta}{2})\sin(\frac{\psi}{2}) - \sin(\frac{\phi}{2})\cos(\frac{\psi}{2})\sin(\frac{\theta}{2}) \end{bmatrix}, \tag{2.24}$$

while the transformation from quaternion to Euler angles

$$\begin{bmatrix}\phi\\ \theta\\ \psi\end{bmatrix} = \begin{bmatrix}atan2(2q_2q_3 + 2q_0q_1,\ {q_0}^2 - {q_1}^2 - {q_2}^2 + {q_3}^2)\\ -asin(2q_1q_3 - 2q_0q_2)\\ atan2(2q_1q_2 + 2q_0q_3,\ {q_0}^2 + {q_1}^2 - {q_2}^2 - {q_3}^2)\end{bmatrix}. \tag{2.25}$$

There is however no problem with relating the Euler angle rotation rates to quaternion rates. The relationship between the quaternions rates $\dot{\boldsymbol{q}} \in \Re^3$ and the angular velocities $\boldsymbol{\omega}(t) = [\omega_x(t)\ \omega_y(t)\ \omega_z(t)]^T \in \Re^3$ in the B-frame can be stated [116] as

$$\boldsymbol{\omega} = \tilde{Z}\,\dot{\boldsymbol{q}}\ ,\quad \begin{bmatrix}\omega_x\\ \omega_y\\ \omega_z\end{bmatrix} = 2\begin{bmatrix}-q_1 & q_0 & q_3 & -q_2\\ -q_2 & -q_3 & q_0 & q_1\\ -q_3 & q_2 & -q_1 & q_0\end{bmatrix}\begin{bmatrix}\dot{q}_0\\ \dot{q}_1\\ \dot{q}_2\\ \dot{q}_3\end{bmatrix}, \tag{2.26}$$

and $\dot{\boldsymbol{q}} = \tilde{Z}^T\boldsymbol{\omega}$ where $\tilde{Z}^T = \tilde{Z}^{-1}$ is an orthogonal matrix. This is useful in control as solid state gyroscopes are available to measure $\omega_x,\ \omega_y,\ \omega_z$, hence giving an opportunity to integrate the attitude changes in realtime.

The transformation from the body coordinates, B-frame, to the world (inertial) coordinates, W-frame, can be expressed using the following matrix [116]

$$R_q = \begin{bmatrix}q_0^2 + q_1^2 - q_2^2 - q_3^2 & 2(q_1q_2 - q_0q_3) & 2(q_0q_2 + q_1q_3)\\ 2(q_1q_2 + q_0q_3) & q_0^2 - q_1^2 + q_2^2 - q_3^2 & 2(q_2q_3 - q_0q_1)\\ 2(q_1q_3 - q_0q_2) & 2(q_0q_1 + q_2q_3) & q_0^2 - q_1^2 - q_2^2 + q_3^2\end{bmatrix}, \tag{2.27}$$

and from W-frame to B-frame using R_q^T where $R_q^T = R_q^{-1}$ is an orthogonal matrix of 3D rotations.

2.2 Feedback Control Theory

This section illustrates the small-gain theorem and dynamic inversion control which are used in this thesis.

2.2.1 Small-Gain Theorem

Mathematical modelling of nonlinear dynamical systems can be described using a number of different approaches. One is the input-output approach, which relates the output of the system to its input without any knowledge of the internal structure of the system. Studying input-output stability is important particularly for interconnected systems in order to ensure that the system is stable. The system is considered stable if a bounded input produces a bounded output. This can be generalised to: *a system is input-output stable if it has finite gain* [58].

One of the ways to measure the stability of interconnected systems is by tracking the variation of signals norms via the gain of the system. This is described by the small-gain theorem. The small-gain theorem is one of the most fundamental and general result for nonlinear feedback systems. It can be used to verify the stability of closed-loop systems under suitable conditions. The theorem has a long history which was first proved by George Zames [134, 135] and developed later (see for example [39, 66, 88]). There are different versions of the small-gain theorem, but the most general which presented in [70] that applies to finite gain input-output stability is chosen in this thesis. The theorem states that for two stable systems H_1 and H_2 in a closed loop feedback structure considering two inputs and four outputs is input-output stable if the product of the gains is lees than 1, i.e. $(\gamma_1\gamma_2 < 1)$. The full proof of the theorem is described in details in Chapter 3.

Dynamical systems usually include modelling uncertainty and if the feedback connection is represented such that H_1 is the stable system and H_2 is the stable disturbance, the condition $\gamma_1\gamma_2 < 1$ is valid when γ_2 is sufficiently small. Therefore, the small-gain theorem is an essential concept for studying the robustness of feedback dynamical systems [39, 70].

The small-gain theorem has been applied to many practical engineering areas. For instance, it is used to derive an attitude control law of spacecrafts [128]. The theorem is used in [79] for stability analysis of hybrid systems. Many other applications of the theorem can be found in [34, 64, 69, 82, 137].

2.2.2 Dynamic Inversion Control

Dynamic inversion control (DIC) [111, 115, 116] is a control technique which provides a straightforward way to derive control laws from the nonlinear dynamics of an aircraft. It is based on the feedback linearisation approach [114] and has been mainly developed to control nonlinear systems and hence there is no need for gain scheduling. The idea of dynamic inversion control is to find a nonlinear control law based on the system's rigid body dynamics. In other words, the nonlinear model that transformed the input-output model into state-space syntax, is converted to a linear model then linearisation control techniques are used for synthesis. It is a method of calculating the torques and/or forces based on the equations of motion and moments of inertia of the system. As each practical nonlinear system has uncertainty, there is an error between the system's dynamics and its nominal control. This error depends on the accuracy of modelling dynamics. The major areas that this control method is used are in robotics and aerospace. This control technique is used in this thesis to design and develop a nonlinear control law of unmanned aerial vehicles.

2.3 Formal Methods

Formal methods can be used to detect and eliminate errors from a designed system. They can be supported by some tools to exhaustively check the complete state space of the design and demonstrate the correctness of its properties. These techniques or tools are based on mathematics which are used for specification, design and verification of hardware and software systems. They can rely on mathematical logic, which consist of formally well-formed statements so that the verification processes are strict deductions in logic and are therefore guaranteed to be correct.

Each formal method essentially consists of one or more of the following three parts; the specification, which is the mathematical model of the desired design's properties, the implementation which is the mathematical model of design's structure, and the verification which is the mathematical representations that describing the relationships between models and using algorithmic analysis or proofs to

verify relations correctness [17].

Due to the variety of applications that need to be checked, each application area requires different modelling methods and different ways of verification. There are many different formal methods available to satisfy the designed system requirements. For instance, theorem provers are used to analyse the validation of a register-transfer level (RTL) description of a Fast Fourier Transform circuit, while algebraic derivational methods used to analyse the validation of the design improvements into a gate-level design. Therefore, there is a wide range of formal methods each of which is used according to the application domain.

The benefits of these techniques are great for many reasons, first of all, their precise semantics can reveal inconsistencies, ambiguities, and incompleteness. They are also considered as excellent guides for defining supporting tools. In addition, properties of the modelled system can be precisely stated, then formally verified [93] [31]. Moreover, these methods could be used to avoid disastrous mistakes especially in safety-critical systems. An example of this is the failure in Ariane 5 rocket which exploded less then forty second after it was launched and the reason of that was due to unverified code that causes a software error which led to computer failure [97]. Another example is from the medical field, where software failures can cause catastrophic damage as in [76] and [53] which led to loss of human-life.

In the following subsections, formal specification and verification used by formal methods are described.

2.3.1 Formal Specification

Formal specification is the process of representing a system and its specifications using a formal language with a mathematically defined syntax and semantics. This process has no proof or analysis while it is used to specify only a system and its requirements. System properties may consist of functional behaviour, functional and/or timing behaviour, performance characteristics, or internal structure of the system. On the other hand, there are formal methods dealing with non-behavioural aspects of the system such as security policies, real-time constraints, and architectural design [31].

2.3.2 Formal Verification

Formal verification is the process of proving or disproving the validity of system's specifications and apply refinement calculus. It is model-based techniques which verify that the mathematical expression satisfy given properties [101]. There are two well-known verification methods: model checking and theorem proving. These techniques are described in the following subsections. There is another proving type called Satisfiability Modulo Theories (SMT) [14], which is an extension of the Boolean satisfiability (SAT), that is a method of deciding the satisfiability of first-order formulas in addition to some background theory with respect to some decidable first-order theory [112]. There are also other formal verification methods such as symbolic simulation and testing, decidable subsets of first order logic, propositional tautology checking, deductive verification, type inference, and data flow analysis [104]. For simulation and testing, for instance, it is cost-efficient method that use to detect the errors in the design, but checking all possible interactions and deadlocks using this technique is rarely possible. Another example is the deductive verification method, which is widely recognized especially in software development, but it is time consuming operation which need experts to work on [30].

2.3.3 Model Checking

One of the well-known tools in testing of designed models and formally verifying and validating their correctness are model checkers [30]. These tools are used to model a system as a finite state transition, like automata or timed automata, and system properties are expressed in the form of proposition temporal logic. Then, the verification problem is reduced to a graph search and an exhaustive exploration of all possible states is accomplished, for instance, using symbolic algorithms [30]. Model checkers are considered as powerful tools in processes checking of different systems, security and communications and complex circuits verifications [45]. Therefore, model checking techniques have a number of advantages. They are fully automatic with no need for supervision or experts. They give counterexamples in case of the design fails to satisfy the required specifications that demonstrate an action to falsifies the specifications. This bug detecting

gives precious insight to know the reasons of faults and fix them. On the other hand, there are some disadvantages of model checking, such as state space explosion problem, one of the main challenges of model checking which occurs in systems that have a large number of components or have many different values in their data structure. This problem is still an issue and it has not be solved by any means in model checking. Moreover, the problem when a program has non-integer parameters or infinite state space designs, where model checkers are not applicable [104]. They can not also used to mathematically prove the derivation correctness of the designed control system.

2.3.4 Theorem Proving

Computer based theorem proving is a computational tool set in some logical system that can be used to prove the soundness and correctness of mathematical arguments. There are two different approaches for theorem proving, *Automated Theorem Proving* (ATP), which automatically proves mathematical formulas by computer software, and *Interactive Theorem Proving* (ITP) which is use to develop formal proofs by human-machine collaboration. It automates steps of formal proofs by aid of a developer guiding the process of proof. The automated steps rely on mathematical logic and automated reasoning techniques. ITPs are proof assistants, which formally define and prove mathematical theorems. Therefore, the user can implement a mathematical theory in an ITP by defining assumptions and some valid logical statements, to start with. Then the ITP procedure will try to prove a sequence of statements, relying on available formal theories, and also by using existing logical methods and techniques or some external resources, such as ATPs and SMT solvers.

The distinction between ITP and ATP systems is not only that ATP systems fully automate proofs but that ATPs tend to have restricted expressivity where, unlike ITPs, they cannot prove higher order mathematical theories. Instead, they are able to prove non complex mathematical formulas that contain inequalities over real numbers, quantified variables, and some mathematical functions. ATPs can be utilised locally in an ITP to prove a step in a proof, by adding their packages to the ITP. This can also be achieved online by using *System on TPTP*

[6], which is a web-based system that includes the most powerful ATPs that can be used to proof mathematical statements automatically. A good example of ATPs especially for control engineering applications is *MetiTariski* [100], a first-order logic (FOL) prover that designed to work over the field of real numbers and inequalities. On the other hand, ITP systems have the ability to support formalising and proving mathematical theories, which involve higher order logic, with the aid of a human supervisor in an interactive way. In contrast to model checkers, ITPs can be applied to an infinite state space design while model checkers can only applied in some settings uner limited applications. However, there are many other features of ITPs such as generality in terms of results and applicability. In addition to modularity as each theory can be defined and then used or modified during theories formalizing and proving. Therefore, the total system is a comprehensive model of correlated theories, i.e., each theory can be built from other theories according to the relations and requirements.

2.3.4.1 MetiTarski Automated Theorem Prover

MetiTarski is an automated theorem prover based on a FOL, which works on the real numbers field. It consists of a resolution theorem prover (Metis) [63] which is works with disjunctions of inequalities and a decision procedure (QEPCAD) [21] which works on finding and removing inconsistent inequalities in the clauses. MetiTarski is able to invoke three reasoning tools which are QEPCAD, Mathematica and Z3 [36] in order to perform proofs. It is designed to solve universally quantified inequalities problems including transcendental and some special functions including log, ln, exp, sin, cos, $sqrt$, etc. This tool is useful especially in control laws as these functions and inequities on real numbers are needed.

As robust controllers are designed with variables constraints and several assumptions which include inequalities on real numbers to bound the variables in the control system in addition to Lyapunov functions which also need such inequalities, MetiTarski is chosen to check the control system stability of unmanned aerial vehicles under the proposed assumptions during the controller design due to its features of proving quantified inequalities including the above functions over real numbers.

2.3.4.2 Isabelle/HOL Interactive Theorem Prover

Isabelle is a generic interactive theorem prover (proof assistant) based on automated reasoning techniques which supports a variety of logics and provides interactive reasoning to prove formal mathematical theories or expressions using logical calculus. It is a specification and verification system written in the ML programming language [92] that represents rules as propositions (not as functions) and constructs proofs by combining rules that comprise a meta-logic based on lambda-calculus [91]. It provides the ability to express the mathematical formulae in a formal language and prove them using different logical tools. Isabelle provides useful proof procedures such as FOL, constructive type theory, Zermelo-Fraenkel set theory (ZF) [54], which offers a formulation of ZF on the top of FOL, and HOL.

The most common platform of Isabelle is *Isabelle/HOL*, which provides a higher-order logic theorem prover environment with quantifiers and semantics. Isabelle has a structured proof language called *Isar* in which proofs are conducted. *Isar* is a mathematics-like proof language that allows proofs to be easily readable and understandable for both users and computers. The mathematical formulas can be formalised and proven in the *Isar* language with the aid of Isabelle's logical tools. Examples of such tools are the *simplifier*, which performs operation and reasoning on equations, the *classical reasoner* that carry out long chains of reasoning procedures to prove statements or theories, automatic proof of *linear arithmetic* statements, *algebraic decision procedures* for decision making verification, advanced *pattern matching*, and *sledgehammer* for automatically finding the proofs based on already proven theorems in Isabelle's library and also calling external FOL provers (ATPs) such as SPASS, Vampire and E-prover; and SMT solvers such as CVC4 and Z3.

Isabelle has been chosen in this research due to its powerful logical techniques and its large library produced by a broad community of applied mathematicians. Isabelle contains most of the formal mathematical theories which are useful for the formalization of control theorems. The most competitive alternative tool to Isabelle is *Coq* [56]. The difference between them is minor from the technical point of view but Isabelle has more useful and larger set of background theories

in its library. For instance, Isabelle's library includes theorems ranging from logics, algebra and type theory such as HOL theory, reals, integers, complex numbers, and functions through spaces definitions such as topological spaces, Euclidian space, vector space and normed space to more complex theories such as derivative, integration, differential equations, high order functions, complex transcendental and operator norm. In addition to other features, for example, there is a code generation feature that allows to transfer the proven specifications from HOL syntax into a corresponding executable code in SML, OCaml, Haskell or the Scala programming languages [91, 92].

There is a wide range of syntax and command types in the Isabelle/HOL, therefore, the most common and useful will be described which are used in Chapter 3 and Chapter 7. Isabelle/HOL expressions and symbols are described in Table 2.1.

2.4 Literature Review

The following subsections are about past studies of UAVs feedback control system and their stability analysis. They include previous efforts and research using different formal methods techniques in control systems verification.

2.4.1 Feedback Control of UAVs

A wide variety of control methods have been proposed in the literature to control and stabilize a multi-rotor UAV. In [77], a now classic approach, a PID controller of the multi-rotor was proposed for regulating the position and orientation of an aircraft. A combination of PID and gain scheduling control approach is presented in [49] to increase robustness. In [133], a cascaded linear PID model-based controller on SO(3) was proposed for quadcopter attitude control to realize complex acrobatic manoeuvres. However successful PID controllers are commercially, they can not guarantee control system stability for various flight conditions with uncertainties and disturbances. In [41], a neural network was used to learn the complete dynamics of the multi-rotor and an output feedback control law is developed to control the translational and rotational motion of the vehicle. The

Table 2.1: Isabelle/HOL symbols and expressions

Expression	Description
$\rightarrow$	mapping from value to value or function to function.
$\longrightarrow$	refers to "*imply*" in HOL.
$\Rightarrow$	used to define a function with its corresponding variables types (e.g., $real \Rightarrow real$) that is a function maps from real to real variable.
$\Longrightarrow$	refers to "*imply*" in *Isar* language in Isabelle. (e.g., $x = 0 \Longrightarrow y = x$) that is x=0 is an assumption and y=x is the statement to be proven.
$\bigwedge$	refers to "*for universal all*" which applies to all assumptions and/or proof statements.
$\forall$	means "*for all*" or "*for any*" and it is for a specific assumption or statement.
$\exists$	means "*there exist*" or "*there is*".
$\exists!$	means "*there is only one*".
$\wedge$	refers to the logical "*and*".
$\vee$	refers to the logical "*or*".
$x^{'}$	the 1^{st} time derivative of x ($x^{''}$ the 2^{nd} time derivative of x).
$\|x\|$	refers to absolute value of x.
$x\$i$	returns the i^{th} element of the vector x.
$\bullet$	an operator for the *dot product* of two vectors.
$*_v$	an operator for the multiplication of a matrix and a vector.
$*_s$	an operator for the multiplication of a scalar value and a vector.
$**$	an operator for the multiplication of two matrices.
$(\lambda\ t.\ x\ t)$	this is equivalent to the function $x(t)$ but under the constraint of an argument (t).
$norm(x)$	the Euclidean norm of a vector or a matrix.
$SUP(x)$	the supremum value of x.

authors in [89] proposed a $PI^{\lambda}D^{\mu}$ neural network aided finite impulse response control scheme for multi-rotor UAVs. In these and similar schemes, it is difficult to quantify whether the controller is near the limits to its performance in order to decide on a modified flight path or landing. Again, it is difficult to know how to use these controllers in real time especially when they are handling an onboard decision-making for flight safety.

A number of robust control schemes have been developed to overcome the modelling uncertainty or disturbances of multi-rotor UAVs. In [108], a robust L_1 optimal control for a multi-rotor was presented and experimentally evaluated. The control objective was to follow the desired trajectory with rejecting persistent disturbances such as sensors errors in the feedback control system by minimizing the L_∞ gain of the plant for these disturbances. Another control method, based on a robust compensation, was proposed in [81] to minimize the effect of aerodynamic disturbances and variable mass distribution.

Several nonlinear control methodologies have been derived by algebraic manipulation in Lyapunov stability derivations. In [16], a nonlinear model-based cascaded controller was proposed by identifying the dynamical parameters of a generic quadcopter. A disturbance based observer for hovering control was proposed in [71]. The authors conducted an extensive analysis of multi-rotor dynamics to provide guidelines for designing a robust control scheme. In [123], a hover mode control based on multi-loop back-stepping design is introduced for a linearized multi-rotor dynamics. An attitude stabilization controller, based on quaternion feedback and integrator backstepping was proposed in [62]. The controller ensures that all the system states are uniformly ultimately bounded with the existence of external disturbances. Similarly, a nonlinear backstepping-based control for multi-rotor aircraft was introduced in [78]. Control system stability was evaluated by Lyapunov methods and LaSalle's invariance theorem with the presence of external disturbances. Other backstepping-based control schemes of multi-rotors can be found in [55, 103, 132]. Sliding mode control method has been used for multi-rotor UAV control. In [131], an adaptive fuzzy gain-scheduling sliding mode controller is introduced for the multi-rotor attitude control. The sliding mode controller is used to control the attitude of the aircraft with the presence of modelling uncertainness and disturbances while the fuzzy logic system is used to

reduce the chattering problem produced by the sliding mode controller. In [118], a robust integral sliding mode controller is developed for attitude control to cope with the parametric uncertainty of quadcopters. A backstepping controller with sliding mode observer is proposed in [87] that overcomes the uncertainty and disturbances of the vehicle. A similar approach was conducted in [119] to reduce external disturbance and load variation effects. The dynamic inversion control method had also been employed to control a quadcopter. In [124], a nonlinear dynamics inversion control scheme was developed for a multi-rotor system to decouple the attitude and position dynamics and maximize the transmission bandwidth of the position control with considering system uncertainty and disturbances. Similarly, a robust dynamic inversion approach was proposed in [33] for controlling and stabilizing under disturbances. A sensor-based incremental nonlinear dynamic inversion controller was developed in [126], with sliding mode disturbance observers for fault-tolerant control, in order to reduce the effects of model uncertainty and disturbances. Control of multi-rotor UAVs with modelling error or flight disturbances has been under various investigations [35, 80, 84].

Although there have been a variety of controllers proposed to control multi-rotors, most of the work available is either concerned with modelling uncertainty or with disturbances. Both *inertial matrix uncertainty and external disturbances* are important factors and can affect the craft at the same time in practice. The upper limits of these need to be known in order to be included in the design, stability proofs and *onboard decision making on flight safety.*

Autonomous helicopter flight control has been widely studied in the last decades. Several controllers have considered the uncertainty and disturbances which are important aspects that affect aircraft stability and performance. However, maintaining attitude stability is still a major control problem due to the aerodynamic mechanism nonlinearity [73]. Previous works on helicopter UAVs control and verification have been reviewed and several are presented. Adaptive inverse dynamic control for an autonomous helicopter is proposed in [68] and [74]. In [107], attitude based model predictive control of an unmanned small helicopter is presented. Robust nonlinear control with considering wind disturbances is proposed in [75] and with H_∞ control in [130].

2.4.2 Formal Methods in Control

Typically, control systems design starts with formal analysis followed by numerical implementation in a simulation tool, then numerical simulations testing for valid behaviour before deploying the implementation. Recently, the use of autocoding generation techniques that produce real-time code from the simulation which reduces manual coding errors have increased. However, nowadays, complex control systems could be designed using digital computation techniques which have been rapidly developed in the last few decades. This enables systems to be formally checked and verified to ensure their validity and reliability. The outcome of this could be significant because system modelling using mathematical derivations can be checked and verified precisely using formal methods like proof assistants can ensure system robustness.

There is a wide range of ITPs including Isabelle/HOL [96], Coq, PVS [98], which are HOL based systems that can be used to verify the stability and performance of control systems with the aid of ATP like MetiTarski [100]. The current development of these techniques enables them to prove the most abstract robust control theories which are used to check systems stability. This thesis is motivated by the need of robust techniques for physical control systems validation and verification. It aims to integrate control theory with ITP techniques by formally proving some of the most important theorems in control theory. This approach will be beneficial especially in safety-critical systems such as flight control, autopilot, autonomous cars and human interactive robots whereby systems stability and performance will be more robustness and safer. Furthermore, information from control theory can be translated into formal mathematical and logical concepts. These concepts then can be proved using proof assistants which can be used later in control systems verification. In particular, for complex systems where computations are very complicated and they are difficult to be handled by a human while it could be done by computer more easily and accurately. To prove that this is applicable, the Small-gain theorem is formally proved in Isabelle/HOL proof assistant.

Due to the importance of the verification of engineering systems in general and control systems stability and performance using formal methods in particu-

lar, several related works and projects in this area are mentioned. There are some projects which have been working on the verification of safety-critical and cyber-physical systems. These include the European project Integrated Tool Chain for Model-based Design of Cyber-Physical Systems (INTO-CPS)[3], where a Functional Mockup Interface (FMI) is developed for integrating the formal verification of Cyber-Physical Systems using PVS [98] theorem prover with model-based software to co-simulation these systems. This approach integrates simulated models in model-based tools such as Modelica, Simulink/Matlab or 20-sim with the FMI interface to verify the control system meets the required specifications using formal methods. This method may produces errors due to interfacing the modelling software with formal method tools instead of using the later directly to verify the intended system. In addition, none of the works which have been verified using this framework targets unmanned areal vehicles or aviation. The proposed verification framework in this thesis is different from this approach as it is verifying the correctness of the derived control law of aviation systems at the design stage before the simulation step directly using ITP then real-time monitoring of their stability by ATP. However, the FMI is also implemented in [136] using Isabelle/UTP [7] framework, where Modelica is used to model the control system of a train then the model is encoded in Isabelle/UTP with FMI framework for co-simulation. Another project is the ERATO Metamathematics for Systems Design (MMSD) [1], where a framework is developed to use formal methods to verify cyber-physical systems of automotive-related applications in industry such as cars. Other projects are conducted by the verification team of NASA Langley Research Center [4] such as integrating MetiTarski with PVS prover [37], air traffic management verification of UAV using formal methods [94] and [95].

Although of these efforts of using formal methods to verify control systems, the derivations of control laws are not covered or verified before implementing them in model-based software, in addition to the absence of onboard real-time stability monitoring of these systems using formal methods. The proposed verification framework presented in this thesis is different from the above approaches as it is dedicated to verify the correctness of the derived control law of aviation systems at the design stage before the simulation step using an ITP, followed by real-time monitoring of stability by ATP.

Hardy [60] developed and implemented a decision procedure to check the validity of a function that has a finite number of inflection points for Nichols plot analysis. This method carried out in the Nichols plot Requirements Verifier (NRV) to implement an automated formal Nichols plot analysis using the computer algebra system (Maple) and PVS proof assistant in addition to the quantifier elimination tool (QEPCAD). NRV is used to verify two control systems: an inverted pendulum and a disk drive reader. Akbarpour and Paulson [8] were also formally proved the control stability, in terms of Nichols plot analysis, of these two systems later on using MetiTarski ATP. The authors used the Maple software to solve the differential equations and obtain problems including the exponential and trigonometric functions which are then passed to the MetiTarski prover. In [38], Denman and his colleagues presented a method to verify the stability of a flight controller with formal Nichols plot analysis by using the MetiTarski automated theorem prover. They extracted the transfer function of a flight control system from Simulink, then defined an exclusion region of the Nichols Plot and proved the unreachability of the exclusion region using MetiTarski. Finally, they applied their proposed method to an autopilot model to check its validity. In [24] the authors presented an approach and tools to translate discrete-time Simulink models to the *LESAR* model checker. These tools have been applied to translate part of Audi's automotive controller. An extension of this work can be found in [109] where further analysis methods are introduced to define a subset of Stateflow for which synchronous semantics can be defined.

Some verification processes can be performed at the design level such as in SimCheck [105] where an implementation of type checking with custom annotations in Simulink blocks was presented. Similar work can be found in Araiza-Illan and her colleagues work [9] where they developed a new approach to automatic translating system's block diagrams modelled in Simulink into the Why3 [52] platform to verify their corresponding properties. The modelled system in Simulink represented high-level properties of stability (Lyapunov stability [114]), feedback gain and robustness. In [10], same authors presented a different approach by performing verification and comparing the results produced by a simulation through assertion checks and the results produced from the Why3 to determine the advantages of the latter.

On the other hand, other verification processes can be accomplished at the code level such as in Feron work [51]. He developed a tool called *credible autocoder* relying on Floyd's and Hoare's proof systems [101] to check Lyapunov-based stability of control systems by producing target C code from Simulink that represents the system specifications in addition to documents that associated with the target code which represent properties of their proofs. Jobredeaux [67], proposed in his thesis an extension of [51] by a credible autocoding framework and tools which are used to develop the state of formal analysis of control software. The framework produced and proved high-level properties of control laws using PVS, such as closed-loop stability, at the code level using the C code.

Other verification approaches were applied on hybrid systems using hybrid theorem proving. For example, in [57], where an introduction of using the hybrid theorem prover *KeYmaera* [102] to prove the control software of aerospace related systems is presented. The authors demonstrate their approach with a case study of lateral collision avoidance maneuver in aviation field.

There have been many attempts made in the same direction of the previously presented works but using different methodologies and various formal methods such as in [19, 20, 26].

Chapter 3

Formal Proof of the Small-Gain Theorem Using Interactive Theorem Proving

3.1 Overview

Control theory can establish properties of systems which hold with all signals of the control system and hence cannot be proven by simulation. The most basic of such properties is the stability of a control subsystem or the overall system. Other examples are statements on robust control performance in the face of dynamical uncertainties and disturbances in sensing and actuation. Until now control theories were developed and checked for their correctness by control scientist manually using their mathematical knowledge. With the emergence of formal methods, there is now the possibility to derive and prove robust control theory by symbolic computation on computers. There is a demand for this approach from industry for the verification of practical control systems with concrete numerical values where the applicability of a control theorem is specialised to an application with given numerical boundaries of parameter variations. This chapter gives an overview of the challenges in the area and demonstrates an example of a computer-based formal proof of the Small-gain theorem using an interactive theorem prover and conclusions are drawn from these initial experiences.

3.2 Mathematical Proof of the Small-Gain Theorem

In order to demonstrate, the Small-gain theorem can be proved in a general way using an interactive theorem proving tool, the version and proof of the theorem presented in Khalil's book [70, Sec. 5.4] has been chosen. From a literature review, it was found that this version was one of the most general proofs using general nonlinear operators and stability concepts. The following are the mathematical procedures of the proof, which need to be presented first in order to comment on the respective steps of a computer-based proof procedure.

If the relation of an input/output system is considered as

$$y = Hu, \tag{3.1}$$

where $H : u \to y$ is an operator that maps the input signal u onto the output signal y. The input signal u belongs to a space of signal functions over the time interval $[0, \infty)$ into the Euclidean space R^m $(u : [0, \infty) \to R^m)$. For the space of piecewise continuous, bounded and square integrable functions, the norm can be defined by

$$\|u\|_{L2} = \sqrt{\int_0^\infty u^T(t)u(t)dt} < \infty, \tag{3.2}$$

where the norm function, which is used to measure the size of the signal, should satisfy the following properties:

- $\|u\| = 0 \iff u = 0$ else $\|u\| > 0$,
- $\|au\| = |a|\|u\|$ for $\forall a \in \Re$,
- $\|u_1 + u_2\| \leq \|u_1\| + \|u_2\|$.

It has been assumed that the input and output signals belong to the same space L so that u, y , $u_\tau, y_\tau \in L$ where u_τ, y_τ are input and output truncated signals, respectively, and all vector space properties are valid. The u_τ is a truncation of

u that is defined by

$$u_\tau(t) = \begin{cases} u(t), & 0 \leq t \leq \tau \\ 0, & t > \tau \end{cases} \tag{3.3}$$

The proof requires some definitions such as system's causality and stability, see [70, Sec. 5.1]. The causality property of an operator $H : L \to L$ is defined by $(Hu)_\tau = (Hu_\tau)_\tau$ for all $\tau \geq 0$. Using this property, the control system stability can be defined

$$\|(Hu)_\tau\| \leq \gamma \|u_\tau\| + \beta, \tag{3.4}$$

where $\gamma, \beta \in \Re$ and $\gamma, \beta > 0$, for all $u \in L$ and $\tau \in [0, \infty)$. For the proof of the Small-gain theorem, suppose that there are two systems $H_1 : L \to L$ and $H_2 : L \to L$, which are both finite-gain stable so that:

$$\|y_{1\tau}\| \leq \gamma_1 \|e_{1\tau}\| + \beta_1, \quad \forall e_1 \in L, \forall \tau \in [0, \infty), \tag{3.5}$$

$$\|y_{2\tau}\| \leq \gamma_2 \|e_{2\tau}\| + \beta_2, \quad \forall e_2 \in L, \forall \tau \in [0, \infty), \tag{3.6}$$

and it is also assumed that for each input $u_1, u_2 \in L$, there exist unique outputs $e_1, y_1, e_2, y_2 \in L$ where $u = [u_1 \quad u_2]^T$, $y = [y_1 \quad y_2]^T$, $e = [e_1 \quad e_2]^T$. The corresponding feedback system is illustrated in Fig. 3.1.

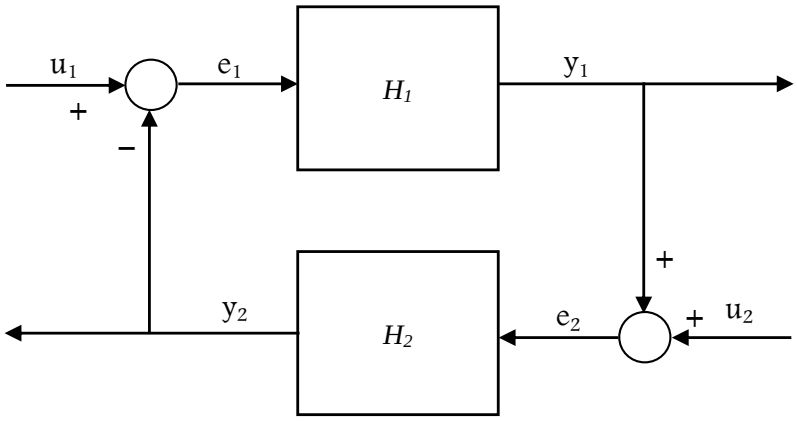


Figure 3.1: Feedback system connection

Theorem 3.1. *Under the above assumptions with finite gains γ_1 for H_1 and γ_2 for H_2, the feedback system is finite-gain stable if $\gamma_1\gamma_2 < 1$.*

Proof. *Assuming existence of the solution, we can write*

$$e_{1\tau} = u_{1\tau} - (H_2 e_2)_\tau, \quad e_{2\tau} = u_{2\tau} + (H_1 e_1)_\tau, \tag{3.7}$$

then,

$$\begin{aligned} \|e_{1\tau}\| &\leq \|u_{1\tau}\| + \|(H_2 e_2)_\tau\| \leq \|u_{1\tau}\| + \gamma_2\|e_{2\tau}\| + \beta_2 \\ &\leq \|u_{1\tau}\| + \gamma_2(\|u_{2\tau}\| + \gamma_1\|e_{1\tau}\| + \beta_1) + \beta_2 \\ &= \gamma_1\gamma_2\|e_{1\tau}\| + (\|u_{1\tau}\| + \gamma_2\|u_{2\tau}\| + \beta_2 + \gamma_2\beta_1), \end{aligned} \tag{3.8}$$

and

$$\begin{aligned} \|e_{2\tau}\| &\leq \|u_{2\tau}\| + \|(H_1 e_1)_\tau\| \leq \|u_{2\tau}\| + \gamma_1\|e_{1\tau}\| + \beta_1 \\ &\leq \|u_{2\tau}\| + \gamma_1(\|u_{1\tau}\| + \gamma_2\|e_{2\tau}\| + \beta_2) + \beta_1 \\ &= \gamma_1\gamma_2\|e_{2\tau}\| + (\|u_{2\tau}\| + \gamma_1\|u_{1\tau}\| + \beta_1 + \gamma_1\beta_2), \end{aligned} \tag{3.9}$$

since $\gamma_1\gamma_2 < 1$,

$$\|e_{1\tau}\| \leq \frac{1}{1-\gamma_1\gamma_2}(\|u_{1\tau}\| + \gamma_2\|u_{2\tau}\| + \beta_2 + \gamma_2\beta_1). \tag{3.10}$$

$$\|e_{2\tau}\| \leq \frac{1}{1-\gamma_1\gamma_2}(\|u_{2\tau}\| + \gamma_1\|u_{1\tau}\| + \beta_1 + \gamma_1\beta_2). \tag{3.11}$$

for all $\tau \in [0, \infty)$. Finally, using the triangle inequality, we have

$$\|e_{1\tau} + e_{2\tau}\| \leq \|e_{1\tau}\| + \|e_{2\tau}\|, \tag{3.12}$$

$$\|e_1 + e_2\| \leq \|e_1\| + \|e_2\|. \tag{3.13}$$

3.3 Formalising and Proving Small-Gain Theorem in Isabelle/HOL Theorem Prover

To describe how the Small-gain theorem has been proved in Isabelle/HOL, this section will show the major steps of the proof procedures starting from the formal definitions of time intervals, signals, truncations of signals, operators causality and stability. A signal's domain and range spaces are also declared in addition to a truncation space and some properties and operations on signals are also declared on these spaces. The definition of an operator space includes the declaration of their properties, which are defined in a general way to provide flexibility and re-usability for the development of other theories in the future.

