Integrated Fault-Tolerant Control System for Unmanned Aerial Systems

Paulin Kantue
(Student number: 0103702W)

School of Mechanical, Industrial and Aeronautical Engineering
University of the Witwatersrand
Johannesburg, South Africa.

Supervisor
Prof. Jimoh O. Pedro

A research thesis submitted to the Faculty of Engineering and the Built Environment, University of the Witwatersrand, in fulfilment of the requirements for the degree of Doctor of Philosophy in Engineering.

22 August 2022

Declaration

I declare that this thesis is my own unaided work. It is being submitted for the degree of Doctor of Philosophy in Engineering at the University of the Witwatersrand, Johannesburg. It has not been submitted before for any degree or examination at any other University.

SIGNED: 

22/08/2022

. P. Kantue Date

Acknowledgements

First and foremost, I have to thank my research supervisor, Prof. Jimoh O. Pedro. Without your consistent support and dedicated involvement throughout these last few years of my research, this thesis would not have been possible. I would like to profoundly thank you for your encouraging words and your expectations for excellence for this research over the past five years.

I have to thank all my former colleagues who were involved in the hardware development and methodology validation for this research: Coert Visser, Stephan Van Der Walt, Prevani Kirsten-Naidoo and Gerrit Viljoen.

Most importantly, none of this would have been possible without my wife, Lerato Kantue, who was my voice of reason when I decided to pursue this degree while still having a full-time career. Throughout this period, which included the current Covid-19 pandemic, she has been patient and supportive even while we were thousands of kilometres apart. My son, Samson Kantue, even though you are too small to comprehend, you played a crucial role in pushing me to be a worthy role model. To my parents and my sister, thank you for your warm messages of encouragement. They were always well-received and made me feel supported.

Lastly, I would like to thank God for being the source of my strength and joy during these last few years. It is through him that I can claim, *for when I am weak, then I am strong*.

Abstract

The susceptibility of unmanned aerial vehicles (UAVs) to faults and errors within critical functions such as flight control and navigation systems, combined with their inability in supporting mechanical redundancy due to their size and weight constraints, has led to the research and development of intelligent and fault accommodating control systems known as fault-tolerant control systems (FTCS). The main objective of this research is to design a fault-tolerant control (FTC) system which makes use of only flight data measurements available in most UAVs, to augment mission integrity against actuator faults. This thesis presents new research into the field of UAV fault-tolerant control. The above-stated data-driven approach in FTC design consisted of using radial basis functions neural networks (RBFNN), combined with a technique of time difference of arrival (TDOA) to detect and identify a particular type of actuator fault called an incipient fault. System identification of a propeller-motor slippage condition enabled the model estimation of such an incipient behaviour. FTC integration issues such as: FTC reliability and implementation in a real-time operating system; fault detection and diagnosis (FDD), and controller reconfiguration delays, were investigated within a development framework which ensured online fault estimation. This was achieved by adopting a modified RBFNN training algorithm with fast convergence and low-memory capabilities. The framework also incorporated a controller reconfiguration mechanism using the extremum seeking control law combined with an optimisation function constructed by utilising a geometric representation for actuator allocation. The integrated FTC requirement to improve the real-time performance of an unmanned quadcopter under various levels of incipient fault was achieved by comparing with a nominal controller within real-time simulation environment. The major contributions of this research can be summarised as follows: (1) The development of a fault-emulation model based on the faulty behaviour of a propeller-motor slippage (incipient) condition validated using a software-in-the-loop (SITL) simulation environment; (2) The development of a TDOA framework and the real-time learning of RBFNN through a meta-heuristic hybrid line search algorithm for real-time FDD. (3) The development and real-time testing of an extremum seeking reconfiguration control algorithm to improve the probability of mission success implemented within an integrated fault-tolerant framework.

Preface

The presented document is a PhD thesis prescribed by the postgraduate course and describes the research investigated. *Integrated Fault-Tolerant Control System for Unmanned Aerial Systems* is undertaken under the supervision of Professor Jimoh O. Pedro.

Contents

List of Figures

List of Tables

Nomenclature

α	Column Index of $\overline{Q}$ Variance
β	ES incircle radius for quadcopter actuator geometric representation
$\Delta\omega_{m_i}$	PWM excitation signal on the ith motor designated as an TDOA emitter
$\Delta\mathbf{J}_{k+1}$	CFA cost function net contribution due to $k+1$th hidden node
$\delta_{lat}, \delta_{lon}$	Lateral and longitudinal cyclic control inputs
$\delta_{mot}, \delta_{rud}$	Heave and yawing control inputs
η	NN learning threshold
γ	ES incircle offset for quadcopter actuator geometric representation
ω_H	ES controller high-pass filter frequency vector
ω_L	ES controller low-pass filter frequency vector
σ	RBFNN node spread
θ^*	ES optimisation parameter minimising $J(\mathbf{y}_\theta)$
$\mathbf{c}$	RBFNN node centre output
$\mathbf{F}$	FDD fault uncertainty gain
$\mathbf{K_a}$	ES controller adaptation gain vector
$\mathbf{P}$	RBFNN cross-correlation regression matrix
$\nabla\Delta\mathbf{J}_{k+1}$	CFA cost function net contribution gradient due to $k+1$th hidden node

$\omega_{k+1}^{(p)}$ CFA $k+1$th hidden node parameter set updated based on pth step of conjugate gradient method

ω_m Motor angular speed

ω_p Propeller angular speed

$\overline{Q}$ Matrix of cross-correlated signals

ϕ, θ, ψ Rolling, pitching and yawing Euler angles

$\mathbf{\Phi}_k$ CFA regressor subset up to the kth hidden node

σ_v Kalman filter measurement noise standard deviation

σ_w Kalman filter process noise standard deviation

$\mathbf{w}$ RBFNN weights vector

A_q^{1i} Longitudinal flapping angle gain of the ith rotor due to pitching body-fixed angular rate

$A_{\delta_{lon}}^{1i}$ Longitudinal flapping angle gain of the ith rotor due to longitudinal cyclic control input

$A_{a_{1i}}^{1i}$ Longitudinal flapping angle gain of the ith rotor due to longitudinal flapping angle of the ith rotor

a_1, b_1 longitudinal and lateral rotor flapping angles

a_x, a_y, a_z axial acceleration along the body-fixed x, y, z axes

B_p^{1i} Lateral flapping angle gain of the ith rotor due to rolling body-fixed angular rate

$B_{\delta_{lon}}^{1i}$ Lateral flapping angle gain of the ith rotor due to longitudinal cyclic control input

$B_{b_{1i}}^{1i}$ Lateral flapping angle gain of the ith rotor due to lateral flapping angle of the ith rotor

C_{T_i} Thrust coefficient of the ith rotor

$G_f(s)$ Laplace transform of a fault injection filter

$H(\theta, \mathbf{F})$ FDD control allocation scheme

I_{xx} Mass moment of Inertia about the body-fixed x-axis

I_{yy} Mass moment of Inertia about the body-fixed y-axis

I_{zz} Mass moment of Inertia about the body-fixed z-axis

$J(\mathbf{y}_\theta)$ ES controller objective function

L, N, M Moments about the body-fixed x, y, z axes

L_p L-moment stability derivative due to rolling body-fixed angular rate

$L_{\delta_{lat}}$ L-moment control derivative due to lateral cyclic control input

$L_{\delta_{rud}}$ L-moment control derivative due to yawing control input

$L_{b_{1i}}$ L-moment stability derivative due to lateral flapping angle of the ith rotor

M_q M-moment stability derivative due to pitching body-fixed angular rate

$M_{\delta_{lon}}$ M-moment control derivative due to longitudinal cyclic control input

$M_{\delta_{mot}}$ M-moment control derivative due to heave control input

$M_{a_{1i}}$ M-moment stability derivative due to longitudinal flapping angle of the ith rotor

N_r N-moment stability derivative due to yawing body-fixed angular rate

$N_{\delta_{lat}}$ N-moment control derivative due to lateral cyclic control input

$N_{\delta_{rud}}$ N-moment control derivative due to yawing control input

p, q, r Rolling, pitching and yawing body-fixed angular rates

P_f, P_i RBFNN model estimation and prediction accuracy metrics

S_{mn} Cross-correlation between m and n signals

Sp_c GSS meta-heuristic algorithm search counter

Sp_H GSS meta-heuristic algorithm search upper limit

Sp_i GSS meta-heuristic algorithm max number of iterations

Sp_L GSS meta-heuristic algorithm search lower limit

T_i Thrust force of the ith Rotor

u, v, w Body-fixed translational velocities

X, Y, Z Forces along the body-fixed x, y, z axes

X_θ X-force stability derivative due to pitching Euler angle

X_q X-force stability derivative due to pitching body-fixed angular rate

X_u X-force stability derivative due to translational velocity along body-fixed x-axis

$X_{\delta_{lon}}$ X-force control derivative due to longitudinal cyclic control input

$X_{a_{1i}}$ X-force stability derivative due to longitudinal flapping angle of the ith rotor

Y_θ Y-force stability derivative due to pitching Euler angle

Y_p Y-force stability derivative due to rolling body-fixed angular rate

Y_v X-force stability derivative due to translational velocity along body-fixed x-axis

$Y_{\delta_{lat}}$ Y-force control derivative due to lateral cyclic control input

$Y_{b_{1i}}$ Y-force stability derivative due to lateral flapping angle of the ith rotor

$Z(P_m, E_n)$ TDOA cross-correlation time delay based on the NN predicted output of the mth motor and NN estimated output nth motor

$Z(P_m, P_n)$ TDOA cross-correlation time delay based on the NN predicted output of the mth motor and NN predicted output nth motor

Z_θ Z-force stability derivative due to pitching Euler angle

Z_q Z-force stability derivative due to pitching body-fixed angular rate

Z_w z-force stability derivative due to translational velocity along body-fixed z-axis

$Z_{\delta_{mot}}$ X-force control derivative due to heave control input

Acronyms

AFTCS Active Fault-Tolerant Control System

CFA Continuous Forward Algorithm

ES Extremum Seeking

ESC Electronic Speed Controller

FDD Fault Detection and Diagnosis

FTC Fault-Tolerant Control

FTCS Fault-Tolerant Control System

GPS Global Positioning System

GSS Golden Section Search

HILS Hardware-in-Loop System

I/O Input/Output Device

IDE Integrated Development Environment

IMU Inertial Measurement Unit

LCF Low Contact Friction

MPC Model Predictive Control

NN Neural Networks

OLS Orthogonal Least Squares

PFTCS Passive Fault-Tolerant Control System

PID Proportional Integral Derivative

PWM Pulse Width Modulation

RBFNN Radial Basis Function Neural Networks

RC Reconfigurable Controller

RPA Remotely Piloted Aircraft

SSE Steady-State Error

SITL Software-in-the-Loop

TDOA Time Difference of Arrival

UAS Unmanned Aerial Systems

UAV Unmanned Aerial Vehicle

Publications Arising from this Research

Published Conference Papers

- [1] Paulin Kantue and Jimoh O. Pedro. "Real-Time Identification of Faulty Systems: Development of an Aerial Platform with Emulated Rotor Faults". In: 2019 4th Conference on Control and Fault Tolerant Systems (SysTol2019), Casablanca, Morocco, 2019, pp. 20–25.
- [2] Paulin Kantue and Jimoh O. Pedro."Grey-box modelling of an Unmanned Quadcopter during Aggressive Maneuvers". In: International Conference on System Theory, Control and Computing (ICSTCC). Sinaia, Romania: IEEE, 2018, pp. 640–645.
- [3] Paulin Kantue and Jimoh O. Pedro. "Nonlinear Identification of an Unmanned Quadcopter Rotor Dynamics using RBF Neural Networks". In: International Conference on System Theory, Control and Computing (ICSTCC). Sinaia, Romania: IEEE, 2018, pp. 292– 298.
- [4] Paulin Kantue and Jimoh O. Pedro. 'Integrated Fault Detection and Diagnosis of an Unmanned Aerial Vehicle using Time Difference of Arrival'. In: International Conference on System Theory, Control and Computing (ICSTCC). Sinaia, Romania: (pp. 336-342), IEEE, 2020.