In this work, some theories, which already exist and have previously been formally proven in Isabelle, have been exploited and used such as *"HOL.thy"*. HOL theory includes the axioms of logic in the higher-order form, the *"Multivariate Analysis.thy"*, which contains, for example, integrations, extended real and algebra theories, *"Bochner Integration.thy"*, which includes Lebesgue integration definition that is used in this work, *"set integral.thy"* which is used for the integration over a specific set or interval, *"Function Algebras.thy"* that includes the properties of functions, for instance, point-wise addition, scalar multiplication, functions addition and multiplication. As theories call other related theories automatically, there are several related theories called and used in this work to carry through the proof of the Small-gain theorem. All the code which are implemented to prove the Small-gain theorem can be found in the web-repository [2].

The steps described previously in Section 3.2 are formalized in Isabelle as follows:

- **Time interval**: Before formalizing the theorem, some definitions are needed to be completed such as the definition of time interval bounds. The overall *time interval* (T) is defined as a real set $[0, \infty)$ such that $t \in T$ where t is a real variable, that is $T = \{t \mid 0 \leq t < \infty\}$. The *truncation time interval* (T_τ) which is a subset of T and $\tau \in T_\tau$ is the period between 0 and τ, where τ is the truncation point, such that $T_\tau = \{\tau \in T \mid 0 \leq t \leq \tau\}$.

Isabelle/HOL code

```
definition T :: "real set" where "T = {t. (0 ≤ t ∧ t < ∞)}"
definition T_τ :: "real set" where "T_τ = {t. (∀τ ∈ T. 0 ≤ t ∧ t ≤ τ)}"
```

- **Signal bounds**: The signal value range is defined over $(-\infty, \infty)$ as

Isabelle/HOL code

```
definition " R = {r. (-∞ < r ∧ r < ∞)}"
```

- **Signal definition**: A straight-forward way to implement input signals is by using ordered pair theory. That approach would not work because the existing ordered pair theory in Isabelle prover is not suitable. Therefore, the following approach of defining the input signal was chosen as a piecewise continuous function by the following general formula:

$$u : T \to R \,;\; u = (\forall t \in T, \exists! u(t) : u(t) \in R).$$

Isabelle/HOL code

```
definition "Signal u = (∀t ∈ T. ∃! x ∈ R. x = u t ∧ u : T → R ∧ u t ∈ u` T ∧
u piecewise_differentiable_on T ∧ continuous_on T u)"
```

- **Domain and range space definition**: The domain and range spaces contain a set of signals, which are declared using the "*locale*" feature in Isabelle which dealing with parametric theories. This feature enables us to form a definition with a set of assumptions in Isabelle. It also gives flexibility in dealing with spaces under certain constraints and properties and provides the possibility to add additional properties when the theory is called and used later. The domain space D and range space G have the same definitions and properties, each of which is defined as a set of signals (functions) under the properties of associativity, commutativity of addition, pointwise

addition, distributivity of scalar multiplication and scalar multiplication over addition.

Isabelle/HOL code

locale *Domain_Space* =

fixes D :: "$(real \Rightarrow real)set$"

assumes *non_empty_D* $[iff, intro?]$: "$D \neq \{\}$"

and *spaceD_mem* $[iff]$: "$range(\lambda t \in T.\ u\ t) \subseteq R \Rightarrow [range(\lambda t \in T.\ u\ t) = (\lambda t \in T.\ u\ t)'\ A \Rightarrow A \subseteq T] \Rightarrow (\lambda t \in T.\ u\ t) \in D$"

and *spaceD_add1* $[iff]$: "$u \in D \Rightarrow s \in D \Rightarrow u + s \in D$"

and *spaceD_add2* $[simp]$: "$u \in D \Rightarrow s \in D \Rightarrow u + s = (\lambda t \in T.\ u\ t + s\ t)$"

and *spaceD_add3* : "$u \in D \Rightarrow s \in D \Rightarrow u + s = s + u$"

and *spaceD_add_assoc* : "$u \in D \Rightarrow s \in D \Rightarrow g \in D \Rightarrow (u + s) + g = u + (s + g)$"

and *spaceD_pointwise* $[simp]$: "$u \in D \Rightarrow s \in D \Rightarrow \forall t \in T.\ (u + s)t = u\ t + s\ t$"

and *spaceD_sub* $[iff]$: "$u \in D \Rightarrow s \in D \Rightarrow u - s \in D$"

and *spaceD_mult1* $[iff]$: "$u \in D \Rightarrow a \in \Re \Rightarrow (a\ .\ u) \in D$"

and *spaceD_mult2* : "$u \in D \Rightarrow a \in \Re \Rightarrow \forall t \in T.\ (a\ .\ u)t = a * u\ t$"

and *spaceD_mult_distr1* : "$u \in D \Rightarrow s \in D \Rightarrow a \in \Re \Rightarrow a\ .\ (u + s) = a\ .\ u + a\ .\ s$"

and *spaceD_mult_distr2* : "$u \in D \Rightarrow s \in D \Rightarrow a \in \Re \Rightarrow \forall t \in T.\ a\ .\ (u + s)t = a * u\ t + a * s\ t$"

and *spaceD_mult_distr3* : "$u \in D \Rightarrow a \in \Re \Rightarrow b \in \Re \Rightarrow (a + b)\ .\ u = a\ .\ u + b\ .\ u$"

and *spaceD_mult_assoc* : "$u \in D \Rightarrow a \in \Re \Rightarrow b \in \Re \Rightarrow (a * b)\ .\ u = a\ .\ (b\ .\ u)$"

Isabelle/HOL code

locale *Range_Space* =

fixes G :: "$(real \Rightarrow real)set$"

assumes *non_empty_G* $[iff, intro?]$: "$G \neq \{\}$"

and *spaceG_mem* $[iff]$: "$range(\lambda t \in T.\ y\ t) \subseteq R \Rightarrow [range(\lambda t \in T.\ y\ t) = (\lambda t \in T.\ y\ t)'B \Rightarrow B \subseteq T] \Rightarrow (\lambda t \in T.\ y\ t) \in G$"

and *spaceG_add1* $[iff]$: "$y \in G \Rightarrow z \in G \Rightarrow y + z \in G$"

and *spaceG_add2* $[simp]$: "$y \in G \Rightarrow z \in G \Rightarrow y + z = (\lambda t \in T.\ y\ t + z\ t)$"

and *spaceG_add3* : "$y \in G \Rightarrow z \in G \Rightarrow y + z = z + y$"

and *spaceG_add_assoc* : "$y \in G \Rightarrow z \in G \Rightarrow j \in G \Rightarrow (y + z) + j = y + (z + j)$"

and *spaceG_pointwise* [*simp*] : "$y \in G \Rightarrow z \in G \Rightarrow \forall t \in T.\ (y + z)t = y\ t + z\ t$"

and *spaceG_mult1* [*iff*] : "$y \in G \Rightarrow a \in \Re \Rightarrow (a\ .\ y) \in G$"

and *spaceG_mult2* : "$y \in G \Rightarrow a \in \Re \Rightarrow \forall t \in T.\ (a\ .\ y)t = a * y\ t$"

and *spaceG_mult_distr1* : "$y \in G \Rightarrow z \in G \Rightarrow a \in \Re \Rightarrow a\ .\ (y + z) = a\ .\ y + a\ .\ z$"

and *spaceG_mult_distr2* : "$y \in G \Rightarrow z \in G \Rightarrow a \in \Re \Rightarrow \forall t \in T.\ a\ .\ (y + z)t = a * y\ t + a * z\ t$"

and *spaceG_mult_distr3* : "$y \in G \Rightarrow a \in \Re \Rightarrow b \in \Re \Rightarrow (a + b)\ .\ y = a\ .\ y + b\ .\ y$"

and *spaceG_mult_assoc* : "$y \in G \Rightarrow a \in \Re \Rightarrow b \in \Re \Rightarrow (a * b)\ .\ y = a\ .\ (b\ .\ y)$"

- **Signal truncation and truncation space definition**: The signal truncation is defined in Isabelle/HOL by declaring a *definition* states truncation concept. It is represented as if there is an input signal u and there is a truncation point τ which belong to the interval $[0, \infty)$ such that all the values in the interval $[0, \tau)$ are valid and the values out of this interval are all set to zero. Afterwards, truncation space TR is declared under specific constraints and all truncated signals should belong to this space.

Isabelle/HOL code

definition *trunc* :: "$(real \Rightarrow real) \Rightarrow (real \Rightarrow real) \Rightarrow real\ set \Rightarrow real\ set \Rightarrow bool$"

where "$trunc\ u\ u_\tau\ U\ U_\tau = (Signal\ u\ \wedge\ u'\ T = U\ \wedge (\forall \tau \in T.\ \forall t \in T.$ **if** $t \leq \tau$ **then** $((u\ t \in U \longrightarrow u\ t \in U_\tau)\ \wedge\ u\ t = u_\tau\ t)$ **else** $((u\ t \in U \longrightarrow 0 \in U_\tau)\ \wedge\ u_\tau\ t = 0)))$"

Isabelle/HOL code

locale *TR_Space* =

fixes TR :: "$(real \Rightarrow real)set$"

assumes *non_empty_TR* [$iff, intro?$] : "$TR \neq \{\}$"

and *spaceTR_mem* [iff] : "$trunc\ u\ u_\tau\ U\ U_\tau \Rightarrow (\lambda t \in T_\tau.\ u_\tau\ t) \in TR$"

and *spaceTR_1* : "$trunc\ u\ u_\tau\ U\ U_\tau \Rightarrow u_\tau \in TR \Rightarrow U_\tau \cap U = u_\tau\ '\ T_\tau$"

and *spaceTR_2* [iff] : "$trunc\ u_1\ u_{1_\tau}\ U_1\ U_{1_\tau} \Rightarrow trunc\ u_2\ u_{2_\tau}\ U_2\ U_{2_\tau} \Rightarrow e_{1_\tau} = u_{1_\tau} - (H_2\ e_{2_\tau}) \Rightarrow e_{2_\tau} = u_{2_\tau} + (H_1\ e_{1_\tau}) \Rightarrow u_{1_\tau} \in TR \Rightarrow u_{2_\tau} \in TR \Rightarrow e_{1_\tau} \in TR$"

and *spaceTR_3* [iff] : "$trunc\ u_1\ u_{1_\tau}\ U_1\ U_{1_\tau} \Rightarrow trunc\ u_2\ u_{2_\tau}\ U_2\ U_{2_\tau} \Rightarrow e_{1_\tau} = u_{1_\tau} - (H_2\ e_{2_\tau}) \Rightarrow e_{2_\tau} = u_{2_\tau} + (H_1\ e_{1_\tau}) \Rightarrow u_{1_\tau} \in TR \Rightarrow u_{2_\tau} \in TR \Rightarrow e_{2_\tau} \in TR$"

and *spaceTR_4* [iff] : "$trunc\ u_1\ u_{1_\tau}\ U_1\ U_{1_\tau} \Rightarrow trunc\ u_2\ u_{2_\tau}\ U_2\ U_{2_\tau} \Rightarrow e_{1_\tau} = u_{1_\tau} - (H_2\ e_{2_\tau}) \Rightarrow$

$e_{2_\tau} = u_{2_\tau} + (H_1\ e_{1_\tau}) \Rightarrow Signal\_Y\ y_{1_\tau}\ H_1\ e_{1_\tau}\ OP \Rightarrow u_{1_\tau} \in TR \Rightarrow u_{2_\tau} \in TR \Rightarrow e_{1_\tau} \in TR \Rightarrow e_{2_\tau} \in TR \Rightarrow y_{1_\tau} \in TR$"

and $spaceTR\_5\ [iff]$: "$trunc\ u_1\ u_{1_\tau}\ U_1\ U_{1_\tau} \Rightarrow trunc\ u_2\ u_{2_\tau}\ U_2\ U_{2_\tau} \Rightarrow e_{1_\tau} = u_{1_\tau} - (H_2\ e_{2_\tau}) \Rightarrow e_{2_\tau} = u_{2_\tau} + (H_1\ e_{1_\tau}) \Rightarrow Signal\_Y\ y_{2_\tau}\ H_2\ e_{2_\tau}\ OP \Rightarrow u_{1_\tau} \in TR \Rightarrow u_{2_\tau} \in TR \Rightarrow e_{1_\tau} \in TR \Rightarrow e_{2_\tau} \in TR \Rightarrow y_{2_\tau} \in TR$"

and $spaceTR\_6$: "$trunc\ u\ u_\tau\ U\ U_\tau \Rightarrow Signal\_Y\ y\ H\ e\ OP \Rightarrow Signal\_Y\ y_\tau\ H\ e_\tau\ OP \Rightarrow e_\tau \in TR \Rightarrow y_\tau \in TR \Rightarrow y_\tau\ '\ T_\tau \subseteq range\ y_\tau \cap range\ y$"

- **Operator causality definition**: Because system stability is required for the proof of the Small-gain theorem and from the fact that the system to be stable should be causal, system causality is defined (see stability and causality in [70]). Causality is an important property of dynamical systems, which is needed to describe practical real-time feedback systems. A system is said to be causal if its output, $y(t)$, at any point depends only on its input, $u(t)$, up to that point. Therefore, with the truncation property the statement will be equivalent to $(Hu)_\tau = (Hu_\tau)_\tau$, which is easily stated in Isabelle.

Isabelle/HOL code

definition *Causality* :: "$(real \Rightarrow real) \Rightarrow (real \Rightarrow real) \Rightarrow (real \Rightarrow real) \Rightarrow (real \Rightarrow real) \Rightarrow real\ set \Rightarrow real\ set \Rightarrow ((real \Rightarrow real) \Rightarrow real \Rightarrow real) \Rightarrow ((real \Rightarrow real) \Rightarrow real \Rightarrow real)set \Rightarrow (real \Rightarrow real)set \Rightarrow bool$"

where "$Causality\ u\ u_\tau\ e\ e_\tau\ U\ U_\tau\ H\ OP\ TR = ((Operator\_Space\ OP \wedge H \in OP \wedge trunc\ u\ u_\tau\ U\ U_\tau \wedge TR\_Space\ TR\ \wedge\ u_\tau \in TR\ \wedge\ e_\tau \in TR) \longrightarrow H(\lambda t \in T_\tau.\ e\ t) = H(\lambda t \in T_\tau.\ e_\tau\ t))$"

- L_2 **norm - Cauchy-Schwarz and Minkowski integral inequalities**: Before defining system stability, there is a need to measure the norm of a signal with its specific properties. Because there is no norm definition in Isabelle/HOL that is suitable for the proof of the Small-gain theorem, it was necessary to formalize and define the norm function. The norm function which should satisfy the properties mentioned in (3.2) is defined. In addition, the Minkowski and Cauchy-Schwarz integral inequalities [59] were

also required to be formalized and derived in Isabelle/HOL using the exist axioms in the software in order to satisfy the required inequality property.

Isabelle/HOL code

definition *L2norm* :: "*real measure* ⇒ (*real* ⇒ *real*) ⇒ *real set* ⇒ *real*"

where "*L2norm M f A* = *sqrt*(*LINT t* : *A*|*M*. (*f t*)^2)"

⋮

Isabelle/HOL code

lemma *schwaz_integral_ineq* :

fixes *f g* :: "*real* ⇒ *real*"

assumes "⋀*t*. *t* ∈ *A*" **and** "*set_integrable M A f*" **and** "*set_integrable M A g*"

and "*set_integrable M A*(λ*t*. (*f t*)^2)" **and** "*set_integrable M A*(λ*t*. (*g t*)^2)"

and "*set_integrable M A*(λ*t*. *f t* ∗ *g t*)" **and** "(*LINT t* : *A*|*M*. (*g t*)^2) > 0"

shows "(*LINT t* : *A*|*M*. *f t* ∗ *g t*) ≤ *sqrt*(*LINT t* : *A*|*M*. (*f t*)^2) ∗ *sqrt*(*LINT t* : *A*|*M*. (*g t*)^2)"

proof

⋮

qed

Isabelle/HOL code

lemma *minkowski_integral_ineq* :

fixes *f g* :: "*real* ⇒ *real*"

assumes "⋀*t*. *t* ∈ *A*" **and** "*set_integrable M A f*" **and** "*set_integrable M A g*"

and "*set_integrable M A*(λ*t*. (*f t*)^2)" **and** "*set_integrable M A*(λ*t*. (*g t*)^2)"

and "*set_integrable M A*(λ*t*. *f t* ∗ *g t*)" **and** "(*LINT t* : *A*|*M*. (*g t*)^2) > 0"

shows "*sqrt*(*LINT t* : *A*|*M*. (*f t* + *g t*)^2) ≤ *sqrt*(*LINT t* : *A*|*M*. (*f t*)^2) + *sqrt*(*LINT t* : *A*|*M*. (*g t*)^2)"

proof

⋮

qed

- **Input/output stability definition**: Input/output (I/O) stability is an essential aspect in the study of interconnected systems stability, where the increasing or decreasing nature of the signals norm can be tracked from the gain of the system. After completing the definitions of signals, truncation of signals, operators causality, and the norm function, it is possible to define I/O stability as in (3.4).

Isabelle/HOL code

definition *Stability* :: "$(real \Rightarrow real) \Rightarrow (real \Rightarrow real) \Rightarrow (real \Rightarrow real) \Rightarrow (real \Rightarrow real) \Rightarrow real\ set \Rightarrow real\ set \Rightarrow real \Rightarrow real \Rightarrow ((real \Rightarrow real) \Rightarrow real \Rightarrow real) \Rightarrow real\ measure \Rightarrow ((real \Rightarrow real) \Rightarrow real \Rightarrow real)set \Rightarrow (real \Rightarrow real)set \Rightarrow bool$"

where "$Stability\ u\ u_\tau\ e\ e_\tau\ U\ U_\tau\ \gamma\ \beta\ H\ M\ OP\ TR = ((Causality\ u\ u_\tau\ e\ e_\tau\ U\ U_\tau\ H\ OP\ TR) \longrightarrow (\exists a.\ \exists b.\ a \in T \wedge b \in T \wedge \gamma = a \wedge \beta = b \wedge ((L2norm\ M\ (\lambda t.\ (H\ e)t)T_\tau) \leq \gamma * (L2norm\ M(\lambda t.\ e_\tau\ t)T_\tau) + \beta)))$"

- **Small-gain theorem formal proof**: After completing the required definitions for formalising the proof, it is possible now to apply the prove procedures step-by-step. The proof steps (3.7)-(3.13) can be applied in Isabelle/HOL under the same assumptions as in [70] in addition to other assumptions listed to perform the proof in Isabelle/HOL. Examples of such assumptions are signals u_1 and u_2 with their truncation, domain space, range space, truncation and operator spaces, causality and stability, and the integrable functions (signals). The proof steps need simple algebra, inequalities, substitutions and some arithmetic operations, which are proved in Isabelle/HOL platform.

Isabelle/HOL code

theorem *Small_Gain_Theorem* :

assumes "$\bigwedge \tau.\ \tau \in T$" **and** "$\bigwedge t.\ t \in T_\tau$" **and** "$Signal\ u_1 \wedge Signal\ u_2$" **and** "$trunc\ u_1\ u_{1_\tau}\ U_1\ U_{1_\tau} \wedge trunc\ u_2\ u_{2_\tau}\ U_2\ U_{2_\tau}$" **and** "$Operator\_Space\ OP \Rightarrow H_1 : D \to G \wedge H_1 \in OP \wedge H_2 : G \to D \wedge H_2 \in OP$" **and**

⋮

shows "$(L2norm\ M\ e_{1_\tau}\ T_\tau) \leq ((L2norm\ M\ u_{1_\tau} T_\tau) + \gamma_2 * (L2norm\ M\ u_{2_\tau}\ T_\tau) + \beta_2 + \gamma_2 * \beta_1)/(1 - \gamma_1 * \gamma_2)$"

and "$(L2norm\ M\ e_{2_\tau} T_\tau) \leq ((L2norm\ M\ u_{2_\tau}\ T_\tau) + \gamma_1 * (L2norm\ M\ u_{1_\tau} T_\tau) + \beta_1 + \gamma_1 * \beta_2)/(1 - \gamma_1 * \gamma_2)$"

and "$(L2norm\ M\ (\lambda t.\ e_1\ t + e_2\ t) T_\tau) \leq (L2norm\ M\ e_1\ T_\tau) + (L2norm\ M\ e_2\ T_\tau)$"

proof

⋮

qed

3.4 Discussion

The interaction process in Isabelle/HOL is carried out in *lemmas* and *theorem* only, where no interaction is needed for *definition* and *locale*. Some *lemmas* and *theorems* can be proven automatically using the supported tools in Isabelle prover such as *try0*, *try*, and *sledgehammer*. If these tools failed to find the proof then the user needs to interact with the prover to find the required pre-existing *lemma* or *theorem* either for the axioms or pre-proven theorems. However, for the Small-gain theorem proof, there was a need for many interactions due to the number of assumptions. It is wealth to mention that when a theorem or lemma has many assumptions then the proof will be difficult to be carried out automatically using the assistant tools, hence interaction is important to finish the proof. Another note form the work conducted so far is that incorrect assumptions will lead to incorrect proof. Moreover, the assistant tools that use to automate the proof may produce incorrect results and therefore the user should check what are the lemmas and theorems used by these tools whether they are related to the statement to be proven or not. If there is any unrelated lemma or theorem, the user can replace it with the related one by finding it manually in Isabelle's proofs repository. Therefore, the user should always check the producing proof to make sure that the proof results are correct. Finally, if the proof is conducted interactively step by step, the prover can not pass any incorrect argument or an argument which is not proved in the prover.

3.5 Shortcomings of Available Methods

Although Isabelle/HOL has an extensive list of proved theories, there was a need for more theories and formalizations to model control systems and their properties. Therefore, some theories and formulas were proved first before proving the Small-gain theorem. The reason for this is that the library of Isabelle is still under development like other interactive theorem prover systems. For instance, the Cauchy-Schwarz's integral inequality, Minkowski's integral inequality and the norm of square integrable function are needed in the proof steps. Therefore, these theories in addition to some related lemmas are formalized and proved (see the web-repository [2]). These theories have been proved as a part of this work because in the proof of the Small-gain theorem, the norm with the integration of a function is needed and the norm definition that already exists in Isabelle library is not applicable. In addition, formalising and proving ZF over HOL platform are needed to work on signals and operators sets. These are some examples of the current limitations of ITPs for proving control theories. Other mathematical concepts are needed to formally prove such theories especially for those dealing with inequalities, which are used extensively in control theory. These concepts are utilised to formalize and prove control theory statements in the formal verification process.

Inequalities involving real-valued special functions are more effective to prove in the MetiTarski theorem prover but it has not yet been integrated with Isabelle/HOL. Moreover, MetiTarski cannot easily be used in association with Isabelle as it is an automated theorem prover. This work required adding and improving theories to Isabelle to deal with control engineering problems. Examples of such improvements are by proving mathematical concepts related to control aspect such as inequalities, convergence concepts, norms, extending ordered-pair theory over HOL, improving set theory over HOL, function algebras, operators, operator norm, etc. Furthermore, there is a need for a collaboration between computer scientists and control engineers to develop and extend theories in theorem proving to improve the formal verification process and this will ultimately lead to assure the robustness of control systems.

3.6 Chapter Summary

The work carried out so far has indicated that even the most theoretical control concepts involving nonlinear operators, causality and normed spaces of signals over the infinite semi-axis of time can be handled by formal languages and theorem proving techniques in higher-order logic using Isabelle/HOL and associated tools. The proof of the Small-gain theorem in Isabelle/HOL indicated that the highly abstract and general control systems can be handled by theorem proving. It was also found that there is a possibility to formulate and prove other control theories using ITPs. This may need to formalise some additional mathematical concepts to prove the intended control theories.

Chapter 4

Nonlinear Attitude Control Design and Verification of a Quadcopter

4.1 Overview

A new attitude controller is presented in this chapter for quadcopters to illustrate the power of controller verification by theorem proving. The example is based on the well known robust inverse dynamics approach [111, 115, 116]. Controller design is analysed using the Lyapunov method to guarantee that the system is asymptotically stable. Then, controller stability is verified by translating the derivative of the Lyapunov function to a FOL formula and applying it in the MetiTarski theorem prover.

4.2 Quadcopter UAV Dynamics

The basic model of the quadcopter is shown in Fig. 4.1. The quadcopter from its name consists of four motors, the front M_1 and rear M_3 motors rotate clockwise while the other two motors, M_2 and M_4, rotate counter-clockwise. This configuration enables the quadcopter vehicle to cancel the effect of the moments produces by each pair of motors. The unmanned quadcopter consists of two

movements: the transitional and rotational. The first determines the vehicle position in the world (inertial) frame while the second, which is considered in this chapter, determines the vehicle attitudes.

The quadcopter moves forwards and backwards when the propeller angular velocity Ω_1 of M_1 reduces/increases and Ω_3 of M_3 increases/reduces by the same amount while keeping the total thrust constant. The forward/backward motion is determined by the pitch angle θ around the Y_B-axis while the right/left motion is determined by the roll angle ϕ around the X_B-axis. Both pitch and roll angles are calculated from the position controller and passed to the attitude controller for calculating the rotational pitch and roll torques τ_θ and τ_ϕ respectively. The rotation around the Z_B-axis is determined according to the given yaw angle ψ by increasing/decreasing the propeller angular velocity of the pair of motors M_1 and M_3 and decreasing/increasing it for the pair of motors M_2 and M_4, since the yaw rotational torque τ_ψ is determined from the given yaw ψ angle.

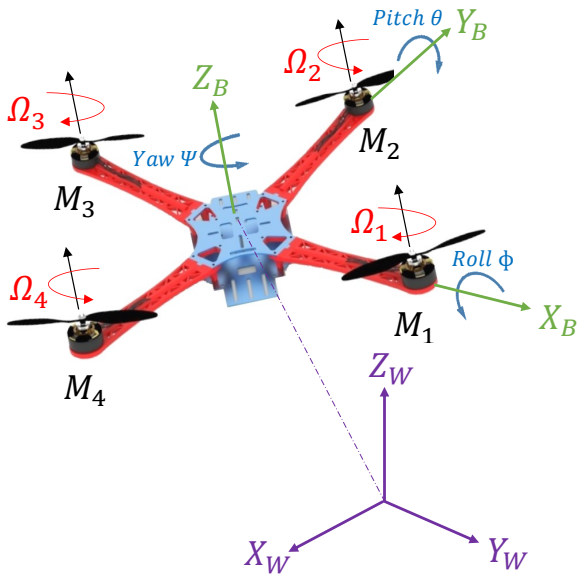


Figure 4.1: A quadcopter illustration in body frame and in inertia frames.

The derivation of the quadcopter attitude dynamics is based on Euler-Lagrange rigid body rotational dynamics (described in Section 2.1.2.2) for controlling the quadcopter rotational motion. The quadcopter attitude dynamics in the B-frame

using Euler-Lagrange equation which as described (2.19)-(2.20) is

$$J(\boldsymbol{\eta})\ddot{\boldsymbol{\eta}} + C(\boldsymbol{\eta}, \dot{\boldsymbol{\eta}})\dot{\boldsymbol{\eta}} + \boldsymbol{d} = \boldsymbol{\tau}, \tag{4.1}$$

where $\boldsymbol{d} \in \Re^3$ is the vector representing the unknown disturbances. Each motor has an angular velocity Ω that produces the vertical force f where

$$f_i = k\Omega_i^2 \tag{4.2}$$

and moments

$$m_i = b\Omega_i^2 \tag{4.3}$$

where k and b are the lift and drag constants respectively. The input to the system, $\boldsymbol{\tau}$, is

$$\boldsymbol{\tau} = \begin{bmatrix} \tau_\phi \\ \tau_\theta \\ \tau_\psi \end{bmatrix} = \begin{bmatrix} \ell k(\Omega_2^2 - \Omega_4^2) \\ \ell k(-\Omega_1^2 + \Omega_3^2) \\ b(-\Omega_1^2 + \Omega_2^2 - \Omega_3^2 + \Omega_4^2) \end{bmatrix}, \tag{4.4}$$

where ℓ is the length from the centre of mass of the quadcopter to each rotor. From (4.1) and (4.4), the attitude dynamics equation becomes

$$\ddot{\boldsymbol{\eta}} = J^{-1}(\boldsymbol{\eta})[\boldsymbol{\tau} - \boldsymbol{n}(\boldsymbol{\eta}, \dot{\boldsymbol{\eta}}) - \boldsymbol{d}], \tag{4.5}$$

where $\boldsymbol{n}(\boldsymbol{\eta}, \dot{\boldsymbol{\eta}}) = C(\boldsymbol{\eta}, \dot{\boldsymbol{\eta}})\dot{\boldsymbol{\eta}}$.

4.3 Control Design

A nonlinear controller is designed for the quadcopter using inverse dynamic control method with considering parameters uncertainty and disturbances. Robust control is also used to bound the uncertainty and then Lyapunov function is used to guarantee asymptotic stability of the control system. Assuming the roll ϕ and pitch θ angles are limited to

$$-\frac{\pi}{2} < \phi < \frac{\pi}{2}\ , \ -\frac{\pi}{2} < \theta < \frac{\pi}{2} \tag{4.6}$$

and by defining the nonlinear control law as

$$\boldsymbol{\tau} = \hat{J}(\boldsymbol{\eta})\boldsymbol{u} + \hat{\boldsymbol{n}}(\boldsymbol{\eta}, \dot{\boldsymbol{\eta}}) + \hat{\boldsymbol{d}} + \boldsymbol{\gamma}, \tag{4.7}$$

where $\boldsymbol{u}$ represents a new input vector to be designed later, $\hat{J}(\boldsymbol{\eta})$ is an estimated matrix of the Jacobian matrix $J(\boldsymbol{\eta})$, $\hat{\boldsymbol{n}}(\boldsymbol{\eta}, \dot{\boldsymbol{\eta}})$ is the nominal vector of $\boldsymbol{n}(\boldsymbol{\eta}, \dot{\boldsymbol{\eta}})$ and the additional term $\boldsymbol{\gamma}$ is added to render the uncertainty of the system which will be defined later; hence from (4.7), equation (4.1) becomes

$$J(\boldsymbol{\eta})\ddot{\boldsymbol{\eta}} + \boldsymbol{n}(\boldsymbol{\eta}, \dot{\boldsymbol{\eta}}) + \boldsymbol{d} = \hat{J}(\boldsymbol{\eta})\boldsymbol{u} + \hat{\boldsymbol{n}}(\boldsymbol{\eta}, \dot{\boldsymbol{\eta}}) + \hat{\boldsymbol{d}} + \boldsymbol{\gamma}. \tag{4.8}$$

Assumption 1: Assume that an estimate $\hat{\boldsymbol{d}}$ of the disturbance $\boldsymbol{d}$ is known (where $\hat{\boldsymbol{d}}$ can be estimated from the maximum wind force that the aircraft may expose), with an error term $\boldsymbol{\Delta d} = \hat{\boldsymbol{d}} - \boldsymbol{d}$ which is known to be bounded by D and $\bar{D}$ where

$$\|\boldsymbol{\Delta d}\| \leq D, \quad \|\boldsymbol{d}\| + D < \bar{D} \tag{4.9}$$

Assumption 2: Assuming that the error between the estimated vector $\hat{\boldsymbol{n}}(\boldsymbol{\eta}, \dot{\boldsymbol{\eta}})$ and the actual $\boldsymbol{n}(\boldsymbol{\eta}, \dot{\boldsymbol{\eta}})$ vector, $\boldsymbol{\Delta n}(\boldsymbol{\eta}, \dot{\boldsymbol{\eta}})$, is also bounded by upper bound as by S as follows

$$\|\boldsymbol{\Delta n}(\boldsymbol{\eta}, \dot{\boldsymbol{\eta}})\| \leq S. \tag{4.10}$$

Suppose that the desired rotational vector is $\boldsymbol{\eta}_d$ and $\dot{\boldsymbol{\eta}}_d$ is to be controlled, then the tracking error defined as,

$$\boldsymbol{e} = \boldsymbol{\eta}_d - \boldsymbol{\eta} \tag{4.11}$$

$$\dot{\boldsymbol{e}} = \dot{\boldsymbol{\eta}}_d - \dot{\boldsymbol{\eta}} \tag{4.12}$$

where $\boldsymbol{\eta}$ and $\dot{\boldsymbol{\eta}}$ are the measured Euler angles and Euler rates respectively. Given $\ddot{\boldsymbol{\eta}}_d$, the $\dot{\boldsymbol{\eta}}_d$ can be obtained by integration and the control input $\boldsymbol{u}$ in (4.7) is defined by

$$\boldsymbol{u} = \ddot{\boldsymbol{\eta}}_d + K_r\dot{\boldsymbol{e}} + K_\eta\boldsymbol{e} = \ddot{\boldsymbol{\eta}}_d + K_r(\dot{\boldsymbol{\eta}}_d - \dot{\boldsymbol{\eta}}) + K_\eta(\boldsymbol{\eta}_d - \boldsymbol{\eta}) \tag{4.13}$$

where $K_r = diag[k_{r_1}\ k_{r_2}\ k_{r_3}] \in \Re^{3\times3}, K_\eta = diag[k_{\eta_1}\ k_{\eta_2}\ k_{\eta_3}] \in \Re^{3\times3}$ are positive-definite diagonal gain matrices. From (4.8), we have

$$\begin{aligned}\ddot{\boldsymbol{\eta}} &= \hat{J}(\boldsymbol{\eta})J^{-1}(\boldsymbol{\eta})\boldsymbol{u} + J^{-1}(\boldsymbol{\eta})[\boldsymbol{\Delta n}(\boldsymbol{\eta},\dot{\boldsymbol{\eta}}) + \boldsymbol{\Delta d}] \\ &\quad + J^{-1}(\boldsymbol{\eta})\boldsymbol{\gamma} \\ &= \boldsymbol{u} + (\hat{J}(\boldsymbol{\eta})J^{-1}(\boldsymbol{\eta}) - I)\boldsymbol{u} + J^{-1}(\boldsymbol{\eta})[\boldsymbol{\Delta n}(\boldsymbol{\eta},\dot{\boldsymbol{\eta}}) + \boldsymbol{\Delta d}] \\ &\quad + J^{-1}(\boldsymbol{\eta})\boldsymbol{\gamma} \\ &= \boldsymbol{u} - \boldsymbol{v} + J^{-1}(\boldsymbol{\eta})\boldsymbol{\gamma}\end{aligned} \tag{4.14}$$

where

$$\boldsymbol{v} = [I - \hat{J}(\boldsymbol{\eta})J^{-1}(\boldsymbol{\eta})]\boldsymbol{u} - J^{-1}(\boldsymbol{\eta})[\boldsymbol{\Delta n}(\boldsymbol{\eta},\dot{\boldsymbol{\eta}}) + \boldsymbol{\Delta d}].$$

From (4.11) - (4.14), we have the error dynamics as

$$\ddot{\boldsymbol{e}} + K_r\dot{\boldsymbol{e}} + K_\eta\boldsymbol{e} = \boldsymbol{v} - J^{-1}(\boldsymbol{\eta})\boldsymbol{\gamma}, \tag{4.15}$$

then by setting $\boldsymbol{E} \in \Re^{6\times1}$ to

$$\boldsymbol{E} = \begin{bmatrix}\boldsymbol{e} \\ \dot{\boldsymbol{e}}\end{bmatrix} \tag{4.16}$$

the following closed-loop error dynamics equation is obtained

$$\dot{\boldsymbol{E}} = A\boldsymbol{E} + B[\boldsymbol{v} - J^{-1}(\boldsymbol{\eta})\boldsymbol{\gamma}] \tag{4.17}$$

where

$$A = \begin{bmatrix} 0^{3\times3} & I^{3\times3} \\ -K_\eta^{3\times3} & -K_r^{3\times3}\end{bmatrix},\ B = \begin{bmatrix}0^{3\times3} \\ I^{3\times3}\end{bmatrix}. \tag{4.18}$$

To bound the error, the uncertainty in $\boldsymbol{v}$ needs to be bounded and this can be achieved by using robust control technique then $\boldsymbol{\gamma}$ needs to be defined using Lyapunov function. The control input $\boldsymbol{u}$ in addition to the term $\boldsymbol{\gamma}$ should guarantee asymptotic stability for any $\boldsymbol{v}$ varying within the bounded range, where $\boldsymbol{v}$ is uncertain but an estimation on its range of variation can be obtained.

Assumption 3: From (4.14), the following assumptions have been chosen in order to bound the term $\boldsymbol{v}$

$$sup(\|\ddot{\boldsymbol{\eta}}_d\|) < H \tag{4.19}$$

$$\|I - \hat{J}(\boldsymbol{\eta})J^{-1}(\boldsymbol{\eta})\| \leq \xi \leq 1, \tag{4.20}$$

and for the matrix $J(\boldsymbol{\eta})$, in addition to the positive-definite matrix property, it should have an upper and lower limited bounds

$$\beta_{min} \leq \|J^{-1}(\boldsymbol{\eta})\| \leq \beta_{max}. \tag{4.21}$$

4.4 Stability Analysis

The Lyapunov direct method is used to define the term $\boldsymbol{\gamma}$ and to guarantee that the system error converges to zero. By setting the equilibrium point $\boldsymbol{E} = 0$ where $V(0) = 0$ and defining the following positive-definite function

$$V(\boldsymbol{E}) = \boldsymbol{E}^T Q \boldsymbol{E} > 0 \ , \ \forall \boldsymbol{E} \neq 0 \tag{4.22}$$

were $Q \in \Re^{6\times 6}$ is a symmetric positive-definite matrix. The time derivative of the function $V(\boldsymbol{E})$ along the trajectory of the error system is

$$\begin{aligned} \dot{V}(\boldsymbol{E}) =& \dot{\boldsymbol{E}}^T Q \boldsymbol{E} + \boldsymbol{E}^T Q \dot{\boldsymbol{E}} \\ =& \boldsymbol{E}^T [A^T Q + QA]\boldsymbol{E} + 2\boldsymbol{E}^T QB(\boldsymbol{v} - J^{-1}(\boldsymbol{\eta})\boldsymbol{\gamma}), \end{aligned} \tag{4.23}$$

since A has eigenvalues with all negative real parts, hence for any symmetric positive-definite matrix P, we have

$$A^T Q + QA = -P, \tag{4.24}$$

which gives a unique solution Q. Therefore, the term $\boldsymbol{E}^T[A^T Q + QA]\boldsymbol{E}$ in (4.23) is negative and the equation can be rewritten as

$$\dot{V}(\boldsymbol{E}) = -\boldsymbol{E}^T P \boldsymbol{E} + 2\boldsymbol{E}^T QB(\boldsymbol{v} - J^{-1}(\boldsymbol{\eta})\boldsymbol{\gamma}). \tag{4.25}$$

As the term $-\boldsymbol{E}^T P \boldsymbol{E}$ in the above equation is negative definite, then if $\boldsymbol{E} \in G(B^T Q)$ the solution converges. If $\boldsymbol{E} \notin G(B^T Q)$ then $\boldsymbol{\gamma}$ must be chosen to

render the second term of the above equation to be less than or equal to zero. The term $\boldsymbol{\gamma}$ has been chosen as

$$\boldsymbol{\gamma} = \begin{cases} \dfrac{\delta(\boldsymbol{E})}{\|B^T Q \boldsymbol{E}\|} B^T Q \boldsymbol{E} & \|B^T Q \boldsymbol{E}\| \geq \sigma \\ \dfrac{\delta(\boldsymbol{E})}{\sigma} B^T Q \boldsymbol{E} & \|B^T Q \boldsymbol{E}\| < \sigma \end{cases} \tag{4.26}$$

where $\delta(\boldsymbol{E})$ is a positive time-varying scalar. Assuming that $\|B^T Q \boldsymbol{E}\| \geq \sigma$, then we have

$$\begin{aligned} \boldsymbol{E}^T Q B(\boldsymbol{v} - J^{-1}(\boldsymbol{\eta})\boldsymbol{\gamma}) \leq & \|B^T Q \boldsymbol{E}\| \|\boldsymbol{v}\| - \beta_{min}\delta(\boldsymbol{E})\|B^T Q \boldsymbol{E}\| \\ = & \|B^T Q \boldsymbol{E}\|(\|\boldsymbol{v}\| - \beta_{min}\delta(\boldsymbol{E})) \end{aligned} \tag{4.27}$$

and if we choose $\delta(\boldsymbol{E})$ as

$$\delta(\boldsymbol{E}) \geq \frac{\|\boldsymbol{v}\|}{\beta_{min}} \tag{4.28}$$

then from (4.9), (4.10), (4.14), (4.19), (4.20), and (4.21), we have

$$\begin{aligned} \|\boldsymbol{v}\| &\leq \|I - \hat{J}(\boldsymbol{\eta})J^{-1}(\boldsymbol{\eta})\|(\|\ddot{\boldsymbol{\eta}}_d\| + \|K_r\|\|\dot{\boldsymbol{e}}\| + \|K_\eta\|\|\boldsymbol{e}\|) \\ &\quad + \|J^{-1}(\boldsymbol{\eta})\|(\|\boldsymbol{\Delta n}(\boldsymbol{\eta}, \dot{\boldsymbol{\eta}})\| + \|\boldsymbol{\Delta d}\|) \\ &\leq \xi(H + \|K_r\|\|\dot{\boldsymbol{e}}\| + \|K_\eta\|\|\boldsymbol{e}\|) + \beta_{max}(S + D) \end{aligned} \tag{4.29}$$

from previous two equations, we get

$$\delta(\boldsymbol{E}) \geq \frac{\xi}{\beta_{min}}(H + \|K_r\|\|\dot{\boldsymbol{e}}\| + \|K_\eta\|\|\boldsymbol{e}\|) + \frac{\beta_{max}}{\beta_{min}}(S + D) \tag{4.30}$$

Finally, (4.25) becomes

$$\dot{V}(\boldsymbol{E}) = -\boldsymbol{E}^T P \boldsymbol{E} + 2\boldsymbol{E}^T Q B(\boldsymbol{v} - J^{-1}(\boldsymbol{\eta})\frac{\delta(\boldsymbol{E})}{\|B^T Q \boldsymbol{E}\|} B^T Q \boldsymbol{E}) < 0 \tag{4.31}$$

or

$$\dot{V}(\boldsymbol{E}) = -\boldsymbol{E}^T P \boldsymbol{E} + 2\boldsymbol{E}^T Q B(\boldsymbol{v} - J^{-1}(\boldsymbol{\eta})\frac{\delta(\boldsymbol{E})}{\sigma} B^T Q \boldsymbol{E}) < 0 \tag{4.32}$$

The next section illustrates the application of these results in Simulink/Matlab.

4.5 Simulation

The controller developed in the previous section has been implemented in Simulink/-Matlab for testing with the nonlinear quadcopter dynamics of (4.1). In order to test the attitude controller, roll and pitch angles need to be passed as inputs to the attitude controller in order to compute the required torque. For this purpose, a simple cascaded P position controller is implemented. The cascaded P position controller takes the given path trajectory as input and the aircraft's measured position and velocity as feedback and calculates the desired roll and pitch angles. Note that the yaw angle is given directly as an input from the pilot or with the given XYZ trajectory. The controllers are simulated with the quadcopter's nonlinear dynamics which are implemented in Simulink/Matlab to achieve better results.

The initial roll ϕ, pitch θ and yaw ψ angles are set to zero. According to the given trajectory, the attitude controller shows that the measured roll, pitch and yaw angles track the references. As can be seen in Fig. 4.2 - 4.7, the measured roll, pitch and yaw (dot-red line) are well followed the reference signal (continuous blue line) in different maneuvers and even with the existence of external disturbances like winds. When any disturbance occur, the controller should produce a counter amount which approximately equals the disturbance value to cope this variation and keep the drone within the stability bounds. The controller parameters are obtained and listed in Table 4.1 which are used in the verification process later.

From (4.24), the positive definite matrix P is chosen then the symmetric positive definite matrix Q is obtained as

$$P = \begin{bmatrix} 9*10^{-12} & 0 & 0 & 0 & 0 & 0 \\ 0 & 9*10^{-12} & 0 & 0 & 0 & 0 \\ 0 & 0 & 5*10^{-9} & 0 & 0 & 0 \\ 0 & 0 & 0 & 3*10^{-8} & 0 & 0 \\ 0 & 0 & 0 & 0 & 3*10^{-8} & 0 \\ 0 & 0 & 0 & 0 & 0 & 8*10^{-4} \end{bmatrix} \tag{4.33}$$

$$Q = \begin{bmatrix} 2*10^{-7} & 0 & 0 & 0 & 0 & 0 \\ 0 & 2*10^{-7} & 0 & 0 & 0 & 0 \\ 0 & 0 & 4.6*10^{-4} & 0 & 0 & 0 \\ 0 & 0 & 0 & 3.8*10^{-6} & 0 & 0 \\ 0 & 0 & 0 & 0 & 3.8*10^{-6} & 0 \\ 0 & 0 & 0 & 0 & 0 & 8.2*10^{-4} \end{bmatrix} \tag{4.34}$$

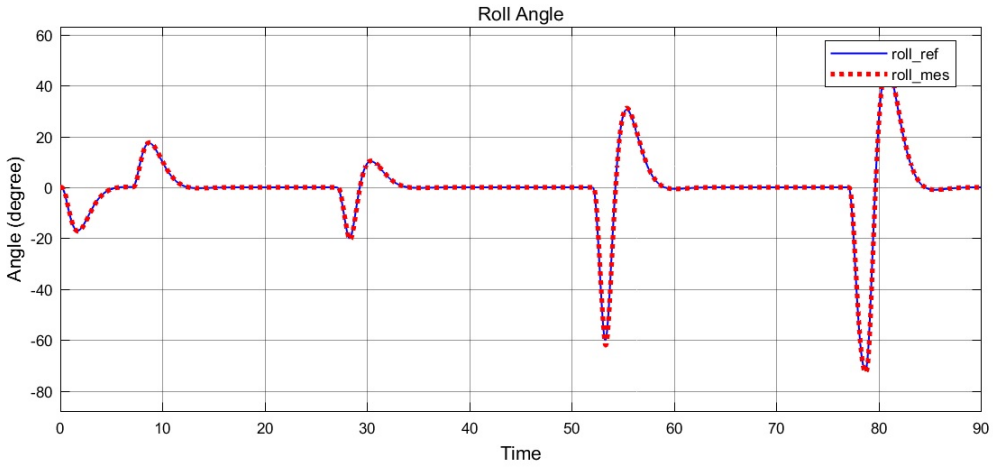


Figure 4.2: Roll angle without disturbances

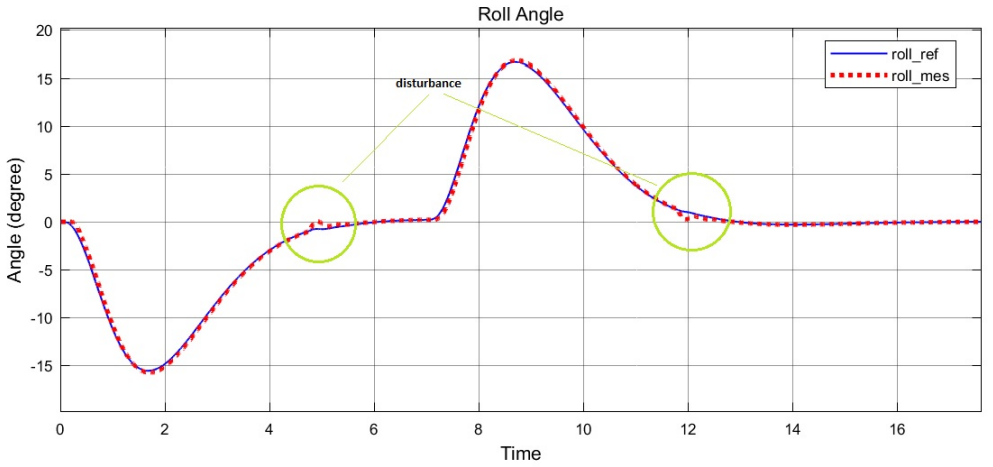


Figure 4.3: Roll angle with disturbances

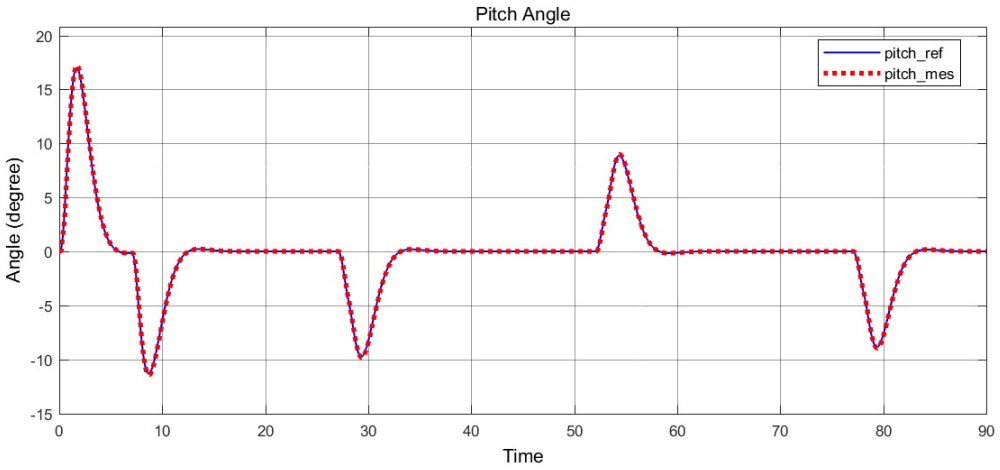


Figure 4.4: Pitch angle without disturbances

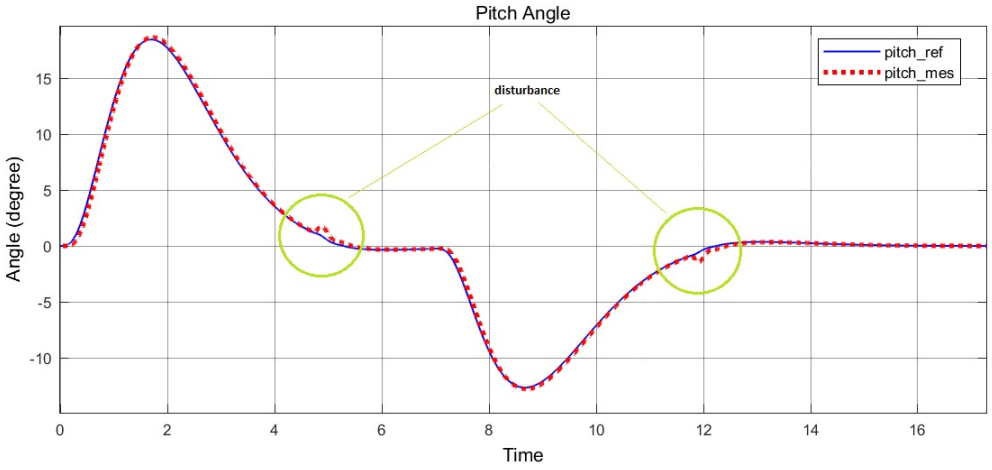


Figure 4.5: Pitch angle with disturbances

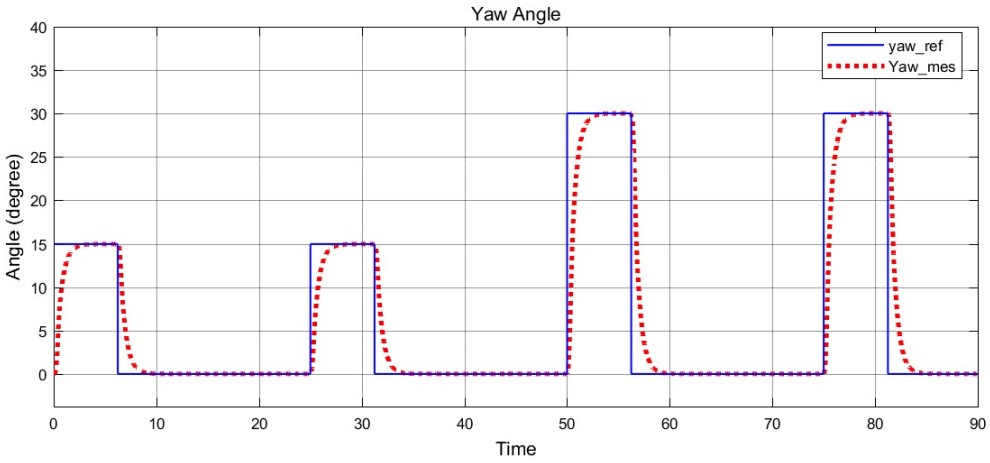


Figure 4.6: Yaw angle without disturbances

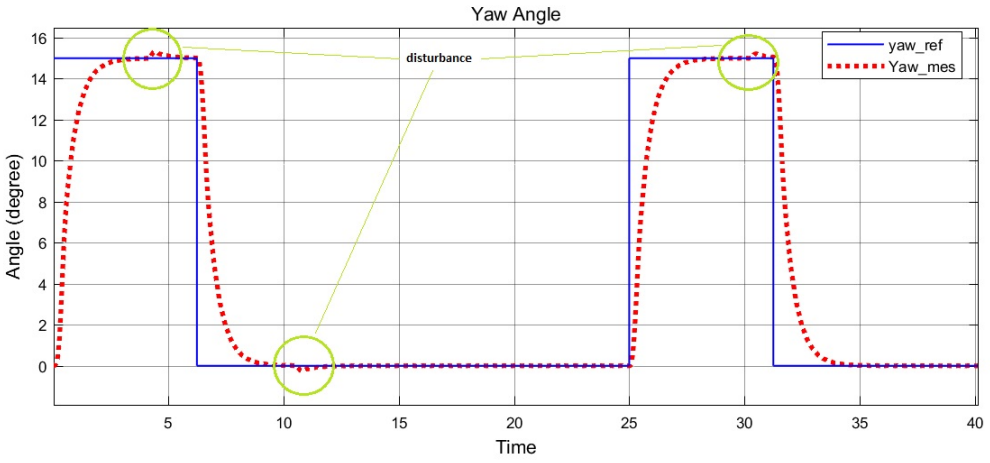


Figure 4.7: Yaw angle with disturbances

Table 4.1: Quadrotor Parameters

Parameter	***Value***	***Parameter***	***Value***
$\hat{I}_x$	$5.831 * 10^{-3}$	σ	$9 * 10^{-13}$
$\hat{I}_y$	$5.831 * 10^{-3}$	k	$12 * 10^{-8}$
$\hat{I}_z$	$1.166 * 10^{-2}$	b	$9 * 10^{-6}$
k_{η_1}	17.5	ξ	0.5
k_{η_2}	17.5	H	1.2
k_{η_3}	1.8	S	$1 * 10^{-3}$
k_{r_1}	0.004	D	$1 * 10^{-3}$
k_{r_2}	0.004	β_{min}	170.5
k_{r_3}	0.4826	β_{max}	173
ℓ	20 cm		

4.6 Controller Stability Verification

To ensure that the control system is asymptotically stable using symbolic computations, equation (4.31) and (4.32) should be strictly negative with the given assumptions. Simulation can not guarantee that this is valid for all possible values as it is relying on numerical computations. Therefore, there is a need to check the validity of Lyapunov stability using symbolic computations. This can be done using theorem provers such as MetiTarski. The following subsections will demonstrate the validity of the controller stability using the MetiTarski prover.

4.6.1 Lyapunov Stability Verification

Due to the limitations of MetiTarski prover as it is a FOL system which means that it works on real scalar values without the ability to work with vectors and matrices, the Lyapunov equations (4.31) and (4.32) have been simplified using the Matlab symbolic toolbox and then formalised to the FOL format to accomplish the verification task. All code which have been formalized in MetiTarski prover to verify the control system stability can be found in the web-repository [2]. An example of the code is shown below:

MetiTarski code

```
fof(QCD_Lyap_eq1_E1,conjecture, ![E_1,E_2, E_3,E_4,E_5,E_6,Phi,Theta,V_1]:
?[Delta_E_1]:
%assumptions
(E_1 != 0 & abs(E_1) <= 0.0987 & abs(E_4) <=1.1071 & E_2 != 0 &
abs(E_2) <= 0.0987 & abs(E_5) <=1.1071 & E_3 != 0 & abs(E_3) <= 1.5533 &
abs(E_6) <= 2.7957 & Phi > -1.5708 & Phi <1.5708 & Theta > -1.5708 &
Theta <1.5708 & V_1 <=(0.5*(1.2+(0.004*abs(E_4))+(17.5*abs(E_1)))
+ (173*(0.001+0.001))) & Delta_E_1 > 0 & Delta_E_1 >= ((0.5/170.5)*
(1.2+(0.004* abs(E_4))+(17.5*abs(E_1)))) + ((173/170.5) *(0.001+0.001))
% implies
=> (... < 0)).
%Note: the above "..." can be seen in the web repository.
```

Table 4.2: Variables and vectors notations in MetiTarski

Variable/Vector	***Notation***
ϕ	Phi
θ	$Theta$
ψ	Psi
$\boldsymbol{E}(i)$	E_i
$\boldsymbol{v}(i)$	V_i
$\delta(\boldsymbol{E})$	$Delta\_E$

As can be seen in the code above, in the first line, fof related to first-order logic and the quantifiers (!) and (?) means *for any* and *for some* respectively, which are used to indicate variables quantification. The symbol => means *implies* which indicate that the lines before this symbol are assumptions and after is the statement to be proven. After *implies*(=>), the Lyapunov equation (4.31) with the first element scalar value of the error vector $\boldsymbol{E}(1)$, which is $E1$ in the above code, is implemented in MetiTarski and it shows that the formula is satisfy the given assumptions for all possible values within the given bounds. The error $\boldsymbol{e}$ and error rate $\dot{\boldsymbol{e}}$ values in $\boldsymbol{E}$ vector are bounded based on the assumption in (4.6) as $0 < |\boldsymbol{E}(1,2)| \leq 0.0987$, $0 < |\boldsymbol{E}(3)| \leq 1.5533$, the error rates $0 < |\boldsymbol{E}(4,5)| \leq$

1.1071 and $0 < |\boldsymbol{E}(6)| \leq 2.7957$, where all values are in radians. The time required for MetiTarski prover to generate the proof in the above code is 0.288 seconds. Note that the proof in MetiTarski was on a Linux Ubuntu operating system, Core i5 1.6 GHz CPU and 8 GB RAM. The variables notation used in MetiTarski is shown in Table (4.2). The verification process performed for all error values in $\boldsymbol{E}$ vector for both (4.31) and (4.32) to complete the controller stability verification process. The approach used above is a first step towards verifying the controller stability while further verification processes can be done by proving the correctness of the controller and it stability derivations. This can not be achieved using FOL provers like MetiTarski due to their limitation but HOL interactive theorem proving like Isabelle/HOL can be used (see Chapter 7). Furthermore, this approach can be implemented onboard in realtime to check the stability during the flight as described in 7.3.2.

4.7 Chapter Summary

A model-based verification technique using symbolic computations is presented to verify quadcopter stability based on Lyapunov's direct method using the MetiTarski automated theorem prover. A nonlinear robust attitude controller is presented using inverse dynamics control method with system uncertainty and disturbances. The control system implemented in Simulink/Matlab and the results have been shown. The verification process results show that control system stability can be verified using ATP to guarantee asymptotic stability of the controller and to ensure that the system works within the given bounds and performance specifications.

Chapter 5

Nonlinear Attitude Control Design and Verification for a Helicopter

5.1 Overview

In this chapter, a robust nonlinear attitude controller is designed which takes into account modelling uncertainty and external wind disturbances for an unmanned small helicopter system. The controller stability is demonstrated using Lyapunov direct method and an invariant set is defined with considering parameters constraints. The controller is then verified since the verification method includes verifying that the control system is asymptotically stable and ensuring that the system states are within the defined control set using formal methods represented by MetiTarski [100] automated theorem prover (ATP). The aircraft control parameters are computed based on a VARIO Benzin Trainer helicopter [120].

The motivation in this research is to work towards automating this verification method and integrating MetiTarski with the autopilot system [12] to perform onboard real-time verification. The parameters required for the verification process can be passed from the autopilot to the MetiTarski prover. The results produced by MetiTarski, proved or not proved, can be then passed back to the autopilot to

check whether the aircraft is unstable or out of the designed constraints. This will allow the autopilot could make decisions to cope with this, for example, avoiding any aggressive maneuvers or performing an emergency landing. By using this approach, the autopilot flight management system will be more trustworthy; i.e. if the vehicle becomes unstable or the controller specifications constraints are violated, the autopilot will either send warnings to the pilot, for semi-autonomous flight or perform an emergency safe landing in case of full-autonomous flight. This verification method is general and can be applied to different kinds of autonomous UAVs that include autopilots such as multicopters and fixed wings crafts. To demonstrate the proposed approach, the nonlinear dynamical model with the designed controller of the helicopter UAV are implemented in Simulink/-Matlab and, as a first step towards the integration, simulation and MetiTarski are employed to illustrate the possibility of implementing the verification method with the autopilot system.

5.2 Helicopter UAV Dynamics

The small helicopter UAV is shown in Fig. 5.1. The helicopter aircraft is controlled by four operating controls: 1) the throttle T_M which determines the amount of thrust generated by the main motor P_M; 2) the throttle T_R which determines the amount of side force produces by the tail motor P_R that required to rotate the aircraft (yaw); 3) collective pitch for controlling the angles of main motor blades hence moving the aircraft up/down vertically; 4) cyclic pitch for determining the flapping angles which are tilting the main rotor blades to move the aircraft forward/backward (pitch) or right/left (roll). More details on the dynamics of a Helicopter can, for example, be found in [99, 116]. However, this chapter will present only the attitude rotational control to illustrate the proposed approach.

The helicopter three-dimensional attitude dynamics are represented in the body-fixed frame B by Euler-Lagrange rigid body rotational dynamics (which were described in Section 2.1.2.2) as follows

$$H(\boldsymbol{q})\ddot{\boldsymbol{q}} + D(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \boldsymbol{w}_d = \boldsymbol{\tau}, \tag{5.1}$$

where $\boldsymbol{q} = [\psi(t)\ \theta(t)\ \phi(t)]^T \in \Re^3$ is the Euler angles vector with yaw, pitch and roll respectively; $\dot{\boldsymbol{q}} \in \Re^3$ represents the Euler rates vector and $\ddot{\boldsymbol{q}} \in \Re^3$ is the Euler acceleration vector; $\boldsymbol{w}_d = [w_{d\psi}(t)\ w_{d\theta}(t)\ w_{d\phi}(t)]^T \in \Re^3$ is the external disturbances vector; $\boldsymbol{\tau} = [\tau_\psi(t)\ \tau_\theta(t)\ \tau_\phi(t)]^T \in \Re^3$ is the torque vector; $D(\boldsymbol{q}, \dot{\boldsymbol{q}}) \in \Re^{3\times3}$ is the Coriolis matrix which the total matrix is shown in equation 5.129 of [25]. $H(\boldsymbol{q}) \in \Re^{3\times3}$ is an invertible Jacobian symmetric positive-definite matrix

$$H(\boldsymbol{q}) = \begin{bmatrix} J_x s_\theta^2 + J_y c_\theta^2 s_\phi^2 + J_z c_\theta^2 c_\phi^2 & J_y c_\theta s_\phi c_\phi - J_z c_\theta s_\phi c_\phi & -J_x s_\theta \\ J_y c_\theta s_\phi c_\phi - J_z c_\theta s_\phi c_\phi & J_y c_\phi^2 + J_z s_\phi^2 & 0 \\ -J_x s_\theta & 0 & J_x \end{bmatrix} \tag{5.2}$$

where $J \in \Re^{3\times3}$ is the symmetric inertia matrix; s and c are short-hand *sin* and *cos* respectively. The relation between the Euler rates $\dot{\boldsymbol{q}}$ and the vehicle angular velocities $\boldsymbol{\omega}$ in B is given by

$$\boldsymbol{\omega} = \Lambda\ \dot{\boldsymbol{q}},\ \begin{bmatrix} \omega_r \\ \omega_q \\ \omega_p \end{bmatrix} = \begin{bmatrix} -s\theta & 0 & 1 \\ c\theta s\phi & c\phi & 0 \\ c\theta c\phi & -s\phi & 0 \end{bmatrix} \begin{bmatrix} \dot{\psi} \\ \dot{\theta} \\ \dot{\phi} \end{bmatrix}, \tag{5.3}$$

where $\dot{\boldsymbol{q}} = \Lambda^{-1}\boldsymbol{\omega}$. The main motor P_M produces a vertical thrust T_M in the Z_B axis and the tail motor P_R produces a lateral thrust T_R in the Y_B axis. The total thrust vector of the main and tail motors are $\boldsymbol{F}_M$ and $\boldsymbol{F}_R$ respectively, where

$$\boldsymbol{F}_M = \frac{|T_M|}{\sqrt{1 - s^2(a).s^2(b)}}. \begin{bmatrix} -c(a).c(b) \\ c(a).s(b) \\ -s(a).c(b) \end{bmatrix},\quad \boldsymbol{F}_R = \begin{bmatrix} 0 \\ T_R \\ 0 \end{bmatrix}, \tag{5.4}$$

where a and b are the longitudinal and lateral flapping angles respectively. The

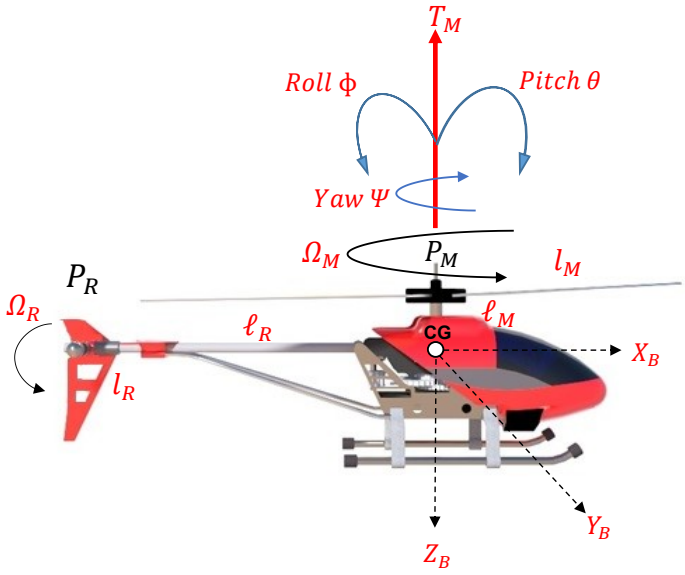


Figure 5.1: Helicopter UAV configuration

total torque vector $\boldsymbol{\tau}$ in B-frame is

$$\begin{aligned}\boldsymbol{\tau} &= \ell_M \times \boldsymbol{F}_M + \ell_R \times \boldsymbol{F}_R + \boldsymbol{E}_M - \boldsymbol{E}_R \\ &= \frac{|T_M|}{\sqrt{1 - s^2(a).s^2(b)}}. \begin{bmatrix} \ell_M^y.s(a).c(b) + \ell_M^x.c(a).s(b) \\ \ell_M^x.c(a).c(b) - \ell_M^z.s(a).c(b) \\ -\ell_M^y.c(a).c(b) - \ell_M^z.c(a).s(b) \end{bmatrix} \\ &+ \begin{bmatrix} \ell_R^x T_R \\ 0 \\ -\ell_R^z T_R \end{bmatrix} + \begin{bmatrix} |\bar{E}_M| \\ 0 \\ 0 \end{bmatrix} - \begin{bmatrix} 0 \\ |\bar{E}_R| \\ 0 \end{bmatrix},\end{aligned} \tag{5.5}$$

where the vectors ℓ_M and ℓ_R represent the length from the centre of mass of the helicopter to the hub of the main and tail motors respectively, $\boldsymbol{E}_M$ and $\boldsymbol{E}_R$ are vectors which represent the anti-torques that acting through the main and tail motors hubs which are generated from the aerodynamic drags on both motors. More details of the above derivations can be found in [25].

5.3 Control System Design

The control system is designed based on the inverse dynamics control [115] with considering parameters uncertainty of the system and external disturbances. Controller stability is illustrated by the Lyapunov second method. Considering the reference trajectory is $\boldsymbol{q}_{ref}$ and $\dot{\boldsymbol{q}}_{ref}$, the trajectory error is defined as

$$\hat{\boldsymbol{q}} = \boldsymbol{q}_{ref} - \boldsymbol{q} \tag{5.6}$$

$$\dot{\hat{\boldsymbol{q}}} = \dot{\boldsymbol{q}}_{ref} - \dot{\boldsymbol{q}} \tag{5.7}$$

$$\ddot{\hat{\boldsymbol{q}}} = \ddot{\boldsymbol{q}}_{ref} - \ddot{\boldsymbol{q}}, \tag{5.8}$$

where $\dot{\boldsymbol{q}}_{ref}$ is obtained such that

$$\dot{\boldsymbol{q}}_{ref} = K_p \, \boldsymbol{q}_{ref}, \tag{5.9}$$

and $\ddot{\boldsymbol{q}}_{ref}$ is the derivative of $\dot{\boldsymbol{q}}_{ref}$. Considering the torque $\boldsymbol{\tau}$ components are the control inputs, the nonlinear control law is defined as

$$\boldsymbol{\tau} = \tilde{H}(\boldsymbol{q})\boldsymbol{u}_c + \boldsymbol{u}_a + \tilde{D}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}}, \tag{5.10}$$

where $\tilde{H}(\boldsymbol{q})$ and $\tilde{D}(\boldsymbol{q}, \dot{\boldsymbol{q}})$ are the nominal matrices of $H(\boldsymbol{q})$ and $D(\boldsymbol{q}, \dot{\boldsymbol{q}})$ respectively. The control input $\boldsymbol{u}_c$ is defined as

$$\boldsymbol{u}_c = \ddot{\boldsymbol{q}}_{ref} + K_d \dot{\hat{\boldsymbol{q}}} + K_p \hat{\boldsymbol{q}}, \tag{5.11}$$

where $K_d = \text{diag}[K_{d1} \; K_{d2} \; K_{d3}] \in \Re^{3\times3}$ and $K_p = \text{diag}[K_{p1} \; K_{p2} \; K_{p3}] \in \Re^{3\times3}$ are positive-definite matrices. The auxiliary input $\boldsymbol{u}_a$ is added to the control law (5.10) to compensate the uncertainty and disturbances in (5.1) which will be chosen depending on the system stability. The following assumptions have been proposed to pursuit the robust control design:
1) As the helicopter actuators have limited rotational speed, the rotational Euler rates and acceleration can be bounded by positive constants $\alpha_1, \alpha_2 > 0$ such that

$$\|\dot{\boldsymbol{q}}_{ref}\| \leq \alpha_1 \tag{5.12}$$

$$\|\ddot{\boldsymbol{q}}_{ref}\| \leq \alpha_2. \tag{5.13}$$

2) The reference Euler angles are varying within limits such that

$$\|\boldsymbol{q}_{ref}\| \leq \beta. \tag{5.14}$$

3) Due to the uncertainty in moments of the inertia matrix J, it is possible to set a lower and upper bound on the Jacobian matrix $H(\boldsymbol{q})$ such that

$$\|H^{-1}(\boldsymbol{q})\| \leq \gamma_1 \tag{5.15}$$

$$\|H^{-1}(\boldsymbol{q})\| \geq \gamma_2 \tag{5.16}$$

$$\|I - H^{-1}(\boldsymbol{q})\tilde{H}(\boldsymbol{q})\| \leq \gamma_3 \tag{5.17}$$

$$\|\tilde{H}(\boldsymbol{q})\| \leq \gamma_4, \tag{5.18}$$

where $\gamma_1, \gamma_2, \gamma_3, \gamma_4 > 0$ and $I \in \Re^{3\times 3}$ is an identity matrix.
4) From (5.12) and (5.13) which require the Euler rates and acceleration to be limited and using the assumptions in (5.15)-(5.18) for the inertia matrix uncertainty and setting $\hat{D}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}}$ as the difference between the actual $D(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}}$ and nominal $\tilde{D}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}}$, the following constant bounds, $\lambda_1, \lambda_2 > 0$, are chosen as

$$\|\tilde{D}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}}\| \leq \lambda_1 \tag{5.19}$$

$$\|\hat{D}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}}\| \leq \lambda_2. \tag{5.20}$$

5) The wind disturbance vector $\boldsymbol{w}_d$ is sufficiently smooth and an upper bound $\delta > 0$ is known such that

$$\|\boldsymbol{w}_d\| \leq \delta, \tag{5.21}$$

where $\delta = \sup \|w(t)\|$; since $w(t)$ is the wind function that could violate the vehicle where its estimated superior value can be computed in practice.