International Journal Papers Currently under Review

- Paulin Kantue and Jimoh O. Pedro. 'Reconfigurable extremum seeking controller design of a quadcopter with incipient actuator faults'. Submitted for review to International Journal of Control, Automation, and Systems, Manuscript ID: JCAS-D-21-00741, August 2021.

International Journal Papers Accepted for Publication

- Paulin Kantue and Jimoh O. Pedro. 'Integrated Fault-Tolerant Control of a Quadcopter UAV with Incipient Actuator Faults'. Submitted for review to International Journal of Applied Mathematics and Computer Science (AMCS), October 2021.

Chapter 1

Introduction

1.1 Research Background and Motivation

Recent industrial activities such as: disaster management, agricultural/terrain mapping, mining exploration, law enforcement surveillance, film and photography, have made use of unmanned aerial vehicles (Unmanned Aerial Vehicle (UAV)s), generally known as drones or remotely piloted aircraft (RPA), as their main tool for sensory data acquisition and analysis. Miniature rotorcraft, a subclass of such vehicles, are of particular interest given their compact size, ability to hover, forward/inverted flight, their suitability to operate in confined/urban environments and above all, their low cost of operation. Unlike their full-scale counterparts, miniature rotorcraft exhibit increased thrust-to-weight ratio enabling them to perform aggressive and aerobatic manoeuvres in confined spaces. The application of unmanned miniature aircraft, and UAVs in general, therefore demands an improvement in their reliability and software robustness to mitigate potential accidents [5].

Manned aircraft achieve airworthiness certification through hardware redundancy and pilot handling qualities assessment, which are unavailable for UAVs due to their size, power-plant and weight constraints. In 2019, a survey which was responded to by more than 150 commercial UAV companies in the United Kingdom confirmed that a critical failure is likely to occur within 100-500 hours of flight time [6]. This is equivalent to a possible fatal accident *every 1-5 month(s)*, given continuous (5hr/day) operations. It has been expressed through numerous studies, that the current emphasis on the reliability and maintainability (compo-

nent complexity, component cost, long-lead items) of unmanned systems is insufficient and poses a major risk to the sustained success of UAVs [7].

Table 1.1 shows the likelihood of various types of failure, the impact on mission success and the associated technologies to prevent critical incidents/accidents. *Likelihood of failure* refers to the probability that a particualr failure is the primary cause of a critical accident. *Impact on mission success* refers to the severity of a particular failure on the completion of UAV mission objectives. *Combined weighting* refers to a means to prioritise a mitigation for the risk associated with a particular fault. It can be seen that the highest combined weighting for mission success deals with preventing firmware (software) failures through the implementation of fault-tolerant stability and control algorithms. The surveyed companies also noted that some mixture of on-board intelligent augmentation and operator training was required in order to improve UAV flight safety [6]. However, most hobby-grade to commercial-grade flight controllers still adopt the Proportional Integral Derivative (PID) controller structure due to its ease of implementation, quick parameter fine-tuning and pilot understanding and training [8]. But given the unstable nature of a rotorcraft such as the quadcopter, such simple controllers quickly degrade in performance outside their area of design [9] and their robustness is further impacted in the event of sensor or actuator faults [10]. Although there are numerous faults that can cause critical accidents, as seen in Table 1.1, this research involves augmenting the default PID controller with fault-tolerant control algorithms with specific focus on actuator faults.

1.2 Fault-Tolerant Control Systems (FTCS)

The lack of self-repairing and smart flight control systems, also known as fault-tolerant control systems (Fault-Tolerant Control System (FTCS)), poses a major roadblock to UAV success in an urban environment. This problem is more pronounced for unmanned rotorcraft given that they are highly unstable, susceptible to environmental conditions and lack aerodynamically-induced stability [6]. There are two types of FTCS namely: passive (Passive Fault-Tolerant Control System (PFTCS)) and active (Active Fault-Tolerant Control System (AFTCS)). Unlike AFTCS, PFTCS make use of fixed and robust controllers without the need for fault detection or control reconfiguration. Throughout this thesis, we will be referring to AFTCS as FTCS.

Table 1.1: UAV critical failure types vs mitigating technologies [6].

Failure Type	Likelihood of failure [1-6]	Mitigating technology	Impact on mission success [1-6]	Combined weighting
Navigation	5.77	Collision Avoidance	3.16	18.2
Firmware	5.6	Fault-tolerant stability & control	3.78	21.1
Battery	5.6	Intelligent power management	3.36	18.8
Rotor blade	3.96	Intelligent warning (mechanical)	4.1	16.2
Brushless motor	3.41	Intelligent warning (electrical)	4.1	13.9

The indisputable necessity for FTCS was driven partly by two commercial aircraft accidents in the late 1970s. On the 12th of April 1977, the Delta Flight 1080 elevator became stuck at 19 degrees without prior pilot warning indication. The experienced pilot was able to reconfigure the remaining actuators and safely land the aircraft. The second case occurred on the 25th of May 1979 when an American Airlines DC-10 pilot only had approximately 15 seconds before the plane crashed due to the left wing's leading-edge slats locked in position. Post-crash investigations demonstrated that an automatic rerouting of hydraulic fluid could have prevented the 273 fatalities on that day [11].

The application of FTCS on miniature helicopters (main rotor diameter less than 1m) or even quadcopters in the same class, is a research area that has become more relevant in recent years [12]. This is mainly due to the increasing use of such systems in civilian airspace by hobbyists and commercial companies alike. Currently, systems such as DJI Phantom series unmanned systems make use of redundant sensors in order to introduce fault-tolerance capabilities. Most platforms make use of a sensor voting mechanism whereby the faulty sensor is detected and isolated, to ensure data from the faulty sensor is ignored, negating the need for controller reconfiguration.

1.2.1 FTCS architecture

A FTCS consists of a fault detection and diagnosis (FDD) module whose main function is the detection, diagnosis and identification of sensor, actuator or system faults. Figure 1.1 shows a typical schematic of a FTCS. This process triggers the modification of a nominal controller via a reconfiguration control (Reconfigurable Controller (RC)) mechanism, which typically involves some form of control reallocation and sensor bypassing, such that the performance and/or stability degradation caused by such faults is mitigated or altogether eliminated. One of the primary goals of this research is the development of a FTCS architecture capable of restoring an acceptable level of system robustness after a particular fault has occurred.

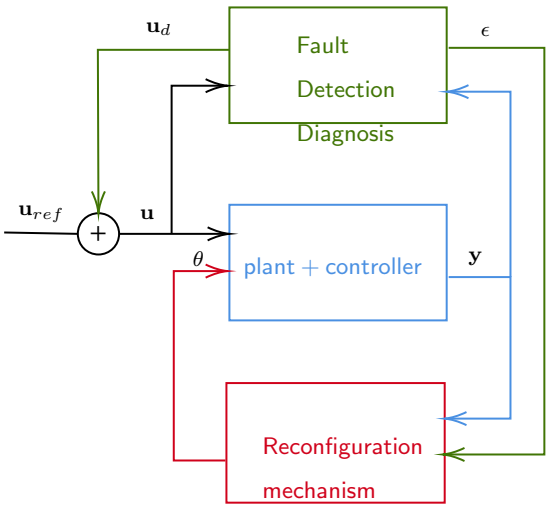


Figure 1.1: Generalised structure of an active FTCS.

The requirement to implement FTCSs exists based on two assumptions: (1) the nominal controller designed based on classical or modern control techniques is incapable of maintaining a satisfactory performance and robustness in the presence of actuator, sensor and system faults, (2) the system architecture has an inherited redundancy that allows the effects of a fault to be in some way circumvented or minimised [13]. These assumptions are found to be correct in Unmanned Aerial Systems (UAS) which require a high level of software and hardware reliability and fault-tolerance for the success of their critical missions.

Consequently, there exists the need to apply analytical redundancy, i.e., to exploit mathematical relations between measured or estimated variables in order to detect possible malfunctions [14]. Combining such methods in the Fault Detection and Diagnosis (FDD) and a reconfigurable controller (RC) based on a black-box architecture such as neural networks,

is the objective of this research. Moreover, it focuses on the real-time performance of such a framework with the potential impact in converting the existing market of low-cost unmanned aircraft into robust and fault-tolerant unmanned systems for commercial use.

1.2.2 Fault classification

Within the context of FTCS for unmanned rotorcraft, three classes of faults are considered: (1) sensor, (2) actuator and (3) structural/components damage. All the above mode of failures are modelled as constant bias, drift or additive-type, multiplicative-type and outlier data failure. Consideration on the time-based transient dynamics in the above models, is the differentiation between abrupt and incipient faults. Most considerations during the synthesis of unmanned quadcopter FDD methods, are based on abrupt faults given their rapid loss of stability and/or performance caused by nonlinear/coupled dynamics [14].

The incipient fault type, often neglected, can cause complete system failure. On the 28th of January 1986, a fatal incident involving the Space Shuttle Challenger resulted from a failure in the O-rings sealing a joint on the right solid rocket booster, releasing pressurised hot gases. This resulted in the solid rocket booster separating from its joint attachment, structural failure of the external tank and eventual mechanical overload due to abnormal aerodynamic forces. In the famous words of theoretical physicist Richard Feynman: *"Once a small hole burns through generates a large hole very fast! Few seconds catastrophic failure."* [15].

Given the increased complexity and payload value of unmanned systems in recent years, the detection and diagnosis of incipient faults have been investigated on such platforms [16]. Given that traditional FDD methods are better suited for systems with significant symptoms, the diagnosis of incipient faults can be problematic given small magnitude in relation to the measured/monitored signals [17]. Therefore, fundamental questions such as: (1) what are the main factors that impact the detection performance of incipient faults? (2) how can you integrate the detection of incipient faults within a closed-loop system? need to be answered. This research is aimed at answering the above questions with specific focus on the development and real-time implementation of an actuator-class incipient fault in Chapter 3 and detection and diagnosis in Chapter 5.

1.2.3 Integrated FTCS

The successful verification and validation of a FTCS can only be achieved by taking into consideration the software and hardware environments in which its algorithms will be executing. The requirements for system safety, reliability, maintainability, and survivability are also imposed on one of its most critical sub-systems, the FTCS. Additional requirements must include the transient and steady-state performance for not only normal operations, but also faulty conditions. An integrated approach to development of a FTCS is therefore crucial to ensure utmost compliance.

In their bibliographical review, Zhang and Jiang stated that one aspect for future FTCS research was to investigate the development of an integrated design methodology for the on-line and real-time application of FDD and RC algorithms [18]. Challenges in this area included: (1) how to deal with the collinearity in identification algorithms; (2) how to obtain accurate parameter estimates on-line and real-time, in the presence of poor input excitation; (3) how to deal with adverse interactions between the identification and the control schemes in a closed-loop setting and (4) how to ensure that the fault detection and controller reconfiguration time delays are minimised given the software and hardware constraints.

Recent research proves that the above concerns still require further investigations [19, 20]. Consequently, the above questions have been incorporated as research objectives in this thesis and act as guidelines throughout the development of a design methodology in order to arrive at an integrated FTCS with real-time capability.

1.3 Plant Description

The FTC architecture developed in this research was implemented on H1 unmanned quadcopter owned by the company Uav4frica (Pty) Ltd shown in Figure 1.2. Due to the critical nature of the actuator-induced faults, initial flight tests were performed and thereafter a pseudo real-time simulation model was developed using the same source code baseline flown by the H1 quadcopter.

1.3.1 Uav4africa H1 quadcopter

The H1 system, shown in Figure 1.2 is an unmanned quadrotor weighing 1.75 kg with four 10 inch (25.4 cm) APC propellers. The Pixhawk flight controller hardware integrates a 10-channel Global Positioning System (GPS) with the ArduCopter software baseline developed by ArduPilot. A shock-absorbing landing gear was designed in anticipation of the testing phase of the FTCS. The H1 actuator system which comprises of the propeller, the speed controller and motor was used to estimate the incipient fault condition. This process and a detailed description of the H1 system and its development is given in Chapter 3. Due to lack of approved destructive test sites and capital for multiple prototypes, flight-test validation process was not completed at the time of writing this thesis.