The vehicle acceleration is obtained from the dynamics in (5.1) and the control law in (5.10) as

$$\ddot{\boldsymbol{q}} = H^{-1}(\boldsymbol{q})\tilde{H}(\boldsymbol{q})\boldsymbol{u}_c + H^{-1}(\boldsymbol{q})[\boldsymbol{u}_a + \boldsymbol{w}_d + \hat{D}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}}], \tag{5.22}$$

and after few simplifications, we get

$$\begin{aligned} &\ddot{\boldsymbol{q}} = H^{-1}(\boldsymbol{q})\boldsymbol{u}_a + \boldsymbol{u}_c - \boldsymbol{b}, \\ \text{where} \quad &\boldsymbol{b} = [I - H^{-1}(\boldsymbol{q})\tilde{H}(\boldsymbol{q})]\boldsymbol{u}_c - H^{-1}(\boldsymbol{q})[\boldsymbol{w}_d + \hat{D}(\boldsymbol{q},\dot{\boldsymbol{q}})\dot{\boldsymbol{q}}]. \end{aligned} \tag{5.23}$$

The error $\hat{\ddot{\boldsymbol{q}}}$ in (5.8) becomes

$$\hat{\ddot{\boldsymbol{q}}} = -K_p\hat{\boldsymbol{q}} - K_d\hat{\dot{\boldsymbol{q}}} - H^{-1}(\boldsymbol{q})\boldsymbol{u}_a + \boldsymbol{b}, \tag{5.24}$$

and in terms of the closed-loop dynamics, the above equation can be rewritten as

$$\begin{aligned} &\dot{\boldsymbol{\xi}} = A\boldsymbol{\xi} + \mathbb{I}[-H^{-1}(\boldsymbol{q})\boldsymbol{u}_a + \boldsymbol{b}], \\ \text{where} \quad &\boldsymbol{\xi} = \begin{bmatrix} \hat{\boldsymbol{q}} \\ \hat{\dot{\boldsymbol{q}}} \end{bmatrix}, \quad A = \begin{bmatrix} 0 & I \\ -K_p & -K_d \end{bmatrix}, \quad \mathbb{I} = \begin{bmatrix} 0 \\ I \end{bmatrix}. \end{aligned} \tag{5.25}$$

Choosing the candidate Lyapunov function $V(\boldsymbol{\xi}) > 0$ for $\forall \boldsymbol{\xi} \neq 0$ as

$$V(\boldsymbol{\xi}) = \boldsymbol{\xi}^T P \boldsymbol{\xi}, \tag{5.26}$$

where $V(0) = 0$ at the equilibrium point, $P \in \Re^{6\times6}$ and $Z \in \Re^{6\times6}$ are positive-definite matrices such that $-Z = A^T P + PA$. The rate of change of $V(\boldsymbol{\xi})$ with respect to the time is

$$\begin{aligned} \dot{V}(\boldsymbol{\xi}) &= \dot{\boldsymbol{\xi}}^T P\boldsymbol{\xi} + \boldsymbol{\xi}^T P\dot{\boldsymbol{\xi}} \\ &= \boldsymbol{\xi}^T[A^T P + MA]\boldsymbol{\xi} + 2\boldsymbol{\xi}^T P\mathbb{I}[-H^{-1}(\boldsymbol{q})\boldsymbol{u}_a + \boldsymbol{b}] \\ &= -\boldsymbol{\xi}^T Z\boldsymbol{\xi} + 2\boldsymbol{\eta}^T[-H^{-1}(\boldsymbol{q})\boldsymbol{u}_a + \boldsymbol{b}], \end{aligned} \tag{5.27}$$

where $\dot{\boldsymbol{\xi}}$ is given by (5.25) and $\boldsymbol{\eta} = \mathbb{I}^T P\boldsymbol{\xi}$. Equation (5.27) is required to be strictly negative to ensure system stability. The first part of (5.27) is negative-definite while the second needs to be negative since it depends on the value of $\boldsymbol{u}_a$. The auxiliary input $\boldsymbol{u}_a$ is defined as

$$\boldsymbol{u}_a = \begin{cases} \nu(\boldsymbol{\xi},t)\|\boldsymbol{\eta}\|^{-1}\boldsymbol{\eta} & \|\boldsymbol{\eta}\| \geq \varrho \\ \nu(\boldsymbol{\xi},t)\varrho^{-1}\boldsymbol{\eta} & \|\boldsymbol{\eta}\| < \varrho \end{cases} \tag{5.28}$$

where $\nu(\xi, t)$ is a time varying scalar function to be defined later and ϱ is a constant for bounding the error in $\boldsymbol{\eta}$. For $\|\boldsymbol{\eta}\| \geq \varrho$ and using (5.16), the second term $2\boldsymbol{\eta}^T[-H^{-1}(\boldsymbol{q})\boldsymbol{u}_a + \boldsymbol{b}]$ in (5.27) is bounded such that

$$2\boldsymbol{\eta}^T[-H^{-1}(\boldsymbol{q})\boldsymbol{u}_a + \boldsymbol{b}] \leq 2\|\boldsymbol{\eta}\|[-\gamma_2\nu(\boldsymbol{\xi}, t) + \|\boldsymbol{b}\|]. \tag{5.29}$$

To ensure that (5.29) is negative hence stable control, $\nu(\boldsymbol{\xi}, t)$ should be chosen such that the term $\gamma_2\nu(\boldsymbol{\xi}, t)$ is semi-positive and greater than or equal to $\|\boldsymbol{b}\|$. Thus, $\nu(\boldsymbol{\xi}, t)$ is defined depending on the superior value of the vector $\boldsymbol{b}$ such that $\|\boldsymbol{b}\| \leq \bar{b}$. From b (5.23) and $\boldsymbol{u}_c$ (5.11), we have

$$\begin{aligned}\|\boldsymbol{b}\| \leq \|I - H^{-1}(\boldsymbol{q})\tilde{H}(\boldsymbol{q})\|[\|\ddot{\boldsymbol{q}}_{ref}\| + \|\mathbb{K}\|\|\boldsymbol{\xi}\|] \\ + \|H^{-1}(\boldsymbol{q})\|[\|\boldsymbol{w}_d\| + \|\hat{D}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}}\|],\end{aligned} \tag{5.30}$$

where $\mathbb{K} = [K_p \; K_d]^T \in \Re^{3\times 6}$ and $\boldsymbol{\xi}$ (5.25). Recalling the assumptions (5.13)-(5.21), we have $\bar{b}$ as

$$\|\boldsymbol{b}\| \leq \gamma_3[\alpha_2 + \|\mathbb{K}\|\|\boldsymbol{\xi}\|] + \gamma_1[\delta + \lambda_2] := \bar{b}. \tag{5.31}$$

From (5.29) where the stability condition should be $\gamma_2\nu(\boldsymbol{\xi}, t) \geq \bar{b}$ and using (5.31), the scalar function $\nu(\boldsymbol{\xi}, t)$ is given by

$$\nu(\boldsymbol{\xi}, t) \geq \gamma_2^{-1}\bar{b} = \gamma_3\gamma_2^{-1}[\alpha_2 + \|\mathbb{K}\|\|\boldsymbol{\xi}\|] + \gamma_1\gamma_2^{-1}[\delta + \lambda_2]. \tag{5.32}$$

Note that $\nu(\boldsymbol{\xi}, t)$ is time dependent because it is relying on the error $\boldsymbol{\xi}$ which varies with time; since $\boldsymbol{\xi}(t)$ is written as $\boldsymbol{\xi}$ for clarity. Finally, the asymptotic stability is guaranteed since by substituting $\boldsymbol{u}_a$ (5.28) (for $\|\boldsymbol{\eta}\| \geq \varrho$) in equation (5.27), we get

$$\dot{V}(\boldsymbol{\xi}) = -\boldsymbol{\xi}^T Z\boldsymbol{\xi} + 2\boldsymbol{\eta}^T[-H^{-1}(\boldsymbol{q})[\nu(\boldsymbol{\xi}, t)\|\boldsymbol{\eta}\|^{-1}\boldsymbol{\eta}] + \boldsymbol{b}] < 0, \tag{5.33}$$

and for $\|\boldsymbol{\eta}\| < \varrho$,

$$\dot{V}(\boldsymbol{\xi}) = -\boldsymbol{\xi}^T Z\boldsymbol{\xi} + 2\boldsymbol{\eta}^T[-H^{-1}(\boldsymbol{q})[\nu(\boldsymbol{\xi}, t)\varrho^{-1}\boldsymbol{\eta}] + \boldsymbol{b}] < 0. \tag{5.34}$$

The following section will illustrate how to determine a robust invariant set of the designed controller which will be used in the verification process to show that all the system trajectories will stay within this set; hence ensure controller stability and robustness.

5.4 Handling of Constraints

This section finds the dynamical state vectors that include the attitude and rotation rates, for which the stable control of the craft is feasible. The control scheme is obtained by suitably chosen references for the guidance derivatives under the constraints of the current state of attitude error and reference for the rotation rate and the current attitude itself. First, the state set is defined where a feasible control input exists under the rotate per minute (rpm) limitations of the motors of the helicopter. Then the state evolution within the set will be verified as illustrate in Section 5.6.

Definition 5.1. *Let* $\boldsymbol{x} = [\boldsymbol{q}\ \dot{\boldsymbol{q}}]^T$*, the helicopter rotational dynamics is defined by*

$$\dot{\boldsymbol{x}} = \begin{bmatrix} \dot{\boldsymbol{q}} \\ \ddot{\boldsymbol{q}} \end{bmatrix} = \begin{bmatrix} \Lambda^{-1}\boldsymbol{\omega} \\ H^{-1}(\boldsymbol{q})\boldsymbol{u}_a + \boldsymbol{u}_c - \boldsymbol{b} \end{bmatrix}, \tag{5.35}$$

and a robust invariant set $\boldsymbol{S}(.) \subset \Re^6$ *is called a* control enabled set*, if for any* $x \in \boldsymbol{S}(.)$ *at current time* t_c *there are continuous guidance functions* $\dot{\boldsymbol{q}}_{ref}$*,* $\ddot{\boldsymbol{q}}_{ref}$ *for any* $t > t_c$*, such that* $\boldsymbol{u}_c$ (5.11) *is realisable by the motors of the vehicle under the constraints of the torque* $\boldsymbol{\tau}$ (5.10) *while considering the constraints of: 1) the torque vector* $\boldsymbol{\tau}$ (5.5) *due to the limits of the thrusts,* T_M *and* T_R *and the flapping angles, a and b; 2) the main motor angular velocity* $0 < \Omega_M < \Omega_M^{max}$*, and rear motor angular velocity* $0 < \Omega_R < \Omega_R^{max}$*; 3) assumptions* (5.12)-(5.21)*.*

The control enabled set can be numerically computed for various values of their guidance parameters $\boldsymbol{q}_{ref}$ and $\dot{\boldsymbol{q}}_{ref}$ with the constraints α_1, α_2, γ_1,γ_2, γ_3, γ_4, λ_1, λ_2, κ_1, κ_2, β and δ. Under the rpm and flapping angles constraints of (5.5), all possible vectors $\boldsymbol{\tau}$ are in a convex set Ψ. The polytope Ψ is reduced due to the bounds of the constraints (5.12)-(5.21) and transformed by feasible values of

$\tilde{H}(\boldsymbol{q})$ and $\tilde{D}(\boldsymbol{q},\dot{\boldsymbol{q}})\dot{\boldsymbol{q}}$ to result in a polytope Ξ for the possible values of $\boldsymbol{u}_c$. Then, for fixed $\dot{\boldsymbol{q}}_{ref}$ and $\ddot{\boldsymbol{q}}_{ref}$ the set of x for which the $\boldsymbol{u}_a \in \Xi$ can be derived as by definition: $\hat{\dot{\boldsymbol{q}}} = \dot{\boldsymbol{q}}_{ref} - \dot{\boldsymbol{q}}$.

Theorem 5.1. *Assuming (5.33) and (5.34) are verified to be satisfied over a control enabled set $\boldsymbol{S}(.) \subset \Re^6$, then the state evolution of $\boldsymbol{x} = [\boldsymbol{q}\ \dot{\boldsymbol{q}}]^T$ defined by $\ddot{\boldsymbol{q}}$ in (5.23) remains in $\boldsymbol{S}(.)$ for any $\|\tilde{H}(\boldsymbol{q})\| \leq \gamma_4$, $\|\hat{D}(\boldsymbol{q},\dot{\boldsymbol{q}})\dot{\boldsymbol{q}}\| \leq \lambda_2$ and $\|\boldsymbol{w}_d\| \leq \delta$, $t > t_c$, with the controller as defined by (5.10) with the constraints of $\boldsymbol{\tau}$ in (5.5) and a suitable choice of adapted references $\dot{\boldsymbol{q}}_{ref}$ and $\ddot{\boldsymbol{q}}_{ref}$.*

Proof. *From the constraints of both motors $0 < \Omega_M < \Omega_M^{max}$ and $0 < \Omega_R < \Omega_R^{max}$ and (5.5), an upper limit of the torque $\boldsymbol{\tau}$ can be computed such that*

$$\|\boldsymbol{\tau}\| \leq \tau_{max}. \tag{5.36}$$

Referring to the control law in (5.10), $\boldsymbol{u}_a$ (5.28), and the assumptions (5.12)-(5.21) and (5.36), we get

$$\begin{aligned} \|\boldsymbol{\tau}\| &\leq \|\tilde{H}(\boldsymbol{q})\|\|\boldsymbol{u}_c\| + \|\boldsymbol{u}_a\| + \|\tilde{D}(\boldsymbol{q},\dot{\boldsymbol{q}})\dot{\boldsymbol{q}}\| \\ \|\tilde{H}(\boldsymbol{q})\|\|\boldsymbol{u}_c\| + \|\boldsymbol{u}_a\| + \|\tilde{D}(\boldsymbol{q},\dot{\boldsymbol{q}})\dot{\boldsymbol{q}}\| &\leq \tau_{max} \\ \|\boldsymbol{u}_c\| &\leq (\tau_{max} - \|\boldsymbol{u}_a\| - \|\tilde{D}(\boldsymbol{q},\dot{\boldsymbol{q}})\dot{\boldsymbol{q}}\|)/\|\tilde{H}(\boldsymbol{q})\| \\ &\leq (\tau_{max} - \nu(\boldsymbol{\xi},t) - \lambda_1)/\gamma_4, \end{aligned} \tag{5.37}$$

then taking (5.32), we have

$$-\nu(\boldsymbol{\xi},t) \leq -\gamma_3\gamma_2^{-1}(\alpha_2 + \|\mathbb{K}\|\|\boldsymbol{\xi}\|) - \gamma_1\gamma_2^{-1}(\delta + \lambda_2). \tag{5.38}$$

Recalling $\boldsymbol{u}_c$ from definition (5.11) and the assumption (5.13),

$$\begin{aligned} -\|\boldsymbol{u}_c\| &\geq -(\alpha_2 + \|K_d\|\|\hat{\dot{\boldsymbol{q}}}\| + \|K_p\|\|\hat{\boldsymbol{q}}\|) \\ &\geq -(\alpha_2 + \|\mathbb{K}\|\|\boldsymbol{\xi}\|), \end{aligned} \tag{5.39}$$

then substituting (5.39) in (5.38), we get

$$-\nu(\boldsymbol{\xi},t) \leq -\gamma_3\gamma_2^{-1}\|\boldsymbol{u}_c\| - \gamma_1\gamma_2^{-1}(\delta + \lambda_2). \tag{5.40}$$

Substituting (5.40) (5.37)*, the maximum control input* $u_{c_{max}}$ *is obtained*

$$\begin{aligned}\|\boldsymbol{u}_c\| &\leq (\tau_{max} - \gamma_3\gamma_2^{-1}\|\boldsymbol{u}_c\| - \gamma_1\gamma_2^{-1}(\delta + \lambda_2) - \lambda_1)/\gamma_4 \\ &\leq (\tau_{max} - \gamma_1\gamma_2^{-1}(\delta + \lambda_2) - \lambda_1)/(\gamma_2^{-1}\gamma_3 + \gamma_4) := u_{c_{max}}.\end{aligned} \tag{5.41}$$

As the upper bound $u_{c_{max}}$ *of the input* $\boldsymbol{u}_c$ *is now known, from* (5.11), (5.13) *and* (5.41) *we have*

$$\begin{aligned}\|\boldsymbol{u}_c\| &\leq \|\ddot{\boldsymbol{q}}_{ref}\| + \|K_d\|\|\dot{\hat{\boldsymbol{q}}}\| + \|K_p\|\|\hat{\boldsymbol{q}}\| \\ \|\ddot{\boldsymbol{q}}_{ref}\| + \|K_d\|\|\dot{\hat{\boldsymbol{q}}}\| + \|K_p\|\|\hat{\boldsymbol{q}}\| &\leq u_{c_{max}} \\ \alpha_2 + \kappa_1\|\dot{\hat{\boldsymbol{q}}}\| + \kappa_2\|\hat{\boldsymbol{q}}\| &\leq u_{c_{max}},\end{aligned} \tag{5.42}$$

where $\|K_d\| \leq \kappa_1$ *and* $\|K_p\| \leq \kappa_2$ *with* $\kappa_1, \kappa_2 > 0$*. Recalling* (5.6), (5.7), (5.12) *and* (5.14)*, we get*

$$\kappa_2\|\boldsymbol{q}\| + \kappa_1\|\dot{\boldsymbol{q}}\| \leq u_{c_{max}} - \alpha_2 - \kappa_1\alpha_1 - \kappa_2\beta. \tag{5.43}$$

Finally, the control enabled set is obtained as

$$\boldsymbol{S}(.) = \{\forall[\boldsymbol{q}\ \dot{\boldsymbol{q}}]^T.\ \kappa_2\|\boldsymbol{q}\| + \kappa_1\|\dot{\boldsymbol{q}}\| \leq u_{c_{max}} - \alpha_2 - \kappa_1\alpha_1 - \kappa_2\beta\}. \tag{5.44}$$

The next section illustrates the application of these results in Simulink/Matlab.

5.5 Simulation

The nonlinear attitude dynamics (5.1) and the designed controller are implemented in Simulink/Matlab for simulation and obtaining numerical parameters required for the verification process. As the some functions like the time derivative is not supported by the MetiTarski prover, the control inputs in $\boldsymbol{\tau}$ and system states $\boldsymbol{q}, \dot{\boldsymbol{q}}$ with parameters are passed from simulation in Simulink to the MetiTarski prover for verification. The simulation is based on a VARIO Benzin-Trainer unmanned helicopter. VARIO numerical parameters are shown in Table 5.1(a). According to the maximum payload of the VARIO helicopter which is approximately $4kg$, the amount of variation of the inertia moments are com-

puted. External disturbances are assumed to vary within a maximum 40% of the maximum torque. Table 5.1(b) states the computed robust parameters of the controller.

The controller results are shown in Fig. 5.2. As can be seen from the step response reference, the roll, pitch and yaw angles are following the reference. The dot-dashed red line shows the measured roll, pitch and yaw with minimum inertia moment values. The dashed green line shows the measured roll, pitch and yaw with maximum inertia moment, where the minimum/maximum values of the inertia moments J are illustrated in Table 5.1(a). It can be noticed from the difference between minimum and maximum inertia moments signals that the the variation of inertia moments has a minor effect using the proposed controller. External disturbances are applied to the signals to test the controller performance. The highest amount of disturbances applied are up to 40% of the maximum torque. The tracking of the reference (continuous blue line) under disturbances are well performed by the controller. The next section illustrates the verification results.

5.6 Control Verification

To ensure the validity of control scheme, the following verification objectives are required to be satisfied: 1) the controller produces torques which are with the maximum torques limits: $\|\tilde{H}(\boldsymbol{q})\boldsymbol{u}_c + \boldsymbol{u}_a + \tilde{D}(\boldsymbol{q},\dot{\boldsymbol{q}})\dot{\boldsymbol{q}}\| \leq \tau_{max}$; 2) the system is stable: $\dot{V}(\boldsymbol{\xi}) < 0$ (5.33) and (5.34); 3) all system states are vary and stay within the control enabled set: $\forall \boldsymbol{x} = [\boldsymbol{q}\ \dot{\boldsymbol{q}}]^T$. $\boldsymbol{x} \in \mathbf{S}(.)$ where $\mathbf{S}(.)$ is defined in (5.44).

Remark 1. *If the translational control is designed then the limits of flapping angles a and b with thrusts T_M and T_R can be checked according to the produce control torques $\boldsymbol{\tau}$.*

MetiTarski ATP is used as a verification tool to prove the above objectives. As the prover is limited to work on real scalar values, all vectors and matrices are simplified to scalar statements. Due to the space limit, only examples of the proof will be presented in this section while the complete code with proofs can be found in the web-repository [2]. Variables notated in MetiTarski is shown in

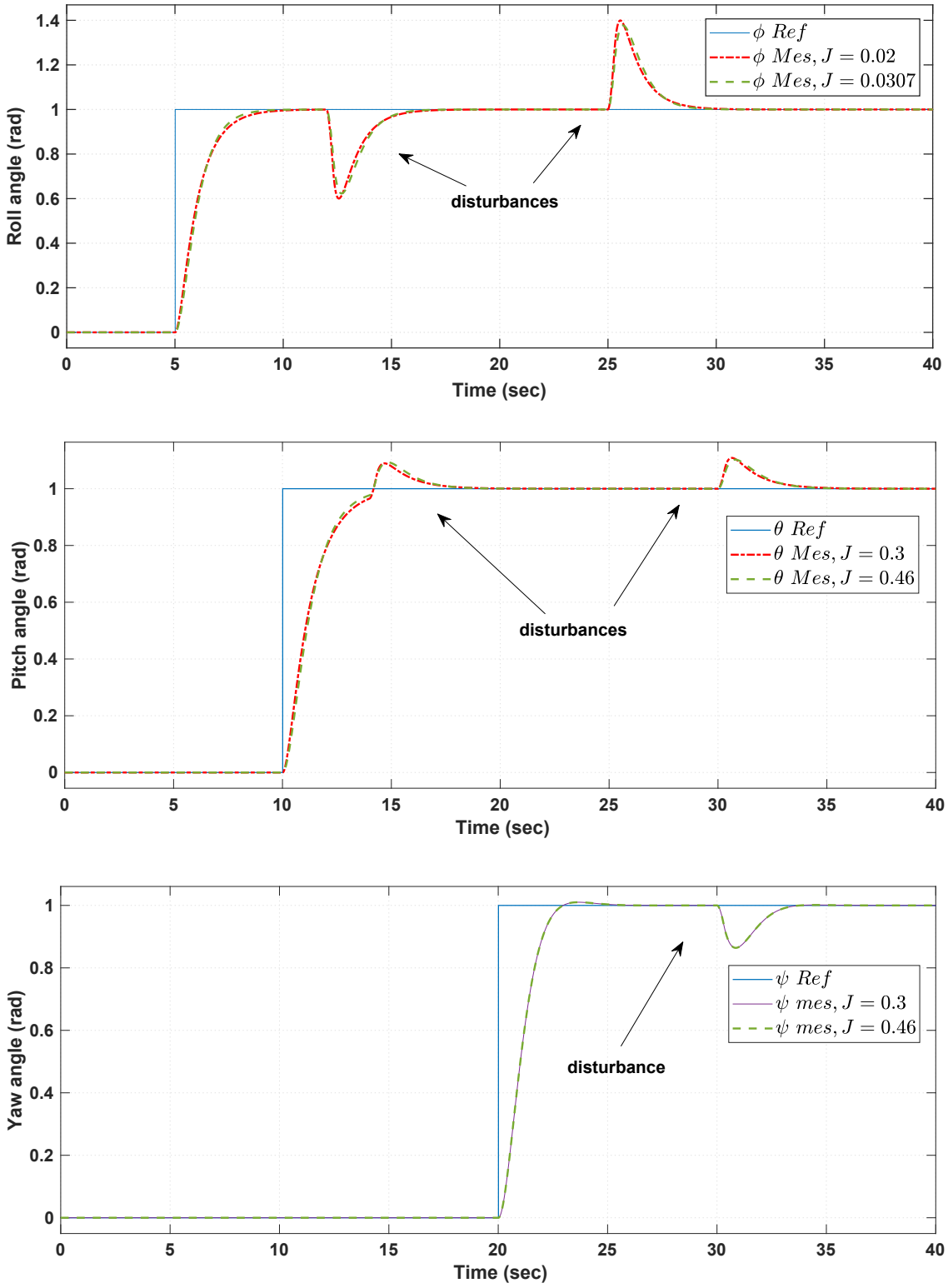


Figure 5.2: Euler angles with disturbances

Table 5.1: Small helicopter UAV parameters and constraints

(a) Aircraft parameters

Parameter	***Value***
m	$7.5\ kg$
l_M	$1.8\ m$
l_R	$0.3\ m$
J_{xx}	$[0.02,\ 0.0307]\ k.gm^2$
J_{yy}	$[0.3,\ 0.46]\ k.gm^2$
J_{zz}	$[0.3,\ 0.46]\ k.gm^2$
ℓ_M^z	$-0.25\ m$
ℓ_R^x	$-0.75\ m$
ℓ_R^y	$-0.05\ m$
$\ell_M^x, \ell_M^y, \ell_R^z$	0
Ω_M^{max}	$132.9941\ rad/s$
Ω_R^{max}	$580.4989\ rad/s$
T_M^{max}	$135.7143\ N$
T_R^{max}	$2.4\ N$
E_M	$0.02\|T_M\|$
E_R	$0.02\|T_R\|$
a	$[-12°, 12°]$
b	$[-14°, 14°]$

(b) Control parameters

Parameter	***Value***
α_1	3.2657
α_2	2.1482
γ_1	60.8276
γ_2	39.0872
γ_3	0.5353
γ_4	0.46
λ_1	0.3337
λ_2	0.1161
β	3.4452
δ	4.2281
κ_1	0.135
κ_2	0.9
$u_{c_{max}}$	7.3385
τ_{max}	10.5703
K_{p1}	0.88
K_{p2}	0.8
K_{p3}	0.9
K_{d1}	0.0013
K_{d2}	0.12
K_{d3}	0.135
ρ	0.5

Table (5.2). The first objective is achieved by taking the torques produce from the controller in (5.10) and $\tau_{i_{max}}$ limits by (5.5) for each element (τ_i) then the objective inequality is formalized in FOL syntax and proved as below: (note that this code is for $|\tau_\psi| \leq \tau_{\psi_{max}}$ only; see the web-repository [2] for other codes)

MetiTarski code

```
fof(Torque_psi,conjecture, ! [T_psi,TM,TR,A,B,Lx_M,Ly_M]:
%assumptions
(T_psi:(=-0.0546,0.581=) & TM=135.7143 & TR=2.4 & A>= -0.2094 & A<=0.2094
& B>=-0.2443 & B<=0.2443 & Lx_M=0 & Ly_M=0
```

```
%implies
=>abs(T_psi)<=(abs(TM)/sqrt(1-sin(A)^2*sin(B)^2))*((Ly_M*sin(A)*cos(B)) +
 (Lx_M*cos(A)*sin(B)))-0.75*TR+0.02*TM)).
```

The implementation of the torque vector τ in MetiTarski is achieved by taking each of the torque vector's element $(\tau_\phi, \tau_\theta, \tau_\psi)$ and prove it separately, where this was due to the limitation of implementing the vectors in the prover. Therefore, Symbolic Math Toolbox in Matlab has been used to simplify the torque vector. Another limitation is that in the prover there is no derivative function to compute the time derivative of the torque vector components. Due to this, the time derivative values have been taken from the simulation to conduct the proof. The longitudinal and lateral flapping angles (a and b) have been implemented in the above code with upper and lower bounds of their values. The τ_ψ value also limited by an upper and lower value according to the actuator limits in (5.5) to ensure that the values of τ_ψ varies within the limits. The time required for MetiTarski prover to generate the proof of the above code is 0.656 seconds. Note that all proofs in MetiTarski were conducted on a Linux Ubuntu operating system, Core i5 1.6 GHz CPU and 8 GB RAM.

The second objective is achieved by formalizing equations (5.33), (5.34) and proving that they are strictly negative for all states under the proposed control scheme. The following code illustrates a part of the stability implementation of (5.33). Symbolic Math Toolbox in Matlab has been used to simplify (5.33) and (5.34). The roll ϕ and pitch θ angles are limited by upper and lower bound to avoid any aggressive rotation that violate the flight stability. The time required for MetiTarski prover to generate the proof of the following code is 0.104 seconds.

MetiTarski code

```
fof(Stability_33,conjecture, ![V,Phi,Theta,Bb_1,Bb_2,Bb_3] :?[Xi1,Xi2,Xi3,
Xi4,Xi5,Xi6]:
%assumptions
(Xi1>0 & Xi1<=1 & Xi2>0 & Xi2<=1 & Xi3!=0 & Xi3:(=-0.0622,1=) & Xi4 !=0
& Xi4:(=-0.0897,0.88=) & Xi5 !=0 & Xi5:(=-0.0394,0.8=) & Xi6!=0
&  Xi6:(=-0.2419,0.9=) & Phi:(=-1,1=) & Theta:(=-1,1=)
& V:(=13.5797,13.6077=) & Bb_1:(=-7.0404*10^(-18),7.5358*10^(-17)=)
& Bb_2:(=-9.8665 *10^(-17), 1.0821*10^(-16)=) & Bb_3:(=-9.8665*10^(-17),
```

```
1.0821*10^(-16)=)
%implies
=>(...<0)).
```

The code below shows formalising of the third objective which is proven by considering the upper and lower variation of the system states $q, \dot{q}$ is complying to the upper bound specified in (5.44). Symbolic Math Toolbox in Matlab has been used to simplify the set in (5.44) and the time derivative values of $\boldsymbol{q}$ are taken from the simulation. The time required for MetiTarski prover to generate the proof of the below code was 3.784 seconds.

MetiTarski code

```
fof(Helicopter_control_enabled_set,conjecture, ![Q1,Q2,Q3,Dot_Q1,Dot_Q2,
Dot_Q3]:
% assumptions
(Q1 >= 0 & Q1 <= 1.0271 & Q2 >= 0 & Q2 <= 1 & Q3 >= 0 & Q3 <= 0.8
& Dot_Q1 >= -0.0735 & Dot_Q1 <= 0.6993 & Dot_Q2 >= 0 & Dot_Q2 <= 0.5933
& Dot_Q3 >= -0.0952 & Dot_Q3 <= 0.7229
% implies
=> ((0.9*sqrt(Q1^2+Q2^2+Q3^2))+(0.135*sqrt(Dot_Q1^2+Dot_Q2^2+Dot_Q3^2))
   <= 1.6487))).
```

The interactive approach between the simulation and the prover was useful since several unproved statements are resolved by retuning the parameters.

5.7 Discussion and Remarks

The proposed approach can be applied to different UAV systems as it is useful in two aspects: control design verification and onboard real-time validation. At the design stage, ensuring controller performance, robustness and stability is essential under physical limitations. This safety analysis cannot be achieved by simulation only as it relies on numerical computations as well as on co-simulation verification with symbolic computations. Regarding the control design, although many control schemes have been proposed in the literature, they are either taking into

Table 5.2: Variables and vectors notations in MetiTarski

Variable/Vector	***Notation***
τ_ψ	T_psi
T_M	TM
T_R	TR
a	A
b	B
ℓ_M^x	Lx_M
ℓ_M^y	Ly_M
$\boldsymbol{\xi}$	Xi
$v(\boldsymbol{\xi}, t)$	V
ϕ	Phi
θ	$Theta$
$\boldsymbol{b}(i)$	Bb_i
$\boldsymbol{q}$	Q
$\dot{\boldsymbol{q}}$	Dot_Q

account parameters uncertainty or disturbances without considering both in the robust design, which are important factors together that affect control system performance. Therefore, we consider both factors in addition to taking into account dynamical actuator constraints based on practical parameters. For control verification, several attempts have been proposed to verify simple control systems such as in [15, 28, 43, 65], and for hybrid verification systems as in [11]. These approaches have been developed based on interactive theorem provers, which need interaction with humans to complete proofs. Other approaches with the MetiTarski prover have been only used at the design stage. However, the remaining issue is how the autopilot knows whether the aircraft's dynamical envelope is violated by external forces such as gusts of wind. Therefore, it has proposed to integrate the MetiTarski ATP with the autopilot system and to perform a real-time verification using the proposed approach as presented. Based on this approach, the autopilot can make decisions based on information from an onboard prover, which can send a warning to perform an emergency landing.

5.8 Chapter Summary

A robust nonlinear attitude controller is presented for a small unmanned helicopter UAVs. Controller stability is demonstrated and verified using formal methods represented by MetiTarski ATP. The control system parameters constraints are computed and system states are verified to be vary within the defined invariant control set. A verification method is proposed by merging the autopilot system with MetiTarski prover. The method is demonstrated in simulation and MetiTarski is used to illustrate its applicability. The method is useful in particular when the vehicle is in a fully autonomous flight. If the controller performance is endangered by gusts of winds beyond its reaction abilities, then the autopilot could perform an emergency landing in a safe place.

Chapter 6

A Robust Controller for Multi Rotor Unmanned Aerial Vehicles

6.1 Overview

Unmanned aerial vehicles are safety-critical systems that often need to fly near buildings and over people under adverse wind conditions and hence require high manoeuvrability, accuracy, fast response abilities to ensure safety. Under extreme conditions, the dynamics of these systems are strongly nonlinear and are exposed to disturbances, which need a robust controller to keep the UAV and its environment safe. In this chapter, a novel robust nonlinear multi-rotor controller (RNDI) is introduced based on essential modifications of the standard dynamic inversion control [111, 115], which makes it insensitive to payload changes and also to large wind gusts. First a robust attitude controller is introduced, followed by lateral and vertical position control in a customary outer loop. The controllers take into account thrust limitations of the UAV and a mathematical proof is provided for robust performance. The control scheme is illustrated in simulation with a realistic nonlinear dynamical model of an UAV that includes rotor dynamics and their speed limitations to show robustness. Lyapunov stability methods are used to prove the stability of the robust control system.