Figure 1.2: Uav4africa H1 drone with shock-absorbing landing gear.

1.4 Thesis Overview

The thesis is divided into six chapters. Each chapter begins with an introduction giving context on the chapter followed by a brief outline where applicable.

Chapter 1 - Introduction

Chapter 1 introduces the FTCS research field and discusses the factors that enable its successful implementation in real-world applications, especially in UAS. A brief introduction into

the research aerial platform is given and a brief outline to the rest of the thesis is provided.

Chapter 2 - Advances in UAV Fault-Tolerant Control

The state-of-the-art information is discussed in detail in chapter 2. It provides a description of the problem addressed in this thesis and aims to establish the research motivation. This chapter takes an in-depth look at what has been achieved in the area of FTCS of UAS thus far and where the research gaps exist.

Chapter 3 - Development of an Unmanned Aerial Vehicle

Chapter 3 presents the development of an aerial quadrotor platform used for the system identification of a common incipient fault-type among multi-rotors: propeller-motor slippage failure or known throughout this thesis as low-contact friction (Low Contact Friction (LCF)) failure. Fault models based on experimental data are developed and implemented in a modified ArduCopter software suite for the development of a pseudo real-time simulation environment. This is then used as a development and verification tool in the subsequent chapters.

Chapter 4 - Nonlinear System Identification of Unmanned Rotorcraft Systems

Chapter 4 presents the concept of grey-box modelling, based on the results from the previous chapter, for the nonlinear system identification of an unmanned quadcopter faulty rotor dynamics. A black-box nonlinear system identification method to estimate rotor dynamics, while addressing the issue of data collinearity, is developed and implemented in real-time simulation system for performance evaluation.

Chapter 5 - Integrated Fault-Tolerant Controller Synthesis

Chapter 5 describes the development of FDD and RC mechanisms through the consideration of various integration factors. The integrated FTCS solution was implemented in real-time simulation environment to evaluate the system closed-loop response during a waypoint

tracking mission. Results based on the real-time desktop simulation has been discussed.

Chapter 6 - Conclusions and Recommendations for Further Work

Chapter 6 summarises the main findings and contributions of this research. The chapter also states some recommendations for further work.

1.5 Delimitations of Scope and Key Assumptions

1.5.1 Delimitations

The following delimitations can be stated as part of this research scope:

- The neural network approach is only based on a Radial Basis Function Neural Networks (RBFNN)-type architecture. Various learning algorithms are investigated without considering their suitability in other types of neural network-based FTCS.
- Only single events of actuator faults are to be considered. Double actuator failures are considered unrecoverable.
- The modification of the baseline controller does not include its architecture.
- Validation of the FTCS with flight test data is not considered.
- No sensor faults are considered.

1.5.2 Assumptions

The following key assumptions are considered within this research:

- The post-fault dynamic model is observable and identifiable.
- The sensors faults that would impact FTCS performance due to degraded post-fault model controllability, were assumed negligible.

- The magnitude of an incipient actuator fault is not time-varying from the moment the FDD module detects a fault.
- The pre-fault dynamic model is closed-loop stable using a PID controller architecture.
- The post-fault dynamic model is closed-loop stable during the synthesis of the FTCS.

Chapter 2

Advances in UAV Fault-Tolerant Control

2.1 Introduction

Fault-Tolerant control systems (FTCS) can be defined as a type of control system with the capability to tolerate faults and malfunctions and the capacity to maintain a desirable level of system stability, robustness and performance. These types of control systems exist based on the premise that classical and modern control system designs may not be able to maintain a satisfactory performance in the presence of actuator, sensor and system faults [21]. This premise is all the more relevant in UAVs which require a high level of software and hardware reliability and fault-tolerance.

Initial research in Fault-Tolerant Control (FTC) (better known then as self-repairing flight control systems) focused on establishing techniques which can be used to either (1) tolerate or (2) detect and compensate for component failures using linear models [11]. In the 1990s, a clear distinction was made between fail-safe systems and fault-tolerant systems. The following properties were proposed for FTCS: (1) prevent any simple fault from developing into system level failure; (2) make use of information (analytical) redundancy to detect faults; (3) make use of reconfiguration in programmable system components to accommodate faults; (4) accept degraded performance due to fault but keep plant availability; (5) low-cost by design given no new hardware is required. Subsequently, several other authors published

papers discussing FTCS robustness/safety issues and learning approaches [22].

Since then, a significant amount of research on FTC has been established. In the last few years, application of FTCS for unmanned systems has also increased. Gain-scheduling techniques on a quadrotor UAV simulator [23], application of Model Predictive Control (MPC) methods on unmanned aircraft [24], construction of a multiple model scheme-based fault detection and identification for an unmanned aircraft and development of neural network-based fault identification method for unmanned helicopter [25] are some of the few publications that have surfaced.

FTCS can be classified in two categories: Passive FTCS and Active FTCS. The main distinction in passive FTCS is the exclusion of a fault detection and identification mechanism and reconfigurable/restructurable control system. On the other hand, passive FTCS have more emphasis on hardware redundancies and establishing a robust control design process upon which various combinations of faults can be tolerated. This is based on a certain pre-defined number of component faults which formed part of the controller design process. The use of actuator pre-compensator design technique combined with state feedback gain matrix was proposed in [26]. Simulated results were demonstrated based on aircraft linear dynamics.

2.2 Fault Detection and Diagnosis (FDD)

Fault detection and diagnosis (FDD) has been described as a crucial aspect to complex and safety-critical systems such as nuclear powerplants, transport and more especially unmanned systems. Important aspects of FDD include: (1) high sensitivity to faults; (2) robustness to model uncertainties; (3) computational complexity; (4) ability to provide quick detection and (5) suitability to FTC [27, 28]. FDD methods can be categorised into model-based and data-based methods which are shown in Figure 2.1. It has been shown that state estimation methods are most suited for fault detection, although they are inefficient in identification of faults without a-priori information. Various survey papers on FDD methods have been published in the last 20 years. [29, 30].

Sensor and actuator faults in unmanned helicopters have been investigated in [31]. Actuator fault detection has been implemented for a stuck actuator type failure. In particular, the collective actuator was simulated to get stuck near the trim hover. An auto regressive exoge-

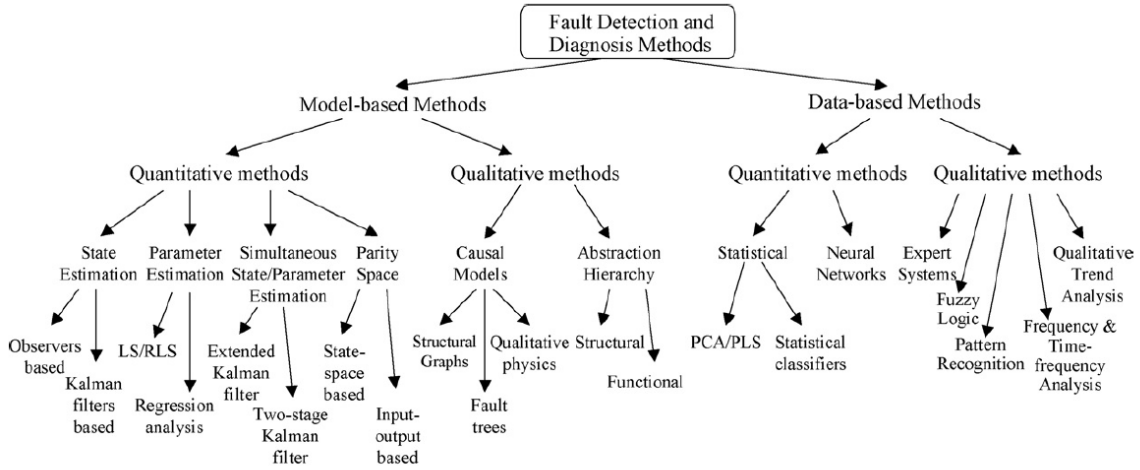


Figure 2.1: Categorisation of FDD methods [18].

nous (ARX) model was chosen to extract localised input-output observer models in a system identification sense upon which the output error would constitute a fault in the system. FDD methods specific for manned and unmanned helicopters were also investigated in [32]. One major drawback of such methods is the requirement of an accurate analytical model, which is not always feasible for small unmanned helicopters. In the same year, an adaptive recursive orthogonal least squares algorithm was presented in [33]. The regression problem was set up through categorising regressor candidates such that parameter (in this case aerodynamic) coupling effects are not lost after a control surface failure. It was demonstrated that the recursive nature with the combination of the model restructure algorithm, could identify the instant upon which the failure occurred even if the fault built up gradually and was only detected afterwards.

Using an unknown input observer to track actuator fault parameters and decouple the effect of faults and unknown inputs was proposed in [34]. It was demonstrated that actuator loss of effectiveness at unknown values and unknown time instants could be described as an unknown vector of parameters belonging to a convex set of values. An observer equation was deduced such that actuator faults and disturbance could be decoupled. The design of the adaptive controller in an H_∞ sense incorporates the convex set of possible faults without much fault information. In order to apply this scheme into real systems, the set of linear matrix inequalities (LMI) were translated into solvable LMIs by using the symmetric properties of semi-definite inequalities and the guaranteed Lyapunov stability. This strategy was applied to form an integrated solution of an H_∞ output feedback controller gain. A switching

scheme allowed the worst-case controller to be used based on the *a-priori* upper bound of the faults to be accommodated. Observer type fault detection was also used in [35]. Using sliding mode observers based on a linear parameter varying system for the development of the fault detection and isolation scheme [36].

Issues on how to deal with model and fault uncertainties and their application to passive and active FTCSs have been researched. It investigates the use of a probabilistic approach in the development of the fault detection mechanism such that boundedness is maintained for all possible faults and model uncertainties. Demonstration of concepts was done using simplistic brushless DC motor. Attitude control of a quadrotor UAV was studied in [37]. Three altitude sensors were used to establish hardware redundancy. Standard deviation of the residuals between the various sensors was used as a quantitative indication that a fault had occurred. An adaptive neural network-based scheme for sensor failure diagnosis of an unmanned rotorcraft was discussed in [29]. The concept was to use a set of online learning neural networks to identify the type of sensors and another set of neural networks to classify the specific sensor output. Output measurements from gyroscopes and accelerometers were considered. Error norms from each NN output are used to detect and identify the sensor faults. In addition, an adaptive threshold was conceived based on the standard deviation and rate of change of the threshold values recorded after each flight. This required an extensive flight test campaign and is applicable to that unmanned helicopter configuration, which in this case is the SIA-Heli-90 aircraft.

An improvement on the multiple model adaptive estimation (MMAE) with extended Kalman filter (EKF) has been proposed in [38] for the fault diagnosis of an unmanned aircraft. This improvement comes with the use of the Euclidean norm to guarantee the filter stability and improve its estimation accuracy. This was achieved through the introduction of multiple fading factors which affected the forward propagation of the state-error covariance matrix. The simulation results were performed where each filter corresponded to a fault condition which was comprised of the elevator and flap actuators. Given lack of the real-time data for validation, the sensor measurements were corrupted by zero-mean white Gaussian noise and injected into the aircraft model. This method is very attractive provided there is enough *a-priori* knowledge of the post-fault dynamics for the effective design of the estimation filters. A similar method was applied to an unmanned aircraft longitudinal dynamics modelled as linear-parameter varying (LPV) multiple model [39]. The aerodynamic coefficients were

augmented with fault models of ice forming on aircraft aerodynamic surfaces. A collection of observers was used to estimate the icing state while considering the various aircraft icing configurations.

The use of an adaptive two-stage extended Kalman filter (EKF) for fault detection and diagnosis (FDD) of an unmanned quadrotor helicopter was investigated in [40]. Given that most commercially available systems make use of low-price and low-precision sensors that suffer from signal bias and drift faults, this was the focus of this paper. A simplistic kinematic model (ignoring actuator dynamics) of a quadcopter model, based on the Quanser Qball-X4, was modified by adding Inertial Measurement Unit (IMU) (Inertial Measurement Unit) fault dynamics into the system state matrix. In order for the EKF to be sensitive to abrupt sensor faults, a dynamic forgetting factor informed by the eigenvectors of the state covariance matrix was introduced. This enables the EKF to estimate the system states and the faults simultaneously. Even though the EKF initialisation process is not defined, a simulation model with normal and faulty conditions (sensor bias and drift) was evaluated, resulting in no false alarm under both scenarios and with the fault estimation error close to zero-mean. This model-based FDD approach is strongly dependent on *a-priori* knowledge of the sensor fault dynamics to prevent the incorrect estimation of faults.

The detection and diagnosis of incipient faults in unmanned quadrotor systems was studied in [16] and method in detail in [17]. The occurrence of this incipient fault was assumed to be of an abrupt nature but small in magnitude compared to the background noise and signal trend. A four-stage approach was used to develop an FDD algorithm for the detection of the abnormal voltage sensor data. A fault-signal-ratio (FSR) was assumed to be a combination of the fault–noise ratio (FNR) and fault–trend ratio (FTR) with the assumption that the measured signal is oscillatory in nature and its trend and the noise are statistically independent. The limitations of a traditional Hotelling method, where the computation of the Mahalanobis distance between the dataset and its past mean value requires a large number of data points and it is susceptible to bad conditioning, was improved through a detrending and denoising algorithm. The monitored signal is assumed that it can be constructed by a least-square-computed polynomial function, where its order is user-defined offline and based on *a-priori* knowledge of the monitored signal. The fault detection algorithm was able to improve the false-alarm rate and act as an early warning system for incipient faults with recovery features.

A relevant and recent concern of unmanned quadcopter propulsion reliability was investigated by Iannace et al., [25]. An acoustics classification model based on experimental measurements was developed to detect unbalanced blades in a UAV propeller. The faulty condition was induced during ground testing (attached to a tripod), by sticking paper tape on the upper surface of the designated faulty propeller blade. This results in a significant impact on the aerodynamics of the propeller and consequently the noise characteristics of the affected blade will be different from the normal conditions. Although the authors did not explicitly state it, but this type of fault condition is of the incipient-type as its magnitude is directly proportional to the rotational speed of the propeller. The resilient backpropagation algorithm was implemented in the training of a feedforward neural network fault classifier. This algorithm determines iteratively the weights of a neural network to minimise the error function finding the local minimum. Integration of this method for real-time fault diagnosis was not considered and as such the successful implementation of this neural network-based fault detection approach is unclear.

The robustness of vertical take-off and landing (VTOL) unmanned systems was investigated from an FDD perspective [41]. A two-stage algorithm based on time-domain and frequency-domain analyses of IMU accelerometer data was developed. A moving window filter, based on the bootstrap method, enabled the extraction and selection of features which were processed by a regularised linear discriminant analysis (LDA) method for fault detection. Validating such a method was achieved by analysing the effects of propeller damage (actuator fault) during the take-off and landing phases of flight. Similarly, the fault detection and dynamics based on small Permanent Magnet Synchronous Motor (PMSM) was investigated in [42]. Given the large number of the mechanical and electrical parameters, experimental data based on step inputs was used to develop transfer function blocks. A high friction and a detached propeller failure were some of the faulty conditions simulated on an experimental bed. Although the fault types are relevant to most available unmanned systems, the user-defined threshold fault detection method was not described and its implementation for FTC is unclear.

2.3 Restructurable/Reconfigurable Flight Control

Most of the reconfigurable controllers for UAS have their design methods stemming from well-established control theory concepts which have been applied in various industrial applications. However, new challenges appear when reconfiguration or control adaptation must be performed in an active fault-tolerant system for UAVs namely: (1) nonlinear dynamics under real-time constraints; (2) automated reconfiguration controller synthesis with little trial-and-error and human interventions; (3) all levels of reconfiguration to prevent further degradation of the system stability while striving for pre-fault level of performance [18]. Various control methods in a reconfigurable sense have been used across many applications, such as: adaptive control [34], feedback linearisation [43], linear quadratic [44, 5], gain-scheduling [23], model predictive control [45], H_∞ robust control [46, 47], linear matrix inequalities (LMI) [48], sliding mode control [49], Youla parametrization [50] and quantitative feedback theory (QFT) [51]. Figure 2.2 illustrates the idea behind controller reconfiguration [52]. d and r are the inputs to the plant and controller respectively, y and y_f are the outputs before and after a fault has occurred respectively. u_c and u_f is the actuator input before and after a fault has occurred respectively.

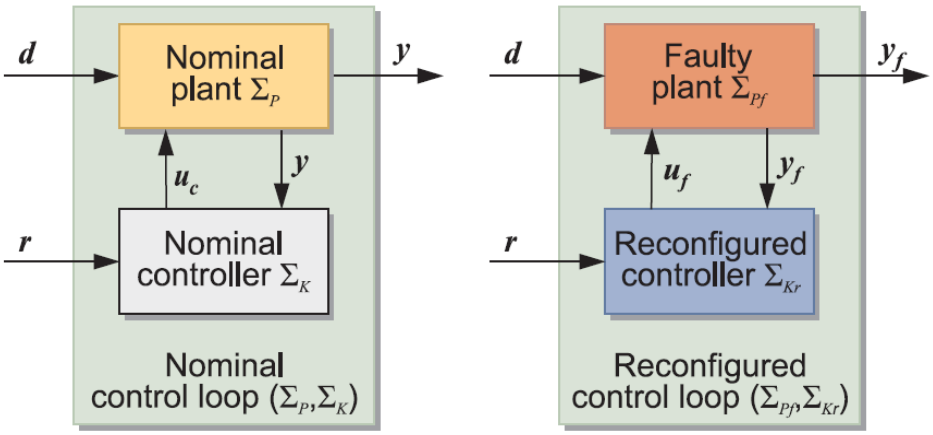


Figure 2.2: Illustration of reconfigurable control [52].

The on-line determination of a control law using neural networks to achieve reconfiguration after control surface damage was investigated in [53]. A distinction between restructurable and reconfigurable controllers was made with the underpinning difference being that the former involves the online restructuring of the control law rather than control gains reconfiguration. The use of an extended back-propagation algorithm (EBPA) for the on-line training of

NN-based controller has been studied. This resulted in the mapping of nonlinear dynamics with method that can be easily implemented in a real-time system given low-memory requirements. A few years later, Lee and Kim [54] investigated a similar approach of using an adaptive neural network controller to compensate for the effect of aerodynamic modelling errors. The control law was synthesised using a three-layer feedforward architecture with sigmoid activation functions. The tracking accuracy of aerodynamic angles was used for controller evaluation and it was found that it resulted in less overshoot and smaller steady-state errors as compared to a traditional backstepping controller.

Flight testing of a reconfigurable control system with the estimated model based on recursive least squares (RLS) algorithm with a forgetting factor was presented in [55]. The controller was based on model reference adaptive control. Given the case of a fixed-wing platform, the reference model mapping of the control variables to output variables were reduced to simple gains with biases. This is deemed applicable to only stable airframe with slow dynamics. The identification of gains was only achieved off-line in a batch process which limits its application on a real-time low-cost microprocessor. The proportional nature of the control law suggests that the elimination of the steady-state error in estimated control gains could be problematic. Linear quadratic (LQ) technique was also chosen to be used for the FTC design with the reformulated algebraic Riccati equation in [44]. The loss of control effectiveness for the rudder and one of the ailerons for an unmanned fixed-wing aircraft is compensated through the controller synthesis taking into account the *a-priori* knowledge of the system linear model which will not always be available in real-time conditions.

A neural network-based adaptive dynamic programming (NN-ADP) controller was investigated for the purpose of reconfigurable control in [56]. The NN-ADP is a cascade of neural networks that provides inner-loop body rate control, attitude control, and outer-loop velocity control. In the case of helicopters, the relationship between the main rotor actuator positions and the swashplate parameters is derived such that an actuator failure accommodation scheme can be established through swashplate reconfiguration. Based on any of the three main rotor actuators failing, the mechanism provides control for the aircraft attitude (pitch and roll) while sacrificing the vertical control [14]. Compensation for loss of vertical control is achieved through the fly-to-trim velocity (FTTV) and rotor speed control (RSC) schemes. The FTTV works with the assumption that there is a longitudinal velocity that the helicopter can reach that will negate the loss in vertical velocity. This is not always practical

in confined spaces especially for unmanned helicopters [57].

The use of an optimal Kalman filter approach for the sensor and actuator fault accommodation of unmanned systems is presented in [58]. The state covariance matrix is reconfigured through the removal of a detected faulty signal, although the authors admit that the computational load of such a technique will increase while improving the state estimation accuracy. A traditional Proportional-Integral (PI) controller is updated by including a feedforward filter to improve reference tracking response while minimising high-frequency dynamics. This method was tested on a time-invariant simulation model with a stuck actuator fault and demonstrated an increase in system stability. The above approach does not explicitly address the successful implementation of online parameter estimation for control reconfiguration under low excitation or the switching mechanism for controller gain tuning. The survey paper presented by Yu and Jiang [22] gives context to such issues as they play a part in the use of flight control reconfiguration in the various aerospace platforms.

Nonlinear model predictive control (NMPC) has also been applied on unmanned fixed-wing aircraft for controller reconfiguration [59, 24]. The NMPC requires the solution of an optimal control problem which was achieved through a pseudo-spectral discretisation method. Thrust and elevator upper and lower constraints were specified and used as inputs in the controller design process with the engine loss as the induced fault. One major drawback of NMPC is that it requires repeated computation to find the solution to the finite prediction horizon optimisation problem in order to generate feedback inputs. This can become computationally expensive and impractical for online implementation. The issue of high computational cost incurred in the general NMPC problem is addressed for the real-time implementation of a MPC controller for the trajectory tracking of an unmanned quadrotor [45]. The full nonlinear dynamics were simplified by exploiting the natural high-gain of the propeller dynamics and the passive characteristics of the attitude dynamics resulting in a linear MPC problem through feedback linearisation. The controller was implemented and evaluated on a Pixhawk autopilot with commanded values transmitted from a ground station using radio waves.

The mitigation of unfavourable switching transients during control reconfiguration and the prevention of healthy actuators from saturation was investigated in [60]. This was achieved through the adjustment of the control input by using the μ-modification technique. The latter implements a virtual safety bound around each remaining actuators by using a weight

factor combined with a saturation duration parameter (which is inversely proportional to the amplitude of the virtual bound). A finite-time control law was used to compensate the error between the faulty aircraft dynamics and the reference model. The state-space matrices were converted to filter states which were solved by computing an auxiliary integrated regressor matrix combined with a forgetting factor. The stability analysis of the closed-loop dynamics was performed and demonstrated that the error dynamics converge to zero while the system states are conserved in a convex set. Although this approach provides an improvement in controller performance while ensuring the safe operation of the remaining actuators, the issue of *a-priori* knowledge for the post-fault reference model was not addressed.

Due to the persistent concern of the UAV safety and reliability for commercial operations, the development of a reconfigurable control system for unmanned aircraft based on a neural network direct adaptive controller was investigated in [61]. The methodology makes use of the NN architecture to provide online adaptation for the dynamic inversion problem. This is achieved through the neural network approximating the inversion error provided that the system matrix can be made into a Hurwitz matrix through the proper choice of positive diagonal matrices by solving a Lyapunov function. The network structure made use of a single-layer of hidden neurons with a sigmoid activation function. Reconfigurable control is achieved through the control allocation of the computed deflection commands. The combination of the genetic algorithms with the neural network structure, has been shown to optimise the gains of the quadcopter PID controller. This enabled the authors to address the issue of implementing non-deterministic methods in the context of UAV flight control certification by keeping the baseline controller architecture deterministic.