6.1.1 Multi Rotor Dynamic Model

This section introduces the dynamical model of a generic multi-rotor using the quaternions (described in Section 2.1.2.3) to avoid the singularity associated with the gimbal lock [121], which is important in high-performance control. The multi-rotor translational dynamics in the B-frame using a Newton equation is

$$m\dot{\boldsymbol{v}} + \Gamma(\boldsymbol{\omega})m\boldsymbol{v} = R_q^T \boldsymbol{f}_G + \boldsymbol{f}_B, \tag{6.1}$$

where $m \in \Re$ is the total mass of the craft, $\boldsymbol{v}(t) = [v_x(t)\ v_y(t)\ v_z(t)]^T \in \Re^3$ is the velocity vector of mass centre, $\dot{\boldsymbol{v}}(t) = [\dot{v}_x(t)\ \dot{v}_y(t)\ \dot{v}_z(t)]^T \in \Re^3$ is the acceleration vector, $\boldsymbol{f}_G = [0\ 0\ -mg]^T$ is the gravitational force, $\boldsymbol{f}_B = [0\ 0\ U]^T \in \Re^3$ is the total force of thrusters, $U = F_1 + F_2 + F_3 + F_4$ (where $F_{1,2,3,4}$ are the propellers forces), and $\Gamma(\boldsymbol{\omega}) \in \Re^{3\times3}$ is the cross-product matrix for the Coriolis forces presented in (2.7). The dynamics in the world frame, W-frame, is then given by

$$\ddot{\boldsymbol{r}} = \frac{1}{m}(\boldsymbol{f}_G + R_q \boldsymbol{f}_B), \tag{6.2}$$

where $\boldsymbol{r}(t) = [x(t)\ y(t)\ z(t)]^T \in \Re^3$ is the position vector in W-frame; since $\ddot{\boldsymbol{r}} = R_q\dot{\boldsymbol{v}}$ with R_q as in (2.27). The multi-rotor rotational dynamics in the B-frame, using a Newton-Euler equation described in (2.9), is

$$I\dot{\boldsymbol{\omega}} + \Gamma(\boldsymbol{\omega})I\boldsymbol{\omega} + \boldsymbol{\tau}_d = \boldsymbol{\tau}, \tag{6.3}$$

where $\boldsymbol{\tau}_d(t) = [\tau_{d\phi}(t)\ \tau_{d\theta}(t)\ \tau_{d\psi}(t)]^T \in \Re^3$ are the unknown disturbances torques with ϕ, θ and ψ are roll, pitch and yaw respectively.

It is assumed that for the multi-rotor each motor is aligned with the vertical main axis of the vehicle and has the angular velocity Ω_i that produces body-aligned forces $F_i = l\Omega_i^2$ and a torques $M_i = b\Omega_i^2$ where l and b are the aerodynamic force and torque constants of the rotors. All angular velocities of the motors are bounded by a known maximum value Ω_{max} so that, $|\Omega_i| < \Omega_{max}$.

The torque output of the onboard control system, $\boldsymbol{\tau}$, for plus-configuration is

$$\boldsymbol{\tau} = \begin{bmatrix} \tau_\phi \\ \tau_\theta \\ \tau_\psi \end{bmatrix} = \begin{bmatrix} \ell l(\Omega_2^2 - \Omega_4^2) \\ \ell l(-\Omega_1^2 + \Omega_3^2) \\ b(-\Omega_1^2 + \Omega_2^2 - \Omega_3^2 + \Omega_4^2) \end{bmatrix}, \tag{6.4}$$

where ℓ is the length from the centre of mass of the multi-rotor to the rotor. For an X-configuration, where propellers 1-2 are on the front, these equations are modified to

$$\boldsymbol{\tau} = \begin{bmatrix} \tau_\phi \\ \tau_\theta \\ \tau_\psi \end{bmatrix} = \begin{bmatrix} \ell l(-\Omega_1^2 + \Omega_2^2 + \Omega_3^2 - \Omega_4^2)/\sqrt{2} \\ \ell l(-\Omega_1^2 - \Omega_2^2 + \Omega_3^2 + \Omega_4^2)/\sqrt{2} \\ b(-\Omega_1^2 + \Omega_2^2 - \Omega_3^2 + \Omega_4^2) \end{bmatrix}, \tag{6.5}$$

For a hexacopter one of the options is, where propellers 1-2 are on the front, to have the attitude control torques generated by

$$\boldsymbol{\tau} = \begin{bmatrix} \tau_\phi \\ \tau_\theta \\ \tau_\psi \end{bmatrix} = \begin{bmatrix} \ell l(-\Omega_1^2/2 + \Omega_2^2/2 + \Omega_3^2 + \Omega_4^2/2 - \Omega_5^2/2 - \Omega_6^2) \\ \ell l(-\Omega_1^2 - \Omega_2^2 + \Omega_4^2 + \Omega_5^2)\sqrt{3}/2 \\ b(-\Omega_1^2 + \Omega_2^2 - \Omega_3^2 + \Omega_4^2 - \Omega_5^2 + \Omega_6^2) \end{bmatrix}, \tag{6.6}$$

The torques can be modelled in a similar manner for other types of mult-irotor configurations, which are out of the scope of this work.

For all cases of multi-rotors, from (6.3), and denoting by $c(\boldsymbol{\omega}) = \Gamma(\boldsymbol{\omega})I\boldsymbol{\omega}$ the torque generated by the rotational moments, the attitude state-space equation derives from

$$\dot{\boldsymbol{\omega}} = I^{-1}[\boldsymbol{\tau} - \boldsymbol{c}(\boldsymbol{\omega}) - \boldsymbol{\tau}_d]. \tag{6.7}$$

6.2 Control System Design

The nonlinear rotational dynamics, when combined with minor inaccuracies in rotor shaft alignments and propeller deficiencies can lead to errors in actuated control torques, the effect of which can be eliminated by an inner-loop feedback controller of the multi-rotor attitude. The same attitude controller can also be used to compensate for external disturbances of wind gusts, aerodynamic interactions with nearby structures and ground effects. Fig. 6.1 shows the proposed

control system. The designed control scheme consists of two loops: the inner and outer loop. The inner loop is a nonlinear robust attitude controller based on dynamic inversion control method which compute the torque vector $\boldsymbol{\tau}$ from the reference quaternions $\boldsymbol{q}_r$. The robust dynamic inversion control scheme is used to overcome the modeling uncertainty and disturbances associated with the rotational movements. This loop is embedded in an outer feedback loop to control lateral and vertical movements. The outer loop includes a PD controller to compute the desired roll and pitch angles in a form of quaternions $\boldsymbol{q}_r$ from the reference position $\boldsymbol{r}_r$ and reference yaw angle $\boldsymbol{\Psi}$ for manoeuvre-goals in the $x,\ y,\ z$ lateral position frame.

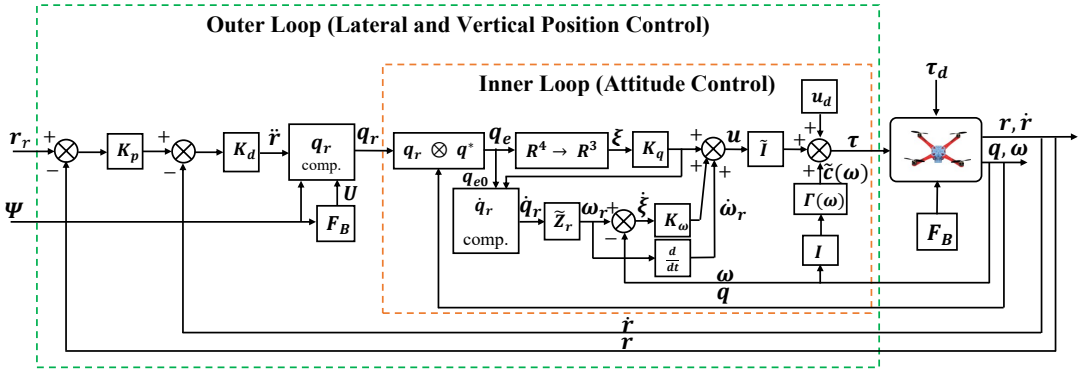


Figure 6.1: The inner and outer control loops of the proposed multi-rotor controller. The notation is explained through equations (6.10)-(6.24).

6.2.1 Position Control

Multi-rotor lateral transition is obtained by tilting the vehicle around the X-axis by (q_0, q_1) and Y-axis by (q_0, q_2) for the quadrotor illustrated in Fig.4.1. These angles are computed based on the reference trajectory of the position controller, which passes them to the inner attitude controller. However, the outer feedback position control loop is chosen as cascaded $P(x), P(y)$ controllers to handle the $\dot{x}$ and $\dot{y}$. Another cascaded $P(z)$ controller is also chosen to control $\dot{z}$ and hence obtaining the required linear movement.

Given the reference trajectory vector $\boldsymbol{r}_r(t) = [x_r(t)\ y_r(t)\ z_r(t)]^T \in \Re^3$ and $q_{r_3}(t)$

as in (2.27) and (6.2) while keeping $\|\boldsymbol{q}\| = 1$, the quaternion reference $\boldsymbol{q}_r$ is computed as

$$\boldsymbol{q}_r = \begin{bmatrix} q_{r_0} \\ q_{r_1} \\ q_{r_2} \\ q_{r_3} \end{bmatrix} = \begin{bmatrix} [(\ddot{z}+g)/(2[\ddot{x}^2+\ddot{y}^2+\ddot{z}^2+2g\ddot{z}+g^2]^{\frac{1}{2}}) + 0.5 - q_{r_3}^2]^{\frac{1}{2}} \\ [m(\ddot{x}q_{r_3} - \ddot{y}q_{r_0})]/[2U(q_{r_0}^2+q_{r_3}^2)] \\ [m(\ddot{x}q_{r_0} + \ddot{y}q_{r_3})]/[2U(q_{r_0}^2+q_{r_3}^2)] \\ q_{r_3} \end{bmatrix}. \tag{6.8}$$

The force $\boldsymbol{f}_B$ including the total thrust U is computed for vehicle altitude control as

$$\boldsymbol{f}_B = \begin{bmatrix} 0 \\ 0 \\ U \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ (m\ddot{z}+mg)/(2q_0^2+2q_3^2-1) \end{bmatrix}. \tag{6.9}$$

Definition 6.1. *The translational motion is controlled by choosing*

$$\ddot{\boldsymbol{r}} = K_d(K_p(\boldsymbol{r}_r - \boldsymbol{r}) - \dot{\boldsymbol{r}}), \tag{6.10}$$

or in terms of components

$$\ddot{\boldsymbol{r}} = \begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix} = \begin{bmatrix} K_{dx}(K_{px}(x_r - x) - \dot{x}) \\ K_{dy}(K_{py}(y_r - y) - \dot{y}) \\ K_{dz}(K_{pz}(z_r - z) - \dot{z}) \end{bmatrix}, \tag{6.11}$$

where $K_p = diag[K_{px}\ K_{py}\ K_{pz}]^T \in \Re^{3\times3}$ *and* $K_d = diag[K_{dx}\ K_{dy}\ K_{dz}]^T \in \Re^{3\times3}$ *are positive-definite diagonal gain matrices.*

The controller represented in (6.10) is implemented using (6.8) to get the quaternion reference required for the multi-rotor attitude control and using (6.9) to compute the total amount of thrust, U.

6.2.2 Attitude Control

The nonlinear control system is designed based on the dynamic inversion control principle [115], for controlling the multi-rotor attitude while accounting for the bounded but uncertain mass distribution of the UAV and external force and

torque disturbances. Lyapunov's method will be used to prove asymptotic stability under these bounded uncertainties for the control system defined as follows.

Definition 6.2. ***Controller Torque Computation.*** *Based on the attitude dynamics in (6.3), the nonlinear control law is defined by*

$$\boldsymbol{\tau} = \hat{I}\boldsymbol{u} + \boldsymbol{u}_d + \hat{\boldsymbol{c}}(\boldsymbol{\omega}). \tag{6.12}$$

where $\hat{I}$ is an estimation of the inertia matrix I of the craft, $\boldsymbol{u}$ represents a new input vector to be designed later on in (6.19), $\hat{\boldsymbol{c}}(\boldsymbol{\omega})$ is an estimate of $\boldsymbol{c}(\boldsymbol{\omega})$ as based on $\hat{I}$ and measured $\boldsymbol{\omega}$. The additional term $\boldsymbol{u}_d$ is added to render the effects of uncertainty and disturbances in addition to guarantee robustness of these effects; $\boldsymbol{u}_d$ will be defined later to counter these effects in (6.36).

Suppose that the attitude reference is $\boldsymbol{q}_r$ and the measured value is $\boldsymbol{q}$, the quaternion error $\boldsymbol{q}_e$ will be defined by

$$\boldsymbol{q}_e = \boldsymbol{q}_r \otimes \boldsymbol{q}^*, \tag{6.13}$$

where $\otimes$ is the Hamiltonian quaternion product and $\boldsymbol{q}^*$ denotes conjugation. Note that $\boldsymbol{q}^{-1} = \boldsymbol{q}^*$ as the attitude quaternion has norm 1. In algebraic detail, the quaternion error $\boldsymbol{q}_e$ is given by

$$\boldsymbol{q}_e = \begin{bmatrix} q_{e_0} \\ q_{e_1} \\ q_{e_2} \\ q_{e_3} \end{bmatrix} = \begin{bmatrix} q_{r_0}q_0 + q_{r_1}q_1 + q_{r_2}q_2 + q_{r_3}q_3 \\ -q_{r_0}q_1 + q_{r_1}q_0 + q_{r_3}q_2 - q_{r_2}q_3 \\ -q_{r_0}q_2 + q_{r_2}q_0 + q_{r_1}q_3 - q_{r_3}q_1 \\ -q_{r_0}q_3 + q_{r_3}q_0 + q_{r_2}q_1 - q_{r_1}q_2 \end{bmatrix}. \tag{6.14}$$

The tracking error vector is given by

$$\boldsymbol{\xi} = [q_{e_1} \;\; q_{e_2} \;\; q_{e_3}]^T, \tag{6.15}$$

since $\boldsymbol{\xi}$ is chosen to reduce the dimensions of $\boldsymbol{q}_e$ by neglecting q_{e0} that is near 1 for small attitude errors and is only indicative of the size of the rotation error. $\boldsymbol{\xi}$ will be used later in (6.19). The correctness of (6.15) can be justified on the grounds that $\boldsymbol{\xi}$ converges to zero when the attitudes of $\boldsymbol{q}$ and $\boldsymbol{q}_r$ converge, as

then q_{e0} converges to 1 and $[q_{e_1}\ q_{e_2}\ q_{e_3}]^T$ converges component-wise to zero. For large rotational-error correction of attitude, the desired reference quaternion rate $\dot{\boldsymbol{q}}_r$ will be defined based on the error $\boldsymbol{q}_e$ as

$$\dot{\boldsymbol{q}}_r = [k_{q_0} q_{e_0}\ [K_q \boldsymbol{\xi}]^T]^T, \tag{6.16}$$

where $k_{q_0} > 0$ and $K_q = diag[k_{q_1} k_{q_2} k_{q_3}] \in \Re^{3\times 3}$ is a positive-definite diagonal gain matrix, and hence large rotational errors through the rate reference is considered. Note that the value of q_{e_0} is not included in (6.15) but it is included in (6.16) to compute the reference quaternion rate.

Using the defined rate $\dot{\boldsymbol{q}}_r$ and the relation in (2.26), the error rate is can be derived as

$$\dot{\boldsymbol{\xi}} = \tilde{Z}_r \dot{\boldsymbol{q}}_r - \tilde{Z} \dot{\boldsymbol{q}} = \boldsymbol{\omega}_r - \boldsymbol{\omega}. \tag{6.17}$$

This choice of a reference rate $\dot{\boldsymbol{q}}_r$ will aid the proofs of control performance. Also note that $\dot{\boldsymbol{\omega}}_r$ can now be obtained from $\boldsymbol{\omega}_r$, as the latter can be made differentiable by a suitable choice of the desired attitude $\boldsymbol{q}_r$. For very small quaternion error, equation (6.17) can be simplified to

$$\dot{\boldsymbol{\xi}} = \tilde{Z}_{q_e} \boldsymbol{\xi} = \mathbb{I}\, \boldsymbol{\xi} = \boldsymbol{\omega}_r - \boldsymbol{\omega}, \tag{6.18}$$

where $\mathbb{I}$ is the 3×3 identity matrix. Note that equation (6.18) is only valid when the attitude error is small enough, i.e. q_e vector values with the maximum of $[1,\ 1.2350 \times 10^{-5},\ 1.241 \times 10^{-3},\ 0.850 \times 10^{-7}]^T$.

Definition 6.3. ***Controller Signal Computation.*** *The control input* $\boldsymbol{u}$ *for equation* (6.12) *is defined by*

$$\boldsymbol{u} = \dot{\boldsymbol{\omega}}_r + K_{\boldsymbol{\omega}} \dot{\boldsymbol{\xi}} + K_{\boldsymbol{q}} \boldsymbol{\xi}, \tag{6.19}$$

where $K_\omega = diag[k_{\omega_1} k_{\omega_2} k_{\omega_3}] \in \Re^{3\times 3}$ *is a positive-definite diagonal gain matrix setting the error gains in feedback.*

By substituting the control torque (6.12) into (6.7), the rotational dynamics in

(6.7) becomes

$$\begin{aligned}\dot{\boldsymbol{\omega}} &= I^{-1}\hat{I}\boldsymbol{u} + I^{-1}\boldsymbol{u}_d + I^{-1}[\boldsymbol{\Delta}(\boldsymbol{\omega}) - \boldsymbol{\tau}_d] \\ &= \boldsymbol{u} + (I^{-1}\hat{I} - \mathbb{I})\boldsymbol{u} + I^{-1}\boldsymbol{u}_d + I^{-1}[\boldsymbol{\Delta}(\boldsymbol{\omega}) - \boldsymbol{\tau}_d] \\ &= \boldsymbol{u} + I^{-1}\boldsymbol{u}_d - \boldsymbol{y}\end{aligned} \tag{6.20}$$

where

$$\boldsymbol{y} = [\mathbb{I} - I^{-1}\hat{I}]\boldsymbol{u} - I^{-1}[\boldsymbol{\Delta}(\boldsymbol{\omega}) - \boldsymbol{\tau}_d] \quad , \quad \boldsymbol{\Delta}(\boldsymbol{\omega}) = \hat{\boldsymbol{c}}(\boldsymbol{\omega}) - \boldsymbol{c}(\boldsymbol{\omega}). \tag{6.21}$$

From equations (6.15)-(6.20), it follows that the error dynamics are given by

$$\ddot{\boldsymbol{\xi}} + K_\omega\dot{\boldsymbol{\xi}} + K_q\boldsymbol{\xi} = \boldsymbol{y} - I^{-1}\boldsymbol{u}_d. \tag{6.22}$$

By setting $\boldsymbol{\eta} = [\boldsymbol{\xi} \;\; \dot{\boldsymbol{\xi}}]^T \in \Re^{6\times 1}$, the closed-loop error dynamics equation is

$$\dot{\boldsymbol{\eta}} = A\boldsymbol{\eta} + G[\boldsymbol{y} - I^{-1}\boldsymbol{u}_d] \tag{6.23}$$

where

$$A = \begin{bmatrix} 0^{3\times 3} & \mathbb{I}^{3\times 3} \\ -K_q^{3\times 3} & -K_\omega^{3\times 3} \end{bmatrix}, \; G = \begin{bmatrix} 0^{3\times 3} \\ \mathbb{I}^{3\times 3} \end{bmatrix}. \tag{6.24}$$

To bound the error $\boldsymbol{\eta}$, it is necessary that the right-hand-side of (6.23) is to be kept small and that will be achieved by (6.36) later. The new control input $\boldsymbol{u}$ needs to guarantee asymptotic stability for any $\boldsymbol{y}$ varying within a bounded range. To ensure this, the following assumptions are made on the circumstances of the flight.

Assumption 6.1. *(**Flight Envelop**): As the motors have limited rotational rates, they have limited angular velocities $|\Omega_i| < \Omega_{max}$. The vehicle angular velocities $\|\boldsymbol{\omega}\| < \omega^{max}$ and angular accelerations $\|\dot{\boldsymbol{\omega}}\| < \dot{\omega}^{max}$ are also limited. It is assumed that a known upper bound $\alpha > 0$ limits the desired vehicle angular accelerations vector $\dot{\boldsymbol{\omega}}_r$ as*

$$\sup(\|\dot{\boldsymbol{\omega}}_r\|) < \alpha. \tag{6.25}$$

Assumption 6.2. *(**Payload Characteristics**): As the moments of inertia and*

mass of the vehicle may change with the payload to dangerous levels, they need to be constrained by limiting the amount of variation in the moments of inertia. The inertia matrix I is assumed to have a lower and upper bound, $\lambda_{min} > 0,\ \lambda_{max} > 0$, hence the requirement made is that

$$\lambda_{min} \leq \|I^{-1}\| \leq \lambda_{max}. \tag{6.26}$$

Consequently, the deviation between the estimated matrix $\hat{I}$ and actual matrix I can also be described with some $\delta > 0$ in the format of

$$\|\mathbb{I} - I^{-1}\hat{I}\| \leq \delta \leq 1. \tag{6.27}$$

Assumption 6.3. *(**Weather and Aerodynamic Disturbances**): The external torque disturbance $\boldsymbol{\tau}_d$ is sufficiently smooth, due to mechanical inertia, and an upper constant bound $\gamma > 0$ is known such that*

$$\|\boldsymbol{\tau}_d\| \leq \gamma, \tag{6.28}$$

where $\gamma = \sup_{t\in[0,\infty)}\ w(t)$; since $w(t)$ is the wind function that could violate the vehicle and its superior value can be estimated in practice.

Lemma 6.1. *Setting $\boldsymbol{\Delta}(\boldsymbol{\omega})$ as the error between the estimated vector $\hat{\boldsymbol{c}}(\boldsymbol{\omega})$ and the actual vector $\boldsymbol{c}(\boldsymbol{\omega})$, there exist $\beta > 0$ such that*

$$\|\boldsymbol{\Delta}(\boldsymbol{\omega})\| \leq \beta. \tag{6.29}$$

Proof. From $\boldsymbol{\Delta}(\boldsymbol{\omega}) = \hat{\boldsymbol{c}}(\boldsymbol{\omega}) - \boldsymbol{c}(\boldsymbol{\omega})$, $\hat{\boldsymbol{c}}_{\boldsymbol{\omega}} = \Gamma(\boldsymbol{\omega})\hat{I}\boldsymbol{\omega}$, and $\boldsymbol{c}(\boldsymbol{\omega}) = \Gamma(\boldsymbol{\omega})I\boldsymbol{\omega}$, we have

$$\begin{aligned} \boldsymbol{\Delta}(\boldsymbol{\omega}) &= \Gamma(\boldsymbol{\omega})\hat{I}\boldsymbol{\omega} - \Gamma(\boldsymbol{\omega})I\boldsymbol{\omega} \\ I^{-1}\boldsymbol{\Delta}(\boldsymbol{\omega}) &= -(\mathbb{I} - I^{-1}\hat{I})\Gamma(\boldsymbol{\omega})\boldsymbol{\omega}, \end{aligned} \tag{6.30}$$

by *Assumption* 6.1, where the upper limit of the angular acceleration is known, it is possible to compute the upper bound of the angular velocity, $\boldsymbol{\omega}$. Hence the angular velocity-dependent matrix, $\Gamma(\boldsymbol{\omega})$, is such that: $\sup(\|\boldsymbol{\omega}\|) \leq \sigma$ and

$\sup(\|\Gamma(\boldsymbol{\omega})\|) \leq \varrho$ where $\sigma > 0$ and $\varrho > 0$; and using *Assumption* 6.2, we get

$$\begin{aligned} \|\boldsymbol{\Delta}(\boldsymbol{\omega})\| &\leq (\|\mathbb{I} - I^{-1}\hat{I}\| \ \|\Gamma(\boldsymbol{\omega})\| \ \|\boldsymbol{\omega}\|)/\|I^{-1}\| \\ &\leq (\delta \ \varrho \ \sigma)/\lambda_{max} \ := \beta. \end{aligned} \tag{6.31}$$

□

6.2.3 Attitude Stability Analysis

The following theorem states the stability of the proposed controller based on Lyapunov's direct method including the definition of the control term $\boldsymbol{u}_d$ in (6.12).

Theorem 6.1. *For the nonlinear dynamics in* (6.3), (6.19) *and considering the control law in* (6.12)*, the close-loop system is asymptotically stable and the control system's errors converge to zero under Assumptions 6.1-6.3.*

Proof. Setting the equilibrium point $\boldsymbol{\eta} = 0$ where $V(0) = 0$ and choosing the following positive-definite function

$$V(\boldsymbol{\eta}) = \boldsymbol{\eta}^T Q \boldsymbol{\eta} > 0 \ , \ \forall \boldsymbol{\eta} \neq 0 \tag{6.32}$$

where $Q \in \Re^{6\times 6}$ is a symmetric positive-definite matrix, the time derivative of $V(\boldsymbol{\eta})$ in (6.32) along the trajectory of the system errors is

$$\begin{aligned} \dot{V}(\boldsymbol{\eta}) =& \dot{\boldsymbol{\eta}}^T Q \boldsymbol{\eta} + \boldsymbol{\eta}^T Q \dot{\boldsymbol{\eta}} \\ =& \boldsymbol{\eta}^T [A^T Q + QA] \boldsymbol{\eta} + 2\boldsymbol{\eta}^T Q G (\boldsymbol{y} - I^{-1}\boldsymbol{u}_d), \end{aligned} \tag{6.33}$$

considering A has eigenvalues with all negative real parts, hence for a symmetric positive-definite matrix P, Lyapunov equation is written as

$$A^T Q + QA = -P. \tag{6.34}$$

This gives a unique solution Q then the term $\boldsymbol{\eta}^T[A^T Q + QA]\boldsymbol{\eta}$ in (6.33) is negative and the equation will be

$$\dot{V}(\eta) = -\boldsymbol{\eta}^T P \boldsymbol{\eta} + 2\boldsymbol{\eta}^T Q G (\boldsymbol{y} - I^{-1}\boldsymbol{u}_d). \tag{6.35}$$

As the first term $-\boldsymbol{\eta}^T P\boldsymbol{\eta}$ is strictly negative, the second term $\boldsymbol{\eta}^T QG(\boldsymbol{y} - I^{-1}\boldsymbol{u}_d)$ need also to be strictly negative to ensure $\dot{V}(\boldsymbol{\eta}) < 0$. Therefore, $\boldsymbol{u}_d$ must be chosen to render the second term.

Definition 6.4. *For a positive time-varying scalar function $\zeta(\boldsymbol{\eta}, t)$ which will be chosen to bound $\boldsymbol{y}$, the term $\boldsymbol{u}_d$ is defined as*

$$\boldsymbol{u}_d = \begin{cases} \dfrac{\zeta(\boldsymbol{\eta}, t)}{\|G^T Q\boldsymbol{\eta}\|} G^T Q\boldsymbol{\eta}, & \text{if} \quad \|G^T Q\boldsymbol{\eta}\| \geq \mu \\ \dfrac{\zeta(\boldsymbol{\eta}, t)}{\mu} G^T Q\boldsymbol{\eta}, & \text{if} \quad \|G^T Q\boldsymbol{\eta}\| < \mu. \end{cases} \tag{6.36}$$

The term $\boldsymbol{u}_d$ is defined as a continuous approximation of the discontinuous control because if $\boldsymbol{u}_d = \dfrac{\zeta(\boldsymbol{\eta}, t)}{\|G^T Q\boldsymbol{\eta}\|} G^T Q\boldsymbol{\eta}$ when $\|G^T Q\boldsymbol{\eta}\| \neq 0$ and $\boldsymbol{u}_d = 0$ at $\|G^T Q\boldsymbol{\eta}\| = 0$, a chattering problem will produce since $\boldsymbol{u}_d$ will be discontinuous which causes trajectories oscillation. To eliminate this problem, the error should vary within the boundary of μ if $\|G^T Q\boldsymbol{\eta}\|$ is less than this value. Note that $\boldsymbol{u}_d$ depends on the error $\boldsymbol{\eta}$ and with (6.36) bounded-norm error will be ensured.

Assuming that $\|G^T Q\boldsymbol{\eta}\| \geq \mu$, using Cauchy-Schwartz inequality we have

$$\begin{aligned} \boldsymbol{\eta}^T QG(\boldsymbol{y} - I^{-1}\boldsymbol{u}_d) \leq & \|G^T Q\boldsymbol{\eta}\|\|\boldsymbol{y}\| - \lambda_{min}\zeta(\boldsymbol{\eta}, t)\|G^T Q\boldsymbol{\eta}\| \\ = & \|G^T Q\boldsymbol{\eta}\|(\|\boldsymbol{y}\| - \lambda_{min}\zeta(\boldsymbol{\eta}, t)), \end{aligned} \tag{6.37}$$

and if $\zeta(\boldsymbol{\eta}, t)$ is chosen such that the above term $\lambda_{min}\zeta(\boldsymbol{\eta}, t)$ is strictly positive and greater than $\|\boldsymbol{y}\|$, then $\dot{V}(\boldsymbol{\eta}) < 0$.

Definition 6.5. *If the term $\boldsymbol{y}$ is bounded such that $\|\boldsymbol{y}\| \leq \varepsilon$ for $\varepsilon > 0$, and for $\lambda_{min} > 0$, $\zeta(\boldsymbol{\eta}, t)$ can be chosen depending on $\boldsymbol{y}$ as*

$$\zeta(\boldsymbol{\eta}, t) \geq \frac{\varepsilon}{\lambda_{min}}. \tag{6.38}$$

From $\boldsymbol{y}$ in (6.21) and the *Assumptions* 6.1 – 6.3 with (6.29), we get

$$\begin{aligned} \|\boldsymbol{y}\| &\leq \|\mathbb{I} - I^{-1}\hat{I}\|(\|\dot{\boldsymbol{\omega}}_r\| + \|K_\omega\|\|\dot{\boldsymbol{\xi}}\| + \|K_q\|\|\boldsymbol{\xi}\|) + \|I^{-1}\|(\|\boldsymbol{\Delta}(\boldsymbol{\omega})\| + \|\boldsymbol{\tau}_d\|) \\ &\leq \delta(\alpha + \|K_\omega\|\|\dot{\boldsymbol{\xi}}\| + \|K_q\|\|\boldsymbol{\xi}\|) + \lambda_{max}(\beta + \gamma) := \varepsilon, \end{aligned} \tag{6.39}$$

from the previous two equations, $\zeta(\boldsymbol{\eta}, t)$ obtained as

$$\zeta(\boldsymbol{\eta}, t) \geq \frac{\delta}{\lambda_{min}}(\alpha + \|K_\omega\|\|\dot{\boldsymbol{\xi}}\| + \|K_q\|\|\boldsymbol{\xi}\|) + \frac{\lambda_{max}}{\lambda_{min}}(\beta + \gamma). \tag{6.40}$$

Finally, for $\|G^T Q\boldsymbol{\eta}\| \geq \mu$, (6.35) becomes

$$\dot{V}(\boldsymbol{\eta}) = -\boldsymbol{\eta}^T P\boldsymbol{\eta} + 2\boldsymbol{\eta}^T QG(\boldsymbol{y} - I^{-1}\frac{\zeta(\boldsymbol{\eta}, t)}{\|G^T Q\boldsymbol{\eta}\|}G^T Q\boldsymbol{\eta}) < 0, \tag{6.41}$$

and for $\|G^T Q\boldsymbol{\eta}\| < \mu$,

$$\begin{aligned} \boldsymbol{\eta}^T QG(\boldsymbol{y} - I^{-1}\boldsymbol{u}_d) \leq & \mu\|\boldsymbol{y}\| - \lambda_{min}\zeta(\boldsymbol{\eta}, t)\mu \\ = & \mu(\|\boldsymbol{y}\| - \lambda_{min}\zeta(\boldsymbol{\eta}, t)), \end{aligned} \tag{6.42}$$

then

$$\dot{V}(\boldsymbol{\eta}) = -\boldsymbol{\eta}^T P\boldsymbol{\eta} + 2\boldsymbol{\eta}^T QG(\boldsymbol{y} - I^{-1}\frac{\zeta(\boldsymbol{\eta}, t)}{\mu}G^T Q\boldsymbol{\eta}) < 0. \tag{6.43}$$

□

6.3 Simulation Studies

In order to test the controller performance in a realistic scenario, simulations have been carried out using the MathWorks team's detailed model [61] in Simulink/-Matlab. The UAV's nonlinear dynamics in (6.1) and (6.3) have been implemented in the model. The DC motors with propeller dynamics were also modelled based on parameters taken from real multi-rotor motor combinations. Moreover, the model includes computations of the motors' angular velocities Ω_{i_r} from the computed thrust U and torques $\boldsymbol{\tau}$ demanded by the control scheme. The computed Ω_{i_r} values have been applied to the motor and propeller dynamics and then realistic thrust U and torques $\boldsymbol{\tau}$ were obtained to approach the behaviour of a real dynamics. The original MathWorks model has been modified with the use of quaternions instead of Euler angles, inertia moments variations, according to the payload change, were considered, disturbances were added to the torques. The proposed nonlinear controller has been compared to a nonlinear adaptive fractional order sliding mode based back-stepping (FRSDBKAD) controller presented

in [119] in terms of robustness and stability.

6.3.1 Nominal Performance

The initial task is to track the desired position trajectory $\boldsymbol{r}_r = [x_r\ y_r\ z_r]^T$ and a desired rotation q_{3_r} without disturbances where all the initial reference x_r, y_r, z_r, q_{3_r} are set to zero. Fig. 6.2 illustrates the desired trajectory of the drone which includes take-off, several manoeuvres and landing. According to the given trajectory, the RNDI controller shows that the measured x, y, z are well followed the reference trajectory as can be seen in Fig. 6.2. The attitude controller results are shown in Fig. 6.3, 6.4 and 6.5, where the attitude controller tracks the reference quaternions produced by the position controller. In nominal flight conditions, the UAV tracked the reference trajectory well and more accurately than the FRSDBKAD controller.