The gain-scheduling control approach within the framework of H_∞ synthesis was investigated in [62]. The issue of compensating for post-fault controller errors has been addressed by using a two-stage Kalman filter for simultaneous state and parameter estimation of a quadrotor UAV affected by actuator loss of effectiveness. The common gain-scheduling problem of storing numerous sets of lookup tables was mitigated by replacing them with parametrised polynomial functions. The tuning of controller gains, for a reduced scheduling scheme, was achieved through multi-objective offline H_∞ optimisation routine which considered both time-domain and frequency-domain requirements. The reduction of tuning space was achieved through parametrisation of the gain surfaces. This reconfigurable control method was tested against single and multiple abrupt actuator faults including single

incipient actuator fault. This method relies on the assumption that the post-fault model can be linearised given the nonlinear coupling effects are negligible and the real-time constraints of hardware memory are not violated.

The combination of control allocation approach and weighted pseudo-inverse technique for UAV control reconfiguration has been presented in [63]. A simulation model with linear dynamics for the blended wing UAV was developed which included various abrupt actuator faults such as floating deflector, loss of effectiveness and lock-in-place. The pseudo-inverse method makes use of the estimated control distribution matrix and a virtual control input vector. A weight factor was added in the minimisation of the cost function to improve the efficiency of the remaining control surfaces by preventing early saturation. The concept of attainable moment subset (AMS), which represents all the moments (roll, pitch and yaw) achievable using all the constrained controls, was used to evaluate the various control allocation techniques. Although this technique has a direct impact on the post-fault model control effectiveness, it is unclear on the real-time performance of the reconfiguration scheme once nonlinear coupling effects have been taken into consideration and multiple optimisation channels have to be considered.

Although a reconfiguration scheme was not activated due to the occurrence of a fault, the development of a control allocation strategy for a tilt tri-rotor vertical take-off and landing UAV is relevant due to the robustness requirements with changing dynamics [64]. The autopilot architecture is developed through adopting a multi-loop structure. The faster inner-loop stabilises and controls the attitude dynamics while the slower outer-loop controls the position dynamics. A dynamic inversion-based control allocation is computed by synthesising the virtual control input through a sliding mode approach. To alleviate the chattering problem often found in sliding mode control, an exponential approach law with a saturation function was defined for both inner and outer loops. Performance analysis was achieved by examining controller rejection to low-frequency disturbance which can be classified as a recoverable time-based incipient fault. Robustness against model parameter variations was also evaluated with the system remaining stable. Real-time constraints and the large number of tuning parameters, are some of the issues that will have to be addressed with this approach.

Recently, an integral terminal sliding mode controller (ITSMC) that guarantees finite-time convergence for a quadrotor UAV suffering simultaneous actuator faults, exogenous disturbances and actuator saturation was investigated in [12]. The control law was constructed

using parameters of the sliding surface such that the asymptotic stability could be proved by Lyapunov stability criteria. Given that not all system states are measured in practice, the ITSMC design makes use of a fuzzy state observer to estimate the unmeasured states. The fuzzy logic system output consists of a singleton fuzzifier, product inference engine and centre average defuzzifier which are used to compute an optimal parameter vector for nonlinear dynamics approximation. Closed-loop asymptotic tracking of the desired trajectory in the presence of loss of effectiveness faults and saturation limits was proven by Lyapunov stability criteria by augmenting the ITSMC control law. This design method was implemented in the inner-loop for attitude and altitude stabilisation while the outer-loop was based on a PID design for position control.

2.4 Integrated Fault-Tolerant Control (FTC)

Over two decades ago, the issue of system safety and reliability was discussed in the context of a single module which contained the design methodology for the fault detection and robust control [65, 66]. It was concluded that the robustness of an FDD process can only be improved if the chosen method takes into consideration the complexity of the system being monitored and the stability margin of the closed-loop system. Due to hardware constraints, the online controller reconfiguration mechanism required to define the approach as an active FTC was not considered which limited the application to industrial plants. A two-stage Kalman filter was proposed as a solution for estimating simultaneously the model faulty states and control effectiveness factors [67]. But this method still did not consider highly-coupled nonlinear dynamics and the complexity of implementing in a real-time environment.

Zhang and Jiang investigated an integrated FDD and reconfigurable control system design approach [68]. A two-stage adaptive Kalman filter was used to perform fault identification and estimation of the state and control matrix for the reconfiguration of an eigenstructure assignment controller. Control effectiveness factors were used as a measure to quantify faults entering control systems through actuators. It was found that the reconfiguration mechanism was activated when the average control effectiveness errors were below a threshold value. Reconfigurable proportional-integral (PI) controller is used to recover steady-state performance and reject unknown disturbances. The above approach was evaluated using a

linearised longitudinal vertical takeoff and landing (VTOL) aircraft model. A short transient response and small steady-state error were used as performance evaluation criteria although coupling effects present in a post-fault model were not considered.

The implementation of a RC as part of a FTCS has mainly been done in isolation from the FDD scheme. Moreover the reconfiguration scheme once a fault has been diagnosed, is modified with the assumption that the post-fault system is actually degraded hence the reconfiguration process assumes a more conservative route over-penalising system performance. An integrated design between the various subsystems is required for FTCS to operate in harmony and to recover the pre-fault system performance as much as possible. A robust integrated controller approach which makes use of the H_∞ optimisation technique is shown in [69]. The four-parameter problem solution was achieved by specifying weight functions for the feedback control and diagnosis residual signals for actuators and sensors. This was applied to LTI models and validated through time simulations of a nonlinear model of the aircraft. No consideration of the reconfigurable nature of this approach was discussed.

An H_∞ output feedback control combined with an improved unknown input observer is proposed to address the issues concerning the integrated design of an active FTCS [34]. An adaptive fault identification scheme is implemented through an adaptive law with projection mapping of the estimation parameter. The output control law in the H_∞ sense does not require the convergence of the fault parameters given the adaptive fault parameters are updated based on the projection matrix. Simultaneous RC performance and accommodation of actuator faults was achieved by solving nonlinear matrix inequalities in a linear sense. The switching of controllers will only occur once the LMIs have been solved, which in the event the fault parameters do not converge or change suddenly could be catastrophic. This concept is tested in a simulation with a helicopter which suffers loss of actuator effectiveness at random values and time events.

A neural network-based fault-tolerant scheme which made use of the implicit function theorem is proposed in [70]. This made use of RBFNNs for the design of an adaptive RC. An integrated approach which combined a passive (robust) FTCS scheme and a threshold value switching logic to activate a simple active FTC scheme without an FDD component. The concept of a fault alarm threshold value was used to determine which controller should be used based on whether a fault was detected or not. The time delay due to fault location and classification was not investigated and the system under test had a simple sinusoidal behaviour

with minimal cross-coupling effects after fault had occurred. It was conceded by the authors that such a method was applicable to a small subset low-order nonlinear systems. A few years later, the authors investigated the impact of fault detection time delays by adopting a controller switching mechanism once a fault is detected but not diagnosed [71].

Integration issues are identified and discussed in [22]. A problem that has been highlighted is that the various subsystems, in a typical active FTCS, are activated in a sequential manner thereby extending the time between when a fault has occurred and the RC issuing a command to the actuators. One concept of an integrated FTCS design is to perform controller reconfiguration even with an imprecise post-fault model provided by the FDD scheme. But this is based on the online synthesis of the robust controller which is traditionally very slow and computationally expensive. It has been suggested that implementing an integrated design also requires a trade-off between fault diagnosis accuracy and reconfigurable control performance. The use of a forgetting factor has been suggested to increase the speed of the FDD scheme with regards to a fault parameter estimation which in turn reduces the transient behaviour during the reconfiguration process.

An integrated design fault detection and estimation and control system reconfiguration using a robust control problem approach based on H_∞ optimisation is proposed in [72]. The direct use of a fault estimation function without the need for a reconfiguration mechanism, results in an inclusive design of an observer for fault estimation and a sliding mode controller state/output feedback control. The authors suggested that there exists bi-directional uncertainties generated by (1) the model mismatch between the observer and the control system and (2) fault and state estimation errors, which substantiates the need for an integrated FTCS design process. This was based on a subset of systems with additive and multiplicative faults. Simulated results based on the stabilisation control of a DC motor was demonstrated. Although the above observations were not validated on UASs, the management of bi-directional uncertainties within an integrated FTC is worth noting.

Adaptive sliding mode control (ASMC) is used in the design of a FTCS which could tolerate actuator failures by limiting the amplitudes and rates of the healthy actuators. This approach made the assumption that the remaining actuators could perform the same function as the failed ones which lends this method towards more passive FTC [60]. The use of H_∞ and $\mu-$ synthesis controller without the need for reconfiguration or adaption was investigated. An integrated approach was introduced by adding a control redistribution algorithm using

fuzzy logic technique. This was heavily based on the assumption that a zero-delay fault detection system was already implemented. A similar approach was taken in Almutairi's thesis where a fault compensation approach was added to the nominal LQR controller [73]. An H_∞/μ synthesis approach combined with fuzzy logic was able to achieve reconfigurable control without *a-priori* knowledge of the post-fault model dynamics.

The issue of bi-directional interactions between the fault estimator and the fault-tolerant controller was investigated by taking into consideration implementation factors for a real-time system [74]. The traditional separated approach in FTC synthesis is presented and compared with an integrated FTC design. The dynamic model of an uncertain nonlinear 3-DOF helicopter system with both actuator faults and saturation was used to demonstrate the degradation of closed-loop stability when an integrated FTC design is ignored. Simulated results show that a separated FTC design will exhibit slow transient response with large overshoot and long settling time driven by poor estimation performance. This can result in overall FTCS instability if the ignored system uncertainty caused saturation in the remaining healthy actuators. Although the above observations motivate for the use of an integrated approach, model-based fault observer will inherently increase the level of uncertainty given the unknown dynamics of a post-fault nonlinear system.

Multiple identification model were developed to formulate an integrated framework for the fault-tolerant of unmanned systems in a formation flight geometry [75]. The nominal controller was designed using robust state feedback H_∞ synthesis which incorporated aerodynamic uncertainties induced by close formation flight vortex effects. A set of RCs were designed for each type of actuator fault and a switching mechanism based on the minimisation of each controller cost function was adopted. Adaptive element was added to the reconfiguration mechanism such that inner (stabilisation) closed-loop system is asymptotically stable provided well-bounded error signal. The above approach assumes linear aircraft dynamics in the estimation of external disturbances and control inputs without making reference to the FDD mechanism and reconfiguration delays due to uncertain estimation errors below defined trigger threshold.

In recent studies on FTCS, more focus has been placed on actuator faults rather than sensor faults given the latter will not modify the system's dynamic response due to the latest flight controllers carrying redundant sensors at negligible cost and weight. The use of an optimisation routine combined with the control allocation technique for an over-actuated aircraft with

actuator faults have been proposed in [76]. An optimisation in quadratic form was implemented to minimise the control effort of the remaining actuators and it is a flexible approach in system with a large number of effectors. A nonlinear dynamic inversion-based control law and a sliding mode observer were combined to compensate for the system nonlinearities and effective fault reconstruction. It was obvious from the results that fault reconstruction process introduced system noise which needed to be filtered in order for fault observer to trigger the control allocation mechanism. This issue of detection delays combined with systems with small time constants (such as under-actuated quadcopters), limits its application.

An insightful study into the effect of FDD uncertainties on the robustness requirements for a RC has been presented in [77]. Such uncertainties will appear in the form of (1) false alarms (erroneous detection), (2) missed detection (positive fault occurrence) and (3) detection delays (time between fault occurrence and detection). Another source of disturbance is the reconfiguration mechanism (especially in switching techniques) inducing unrecoverable instability. An H_∞/μ synthesis approach combined with an adapted DK-iteration is used to design a bank of robust controllers with the ability to compensate a subset of bounded actuator faults which might be undetected. To minimise undesirable transient response of the closed-loop system during reconfiguration, an objective function which is minimised for the closed-loop is used to initialise system with an optimal initial state. Experimental validation achieved good results although *a-priori* knowledge of the post-fault model does not incorporate coupling dynamics.

2.5 Identified Gaps in the Existing Literature

- As stated by [28], the lack of a real-time application of knowledge-based FDD methods, specifically using artificial neural networks in unmanned aerial rotorcraft, is clearly evident in the literature. Moreover, the function approximation capabilities of nonlinear systems, which embodies the behavioural pattern of a system at fault, is at the centre of neural networks' capabilities. The main drawback in their application is the speed of adaptation, convergence and computational throughput especially in low-cost electronics.

- The development and real-time implementation of a learning alogrithm and simpler network architecture, such as the extended backpropagation algorithm and a RBFNN

respectively for FDD of a miniature unmanned rotorcraft has not been investigated and is a focal point in this research.

- The investigation of various actuation and sensor faults for unmanned rotorcraft have been investigated in various forms [28] for the purpose of reconfigurable control. However, an integrated approach which takes into account the use of fault accommodation controllers and RCs has yet to be investigated for unmanned rotorcraft without *a-priori* knowledge or assumption of the post-model dynamics.
- The use of a neural-network parameter estimation approach based on system identification manoeuvres, which has been shown to be successful in the online estimation of aerodynamic parameters [78], to detect actuator fault information necessary for online controller reconfiguration, has not been investigated and is another focal point in this research.

2.6 Research Problem and Research Question

The application of FTCSs on miniature unmanned rotorcraft has far reaching implications on their safety and reliability given their increasing operation in populated areas and the scalability of their technologies on manned rotorcraft. Currently, this area of research still requires further exploration. Unlike fixed-wing aircraft and large-scale helicopters, miniature rotorcraft exhibit higher thrust-to-inertia ratios making them more agile and requiring high-bandwidth control effort [14]. And yet, successful control techniques can be easily applied to their manned counterparts. A combination of theoretical and experimental research is deemed essential to demonstrate the potential of such systems.

2.6.1 Problem statement

Unmanned rotorcraft (specifically quadcopters) are inherently unstable and have a complex electrical/electronic architecture. This makes their applicability in an urban environment almost impossible without some form of intelligent control. There has been efforts to apply fault-tolerant control methodologies to address this problem but the focus has been either on controller reconfiguration or FDD. Investigation into an integrated fault-tolerant approach has been limited in the literature. The task is to develop a neural network-based integrated

FTCS that will improve the stability and mission success of an unmanned rotorcraft in the presence of actuator faults.

2.6.2 Research questions

Does the development of an integrated FTCS neural network-based approach have the potential to be easily implementable on different configurations without the need for excessive fault controller redesign and for accurate mathematical models?

Can such a system prove to enhance robustness and minimise performance degradation while being more forgiving in the identification of specific faults given its ability to approximate the faulty nonlinear behaviour of system in question?

Will the adoption of such a system be able to increase the capability and reliability of unmanned rotorcraft systems in various tasks such as: border patrol, traffic monitoring, search and rescue, power line inspection and broadcasting of mobile telecommunication within a dynamic environment?

2.7 Research Objectives

The main objective of this research is to investigate the use of a neural network-based integrated FTCS to improve the resilience of an unmanned miniature rotorcraft in the presence of faults, with specific focus on incipient actuator faults. The following sub-objectives are applicable:

- To develop the dynamics of a miniature unmanned quadrotor in the presence of incipient actuator faults.
- To develop and test a quadrotor hardware and software system with the capability of evaluating incipient actuator faults models and FTC algorithms.
- To synthesise a neural network-based integrated FTC architecture for an unmanned aerial rotorcraft.

- To quantify the robustness and performance of such an integrated neural network-based FTCS within real-time constraints.

2.8 Research Methodology

The methodology followed in this thesis is derived as means to achieve the research objectives described in Section 2.7. Given that the identification of incipient actuator fault models is required for the simulation of a high-fidelity quadrotor model, the acquisition or development of a quadcopter hardware platform with adaptable flight controller software is a requisite.

Moreover, such a flight controller software should be able to evaluate the effectiveness and suitability of the developed FTC algorithms both within a real-time environment and through offline analysis. The software tools should also be flexible enough to assess the individual parts of a FTCS and the integrated FTC architecture.

2.8.1 Research platform

In order to study the underlying dynamics of quadcopter post actuator fault, the physical model will have to be studied including the impact of each part on the overall behaviour. The acquisition of a flight test vehicle could ensure this is possible. Given the objective for a demonstration flight test vehicle (FTV), the following requirements have been identified:

- The generation of flight data from the FTV shall be largely automated as far as possible with minimal human interaction (apart from take-off and landing).
- The FTV shall incorporate (analytical or hardware) redundancy systems on both critical sensors and actuators to ensure the safety during the flight testing campaign.
- The FTV shall be in the case of a rotorcraft.
- The FTV shall have a take-off weight of 900 grams to 2 kg.
- The FTV shall have a maximum main rotor diameter of 1 metre (in the case of a rotorcraft).
- The FTV shall make use of ground control station if necessary.
- The FTV shall have transmitter manual override capability.
- The complete FTV system shall cost not more than R10000.

An unmanned rotorcraft with an off-the-shelf flight control software (such as ArduCopter) shall be used as a risk-reducing platform for all new-algorithms after code verification and validation in desktop simulation and hardware-in-the-loop simulation has been performed.

2.8.2 Flight test instrumentation

The collection of data to perform various data-driven development activities will enable a better understanding of the underlying system dynamics. A flight instrumentation system could be used to ensure the data collection is repetitive and free from measurement glitches. The following design requirements specifications for a flight test instrumentation system is recommended:

1. The instrumentation system shall be able to capture complete flight vehicle state information as specified.
2. The instrumentation system shall be effective in the isolation of flight-induced vibrations.
3. The instrumentation system shall accommodate both automatic and manual control of the flight vehicle.
4. The vehicle shall be able to transport the full-suite of instrumentation and have enough power capacity to perform aggressive manoeuvres such as circuits and pirouettes.
5. The instrumented vehicle shall be able to obtain uninterrupted data for minimum period of 120 seconds.
6. The instrumented vehicle shall have a maximum Take-Off Weight (TOW) of 2kg in any configuration.
7. The instrumented vehicle shall have minimum electronics noise and interference caused by poor grounding and electromagnetic interference.

The Pixhawk PX4 2.4.8 32-Bit ARM Flight Controller is potentially the most suitable candidate to be integrated while meeting most of the requirements. The APM 2.6 Mega flight controller could also be considered because of its smaller size, weight and cost.

2.8.3 Flight test campaign design

The identification of the underlying dynamics of a highly-nonlinear system such as quadrotor requires exposing such a model to various conditions that will enable a better assessment of the chosen fault-tolerant methods. Such conditions can be compiled into flight testing or simulation plan that will be ensure adequate exposure of the underlying dynamics of physical model. The requirements to meet the objectives of a flight test/simulated FTCS concept is given below:

- The following flight conditions shall be attained during the triggering of the FTCS:
  - Hover 20m above ground.
  - Forward speed (GPS measured) of 10 m/s
  - Forward speed (GPS measured) of 20 m/s
- Pre-flight and post-flight checks shall be developed and implemented for each flight.
- The video recording shall be used either on-board the FTV or on the ground. If on-board, it should be of minimal weight and size.
- The flight test campaign shall be done outdoors and within the confines of a 100m x 100m flight testing field. This is to illustrate confined urban spaces.
- Post data filtering shall be performed if necessary.
- All data analysis shall be done using an industry-standard software (MATLAB).
- A constrained environment which will minimise the number of crashes without compromising the flight test objectives should be investigated.
- Investigation into using flight data to perform online system identification for the purpose of verifying the mathematical model, will be performed.

2.8.4 Development and implementation of an integrated fault-detection and diagnosis scheme

A critical component to a FTCS is its ability to detect, isolate and identify a set number of faults. An integrated FTCS design can be achieved provided such a routine exist and

operates within a real environment. The following requirements for such a routine can be specified:

- The fault detection algorithm shall be implemented into a microprocessor and executed in real-time without data overload.
- The fault detection algorithm shall be able to detect, identify and time-stamp the fault that occurred.
- The following faults shall be investigated: rotor speed failure, propeller slippage failure, broken propeller failure.
- The following sensor faults shall be investigated: sensor bias and multiplicative faults (both accelerometer and gyroscope).
- The following types of fault detection schemes shall be tested: two-stage adaptive Kalman filter, a novel RBFNN-based parameter estimation scheme.
- The following neural network architectures shall be investigated: RBFNN and MLPNN.

Note: The implementation of the FDD scheme will largely depend on the real-time performance requirements of the developed algorithms.

Proposed integrated FTCS architecture

Figure 2.3 shows a possible concept for a neural network-based integrated approach to fault-tolerant control. The premise behind the design is that online learning happens simultaneously for the neural network-based controller and fault estimator. The outputs from the fault estimator, in the form of weight and biases, are used in a reconfiguration mechanism to determine the augmentation factors required for the controller to reconfigure for augmented control inputs in the event a fault has occurred.

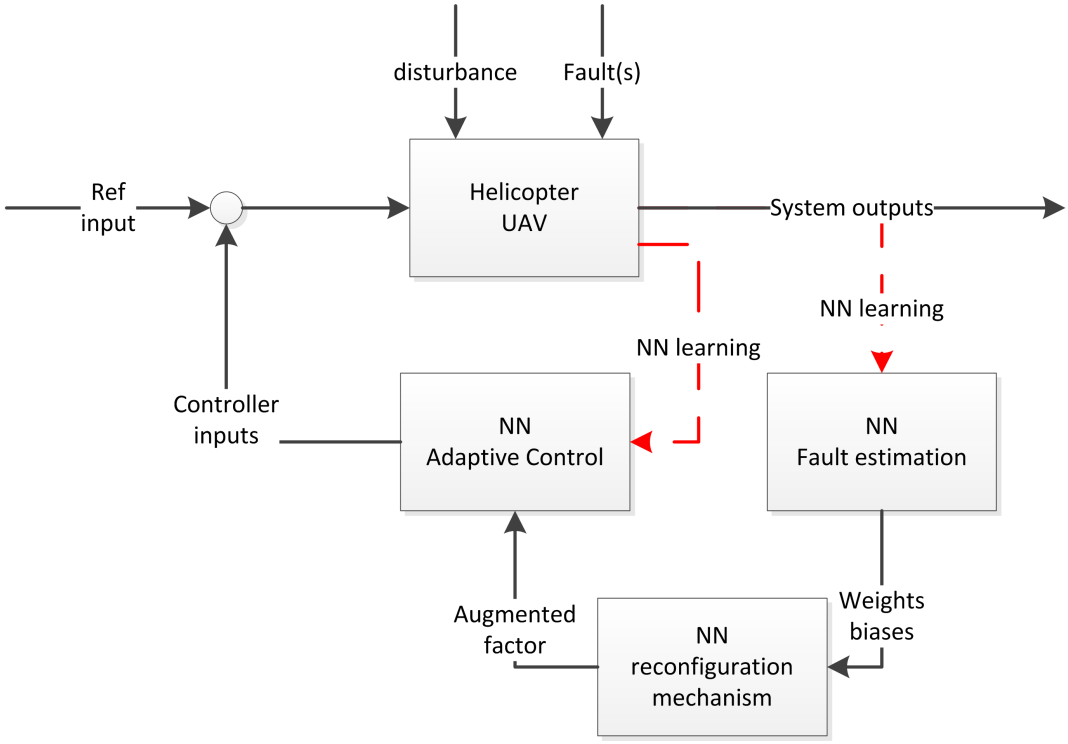


Figure 2.3: Proposed structure of an integrated FTCS.

2.9 Requirements for the Development and Implementation of the Reconfigurable Mechanism and Controller

The successful reconfiguration of a controller is an important aspect of system stability and performance. Given that it will form part of an integrated FTCS architecture, the following requirements can be specified:

- The controller reconfiguration algorithm shall be implemented on a microprocessor and executed in real-time without data overload.
- Actuator control signal reallocation shall be investigated.
- The reconfiguration and switching logic of the nominal controller shall be investigated only based on the design outcome of the fault detection algorithm.
- The following types of controller reconfiguration schemes shall be tested: reconfigurable PID controller, a novel neural network-based adaptive controller.
- The following neural network architectures shall be investigated: RBFNN.