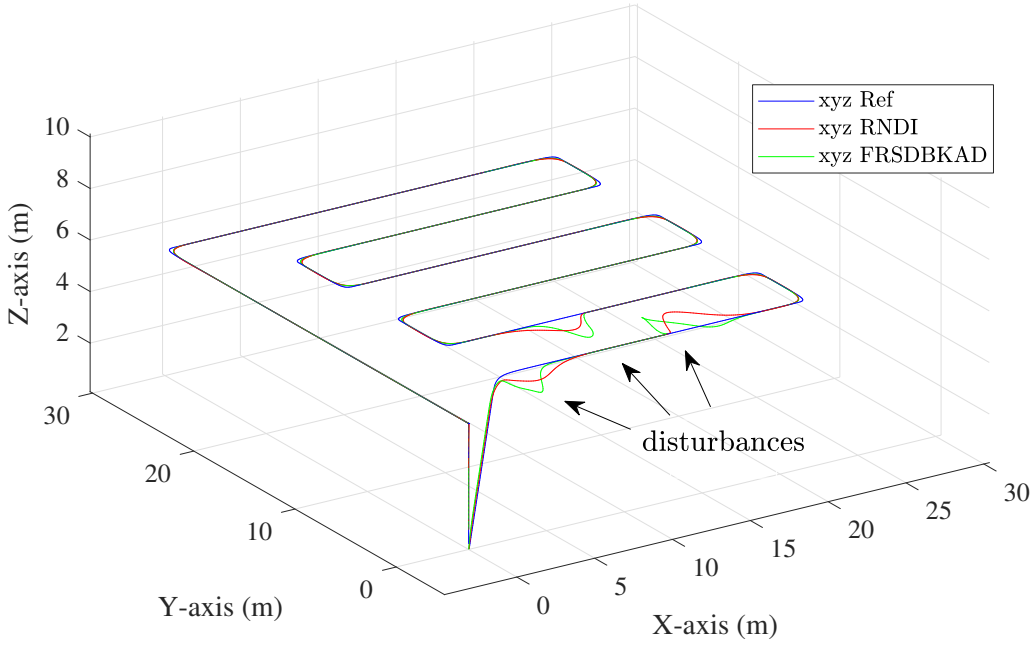
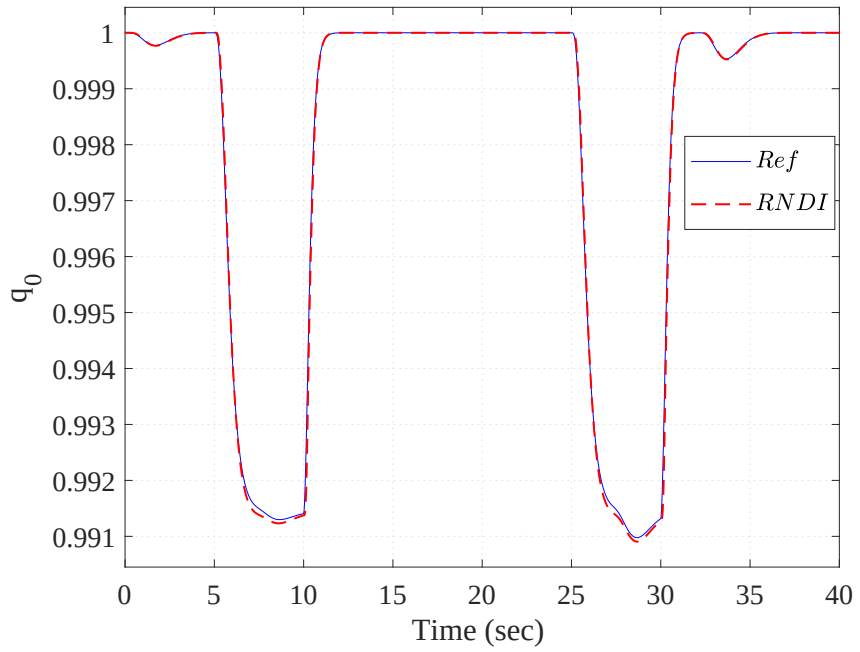
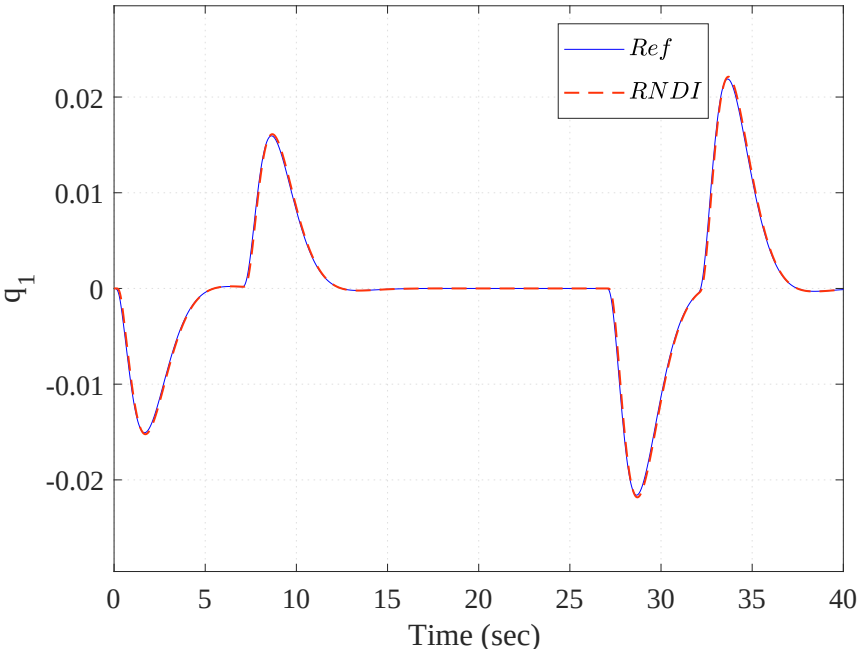


Figure 6.2: Three dimensional xyz trajectory in the W-frame. Ref: reference trajectory, RNDI: the proposed dynamic inverse controller, and FRSDBKAD: adaptive fractional order sliding mode based back-stepping controller. Differences can be seen under wind disturbances.


1
0.999
0.998
0.997
0.996
0.995
0.994
0.993
0.992
0.991
q_0
Ref
RNDI
0
5
10
15
20
25
30
35
40
Time (sec)


0.02
0.01
0
-0.01
-0.02
q_1
Ref
RNDI
0
5
10
15
20
25
30
35
40
Time (sec)

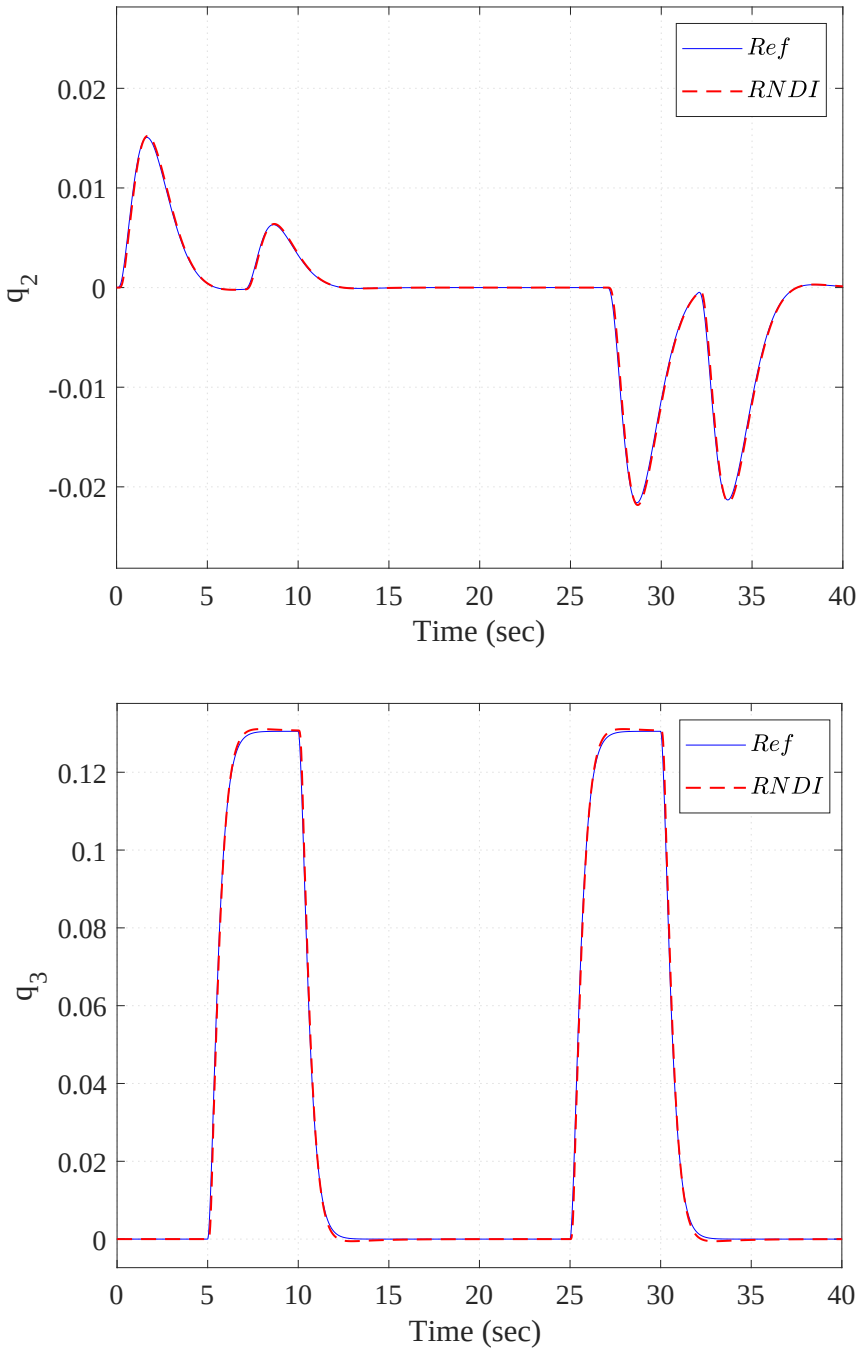
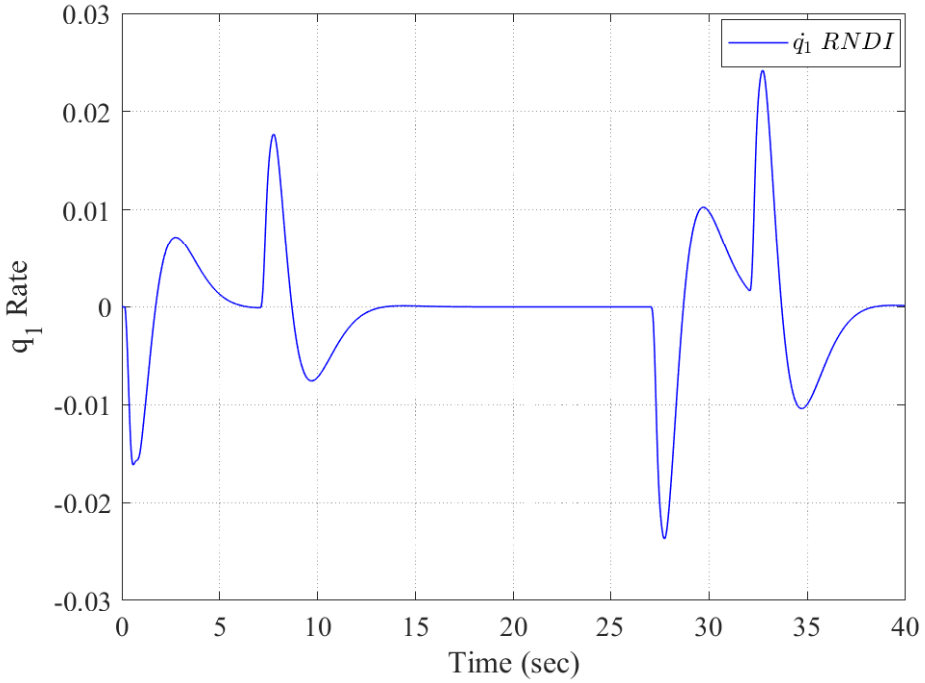
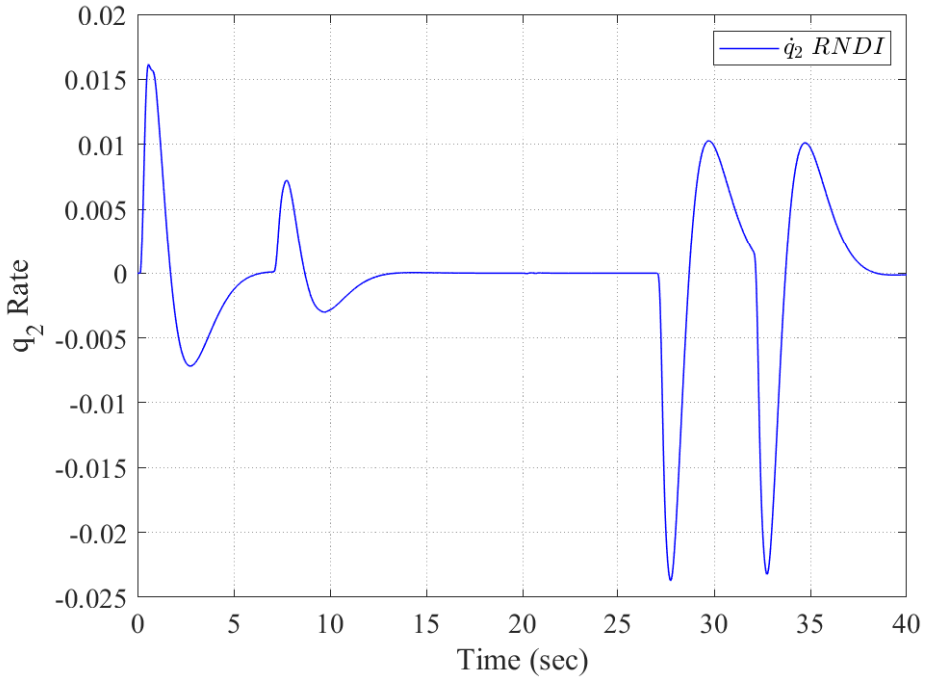


Figure 6.3: The measured quaternions track the reference attitude by robust nonlinear dynamics inversion (RNDI) control. The q_0 shows the attitude angle and $q_1,\ q_2,\ q_3$ show the attitude-axis components: the blue continuous reference line almost coincides with the dashed RNDI controller.


$\dot{q}_1$ RNDI
q_1 Rate
Time (sec)
0.03
0.02
0.01
0
-0.01
-0.02
-0.03
0
5
10
15
20
25
30
35
40


$\dot{q}_2$ RNDI
q_2 Rate
Time (sec)
0.02
0.015
0.01
0.005
0
-0.005
-0.01
-0.015
-0.02
-0.025
0
5
10
15
20
25
30
35
40

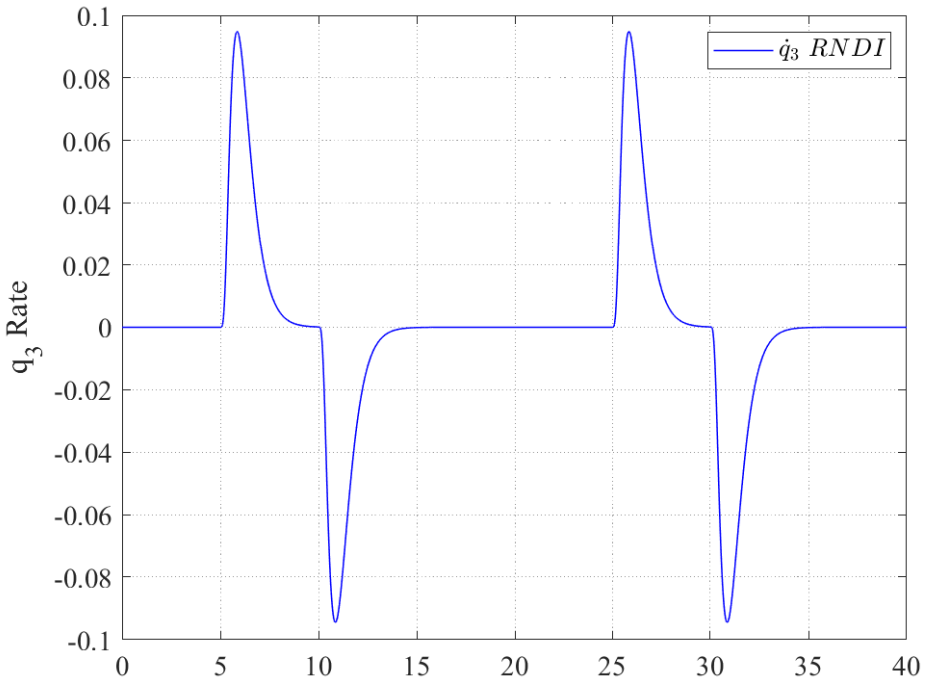


Figure 6.4: The measured quaternions rates for the RNDI controller.

The controller parameters obtained are listed in Table 6.1. From (6.34), the positive definite diagonal matrix $P \in \Re^{6\times 6}$ is chosen as

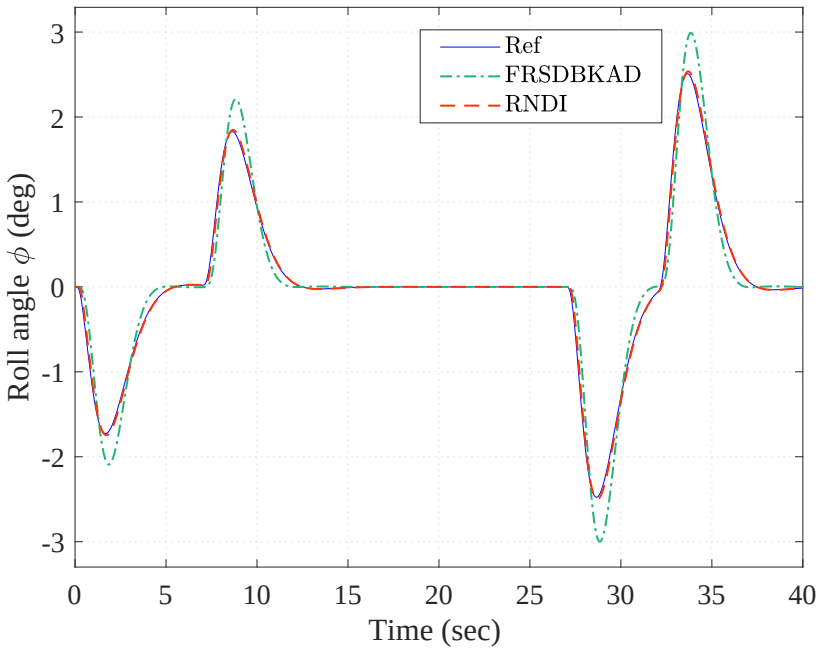
$$P = diag[9 * 10^{-12} \quad 9 * 10^{-12} \quad 5 * 10^{-13} \quad 3 * 10^{-10} \quad 3 * 10^{-10} \quad 8 * 10^{-10}], \quad (6.44)$$

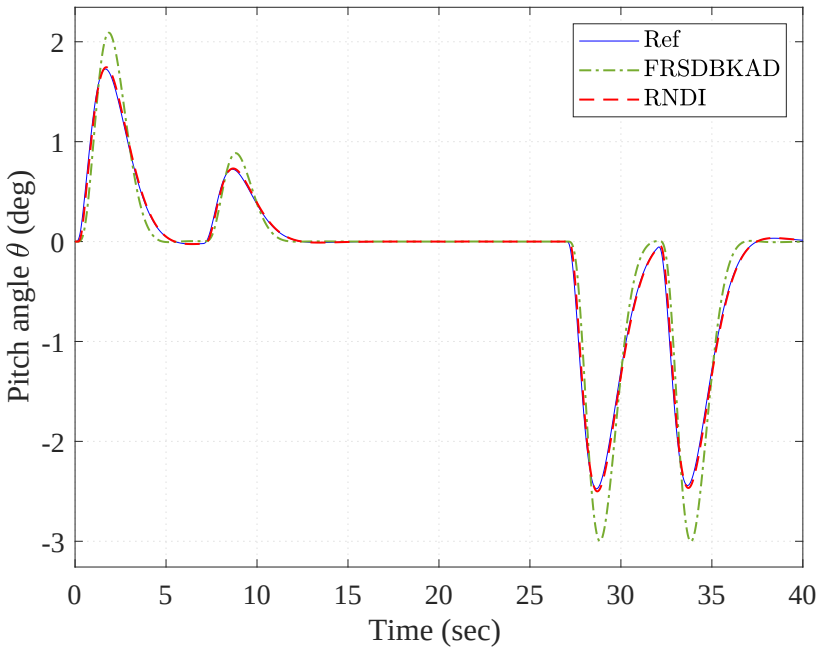
and the symmetric positive definite matrix Q is obtained

$$Q = \begin{bmatrix} 1.566 * 10^{-11} & 0 & 0 & -4.5 * 10^{-12} & 0 & 0 \\ 0 & 1.566 * 10^{-11} & 0 & 0 & -4.5 * 10^{-12} & 0 \\ 0 & 0 & 2.539 * 10^{-9} & 0 & 0 & -2.5 * 10^{-13} \\ -4.5 * 10^{-12} & 0 & 0 & 2.466 * 10^{-10} & 0 & 0 \\ 0 & -4.5 * 10^{-12} & 0 & 0 & 2.466 * 10^{-10} & 0 \\ 0 & 0 & -2.5 * 10^{-13} & 0 & 0 & 6.347 * 10^{-8} \end{bmatrix}. \quad (6.45)$$

6.3.2 Performance under Payload Uncertainties

The multi-rotor's flight controller should maintain the stability of the UAV if its total mass changes due to adding payload, which causes a shift in the centre of gravity (CG) and changes the inertia matrix. In this subsection, this problem is tackled by testing the proposed control scheme under a mass distribution change.


Ref
FRSDBKAD
RNDI
Roll angle ϕ (deg)
Time (sec)
3
2
1
0
-1
-2
-3
0
5
10
15
20
25
30
35
40


Ref
FRSDBKAD
RNDI
Pitch angle θ (deg)
Time (sec)
2
1
0
-1
-2
-3
0
5
10
15
20
25
30
35
40

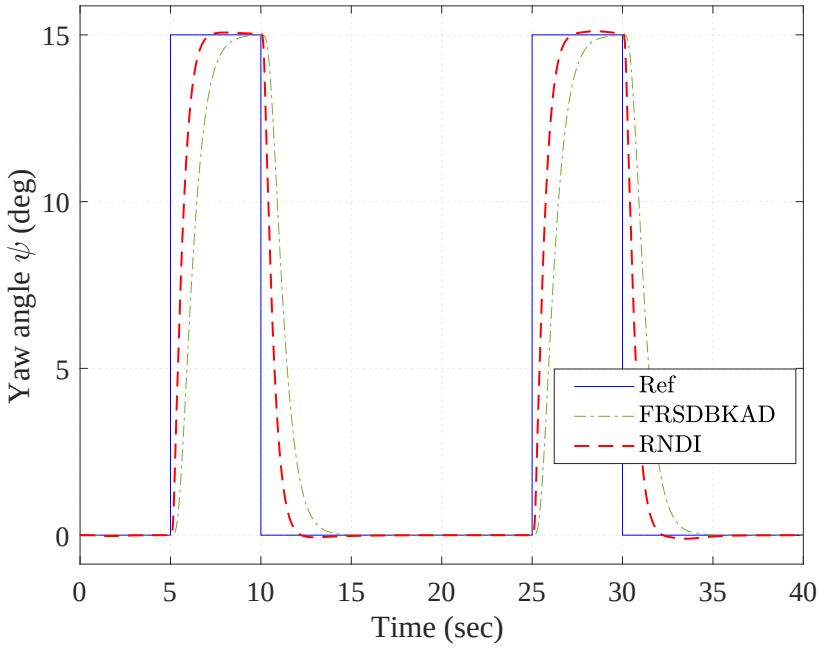


Figure 6.5: The measured angles track the reference attitude by adaptive fractional order sliding mode based back-stepping control (FRSDBKAD) and by robust nonlinear dynamics inversion (RNDI) control. The "*Roll angle* ϕ" shows the roll rotation around X-axis, "*Pitch angle* θ" shows the pitch rotation around Y-axis and "*Yaw angle* ψ" shows the yaw rotation around Z-axis; The blue continuous reference line almost coincides with the dashed RNDI controller proposed in this chapter, while the dot-dashed FRSDBKAD controller is far from achieving that.

Table 6.1: Multi-rotor Parameters

Parameter	Value	Parameter	Value
$\hat{I}_x$	$5.831 * 10^{-3}\ kg.m^2$	b	$12 * 10^{-8}\ N.m/(rad/sec)^2$
$\hat{I}_y$	$5.831 * 10^{-3}\ kg.m^2$	l	$9 * 10^{-6}\ N/(rad/sec)^2$
$\hat{I}_z$	$1.166 * 10^{-2}\ kg.m^2$	α	180.7904
k_{q_0}	0.01	σ	36.3485
k_{q_1}	16	δ	0.04231
k_{q_2}	16	β	0.332
k_{q_3}	25	γ	0.4231
k_{ω_1}	0.9	μ	0.0095
k_{ω_2}	0.9	ϱ	36.3485
k_{ω_3}	0.0064	λ_{min}	171.045
ℓ	0.2 m	λ_{max}	171.47
m	0.9272 kg	Ω_{max}	707.1068 rad/sec

Referring to *Assumption* 6.2, the maximum payload of the proposed multi-rotor has been set to 300 grams. Due to this mass distribution change, the moments of inertia will be altered. Considering the specified payload capacity that the multi-rotor can hold, the range of variation in the inertia moments is computed, hence the values of $\lambda_{min}, \lambda_{max}, \delta$ (6.26) and (6.27) can be derived. By knowing these bounds, the proposed controller can compensate for any variation of inertia moments within the specified range. Where any change in inertia components due to payload variation or even inaccurate values of the inertia moments or centre of the mass in modelling can be compensated by the proposed term u_d in (6.36) hence the UAV will stay in the stable region.

The CG is computed by assuming the geometric CG is at the centre of the UAV's hub, i.e. at point (0, 0, 0). Then the nominal diagonal inertia matrix components are computed. For any additional payload of up to 300 grams located within the hub (centre of the vehicle's body) of $10 \times 10 \times 4$ cm, for instance if the UAV equipped with an omnidirectional camera or an arm to pick up and deposit objects, the inertia matrix components (not diagonal) are computed for testing the controller with any payload change within the specified limits. Figure 6.6 illustrates the simulation which is conducted to test the performance of the proposed control scheme when different payloads are applied. This test is con-

ducted by changing the UAV's mass since different payloads were added to the UAV's hub for up to 300 grams and consequently the CG and inertia moments were varied. The results show that the controller can cope well with any mass, CG and inertia matrix change within the specified bounds of $\lambda_{min}, \lambda_{max}$ and δ which have been formulated in *Assumption* 6.2.

To further increase the robustness of the control scheme for more reliable performance, a test can be executed before the flight to make an estimation of the range of uncertainty in terms of the payload changes, i.e. more accurate estimation of $\lambda_{min}, \lambda_{max}$, and δ. Known methods such as in [72, 83, 129] can be used to estimate the inertia matrix while in flight and disallow the flight if the $\lambda_{min}, \lambda_{max}, \delta$ are violated.

6.3.3 Performance under Aerodynamic Disturbances

In this section we expose the multi-rotor to external torque disturbances to test the controller's behaviour and stability. External disturbances have been applied to the nominal torques and their effects on vehicle attitudes are illustrated in Fig. 6.7. It is assumed that the disturbances are varying within 40% of the minimum/maximum torque $\boldsymbol{\tau}_{min/max} = [\pm 0.7446 \ \pm 0.7446 \ \pm 0.0993]^T \ Nm$; where the range of disturbances for both roll and pitch is $\tau_{d\phi}, \tau_{d\theta} = [-0.2978, 0.2978] \ Nm$ and for yaw $\tau_{d\psi} = [-0.0397, 0.0397] \ Nm$. The results in Fig. 6.7 illustrate how the controllers are reacting to the disturbances by counter acting the extra torques with some success in order to return the vehicle to follow the reference trajectory. The figures show the UAV's attitudes in terms angles, where quaternions have been transferred to Euler angles using the relation (2.25) for illustration. A comparison between FRSDBKAD control and the proposed robust RNDI control is conducted to show how this controller is performing well, especially under high external disturbances for roll and pitch motion where the FRSDBKAD control performed less well with some oscillations. The robust RNDI controller also does not hit the limits of the maximum actuator ($\Omega_{i\ max} = \ 707.1068 \ rad/sec$) even under high disturbances as can be seen from the measured angular velocities of the motors, Ω_i, in Fig. 6.8.

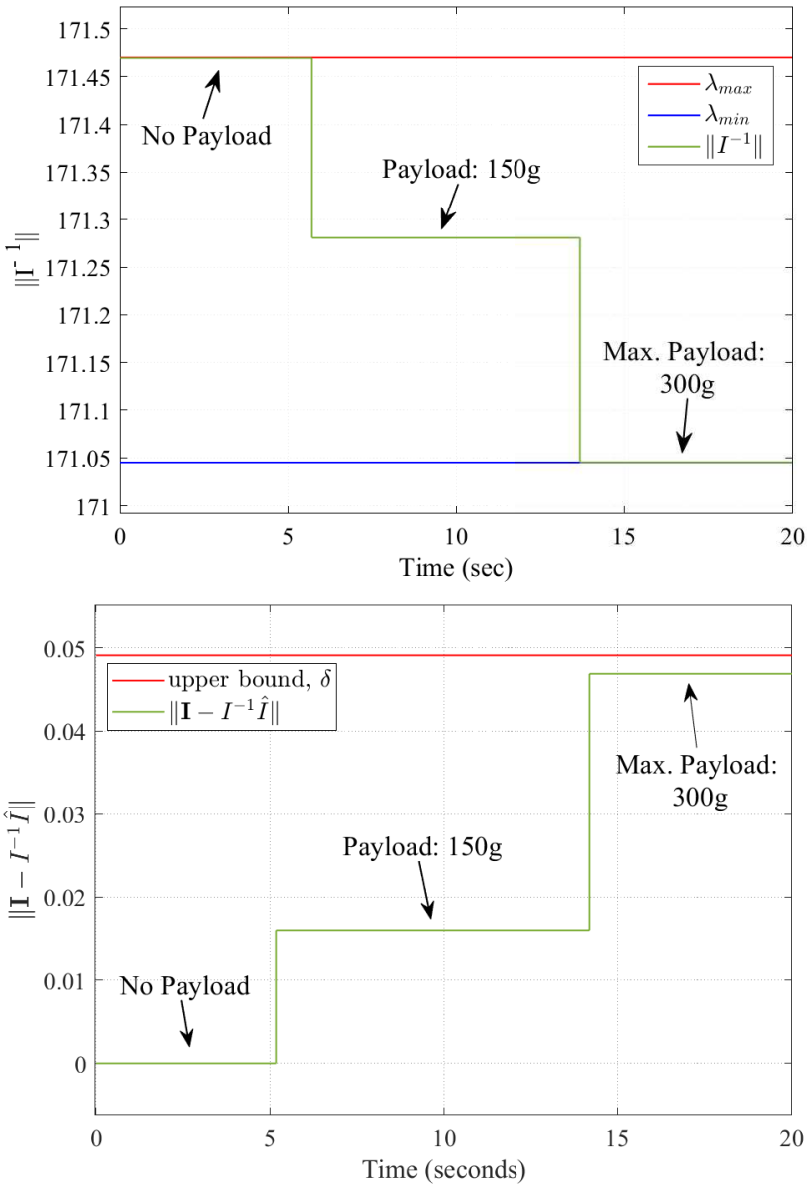
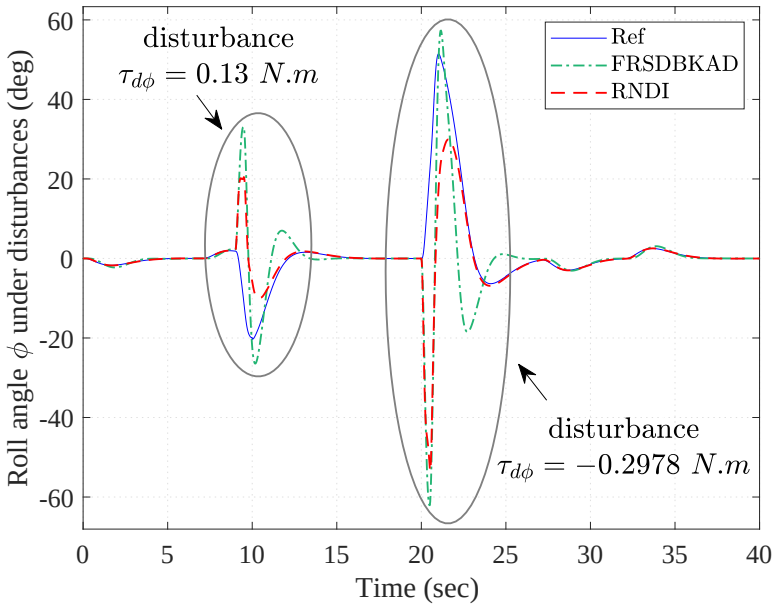
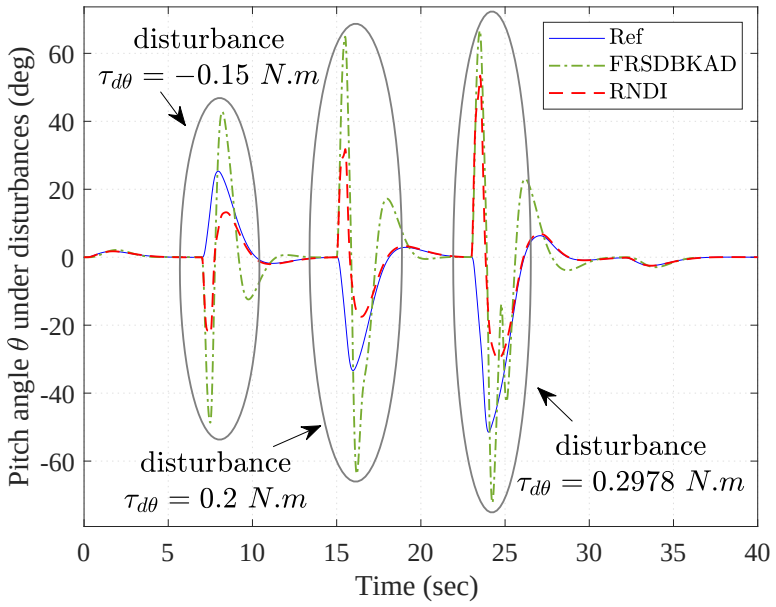


Figure 6.6: The first graph illustrates the norm of inertia matrix inverse $\|I^{-1}\|$ variation with payload change within the UAV's hub (*Assumption* 6.2 - equation (6.26)). The term $\|I^{-1}\|$ varies within the specified upper limit λ_{max} and lower limit λ_{min}. The second graph shows the effect of payload variation on the term $\|\mathbb{I} - I^{-1}\hat{I}\|$ which stays below the specified upper bound δ (*Assumption* 6.2 - equation (6.27)).


disturbance
$\tau_{d\phi} = 0.13\ N.m$
Ref
FRSDBKAD
RNDI
disturbance
$\tau_{d\phi} = -0.2978\ N.m$
Roll angle ϕ under disturbances (deg)
Time (sec)


disturbance
$\tau_{d\theta} = -0.15\ N.m$
Ref
FRSDBKAD
RNDI
disturbance
$\tau_{d\theta} = 0.2\ N.m$
disturbance
$\tau_{d\theta} = 0.2978\ N.m$
Pitch angle θ under disturbances (deg)
Time (sec)

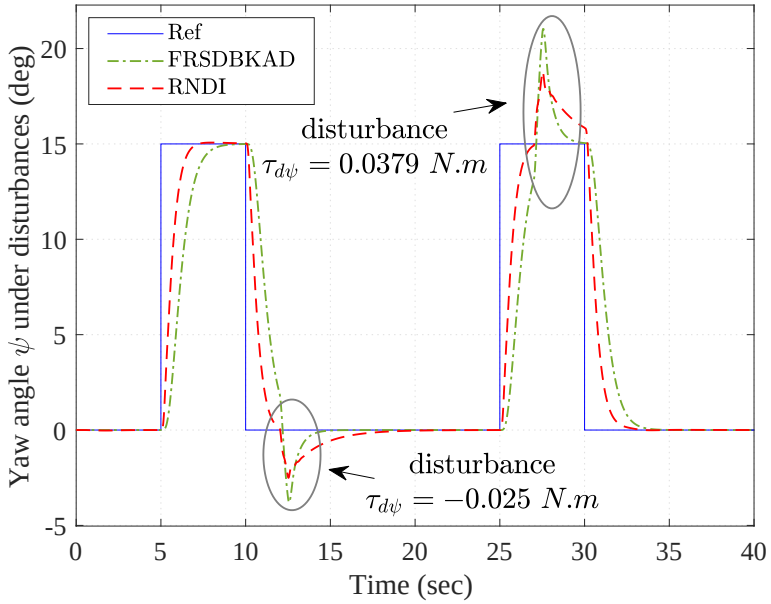


Figure 6.7: Attitudes under external disturbances show some oscillation in roll, ϕ, and pitch, θ, motion of the FRSDBKAD controller (dot-dashed green line) with less deviation in yaw, ψ, but not so for the RNDI (dashed red line) controller.