2.10 FTCS Hardware-in-Loop System

Hardware-in-the-loop-simulation (HILS) is a well known process applied during the test and evaluation of complex systems [79]. The main reason for requiring a Hardware-in-Loop System (HILS) system is to test and evaluate the performance of the FTCS given that real-time simulation introduces computational delays and signal discretisation which negatively affect the robustness and performance of the controller. HILS can also act as a cost-effective verification gate prior to commencing flight testing. The implementation requirements for a HILS system are as follows:

1. A microprocessor card(s) with floating-point capability shall be used. This will host the fault-tolerant control embedded algorithms and state logic.
2. A sampling time (for the controller and plant dynamics) from an Input/Output card shall be at least 50 Hz.
3. A transmitter shall be incorporated into the HILS environment and its Pulse Width Modulation (PWM) signals captured with a sampling rate of 50 Hz. This is required to evaluate the design of a flight plan procedure required to perform an autonomous mission.
4. The emulation of GPS and IMU sensors shall be performed on a HILS interface board. It will be investigated at what level of complexity this emulation will need to be performed.
5. The transmission data to the on-board computer shall be in the native format of the respective microprocessor hardware. This is vital to ensure the embedded software in the microprocessor is compatible with flight test data.
6. The communication protocol shall be synchronous serial communication, such as ARCNET/CAN, or asynchronous serial with a baud rate of at least 9600 bits/per second. This is required to prevent additional delays.
7. A UART/serial link shall be used for the GPS transmission/emulation.

2.11 Criteria of Verification and Validation

The following verification and validation (V & V) criteria will be used for the FTCS algorithms developed:

- Demonstrate FTCS robustness and performance (in classical control theory terms: at least 3dB gain margin and 35 deg phase margin throughout the reconfiguration transition period).
- Demonstrate overall FTCS stability.
- Demonstrate command tracking (trajectory or acceleration tracking).
- Demonstrate recovery of attitude control (regulation control).
- Assess the effectiveness of measurements to quantify the occurrence of actuator, sensor or systems faults within the hardware-in-loop simulation environment.
- All the above metrics shall be implemented in a nonlinear 6DOF desktop and HILS simulation model.

2.12 Envisaged Contributions to Knowledge

The following contributions are envisaged as a result of undertaking this research:

- A better understanding of the dynamic behaviour of a highly manoeuvrable and unstable nonlinear system, such as a miniature rotorcraft in the presence of faults during an autonomous mission.
- The design and implementation of a learning algorithm, applied to both fault detection and control reconfiguration, for a neural network-based integrated FTCS with specific application to miniature unmanned rotorcraft.
- The development of an evaluation framework that would enable the verification and validation of integrated FTCSs with specific application to miniature unmanned rotorcraft.

Chapter 3

Development of an Unmanned Aerial Vehicle with Faulty Dynamics

3.1 Introduction

The development of complex aerospace systems such as: UAVs, commercial aircraft, lunar landing modules, etc., often requires experimental testing using representative hardware prior to operational use or product launch. The system design, build and integration based on analytical or numerical methods is based on simplified assumptions which need to be validated through observing the system under-test within its real environment. The process of using perturbations as input signals into the system in order to study its behaviour in areas which require better knowledge such as: cross-coupling effects, actuator failure dynamics and more, has resulted in decades of research into the field of system identification [80, 18, 81, 82].

Unlike white-box modelling which makes use of only *a-priori* information (both for the type of functions relating to various variables and the theoretical parameter values in those functions), the system identification process (which forms part of data-driven modelling techniques) can be used within a grey-box or black-box modelling framework. The main difference between these two approaches is that black-box modelling (such as neural networks) completely disregards the availability of *a-priori* information and grey-box modelling (such as polynomial regression) assumes a theoretical model structure and uses experimental or

empirical data to completely describe the system dynamics. Chapter 4 describes in detail the application of nonlinear black-box modelling for the system identification of an unmanned quadcopter system with actuator faults.

Moreover, the introduction of low-cost unmanned systems, specifically quadrotor drones, to the mainstream consumer electronics market has redefined performing tasks such as: capturing of a self-portrait, showcasing a home or hiking in remote areas. It has created new possibilities which were previously not available to the average sport enthusiast such as: video capture of high-speed trail biking, close-up videos while surfing or wide-angle shots while rock-climbing [83]. Unfortunately, the mechanical architecture of most of these drones are such that they are prone to failure where the main cause is due to a defective propeller or motor. Such a fault is usually not accounted for or easily detected by the on-board flight computer and its dynamic behaviour under such a faulty condition is poorly understood, resulting in a catastrophic event [32]. The use of system identification to better understand the behaviour of a system under fault, is an essential tool in the improvement of flight computers for safe operational conditions.

3.1.1 System identification architecture

Adapted from the Quad-M flight vehicle system identification framework described in [84], Figure 3.1 introduces the system identification framework used throughout this thesis. The three major E's, namely Experiment, Estimate and Evidence form part of the system identification of an UAV affected by incipient fault dynamics. The *Experiment* stage involves the building of a test-bench platform representative of real-time conditions, for the generation of repeatable data. In order to achieve satisfactory prediction of system behaviour under faulty conditions, two representative test-bench platforms were developed: (1) An UAV nonlinear model, executing within a real-time simulation environment and representative of the complete avionics setup described in Section 3.2, (2) A rotor-propeller hardware configuration, representative of an UAV under incipient actuator faults described in Section 3.4.

The *Estimation* and *Evidence* stages, described in more detail in Chapter 4, are used to estimate and validate the predicted behaviour of the above-mentioned test-bench platforms under incipient fault conditions. The rotor-propeller estimation model is used as a fault emulator input in the system identification of UAS test-bench platform under normal and

faulty conditions as described in Chapter 4. The rationale behind using a validated real-time software-based system identification architecture versus a traditional hardware architecture for generating identification data was to mitigate hardware failures due to unrecoverable induced actuator faults and provide a deterministic process enabling the development of FDD algorithms and fault-tolerant controllers described in Chapter 5.

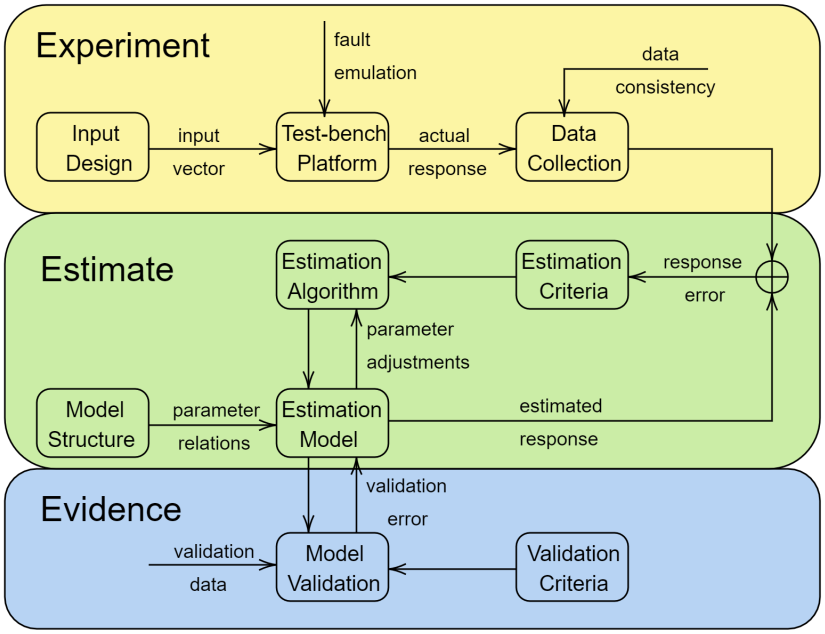


Figure 3.1: System identification framework

3.1.2 Outline

This chapter is organised as follows. Section 3.2 discusses the avionics setup used in the development of the aerial platform. Section 3.3 introduces the quadcopter dynamics and fault emulation algorithm used for the software development of the real-time test-bench platform representative of the avionics hardware. The system identification of the incipient fault condition using the above avionic setup is discussed in Section 3.4.

3.2 Platform Hardware Development

3.2.1 Airframe

The airframe is an H-frame design configuration developed by Uav4africa (Pty) Ltd [2] with improved forward flight stability and larger electronics bay as compared to the more popular X-frame design. This can be seen in Figure 3.2.

Figure 3.2: Uav4africa research platform: H-1 quadrotor - without high-impact landing gear

Two carbon-fibre plates sandwiching aluminium square tubing constitute the airframe body. This created space for high voltage electrical wires. It is powered by four RCTimer 5010-620KV motors and 10×4.7 inch propellers. A 4-cell 14.8V LiPo battery powers the flight controller and the power train. A separate 3-cell 11.1V battery powers a 3-axis camera gimbal. The completed airframe was then used for measuring the mass moment of inertia on all 3 axes [2] to be incorporated in the nonlinear model.

Table 3.1: H-1 quadcopter mass properties

Parameter	Min	Mean	Max
Mass (kg)	-	1.75 kg	-
I_{xx} $(kg.m^2)$	0.03259	0.03274	0.03298
I_{yy} $(kg.m^2)$	0.04117	0.04202	0.04295
I_{zz} $(kg.m^2)$	0.07270	0.07327	0.07467

The mass properties, obtained using a bifilar pendulum method [85], are given in Table 3.1. The off-diagonal mass moment of inertia terms were ignored as they were found to be quite small in comparison to the principal diagonal terms. The values formed part of the airframe parameters used in the test-bench platform model which also included estimated drag coefficients and centre of gravity offsets.

3.2.2 Flight controller

The validity of using a real-time simulation environment as a system identification test-bench platform, required a flight control computer and its associated firmware/software to be flexible enough to adopt extra functionality and have mature emulation software functions. The Pixhawk 2.4.8 flight controller, shown in Figure 3.3, was chosen as the primary flight controller. A modified version of the ArduCopter 3.4 software enabled the integration of system identification input manoeuvres. The modification architecture is described in Section 3.3.1.

The real-time emulation of the 32-bit ARM Cortex M4 core with floating point unit (FPU) occurs through timing callback functionality which prioritises each function call based on its interface to the I/O (input/output) devices.The IMU emulation was simply done with superimposed signals errors. Virtual allocation of the Pixhawk 256 KB of RAM is also emulated to ensure that the implemented system identification routines can run in real-time. The emulation of General Purpose Input/Outputs (GPIOs), which are used to control actuators (motors), servos, LEDs and relays were also implemented so that the state status transmission to the Mission Controller described in Section 3.2.5 was achieved.

3.2.3 Powertrain

The airframe was powered by a set of four brushless RCTimer 5010-620KV motors (shown in Figure 3.4), APC 10×4.7 inch (25.4 by 11.43 cm) multirotor propellers and 30A electronic speed controllers with BEC (Battery Eliminator Circuit) support. Contact friction between the motor shaft and propeller using a torqued nut and washer was the attachment method. A power module was also installed to ensure a stable power supply to the flight controller and while monitoring the 4000mah battery (Lithium-Polymer-Ion) voltage and total current draw from the motors.

Figure 3.3: Pixhawk 2.4.8 flight controller

In order to include high-order dynamics of the propeller, wind tunnel data for a 10 by 4.5 inch propeller was processed and modelled [86]. Although it does not match the propeller pitch angle for the research platform used in this research, this difference was deemed negligible. This is shown in Figure 3.5. The propeller thrust and torque characteristics are a function of the forward flight and rotor speed. μ is the advance ratio. This was utilised in the flapping dynamics model used within the test-bench platform which is described in Section 3.3.2. The modelling of the incipient fault condition, as introduced in the Section 2.2, made use of the above powertrain configuration. The methodology and resulting models used in the test-bench platform are described in Section 3.4.