6.4 Discussion of Applicability

The ultimate aim of this work is to design a robust control scheme for multi-rotor UAVs that can provide a good or at least an acceptable performance and able to deal with different flight conditions such as payload change during the flight or when the vehicle is exposed to external forces, e.g. winds. These two conditions are very common in practice which may force the UAV to enter unstable state-space regions, and as a consequence, the craft may crash and potentially cause damage to property, humans and privacy. However, in this work these conditions are tackled in the modelling and design of a robust nonlinear controller for multi-rotor unmanned aircraft.

6.4.1 Environmental Conditions

The main two environmental conditions, which the UAV may be exposed to, are the payload change and wind disturbances. The first considered condition, the

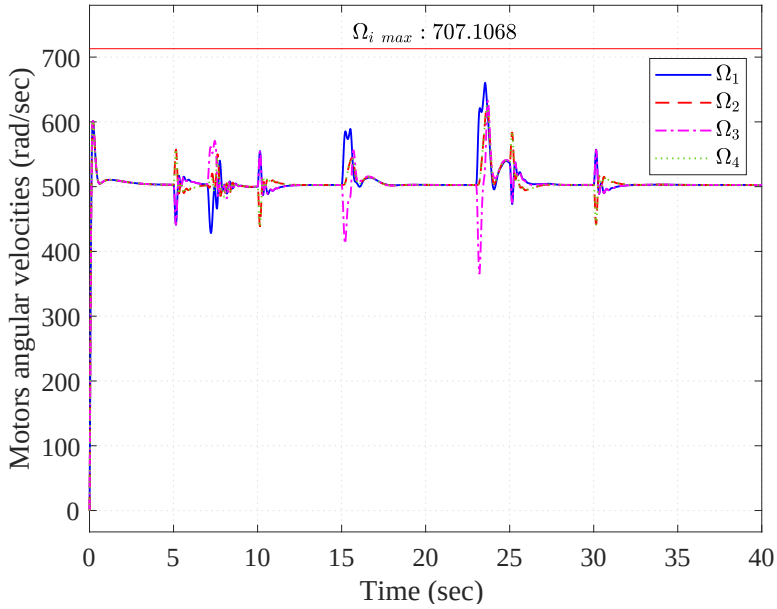


Figure 6.8: Actuators angular velocities computed from the RNDI control. It can be seen that the actuators limit, $\Omega_{i\ max}$, has not been reached even with the presence of disturbances.

payload variation, leads to a change in the mass of the UAV, hence in its inertia moments can change. The range of these variations can be computed using the fact that the additional mass or payload is limited by the rotors lifting limits. Therefore, the UAV should have a limited payload that the actuators can handle. Knowing the possible range of the vehicle's mass variations, one can set the lower and upper bounds of inertia moments following (6.26) and (6.27). This way any change in the payload within the specified range will produce stable control. For the second disturbance, wind disturbances, knowing the range of wind strengths, which the craft may be exposed to during its flight, leads to the design of a controller that accounts for additional torques that represent these disturbances for up to the maximum specified limit. The nonlinear term u_d defined in (6.36) compensates the variation of these conditions based on the specified bounds from (6.26) and (6.28). Hence any variation in these two disturbances under specified bounds results in the stable control of the UAV. Note that u_d is defined based on the attitude errors under stability conditions to compensate for any external variation caused by winds or payload change.

In terms of inertia moment changes, which can be attributed to payload variation, the RNDI control performs well by compensating for the moments change through the u_d term for any mass change that is within the specified limits as illustrated in Fig. 6.6. The RNDI controller has less deviation and oscillation in comparison with FRSDBKAD especially for roll and pitch for dealing with external wind disturbances as can be seen from Fig. 6.7. Keeping this deviation in attitude at the minimum will reduce the deviation from the reference trajectory, as can be seen in Fig. 6.2. It is also essential to avoid reaching the maximum motors' speed which has been considered in RNDI control scheme as illustrated in Fig. 6.8 to preserve UAV stability. Note that both payload change and wind disturbances have been applied at the same time to the UAV in order to test the controller performance. The simulation results show that the RNDI controller can cope well even if both conditions occur within the specified limits stated in the proposed assumptions. This is a more realistic scenario that happens in practice and with this controller the UAV can preserve its stability and tracking the given trajectory more effectively.

6.4.2 Multi-rotor UAVs Supported with Decision Making Strategies

The remaining question is how to address the situations when the maximum payload is reached or when the UAV is exposed to extreme gusts of wind beyond the craft abilities, i.e. exceeding the maximum disturbance torques bounds that considered during the control design. Answering these questions is essential for a safe and reliable flight of unmanned vehicles in general and for autonomy in particular. Several studies have been conducted to provide the UAV's autopilot with the ability to monitor its flight condition [42, 122, 127]. Other studies in [13, 18, 48, 106, 113, 117] have implemented intelligent agents supported by decision-making abilities to supervise the variations in the environmental conditions and to see whether they go beyond the specified limits then take the appropriate decisions.

The advantages of these studies can be exploited by providing the autopilot with a software agent, which is able to monitor whether the term u_d in *Definition* 6.4 reaches its bounds or stay within the safe (stable) region. Another approach can be implemented by detecting the out of bounds status by monitoring the limits of the actuators, i.e. observing the angular velocities of motors against their maximum boundaries ($\Omega_{i\ max}$); see Fig. 6.8. If these boundaries are reached for some period of time (which can be tested and computed in practice), the agent can make the required decisions and perform emergency procedures to prevent incidents or reduce the risk of a crash. The agent may also inform the pilot or send warning messages to the nearest station to inform the need for an emergency landing, for instance. This approach increases flight safety and reduces the risk of collision or causing material damage.

Using the proposed RNDI control scheme under mild disturbances, the UAV's autopilot does not need to estimate the inertia moments or wind disturbances on board as any variation of the conditions within the limits will be handled by the RNDI controller. When combined with inertia estimation and an onboard decision agent, high levels of robustness and safety can be achieved.

6.5 Chapter Summary

This chapter has introduced a novel robust multi-rotor controller that accounts for both inertial uncertainty and disturbances. The proposed control system consists of two loops: an inner and outer loop. The inner loop is a nonlinear attitude controller, which is designed based on dynamic inversion control by taking into account dynamical uncertainty and external disturbances. The outer loop is a feedback position controller that computes the total thrust and reference quaternion values, which are passed to the inner loop. Lyapunov's second method is used as part of the control design to compute an additional nonlinear term that compensates for the uncertainty and disturbances and ultimately ensures stability under well-defined conditions in practice. The control system has been simulated based on a nonlinear multi-rotor model developed by MathWorks to test the control performance and it was compared with a competitive nonlinear controller. Ultimately, the results of this work may enhance the safety of multi-rotor unmanned aerial vehicles.

Chapter 7

Verification Framework for Control System Functionality of Unmanned Aerial Vehicles

7.1 Overview

A functional verification framework is proposed and presented in this chapter for unmanned aerial vehicles using theorem proving. The framework's aim is to provide a procedure for proving that the theoretically designed control system of the UAV satisfies robustness requirements to ensure safe performance under varying environmental conditions. Extensive manual mathematical and numerical derivations, which have formerly been carried out manually, are checked for their correctness on a computer. To illustrate the applicability of the framework, a higher-order logic interactive theorem prover and an automated theorem prover are employed to formally verify the nonlinear attitude control system of a generic multi-rotor UAV presented in Chapter 6, over a stability domain within the dynamical state space of the drone. Further benefits of the framework are that some of the methods can be implemented onboard the aircraft to detect when its controller reaches its flight envelop limits due to severe weather conditions. Such a detection procedure can be used to advise the remote pilot or an onboard intelligent agent to decide on alterations of the planned flight path.

7.2 An Aircraft Verification Framework

The proposed framework is developed to conduct further verification steps for aircraft at the design stage of its control system. After an engineer designed the control system of an UAV, the verification process is strated using our framework to verify the correctness of the designed control system and stability analysis with considering the aircraft dynamics and actuaters constrains. The framework also includes safety procedures of onboard stability monitoring of aircraft during the flight using formal methods. This is considered a complementary work of the conventional verification processes such as software and code verification.

This section presents the verification framework which is shown in Fig.7.1. The framework consists of two stages: the ITP represented by *Isabelle/HOL* to prove the mathematical derivation of the designed control system and its stability analysis, and the ATP represented by *MetiTarski* prover for continuously checking the validity of aircraft's stability onboard during the flight. To perform the first stage, the aircraft's components need to be included in the Isabelle/HOL prover to carry out the verification process. Therfore, the framework starts with formalising the aircraft's components in the HOL syntax of the Isabelle prover. The aircraft's componets that need to be included and formalised in the framework are the dynamical equations of motion, the coordinate system in the rigid body frame, the transformations between the world and body frame, the controller design, and stability analysis. Other properties needed for this work are also formalised in HOL such as time domain, signal definition, real vectos and matrices with their properties, etc. For the second stage, the aircraft's stability analysis is needed to be formalised in the FOL syntax of the MetiTarski prover for possible verification of stability onboard the craft. Therefore, the derivative of Lyapunov function is formalised in FOL to be used in the MetiTarski prover for checking the negation, i.e. the Lyapunov derivative is always negative otherwise the aircraft is out of the stable region. Despite that this work aims to verify control systems in the aerospace field, in particular UAVs, but it is worth to mention that with minor developments the framework can also be used in other fields such as robotics, automotive, offshore, autonomous systems and safety-critical systems. The next section will illustrate the two verification stages of the framework in detail.

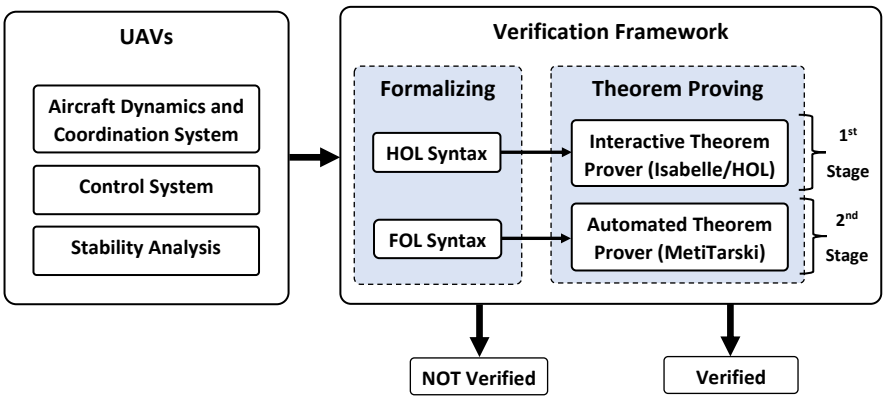


Figure 7.1: UAVs verification framework

7.3 Case Study: Multirotor Verification

7.3.1 Verification in Isabelle/HOL Prover

In this subsection, the first stage of the verification framework shown in Fig. 7.1 will be demonstrated using the attitude controller of a generic quadcopter UAV proposed in Chapter 6 including the assumptions and flight conditions made. In this controller, the quadcopter's rotational dynamics are controlled using a robust nonlinear controller that takes into account the modelling uncertainty and external disturbances. To ensure correctness of the designed attitude control, the design's derivations and stability analysis have been verified using Isabelle/HOL prover. Isabelle/HOL is chosen for this purpose due to its rich library of mathematical theorems which are required to perform the verification.

The verification process using Isabelle/HOL is illustrated in Fig. 7.2 which consist of two stages: formalising and proving procedures. The first stage starts by formalising the quadcopter UAV system into the Isabelle/HOL syntax such as the coordinate system, rotational dynamics, time-domain functions, proposed assumptions and aircraft's stability analysis. The implementation of the control design and aircraft's dynamics includes a series of *definition*, *lemma*, and *theorem* items. Some assistant lemmas were needed to be formalised and proven, which did not exist in Isabelle due to the fact that prover library is still under development. The formalisation also needs to import some pre-proven mathematical theories

and lemmas from the prover library, which are used in formalising and proving the control system equations with the proposed assumptions and definitions.

The theories that used under HOL platform will be described here to illustrate the formalisation and proof procedures. First of all, the *Quaternions.thy* for the quaternion definition and operations that represents aircraft's coordination. The main multi-variable analysis package which includes *Analysis.thy* for functions operations over real field, *Finite_Cartesian_Product.thy* and *Inner_Product.thy* for definitions and operations of real vectors, *L2_norm.thy* and *Norm_Arth.thy* for real vector norms and their operations, etc. The *HOL.thy* is the core of HOL platform which includes definitions of real numbers (*real.thy*), functions (*fun.thy*), sets (*set.thy*), etc., which are necessary in all the formalising procedures. The main multi-variable analysis theory, *Analysis.thy*, which includes definitions of real vectors (*Finite_Cartesian_Product.thy*), vector norms (*L2_norm.thy, Norm_Arth.thy*) and their operations. These theories are used to define the aircraft's three-dimensional rotation vectors and their norms including the torque, angular velocity and acceleration vectors where each component of a vector represented as a continuous time-domain function $f(t)$; the continuous function defined in $Fun.thy$, $Function\_Algebras.thy$ and $Topological\_Spaces$ theories are utilised for this purpose. The time sub-domain is defined by a time set $T = \{t.\ t \in \{0..\infty\}\}$ and is followed by definitions of sets of vectors, which are working within T.

The matrices components are formalised using $Matrix.thy$ and their operations using $Analysis.thy$, $Finite\_Cartesian\_Product.thy$ and $Real\_Vector\_Spaces.thy$. The rate change of the quadcopter attitudes, i.e. velocities and accelerations, are formalised by time derivation using *Deriv.thy* and *derivative.thy* theories. The quadcopter controller design includes several robust assumptions which need inequalities over the real-numbers field. Fortunately, such inequalities have been defined in Isabelle prover under *Orderings.thy* theory. This is an important feature for any robust control design to be proven. However, the second stage (proving procedures) is an interactive process between the designer/engineer and the automated proving tools that Isabelle prover has or supported. The role of designer/engineer is to help the prover to step-by-step prove the statement in case that the prover is not able to solve the proof automatically by simplifying the statement into several steps. Each step should be proven using the provided

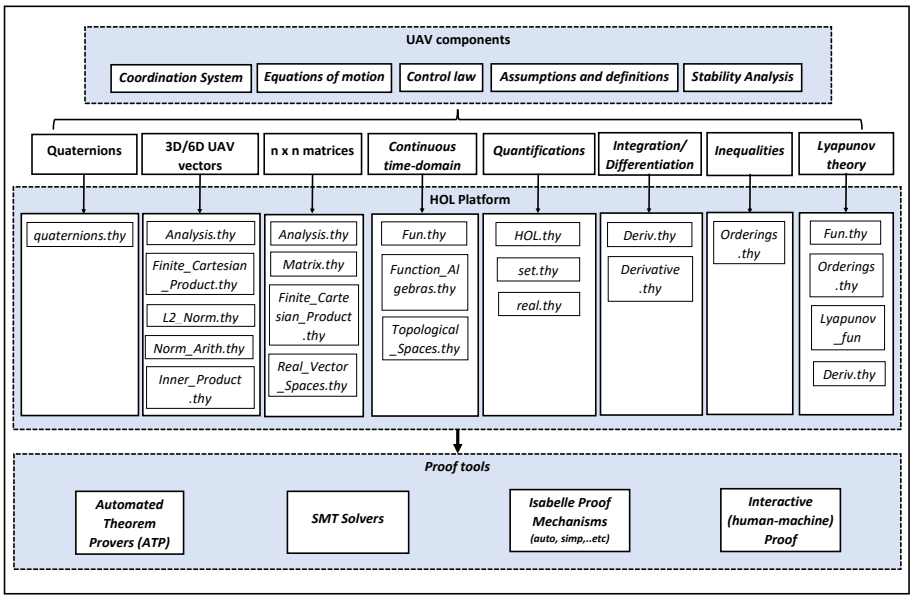


Figure 7.2: Formalising and proving UAV's controller in Isabelle/HOL theorem prover

automated tools before moving to the next one otherwise the prover will not pass the statement. Examples of the automated tools supported in Isabelle are: *CVC4*, *Z3*, *SPASS*, *E prover*, *Remote_Vampire* and *SMT sovlers*. In addition, Isabelle has its own automatic proving tools such as *auto*, *simp*, *blast*, etc. Most of the control system that have been verified in this work required an interaction with the prover due to the design complexity making it not possible for the prover to solve them automatically. The Isabelle code is too long to be stated here, and instead only the important definitions and proofs are shown, while the complete code can be found in the online repository [2].

The quadcopter attitude dynamics (6.3) is formalised in Isabelle/HOL as can be seen the following code

Isabelle/HOL code

definition "$att\_dyms\ \omega\ \omega'\ I\ C\ \Gamma\ \tau\ \tau_d = (\forall t \in T.\ (\forall \omega.\ \omega \in D3\_vec\_set) \wedge (\forall i.((\lambda t.\ \omega\$i)\ has\_derivative\ (\lambda t.\ \omega'\$i))(at\ t\ within\ T)) \wedge I\_mat\ I \wedge C\_fun\ C\ \Gamma\ I\ \omega\ \wedge \tau = I *_v \omega' + C + \tau_d)$"

and the torque vector $\boldsymbol{\tau}$ (6.4) is defined by bounding all propeller angular velocities Ω_i with their maximum value Ω_{max} as

Isabelle/HOL code

definition "*torq_fun* $\tau = ((\exists\ \Omega_1\ \Omega_2\ \Omega_3\ \Omega_4.\ |\ \Omega_1\ |< \Omega_{max} \wedge\ |\ \Omega_2\ |< \Omega_{max} \wedge\ |\ \Omega_3\ |< \Omega_{max} \wedge\ |\ \Omega_4\ |< \Omega_{max} \wedge \tau \in D3\_vec\_set \wedge \tau\$1 = \ell * l * (\Omega_2^2 - \Omega_4^2) \wedge \tau\$2 = \ell * l * (-\Omega_1^2 + \Omega_3^2) \wedge \tau\$3 = b * (-\Omega_1^2 + \Omega_2^2 - \Omega_3^2 + \Omega_4^2)))$"

The control law (6.12) and the control input $\boldsymbol{u}$ (6.19) are defined in the prover through the following code,

Isabelle/HOL code

definition "*cont_law* $(\tau :: (real, 3)vec)\ I_{hat}\ u\ u_d\ C_{hat} = (\tau = I_{hat} *_v u + u_d + C_{hat})$"

definition "*cont_u_def* $(u :: (real, 3)vec)\ \omega'_{ref}\ K_\omega\ K_q\ \xi'\ \xi = (u = \omega'_{ref} + K_\omega *_v \xi' + K_q *_v \xi)$"

The derivation (6.20) and (6.21) are formalised and proved in Isabelle/HOL based on the *att_dyms*, *cont_u* and *cont_law* (see the proof in the repository [2]). The closed-loop error dynamic (6.23) and (6.24) are implemented as

Isabelle/HOL code

lemma *Eq_6_23* :

assumes "$\forall t.\ t \in T$" **and** "*set_of_definitions* $\omega\ \omega_{ref}\ \omega'\ \omega'_{ref}\ u\ u_d\ \xi\ \xi'\ \xi''\ q\ q'\ q_r\ q'_r\ q_e\ \tau\ \tau_d\ \eta\ y\ \zeta\ C\ C_{hat}\ \Delta\ A\ G\ \Gamma\ Z_t\ Q\ P\ K_q\ K_\omega\ I\ I_{hat}$"

shows "$\eta' = A *_v \eta + G *_v (y - (matrix\_inv(I) *_v u_d))$"

proof -

have "$\xi'' = \omega'_{ref} - \omega'$" **using** *assms ddot_error_fun_def set_of_definitions_def* **by** *metis*

thus *? thesis* **by** (*smt G_mat_def assms*(2) *exhaust_3 set_of_definitions_def*)

qed

The *set_of_definitions* in the above code is a definition which is used to call all the pre-defined definitions. The assumptions (6.1-6.3) proposed in (6.25)-(6.28) are formalised in Isabelle/HOL as follow:

Isabelle/HOL code

definition "*assump1* $\omega'_{ref} = ((\text{SUP } t \in T.\ norm(\omega'_{ref})) < \alpha)$"

definition "*assump2* $I\ I_{hat} = (I\_mat\ I \ \wedge \ I_{hat}\_mat\ I_{hat} \ \wedge \ \lambda_{min} \leq norm(matrix\_inv(I)) \ \wedge$
$norm(matrix\_inv(I)) \leq \lambda_{max} \ \wedge \ norm(mat\ 1 - ((matrix\_inv(I)) ** I_{hat})) \leq \delta)$"

definition "*assump3* $(\tau_d :: (real, 3)vec) = (norm(\tau_d) \leq \gamma)$"

Stability analysis of the attitude controller as stated in (6.32)-(6.43) is implemented in Isabelle/HOL using a set of definitions (*definition*), several lemmas, (*lemma*), and short theorems in terms of theorem, (*theorem*). This structure of using several *lemmas* and *theorems* during the proof is due to the fact that the reasoning system of the theorem prover cannot handle long proofs with many assumptions, i.e. Isabelle system is unable to prove statements whcih have many equations if they are formalised in only one *lemma* or *theorem* style. However, the stability analysis starts by defining the candidate Lyapunov function V (6.32) is formalised as a *definition* in Isabelle/HOL:

Isabelle/HOL code

definition "*Lyapunov* $V\eta = (\forall t \in T.$ ***if*** $(\eta :: (real, 6)vec) \neq 0$ ***then*** $(\exists a.\ V(\eta) = (a :: real) \ \wedge$
$continuous\_on\ \ D6\_vec\_set\ V \ \wedge \ V(\eta) > 0)$ ***else*** $V(\eta) = 0)$"

Taking the candidate Lyapunov function V, the time derivative of Lyapunov function is derived and the derivations in (6.33)-(6.35) are proven symbolically and detailed in the online repository [2].

Isabelle/HOL code

theorem *Stb_Eq_6_33_6_35* :

assumes "$\forall \eta.\ \eta \neq 0$" **and** "$Lyapunov\ V\ \eta$" **and** "$V(\eta) = \eta \bullet (Q *_v \eta)$" **and** "$A\_mat\ A$" **and** "$\eta' = A *_v \eta +$
$G *_v (y - (matrix\_inv(I) *_v u_d))$" **and** "$(\forall t \in T.\ ((\lambda t.\ V(\eta))\ \ has\_derivative\ \ (\lambda t.\ V'(\eta)))(at\ t\ within\ T))$"

shows "$V'(\eta) = -(\eta \bullet (P *_v \eta)) + 2 * (((\eta\ v^*\ Q)\ v^*\ G) \bullet (y - matrix\_inv(I)\ *_v\ u_d))$"

proof - ... **qed**

The term $\boldsymbol{u}_d$ (6.36) is defined then the derivation in (6.37) is performed using Cauchy-Schwartz inequality (see "*theorem Eq_6_37*" in the repository [2]). Based on (6.38) and the upper bound of norm of $\boldsymbol{y}$ derived in (6.39) (see "*theorem Eq_6_39*" in the repository [2]), $\zeta(\boldsymbol{\eta}, t)$ (6.40) is obtained (see "*theorem Eq_6_40*" in the repository [2]). The terms $\boldsymbol{u}_d$ and $\zeta(\boldsymbol{\eta}, t)$ are implemented in the prover as "u_d_def" and "$zeta\_def$" respectively,

Isabelle/HOL code

definition "$u_d\_def\ u_d\ G\ Q\ \zeta\ \eta = (\forall t \in T.\ \textbf{\textit{if}}(norm(transpose(G) *_v (Q *_v \eta)) \geq \mu)\ \textbf{\textit{then}}\ (u_d = (\zeta/norm(transpose(G) *_v (Q *_v \eta))) *_s (transpose(G) *_v (Q *_v \eta)))\ \textbf{\textit{else}}\ (u_d = (\zeta/\mu) *_s (transpose(G) *_v (Q *_v \eta))))$"

definition "$zeta\_def\ \zeta\ (y :: (real, 3)vec) = (\forall\ t \in T.\ \exists \varepsilon.\ \varepsilon > 0\ \ \wedge\ \ norm(y) \leq \varepsilon \rightarrow\ \zeta \geq \varepsilon/\lambda_{min})$"

Note that the short arrow $\rightarrow$ in the code refers to *implies* while the longer $\longrightarrow$ refers to *convergence* in HOL.

Finally, based on all the above definitions and assumptions, it has been verified that the proposed control system is asymptotically stable since the time derivative of Lyapunov function in (6.41) and (6.43) is strictly negative for $\forall \boldsymbol{\eta} \neq 0$. It has also been proven that the tracking error converges to zero as the time converges to infinity, ($\|\boldsymbol{\eta}\| \longrightarrow 0$). The code below illustrates the symbolic proof in the Isabelle theorem prover.

Isabelle/HOL code

theorem *Stb_Eq_6_41_6_43* :

assumes "$\forall t.\ t \in T$" **and** "$(set\_of\_definitions\ \omega\ \omega_{ref}\ \omega^{'}\ \omega^{'}_{ref}\ u\ u_d\ \xi\ \xi^{'}\ \xi^{''}\ q\ q^{'}\ q_r\ q^{'}_r\ q_e\ \tau\ \tau_d\ \eta\ y\ \zeta\ C\ C_{hat}\ \Delta\ A\ G\ \Gamma\ Z_t\ Q\ P\ K_q\ K_\omega\ I\ I_{hat})$" **and** "$assump1\ \omega^{'}_{ref}$" **and** "$assump2\ I\ I_{hat}$" **and** "$assump3\ \tau_d$" **and** "$\forall \eta.\ \eta \neq 0$" **and** "$Lyapunov\ V\ \eta$" **and** "$V(\eta) = \eta \bullet (Q *_v \eta)$" **and** "$(\forall t \in T.\ ((\lambda t.\ V(\eta))\ has\_derivative\ (\lambda t.\ V^{'}(\eta)))(at\ t\ within\ T))$" **and** "$\omega^{'} = u - y + matrix\_inv(I) *_v u_d$" **and** "$\eta^{'} = A *_v \eta + G *_v (y - (matrix\_inv(I) *_v u_d))$" **and** "$V^{'}(\eta) = -(\eta \bullet (P *_v \eta)) + 2 * (((\eta v^* Q) v^* G) \bullet (y - matrix\_inv(I) *_v u_d))$"

shows "$norm(transpose(G) *_v (Q *_v \eta)) \geq \mu \Longrightarrow V^{'}(\eta) < 0$"

and "$norm(transpose(G) *_v (Q *_v \eta)) < \mu \Longrightarrow V^{'}(\eta) < 0$" **and** "$((\lambda t.norm(\eta)) \longrightarrow 0)(at\ t\ within\ T)$"

proof -

show "$norm(transpose(G) *_v (Q *_v \eta)) \geq \mu \Longrightarrow V^{'}(\eta) < 0$" **using** *assms Eq_19 rel_simps*(93) **by**

```
metis
then show "norm(transpose(G) *v (Q *v η)) < μ ⟹ V'(η) < 0" using assms Eq_19 rel_simps(93)
by metis
show "((λt.norm(η)) ⟶ 0)(at t within T)" using assms by auto
qed
```

7.3.2 Onboard Verification for a Safe Flight using MetiTarski prover

The control system of the UAV can be designed, simulated and verified at the model/design stage. The designed controller is then formalised to code and implemented into the autopilot system, which controls the UAV trajectory. The UAV controlled by the autopilot can be exposed to gusts of wind which may cause unstable flight. In this case, the autopilot system cannot be informed if the UAV has entered an unstable region which may cause a crash or the loss of human(s) life. Therefore, we have proposed the use of ATP tool represented by MetiTarski prover for onboard verification of the stability state of the UAV and to inform the autopilot in case of any unstable behaviour detected.

The autopilot then can send warning messages to the user/pilot or base station to perform, for instance, an emergency safe landing using autolanding techniques such as in [86]. This will ensure safer flight and may avoid losing the UAV or any harm to humans and properties. The MetiTarski ATP is chosen in this work to verify the controller stability of the UAV due to its ability of deal with inequalities of numerical real numbers. Unlike the previous verification stage using Isabelle, MetiTarski proves the statements automatically without the need to any interaction with the designer/engineer.

MetiTarski can be implemented on the autopilot's electronic board such as PixHawk [5] or Navio2 [46] Raspberry Pi. These electronic autopilots use the Linux operating system where MetiTarski can also be installed. Therefore, an interface between the two systems (autopilot and MetiTarski) is easy to create in practice. The proposed onboard framework is illustrated in Fig.7.3. The applicability of this approach is illustrated in simulation by interfacing the Simulink/Matlab

model with MetiTarski and test the stability of the control system. Note that the quadcopter that has been implemented in Simulink considers the nonlinear dynamics of the craft based on the MathWorks model in [61], which is widely used.

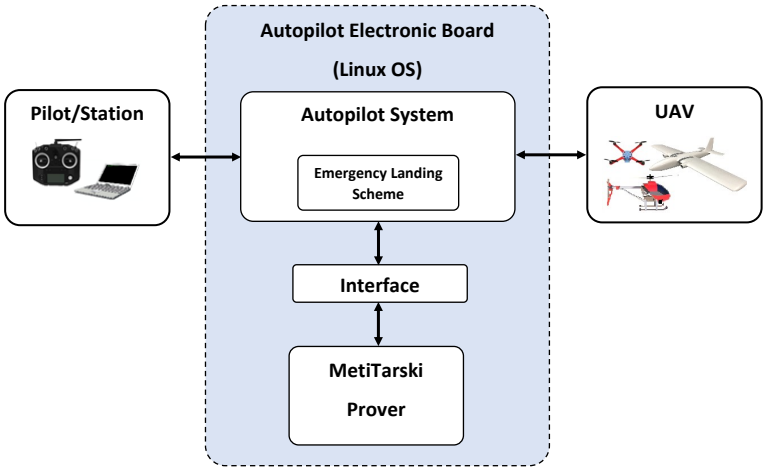


Figure 7.3: Onboard verification framework of UAVs

Considering the stability analysis stated in (6.41) and (6.43), the time derivative of the Lyapunov function will be tested to check whether it is negative definite or not. If it is not negative definite, then this indicates that the control system is out of its stability region, hence the autopilot can pass warning messages to the pilot or station to take an action or perform an emergency landing. The verification process starts by formalising the stability equations (6.41) and (6.43) into a FOL syntax. The parameters of these equations are passed from the autopilot system to MetiTarski prover via an interface scheme. Afterwards, the test is conducted in the MetiTarski prover such that the derivative of the Lyapunov function is negative ($\dot{V}(\boldsymbol{\eta}) < 0$). The above procedures are simulated in Simulink/Matlab to illustrate its applicability. The stability equations (6.41) and (6.43), are simplified using symbolic computations in Matlab before formalising them into FOL syntax. The parameters included in both stability equations are passed from Simulink/Matlab to MetiTarski to perform monitor testing. The following code shows an example of stability check in MetiTarski prover for (6.41) (note that the full code can be found in the online repository [2]; where the notations used in

the code are described in Table 7.1):

MetiTarski code

$fof(Stability\_Eq6\_41, conjecture, ![E\_1, E\_2, E\_3, E\_4, E\_5, E\_6, Phi, Theta] :?[Y\_1, Y\_2, Y\_3, Zeta\_E] :$

$\%assumptions$

$(E\_1 = 0.0037$ & $E\_2 = 0.004964$ & $E\_3 = 0.014124$ & $E\_4 = 0.0504$ & $E\_5 = 0.05748$ & $E\_6 = 0.03166$

& $Phi > -1.5708$ & $Phi < 1.5708$ & $Theta > -1.5708$ & $Theta < 1.5708$

& $abs(Y\_1) <= (0.04231 * (180.7904 + (0.9 * abs(E\_4)) + (16 * abs(E\_1))) + (171.47 * (0.332 + 0.4231)))$

& $abs(Y\_2) <= (0.04231 * (180.7904 + (0.9 * abs(E\_5)) + (16 * abs(E\_2))) + (171.47 * (0.332 + 0.4231)))$

& $abs(Y\_3) <= (0.04231 * (180.7904 + (0.0064 * abs(E\_6)) + (25 * abs(E\_3))) + (171.47 * (0.332 + 0.4231)))$

& $Zeta\_E > 0$ & $Zeta\_E >= sqrt(Y\_1^2 + Y\_2^2 + Y\_3^2)/171.045$

$\%implies$

$=> .... < 0)).$

The time required for MetiTarski prover to generate the proof for (6.41) was 0.324 seconds. Note that all proofs in MetiTarski were on a Linux Ubuntu operating system, Core i5 1.6 GHz CPU and 8 GB RAM. For (6.43), the code as below, where the time required to generate the proof was 0.240 seconds.

MetiTarski code

$fof(Stability\_Eq6\_43, conjecture, ![E\_1, E\_2, E\_3, E\_4, E\_5, E\_6, Phi, Theta] :?[Y\_1, Y\_2, Y\_3, Zeta\_E] :$

$\%assumptions$

$(E\_1 = 0.001227$ & $E\_2 = 0.001241$ & $E\_3 = 0.007062$ & $E\_4 = 0.0168$ & $E\_5 = 0.01437$ & $E\_6 = 0.01583$

& $Phi > -1.5708$ & $Phi < 1.5708$ & $Theta > -1.5708$ & $Theta < 1.5708$

& $abs(Y\_1) <= (0.04231 * (180.7904 + (0.9 * abs(E\_4)) + (16 * abs(E\_1))) + (171.47 * (0.332 + 0.4231)))$

& $abs(Y\_2) <= (0.04231 * (180.7904 + (0.9 * abs(E\_5)) + (16 * abs(E\_2))) + (171.47 * (0.332 + 0.4231)))$

& $abs(Y\_3) <= (0.04231 * (180.7904 + (0.0064 * abs(E\_6)) + (25 * abs(E\_3))) + (171.47 * (0.332 + 0.4231)))$

& $Zeta\_E > 0$ & $Zeta\_E >= sqrt(Y\_1^2 + Y\_2^2 + Y\_3^2)/171.045$

$\%implies$

$=> .... < 0)).$

Table 7.1: Variables and vectors notations in MetiTarski

Variable/Vector	***Notation***
ϕ	Phi
θ	$Theta$
$\boldsymbol{\eta}$	E
$\boldsymbol{y}$	Y
$\zeta(\boldsymbol{\eta}, t)$	$Zeta\_E$

7.4 Discussion

From the work conducted so far, it has been found that Isabelle/HOL prover is a powerful tool to verify control systems. However, there are several drawbacks such as many of control concepts and theories need to be implemented to the prover in order to carry out the proof. For example, the concepts of stability analysis, norms of real numbers with their properties, signals properties, time and frequency domain, Laplace and Z transforms, inequality properties over real and complex numbers, etc. Moreover, the automation tools in Isabelle required to be enhanced as a lot of human-machine interactions were needed to conduct the proofs. For the MetiTarski prover, it is a good tool to conduct inequalities checking and verification over the real numbers field. However, there are several limitations to the use of MetiTarski prover in control systems verification. For instance, it supports a limited number of variables during the prove which make it impossible to prove controllers with a high number of variables such as more than ten variables. In addition, it is a FOL and therefore can only support scalar number (no vectors or matrices are supported), where the term to be verified which includes vectors and matrices need to be simplified into scalars first then implemented in the MetiTarski prover. All the above drawbacks can be overtaken by collaborative work between control engineers and computer science experts for further developing and enhancing of the Isabelle/HOL and MetiTarski to be utilised to verify more control system such as complex, intelligent and adaptive controllers.

7.5 Chapter Summary

This chapter has introduced a new verification framework for safety-critical control systems by applying the power of a higher-order-logic-based interactive theorem provers and a first-order logic-based automated theorem prover to verify the control system of unmanned aerial vehicles and to ensure UAV safety during the flight. The framework relies on two stages, the first is for verifying the design of the control system and its stability and the second is for onboard monitoring the UAV's stability to ensure flight safety. The framework has been demonstrated on a robust attitude controller of a generic quadcopter UAV to verify the correctness of the design and stability analysis in addition to onboard monitoring the conditions of its dynamical stability while the UAV is flying. The UAV's attitudes are controlled by a nonlinear robust controller, which is designed using inverse dynamics control and it takes into account dynamical uncertainty and external disturbances.

The methods used in the verification stages go significantly beyond symbolic computation of inequalities for the Lyapunov theory as concepts of convergence as mappings of functions and quantifications over sets of functions are used in Isabelle and as such they were not be found in prior literature in aviation control systems.

Chapter 8

Conclusions and Future Work

8.1 Overview

This chapter presents the conclusions of the thesis contributions and outlines the possible future work to extend the research.

8.2 Conclusions

This thesis presented new verification schemes for safety-critical systems such as unmanned aerial vehicles. It is intended to fill the gap between control engineering and existing verification methods of control code. The motivation is to solve verification problems of digital control systems on physical plants using formal methods for symbolic computations, which includes verification of control theory as well. The thesis illustrated the applicability and the power of interactive theorem provers relying on HOL and automated theorem prover represented by FOL for control theory.

Formal methods represented by interactive theorem proving are used in this thesis to show the possibility of formally prove of control theories on computers. This is illustrated by an example using Isabelle/HOL (Higher-Order Logic) proof assistant to formally proof a general version of the Small-Gain Theorem for feedback control systems. This work has indicated that even the most theoretical control concepts involving nonlinear operators, causality and normed spaces of

signals over the infinite semi-axis of time can be formally handled by theorem proving techniques. It is also found that other control theories can be formalised and proved using these tools.

Another verification approach is presented by verifying the stability of unmanned aerial vehicles based on Lyapunov's direct method using the MetiTarski automated theorem prover. This is illustrated by designing a nonlinear attitude controllers for a quadcopter and a small-scale helicopter unmanned aerial vehicle and their stability proven using the MetiTarski prover. The control systems were implemented in Simulink/Matlab and the simulation results have been shown. The verification results show that control system stability can be verified using automated theorem provers to guarantee asymptotic stability of the controller and to ensure that the system works within the given bounds and performance specifications.

A new verification framework of unmanned aerial vehicles is introduced. The framework includes two stages. The first stage is concerned with formally verifying the correctness of the controller design and stability analysis at the design stage using interactive theorem proving tools. This includes checking the validity of mathematical derivations of the control law and its stability, hence ensure system performance and robustness. The second stage is for onboard stability monitoring of the aircraft during the flight. This stage is developed to monitor the vehicle's stability and if the aircraft violated by gusts of wind which affect its stability, the autopilot can avoid aggressive manoeuvres or may perform an emergency landing in a safe place.

A novel robust nonlinear dynamic inversion controller (RNDI) is designed and presented for multi-rotor unmanned aerial vehicles. The controller consists of two loops: an inner loop for attitude control and an outer loop for lateral and vertical position control. The controller considered both inertial modelling uncertainty and external disturbances. The control scheme has been simulated in Simulink/-Matlab based on a nonlinear multi-rotor model developed by MathWorks in order to test the control performance. The RNDI controller has been compared with a competitive nonlinear controller to illustrate its performance. The results indicate that the new controller can make a craft tolerant to payload changes and to large wind gusts. Ultimately, the results of this work may enhance the safety of

multi-rotor unmanned aerial vehicles.

To implement the proposed verification framework, formal verification of the RNDI controller is addressed and solved in this thesis for unmanned aerial vehicles. An interactive theorem prover is applied to test the validity of the novel and robust nonlinear control law within a controllability domain that describes its flight envelop. Innovative symbolic computation is used to prove the validity of a highly abstract control theory that verifies the robustness of the controller. The presented symbolic computational technique is able to uncover inaccuracies in the mathematical arguments of pen and paper based proofs with numerical values on limits of performance. The technique makes the verification of the entire control system, including the control scheme and its software, more reliable. As such, the approach can point the way to formal verification of safety critical aviation systems in general. Equipped with with controller verification, aircraft can be programmed to prevent its crash under challenging environmental conditions, by deciding or proposing.

From the work conducted so far, it has been found that there are some limitations of using formal methods for control system verification. First of all, many control concepts and theories need to be developed and implemented in theorem proving tools in order to make the verification of control system easier in addition to speeding up the process. Moreover, complex control schemes may be difficult to implement in theorem proving such as intelligent and adaptive controllers, which is due to the limitation of current techniques in theorem proving. This could be overtaken, if possible, by either taking abstract of the design or verifying some part of it. Furthermore, the verification process needs the dynamical model of the system to be controlled to perform the verification with the designed control scheme. Finally, control engineers need to be familiar with formal methods tools in order to conduct the formal verification process as a part of the control system design and analysis.

The novelty of this thesis is to demonstrate that formal methods in some theorem provers are suitable to verify and prove the correctness of robust control theory for prescribed flight envelop of multi-rotor unmanned vehicles. Although prior work suggested this may be a possibility, this is a first evidence of this kind. This involves formal stability analysis to guarantee system's robustness then en-

sure aircraft's safety by conducting continuous onboard stability monitoring using interactive and automated theorem provers. The methods are implemented in Isabelle and MetiTarski, and the codes have been made available online. This is promising and may encourage the use of such methods in control system verification of safety-critical systems in general. The symbolic methods are generic and potentially generalise to verification to a variety of industrial control systems, where performance loss is damaging and therefore analysis is important to be carried out formally.

8.3 Future Work

There is the prospect to address various topics and challenges as follows:

- Formally prove more control theories that are useful for verifying practical control systems using interactive theorem proving techniques. This will enrich the library of control theories in these provers. Hence, if some control theories are formally proven using these techniques, the results will be significant for robustness and safety of safety-critical systems.
- Implementing the proposed robust nonlinear controller of quadcopter UAV presented in Chapter 6 in practice.
- Implementing *Stage 2* of the proposed verification framework presented in Chapter 7 in practice for stability monitoring of the aircraft.
- Extending the verification framework by adding code verification to the procedures to ensure that the code complies with the design.
- Extending the verification framework by adding software verification to the procedures using theorem proving in addition to using other formal methods techniques such as abstract interpretation and model checking.
- Extending the verification framework by adding software verification to the procedures using theorem proving in addition to model checking techniques.

- Using the verification framework in other safety-critical applications such as autonomous cars, surgical robotics, field robotics, etc.

Bibliography

[1] ERATO Metamathematics for Systems Design (MMSD) project. http://www.jst.go.jp/erato/hasuo/en/. Accessed: 11 February 2020. 33

[2] Formal methods in control engineering: web-repository. https://figshare.com/s/b996fbf3bed1d624a70b. 41, 49, 62, 76, 78, 115, 116, 117, 118, 120

[3] Integrated Tool Chain for Model-based Design of Cyber-Physical Systems (INTO-CPS) Project. http://projects.au.dk/into-cps/. Accessed: 11 February 2020. 33

[4] NASA Langley's Formal Methods Research Program, NASA. https://shemesh.larc.nasa.gov/fm/index.html. Accessed: 11 February 2020. 33

[5] PixHawk. http://pixhawk.org/. Accessed: 8 January 2020. 119

[6] System on TPTP. http://www.tptp.org/cgi-bin/SystemOnTPTP. Accessed: 4 January 2020. 26

[7] A verification toolbox for Isabelle/HOL based on unifying theories of programming (Isabelle/UTP). https://www-users.cs.york.ac.uk/~simonf/utp-isabelle/. Accessed: 11 February 2020. 33

[8] Behzad Akbarpour and Lawrence C Paulson. Applications of metitarski in the verification of control and hybrid systems. In *International Workshop on Hybrid Systems: Computation and Control*, pages 1–15. Springer, 2009. 34

[9] D. Araiza-Illan, K. Eder, and A. Richards. Formal verification of control systems' properties with theorem proving. In *2014 UKACC International Conference on Control (CONTROL)*, pages 244–249, July 2014. 34

[10] D. Araiza-Illan, K. Eder, and A. Richards. Verification of control systems implemented in simulink with assertion checks and theorem proving: A case study. In *2015 European Control Conference (ECC)*, pages 2670–2675, July 2015. 34

[11] N. Aréchiga, S. M. Loos, A. Platzer, and B. H. Krogh. Using theorem provers to guarantee closed-loop system properties. In *2012 American Control Conference (ACC)*, pages 3573–3580. IEEE, June 2012. 81

[12] Ardupilot. Ardupilot: Copter Home. http://ardupilot.org/copter/index.html. Accessed: 15 January 2020. 65

[13] SF Armanini, M Polak, James E Gautrey, A Lucas, and James F Whidborne. Decision-making for unmanned aerial vehicle operation in icing conditions. *CEAS Aeronautical Journal*, **7**[4]:663–675, 2016. 109

[14] Clark W Barrett, Roberto Sebastiani, Sanjit A Seshia, and Cesare Tinelli. Satisfiability modulo theories. *Handbook of satisfiability*, **185**:825–885, 2009. 24

[15] Cinzia Bernardeschi, Andrea Domenici, and Paolo Masci. A PVS-simulink integrated environment for model-based analysis of cyber-physical systems. *IEEE Transactions on Software Engineering*, **44**[6]:512–533, 2018. 81

[16] Marius Beul, Rainer Worst, and Sven Behnke. Nonlinear model-based position control for quadrotor UAVs. In *ISR/Robotik 2014; 41st International Symposium on Robotics*, pages 1–6. VDE, 2014. 30

[17] P. E. Black, K. M. Hall, M. D. Jones, T. N. Larson, and P. J. Windley. A brief introduction to formal methods [hardware design]. In

Proceedings of Custom Integrated Circuits Conference, pages 377–380, May 1996. 23

[18] J. Boubeta-Puig, E. Moguel, F. Sánchez-Figueroa, J. Hernández, and J. Carlos Preciado. An autonomous UAV architecture for remote sensing and intelligent decision-making. *IEEE Internet Computing*, **22**[3]:6–15, May 2018. 109

[19] Richard J Boulton, Hanne Gottliebsen, Ruth Hardy, Tom Kelsey, and Ursula Martin. Design verification for control engineering. In *International Conference on Integrated Formal Methods*, pages 21–35. Springer, 2004. 35

[20] Guillaume Brat, David Bushnell, Misty Davies, Dimitra Giannakopoulou, Falk Howar, and Temesghen Kahsai. Verifying the safety of a flight-critical system. In *International Symposium on Formal Methods*, pages 308–324. Springer, 2015. 35

[21] Christopher W Brown. QEPCAD B: a program for computing with semi-algebraic sets using cads. *ACM SIGSAM Bulletin*, **37**[4]:97–108, 2003. 26

[22] James Buffington, Vince Crum, Bruce Krogh, Clinton Plaisted, Ravi Prasanth, Prasanta Bose, and Tim Johnson. Validation & verification of intelligent and adaptive control systems. In *2nd AIAA" Unmanned Unlimited" Conf. and Workshop & Exhibit*, page 6603, 2003. 1

[23] Elisa Capello, Giorgio Guglieri, Fulvia Quagliotti, and Daniele Sartori. Design and validation of an $\mathcal{L}_1$ adaptive controller for mini-uav autopilot. *Journal of Intelligent & Robotic Systems*, **69**[1-4]:109–118, 2013. 3

[24] Paul Caspi, Adrian Curic, Aude Maignan, Christos Sofronis, and Stavros Tripakis. Translating discrete-time simulink to lustre. In *International Workshop on Embedded Software*, pages 84–99. Springer, 2003. 34

[25] Pedro Castillo, Rogelio Lozano, and Alejandro E Dzul. *Modelling and control of mini-flying machines*. Springer-Verlag, 2005. 67, 68

[26] A. Cavalcanti and P. Clayton. Verification of control systems using circus. In *11th IEEE International Conference on Engineering of Complex Computer Systems (ICECCS'06)*, pages 10 pp.–, 2006. 35

[27] HaiYang Chao, YongCan Cao, and YangQuan Chen. Autopilots for small unmanned aerial vehicles: a survey. *International Journal of Control, Automation and Systems*, **8**[1]:36–44, 2010. 3

[28] Xi Chen and Gang Chen. Formal verification of helicopter automatic landing control algorithm in theorem prover coq. *International Journal of Performability Engineering*, **14**[9]:1947, 2018. 81

[29] Jack CK Chou. Quaternion kinematic and dynamic differential equations. *IEEE Transactions on robotics and automation*, **8**[1]:53–64, 1992. 18

[30] Edmund M Clarke, Orna Grumberg, and Doron Peled. *Model checking*. London: MIT press, 1999. 24

[31] Edmund M Clarke and Jeannette M Wing. Formal methods: State of the art and future directions. *ACM Computing Surveys (CSUR)*, **28**[4]:626–643, 1996. 23

[32] Michael V Cook. *Flight dynamics principles*. Elsevier, 2007. 13

[33] Abhijit Das, Kamesh Subbarao, and Frank Lewis. Dynamic inversion with zero-dynamics stabilisation for quadrotor control. *IET control theory & applications*, **3**[3]:303–314, 2009. 31

[34] Sergey N Dashkovskiy, Björn S Rüffer, and Fabian R Wirth. Small gain theorems for large scale systems and construction of iss lyapunov functions. *SIAM Journal on Control and Optimization*, **48**[6]:4089–4118, 2010. 21

[35] Emanuele L de Angelis, Fabrizio Giulietti, and Goele Pipeleers. Two-time-scale control of a multirotor aircraft for suspended load transportation. *Aerospace Science and Technology*, **84**:193–203, 2019. 31

[36] Leonardo De Moura and Nikolaj Bjørner. Z3: An efficient smt solver. *Tools and Algorithms for the Construction and Analysis of Systems*, pages 337–340, 2008. 26

[37] William Denman and César Muñoz. Automated real proving in PVS via MetiTarski. In *International Symposium on Formal Methods*, pages 194–199. Springer, 2014. 33

[38] William Denman, Mohamed H. Zaki, Sofiène Tahar, and Luis Rodrigues. Towards flight control verification using automated theorem proving. In Mihaela Bobaru, Klaus Havelund, Gerard J. Holzmann, and Rajeev Joshi, editors, *NASA Formal Methods*, pages 89–100, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg. 34

[39] Charles A Desoer and Mathukumalli Vidyasagar. *Feedback systems: input-output properties*. SIAM, 2nd edition, 2009. 21

[40] James Diebel. Representing attitude: Euler angles, unit quaternions, and rotation vectors. *Stanford University Report*, 2006. 19

[41] T. Dierks and S. Jagannathan. Output feedback control of a quadrotor UAV using neural networks. *IEEE Transactions on Neural Networks*, **21**[1]:50–66, Jan 2010. 28

[42] Patrick Doherty, Jonas Kvarnström, and Fredrik Heintz. A temporal logic-based planning and execution monitoring framework for unmanned aircraft systems. *Autonomous Agents and Multi-Agent Systems*, **19**[3]:332–377, 2009. 109

[43] Andrea Domenici, Adriano Fagiolini, and Maurizio Palmieri. Integrated simulation and formal verification of a simple autonomous vehicle. In Antonio Cerone and Marco Roveri, editors, *Software Engineering*

and Formal Methods, pages 300–314, Cham, 2018. Springer International Publishing. 81

[44] Richard C Dorf and Robert H Bishop. *Modern control systems*. New Jersey: Pearson Prentice-Hall, 11th edition, 2008. 4

[45] Rolf Drechsler. *Formal verification of circuits*. Springer Science & Business Media, 2013. 24

[46] Emlid. Navio2. https://emlid.com/navio/. Accessed: 8 January 2020. 119

[47] Bernard Etkin and Lloyd Duff Reid. *Dynamics of flight: stability and control*, **3**. Wiley New York, 1996. 17

[48] Rick Evertsz, John Thangarajah, Nitin Yadav, and Thanh Ly. A framework for modelling tactical decision-making in autonomous systems. *Journal of Systems and Software*, **110**:222–238, 2015. 109

[49] S. Fang, Y. Xu, J. Jiang, B. Hu, and X. Que. The analysis on posture control of micro quadrotor based on PID. In *2011 Fourth International Symposium on Computational Intelligence and Design*, **2**, pages 283–286, Oct 2011. 28

[50] E. Feron. From control systems to control software. *IEEE Control Systems Magazine*, **30**[6]:50–71, Dec 2010. 5

[51] Eric Feron. From control systems to control software. *IEEE Control Systems*, **30**[6]:50–71, 2010. 35

[52] Jean-Christophe Filliâtre and Andrei Paskevich. Why3 — where programs meet provers. In Matthias Felleisen and Philippa Gardner, editors, *Programming Languages and Systems*, pages 125–128, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg. 34

[53] US Food, Drug Administration, et al. Medical device recall report fy2003 to fy2012. *Center for Devices and Radiological Health*, 2012. 23

[54] Abraham Adolf Fraenkel, Yehoshua Bar-Hillel, and Azriel Levy. *Foundations of set theory*, **67**. Elsevier, 1973. 27

[55] Chunyang Fu, Wei Hong, Huiqiu Lu, Lei Zhang, Xiaojun Guo, and Yantao Tian. Adaptive robust backstepping attitude control for a multi-rotor unmanned aerial vehicle with time-varying output constraints. *Aerospace Science and Technology*, **78**:593–603, 2018. 30

[56] G. Kahn G. Huet and C. Paulin-Mohring. The coq proof assistant: A tutorial. In *Technical Report 178 [Online]. Available: http://cs.swan.ac.uk/ csoliver/ok-sat-library/OKplatform/ExternalSources/sources/Coq/Tutorial.pdf*. National Institute of Research in Information and Automation (INRIA), 2009. 27

[57] Khalil Ghorbal, Jean-Baptiste Jeannin, Erik Zawadzki, André Platzer, Geoffrey J Gordon, and Peter Capell. Hybrid theorem proving of aerospace systems: Applications and challenges. *Journal of Aerospace Information Systems*, **11**[10]:702–713, 2014. 35

[58] Torkel Glad and Lennart Ljung. *Control Theory*. CRC Press, 2000. 21

[59] Godfrey Harold Hardy, John Edensor Littlewood, and George Pólya. *Inequalities*. Cambridge university press, 1952. 45

[60] Ruth Hardy. *Formal methods for control engineering: A validated decision procedure for Nichols Plot analysis*. PhD thesis, University of St Andrews, 2006. 34

[61] Brad Horton and MathWorks Australia. Modelling, simulation and control of a quadcopter. In *MATLAB Academic Conference. Australia and New Zealand*, pages 4–14, 2016. [Accessed: 8 November 2018]. 94, 120

[62] Xing Huo, Mingyi Huo, and Hamid Reza Karimi. Attitude stabilization control of a quadrotor uav by using backstepping approach. *Mathematical Problems in Engineering*, **2014**, 2014. 30

[63] Joe Hurd. First-order proof tactics in higher-order logic theorem provers. *Design and Application of Strategies/Tactics in Higher Order Logics, number NASA/CP-2003-212448 in NASA Technical Reports*, pages 56–68, 2003. 26

[64] Hiroshi Ito, Sergey Dashkovskiy, and Fabian Wirth. On a small gain theorem for networks of iISS systems. In *Proceedings of the 48h IEEE Conference on Decision and Control (CDC) held jointly with 2009 28th Chinese Control Conference*, pages 4210–4215. IEEE, 2009. 21

[65] O. A. Jasim and S. M. Veres. Towards formal proofs of feedback control theory. In *2017 21st International Conference on System Theory, Control and Computing (ICSTCC)*, pages 43–48, Oct 2017. 81

[66] Z-P Jiang, Andrew R Teel, and Laurent Praly. Small-gain theorem for ISS systems and applications. *Mathematics of Control, Signals and Systems*, **7**[2]:95–120, 1994. 21

[67] Romain J Jobredeaux. *Formal verification of control software*. PhD dissertation, Georgia Institute of Technology, 2015. 5, 35

[68] Eric N Johnson and Suresh K Kannan. Adaptive trajectory control for autonomous helicopters. *Journal of Guidance, Control, and Dynamics*, **28**[3]:524–538, 2005. 31

[69] Iasson Karafyllis and Zhong-Ping Jiang. A small-gain theorem for a wide class of feedback systems with control applications. *SIAM Journal on Control and Optimization*, **46**[4]:1483–1517, 2007. 21

[70] Hassan K Khalil. *Noninear Systems*. Prentice-Hall, New Jersey, 1996. 4, 21, 38, 39, 45, 47

[71] Jinhyun Kim, Min-Sung Kang, and Sangdeok Park. Accurate modeling and robust hovering control for a quad-rotor VTOL aircraft. In *Selected papers from the 2nd International Symposium on UAVs, Reno, Nevada, USA June 8–10, 2009*, pages 9–26. Springer, 2009. 30

[72] Matija Krznar, Denis Kotarski, Petar Piljek, and Danijel Pavković. On-line inertia measurement of unmanned aerial vehicles using on board sensors and bifilar pendulum. *Interdisciplinary Description of Complex Systems: INDECS*, **16**[1]:149–161, 2018. 103

[73] Dong-Ah Lee, Sangkyung Sung, Junbeom Yoo, and Doo-Hyun Kim. Formal modeling and verification of operational flight program in a small-scale unmanned helicopter. *Journal of Aerospace Engineering*, **25**[4]:530–540, 2011. 31

[74] S Lee, C Ha, and BS Kim. Adaptive nonlinear control system design for helicopter robust command augmentation. *Aerospace science and technology*, **9**[3]:241–251, 2005. 31

[75] François Léonard, Adnan Martini, and Gabriel Abba. Robust nonlinear controls of model-scale helicopters under lateral and vertical wind gusts. *IEEE Transactions on Control Systems Technology*, **20**[1]:154–163, 2012. 31

[76] N. G. Leveson and C. S. Turner. An investigation of the therac-25 accidents. *Computer*, **26**[7]:18–41, July 1993. 23

[77] J. Li and Y. Li. Dynamic analysis and PID control for a quadrotor. In *2011 IEEE International Conference on Mechatronics and Automation*, pages 573–578, Aug 2011. 28

[78] X. Liang, Y. Fang, and N. Sun. A novel nonlinear backstepping-based control approach for quadrotor unmanned aerial vehicle transportation systems. In *2017 36th Chinese Control Conference (CCC)*, pages 884–889, July 2017. 30

[79] Daniel Liberzon and Dragan Nešić. Stability analysis of hybrid systems via small-gain theorems. In *International Workshop on Hybrid Systems: Computation and Control*, pages 421–435. Springer, 2006. 21

[80] Hao Liu, Dafizal Derawi, Jonghyuk Kim, and Yisheng Zhong. Robust optimal attitude control of hexarotor robotic vehicles. *Nonlinear dynamics*, **74**[4]:1155–1168, 2013. 31

[81] Hao Liu, Jianxiang Xi, and Yisheng Zhong. Robust motion control of quadrotors. *Journal of the Franklin Institute*, **351**[12]:5494–5510, 2014. 30

[82] Lijun Long and Jun Zhao. A small-gain theorem for switched interconnected nonlinear systems and its applications. *IEEE Transactions on Automatic Control*, **59**[4]:1082–1088, 2013. 21

[83] R Lopéz, I Gonzalez, J Flores, J Ordaz, Sergio Salazar, and Rogelio Lozano. Real time parameter identification of the inertia tensor for a quad-rotor mini-aircraft using adaptive control. *IFAC Proceedings Volumes*, **46**[30]:32–37, 2013. 103

[84] Ricardo López-Gutiérrez, Abraham Efraim Rodriguez-Mata, Sergio Salazar, Ivan González-Hernández, and Rogelio Lozano. Robust quadrotor control: attitude and altitude real-time results. *Journal of Intelligent & Robotic Systems*, **88**[2-4]:299–312, 2017. 31

[85] Rogelio Lozano. *Unmanned aerial vehicles: Embedded control*. John Wiley & Sons, 2013. 16

[86] Parker C Lusk, Patricia C Glaab, Louis J Glaab, and Randal W Beard. Safe2ditch: Emergency landing for small unmanned aircraft systems. *Journal of Aerospace Information Systems*, pages 1–13, 2019. 119

[87] T. Madani and A. Benallegue. Sliding mode observer and backstepping control for a quadrotor unmanned aerial vehicles. In *2007 American Control Conference*, pages 5887–5892, July 2007. 31

[88] Iven MY Mareels. Monotone stability of nonlinear feedback systems. *Journal of Mathematical Systems, Estimation and Control*, **2**:275–291, 1992. 21

[89] Mehmet O. Efe. Neural network assisted computationally simple $PI^{\lambda}D^{\mu}$ control of a quadrotor UAV. *IEEE Transactions on Industrial Informatics*, **7**[2]:354–361, May 2011. 30

[90] Tom F Melham. *Higher order logic and hardware verification*, **31**. Cambridge University Press, 2009. 2

[91] Greg Michaelson. *An introduction to functional programming through lambda calculus*. Courier Corporation, 2011. 27, 28

[92] Robin Milner. *The definition of standard ML: revised*. MIT press, 1997. 27, 28

[93] Jean-François Monin. *Understanding formal methods*. Springer Science & Business Media, 2012. 23

[94] César A Muñoz. Formal methods in air traffic management: The case of unmanned aircraft systems (invited lecture). In *International Colloquium on Theoretical Aspects of Computing*, pages 58–62. Springer, 2015. 33

[95] César A Munoz, Aaron Dutle, Anthony Narkawicz, and Jason Upchurch. Unmanned aircraft systems in the national airspace system: a formal methods perspective. *ACM SIGLOG News*, **3**[3]:67–76, 2016. 33

[96] Tobias Nipkow, Lawrence C Paulson, and Markus Wenzel. *Isabelle/HOL: a proof assistant for higher-order logic*, **2283**. Springer Science and Business Media, 2002. 32

[97] B. Nuseibeh. Ariane 5: Who dunnit? *IEEE Software*, **14**[3]:15–16, May 1997. 23

[98] S. Owre, J. M. Rushby, and N. Shankar. PVS: A prototype verification system. In Deepak Kapur, editor, *Automated Deduction—CADE-11*, pages 748–752, Berlin, Heidelberg, 1992. Springer Berlin Heidelberg. 32, 33

[99] Gareth D Padfield. *Helicopter flight dynamics: the theory and application of flying qualities and simulation modelling*. John Wiley & Sons, 2008. 66

[100] Lawrence C Paulson. Metitarski: Past and future. In *International Conference on Interactive Theorem Proving*, pages 1–10. Springer, 2012. 26, 32, 65

[101] Doron A Peled. *Software reliability methods*. Springer Science & Business Media, 2013. 24, 35

[102] André Platzer and Jan-David Quesel. KeYmaera: A hybrid theorem prover for hybrid systems (system description). In *International Joint Conference on Automated Reasoning*, pages 171–178. Springer, 2008. 35

[103] Heriberto Ramirez-Rodriguez, Vicente Parra-Vega, Anand Sanchez-Orta, and Octavio Garcia-Salazar. Robust backstepping control based on integral sliding modes for tracking of quadrotors. *Journal of Intelligent & Robotic Systems*, **73**[1-4]:51–66, 2014. 30

[104] Mardavij Roozbehani, Alexandre Megretski, and Eric Feron. Optimization of lyapunov invariants in verification of software systems. *IEEE Transactions on Automatic Control*, **58**[3]:696–711, 2013. 24, 25

[105] Pritam Roy and Natarajan Shankar. Simcheck: a contract type system for simulink. *Innovations in Systems and Software Engineering*, **7**[2]:73–83, 2011. 34

[106] J. J. Ruz, O. Arevalo, G. Pajares, and J. M. de la Cruz. Decision making among alternative routes for UAVs in dynamic environments. In *2007 IEEE Conference on Emerging Technologies and Factory Automation (EFTA 2007)*, pages 997–1004, Sep. 2007. 109

[107] Mahendra Kumar Samal, Matthew Garratt, Hemanshu Pota, and Hamid Teimoori Sangani. Model predictive attitude control of vario unmanned helicopter. In *IECON 2011-37th Annual Conference on IEEE Industrial Electronics Society*, pages 622–627. IEEE, 2011. 31

[108] Aykut C Satici, Hasan Poonawala, and Mark W Spong. Robust optimal control of quadrotor UAVs. *IEEE Access*, **1**:79–93, 2013. 30

[109] N. Scaife, C. Sofronis, P. Caspi, S. Tripakis, and F. Maraninchi. Defining and translating a "safe" subset of simulink/stateflow into lustre. In *Proceedings of the 4th ACM International Conference on Embedded Software*, EMSOFT '04, pages 259–268, New York, NY, USA, 2004. ACM. 34

[110] Lorenzo Sciavicco and Bruno Siciliano. *Modelling and control of robot manipulators*. Springer Science & Business Media, 2012. 14

[111] Lorenzo Sciavicco and Bruno Siciliano. *Modelling and control of robot manipulators*. Springer Science and Business Media, 2012. 22, 51, 83

[112] Roberto Sebastiani. Lazy satisfiability modulo theories. *Journal on Satisfiability, Boolean Modeling and Computation*, **3**:141–224, 2007. 24

[113] A. G. Shem, T. A. Mazzuchi, and S. Sarkani. Addressing uncertainty in UAV navigation decision-making. *IEEE Transactions on Aerospace and Electronic Systems*, **44**[1]:295–313, January 2008. 109

[114] Jean-Jacques E Slotine, Weiping Li, et al. *Applied nonlinear control*, **199**. prentice-Hall Englewood Cliffs, NJ, 1991. 22, 34

[115] Mark W Spong, Seth Hutchinson, and Mathukumalli Vidyasagar. *Robot modeling and control*, **3**. Wiley New York, 2006. 22, 51, 69, 83, 87

[116] Brian L Stevens, Frank L Lewis, and Eric N Johnson. *Aircraft control and simulation: dynamics, controls design, and autonomous systems*. John Wiley and Sons, 2015. 19, 20, 22, 51, 66

[117] Suradet Tantrairatn and Sandor M Veres. A rational agent framework for adaptive flight control of UAVs. In *2015 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 147–156. IEEE, 2015. 109

[118] C. T. Ton and W. MacKunis. Robust attitude tracking control of a quadrotor helicopter in the presence of uncertainty. In *2012 IEEE 51st IEEE Conference on Decision and Control (CDC)*, pages 937–942, Dec 2012. 31

[119] Mohsen Vahdanipour and Mahdi Khodabandeh. Adaptive fractional order sliding mode control for a quadrotor with a varying load. *Aerospace Science and Technology*, **86**:737–747, 2019. 31, 95

[120] VARIO Helicopter. NVARIO Helicopter Benzin Trainer: User Guide. https://www.vario-helikopter.de/uk/d/manual/8301-8330.pdf. Accessed: 8 January 2020. 65

[121] Gergely Vass. Avoiding gimbal lock. *Comput. Graph. World*, **32**[6]:10–11, 2009. 18, 84

[122] Michael Vierhauser, Jane Cleland-Huang, Sean Bayley, Thomas Krismayer, Rick Rabiser, and Pau Grünbacher. Monitoring CPS at runtime- a case study in the UAV domain. In *2018 44th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, pages 73–80. IEEE, 2018. 109

[123] E. De Vries and K. Subbarao. Backstepping based nested multi-loop control laws for a quadrotor. In *2010 11th International Conference on Control Automation Robotics Vision*, pages 1911–1916, Dec 2010. 30

[124] Jian Wang, Thomas Bierling, Michael Achtelik, Leonhard Hocht, Florian Holzapfel, Weihua Zhao, and Tiauw Hiong Go. Attitude free position control of a quadcopter using dynamic inversion. *AIAA Infotech@ Aerospace*, :1583, 2011. 31

[125] Timothy Wang. *Credible autocoding of control software*. PhD thesis, Georgia Institute of Technology, 2015. 5

[126] Xuerui Wang, Sihao Sun, Erik-Jan van Kampen, and Qiping Chu. Quadrotor fault tolerant incremental sliding mode control driven by sliding mode disturbance observers. *Aerospace Science and Technology*, 2019. 31

[127] D. Wolfram, F. Vogel, and D. Stauder. Condition monitoring for flight performance estimation of small multirotor unmanned aerial vehicles. In *2018 IEEE Aerospace Conference*, pages 1–17, March 2018. 109

[128] Han Yan. Attitude control of spacecrafts based on small-gain theorem. In *Proceedings of the 33rd Chinese Control Conference*, pages 3494–3499. IEEE, 2014. 21

[129] Sungwook Yang, Sangchul Lee, Jung-Hyung Lee, and Hwa-Suk Oh. New real-time estimation method for inertia properties of STSAT-3 using gyro data. *Transactions of the Japan Society for Aeronautical and Space Sciences*, **58**[4]:247–249, 2015. 103

[130] Xilin Yang, Matt Garratt, and Hemanshu Pota. Non-linear position control for hover and automatic landing of unmanned aerial vehicles. *IET control Theory & Applications*, **6**[7]:911–920, 2012. 31

[131] Yueneng Yang and Ye Yan. Attitude regulation for unmanned quadrotors using adaptive fuzzy gain-scheduling sliding mode control. *Aerospace Science and Technology*, **54**:208–217, 2016. 30

[132] Y. Yu, Y. Guo, X. Pan, and C. Sun. Robust backstepping tracking control of uncertain mimo nonlinear systems with application to quadrotor uavs. In *2015 IEEE International Conference on Information and Automation*, pages 2868–2873, Aug 2015. 30

[133] Yun Yu, Shuo Yang, Mingxi Wang, Cheng Li, and Zexiang Li. High performance full attitude control of a quadrotor on SO(3). In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1698–1703, May 2015. 28

[134] George Zames. On the input-output stability of time-varying nonlinear feedback systems–part ii: Conditions involving circles in the frequency plane and sector nonlinearities. *IEEE transactions on automatic control*, **11**[3]:465–476, 1966. 21

[135] George Zames. On the input-output stability of time-varying nonlinear feedback systems part one: Conditions derived using concepts of loop gain, conicity, and positivity. *IEEE transactions on automatic control*, **11**[2]:228–238, 1966. 21

[136] Frank Zeyda, Julien Ouy, Simon Foster, and Ana Cavalcanti. Formalising cosimulation models. In *International Conference on Software Engineering and Formal Methods*, pages 453–468. Springer, 2017. 33

[137] Jiang Zhongping, Lin Yuandan, and Wang Yuan. Nonlinear small-gain theorems for discrete-time large-scale systems. In *2008 27th Chinese Control Conference*, pages 704–708. IEEE, 2008. 21