3.2.4 GPS

The Readytosky Ublox NEO-M8N GPS and compass module, shown in Figure 3.6, was mounted on the airframe such that the electromagnetic interference from the power electronics was minimised. This module was chosen due to its ability to process multiple GNSS (Global Navigation Satellite System) signals such as: GPS, GLONASS, Galileo-ready, BeiDou and QZSS. It also contains a compass which is a HMC5883L digital triple-axis magnetometer module.

The emulation of the GPS module within the test-bench platform was achieved by incorporating various functionalities such as: sample delay, transmission byte loss percentage, glitch

Figure 3.4: RCTimer brushless DC motor.

offsets in latitude and longitude, number of visible satellites and amplitude on the altitude error which is used in the trajectory tracking controller.

3.2.5 Ground station

A ground control station (GCS) software called Mission Planner, as seen in Figure 3.7, was used for the planning and monitoring of UAS execution of a system identification manoeuvre profile. This was installed on a Windows 7 computer. The MAVLINK (Micro Air Vehicle Link) communication protocol is chosen combined with a 433 MHz telemetry receiver to transfer data packets from the GCS and the UAS.

Given the simplicity of the MAVLINK protocol architecture, the emulation of the communication channel between the test-bench platform and the GCS was achieved by passing the same data packets through a virtual TCP (Transmission Control Protocol) port with signal latency emulation. The logging capability of the GCS was also used to analyse data recorded by the test-bench platform prior to further post-processing.

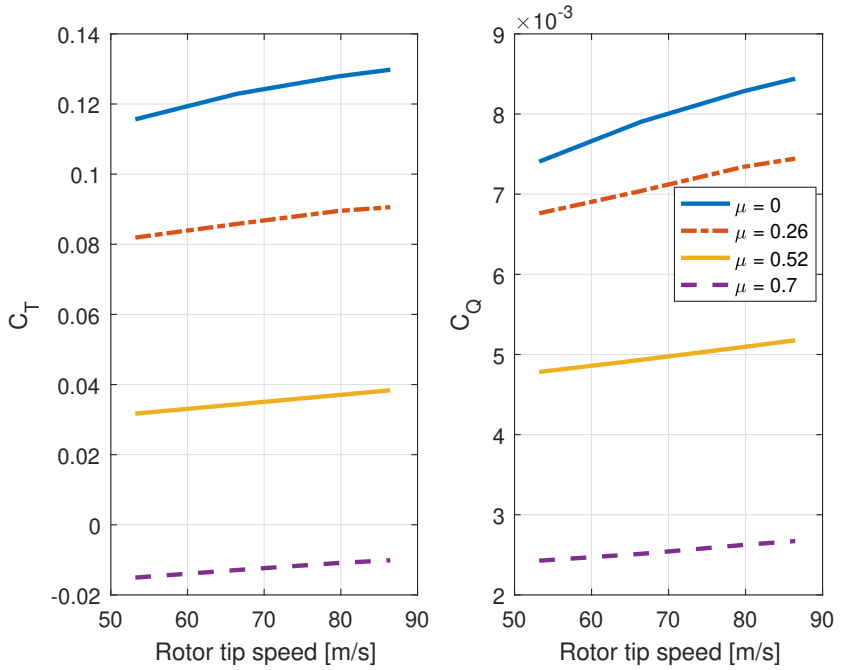


Figure 3.5: Propeller thrust/torque coefficients - wind tunnel data

3.3 Platform Software Development

3.3.1 Software architecture

The test-bench platform software architecture was developed such that it will be able to:

- Introduce faulty rotor dynamics (as described in Section 3.4) through an emulation routine ensuring repeatable system behaviour to be used by model identification algorithms.
- Support the generation of system identification data for the representative hardware of a UAV.
- Interface with a nonlinear identification routine which will use the generated data in real-time as discussed in Chapter 4
- Integrate an FDD routine as discussed in Chapter 5.
- Accommodate a reconfiguration mechanism which will influence the performance of the test-bench platform in the event of a faulty condition as discussed in Chapter 5.

Figure 3.6: M8N GPS/Compass module with folding mount.

The various routines were incorporated into the test-bench platform framework for the purpose of the activities described above without the need to manage separate software baselines. This is shown in Figure 3.8.

The Codeblocks IDE (Integrated Development Environment) was used to implement the above routines in the ArduCopter 3.4 baseline. The real-time memory requirements for the nonlinear identification routine (described in Chapter 4) required a separate microcontroller to be used for its validation under real-time constraints. A I2C interface was then used to transmit system data from the test-bench platform.

3.3.2 Quadrotor dynamics

The purpose of quantifying the flight dynamics for quadrotor system as described in the previous sections is to capture the relations between the salient parameters describing its behaviour. These functions can then be integrated in software to form part of the test-bench platform mathematical engine as described in detail in this Chapter. Such a platform can then be used as a source of system data required for the verification and validation of the nonlinear identification methods as described in Chapter 4 and fault-tolerant algorithms as described in Chapter 5.

Figure 3.7: Ground control station running the mission planner software.

Assumptions

The assumptions used to develop the quadcopter dynamics model are [87]:

1. The non-diagonal mass moment of inertia terms are small and can be neglected. Moreover the principal axes, defining the principal mass moment of inertia, coincide with body-fixed reference frame.
2. The induced moments created by aerodynamic forces from the fuselage are neglected as the centre of pressure is assumed to coincide with the centre of gravity.
3. The tip-path-plane flapping dynamics are described as a reduced-order model with two states.
4. The direction of the thrust generated by the rotor is orthogonal to the tip-path-plane.
5. The rotor blades mass moment of inertia is small enough that no cross-coupling effects in the flapping dynamics are considered.
6. Aerodynamic moments generated by the quadcopter rotors are represented by a linear model with a constant torsional stiffness value.
7. The dynamic response of the brushless motor to controller commands is considered negligible.

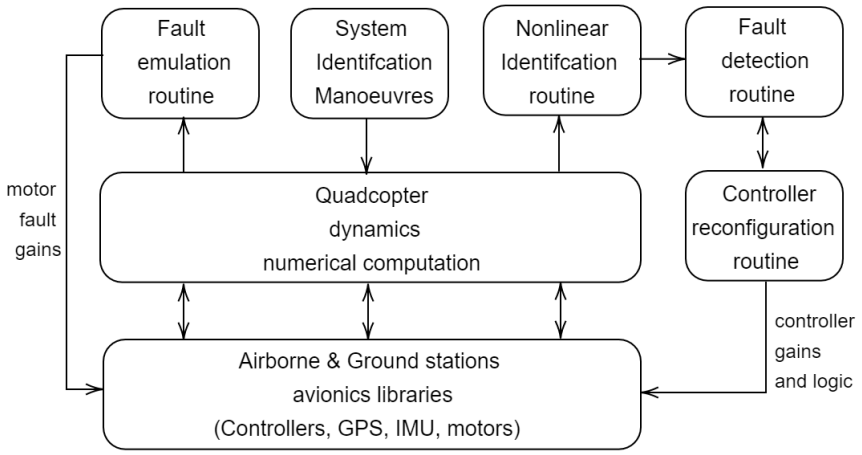


Figure 3.8: Software routines of the test-bench platform developed for this research.

Equations of motion

By considering a coordinate system located at the quadcopter centre of gravity and aligned with the quadcopter body-axis, the following equations of motion can be described [88, 3]:

$$m\dot{\mathbf{v}} + m(\bar{\omega} \times \mathbf{v}) = \mathbf{F} \tag{3.1}$$

$$\mathbf{I}\dot{\bar{\omega}} + (\bar{\omega} \times \mathbf{I}\bar{\omega}) = \mathbf{M} \tag{3.2}$$

where the body-fixed linear and angular velocities are defined by $\mathbf{v} = [u \ v \ w]^T$ and $\bar{\omega} = [p \ q \ r]^T$ respectively. External forces such as: gravity, aerodynamics and propulsion, are assumed to act at the quadcopter centre of gravity with a measured mass of m, are defined as $\mathbf{F} = [X \ Y \ Z]^T$. Based on geometric level arms and the mass moment of inertia defined as $\mathbf{I} = [I_{xx} \ I_{yy} \ I_{zz}]$, external moments are defined as $\mathbf{M} = [L \ M \ N]^T$.

For a given rotor speed Ω and ambient density ρ, the generated propulsive force of quadcopter rotor system is given as [88]:

$$T_i = C_{T_i}\rho(\Omega_i R_i)^2 \pi R_r^2 \tag{3.3}$$

where the number of rotors are represented by $i = [1 - 4]$. The thrust coefficient of the ith rotor defined as C_{T_i}, can be expressed as:

$$C_{T_i} = \frac{a_r \sigma_r}{2}\left(\theta_0\left(\frac{1}{3} + \frac{\mu_{ri}^2}{2}\right) + \frac{\mu_{z_{ri}}\lambda_{0i}}{2}\right) \tag{3.4}$$

where the definition of the ith rotor inflow velocity λ_{0i} can be expressed as:

$$\lambda_{0i} = \frac{C_{T_i}}{2\eta_w \sqrt{\mu_{r_i}^2 + (\lambda_{0i} - \mu_{z_{ri}})^2}} \tag{3.5}$$

$$\mu_{ri} = \frac{\sqrt{(u_i - u_{wind})^2 + (v_i - v_{wind})^2}}{\Omega_i R_r} \tag{3.6}$$

$$\mu_{z_{ri}} = \frac{w_i - w_{wind}}{\Omega_i R_r} \tag{3.7}$$

$$\sigma_r = \frac{2c_r}{\pi R_r} \tag{3.8}$$

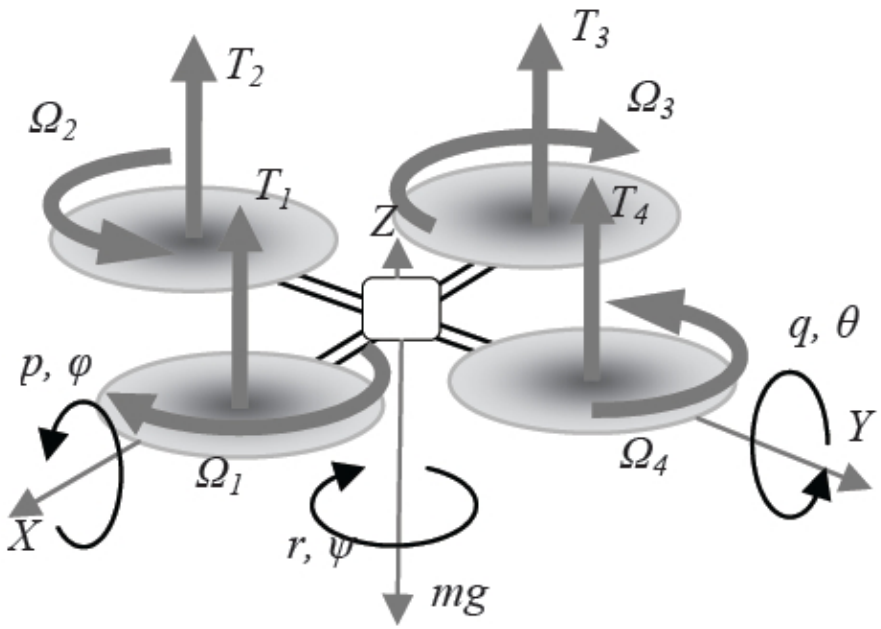


Figure 3.9: Quadcopter rotor forces and moments

where the propeller lift-curve slope is a_r, c_r is the propeller blade root chord (this is approximated as a constant value), R_r and θ_0 are the propeller blade length and the approximated pitch angle respectively (blade twist is assumed to be negligible given the ratio of generated thrust over the span of the blade). $\mu_{z_{ri}}$ and μ_{ri} represent the inflow velocities to be solved based on aerodynamic angles between the quadcopter body-axis and resulting velocity vector [87]. Velocity components based on a local reference frame at each ith rotor is given as

u_i, v_i, w_i. These components can be computed as:

$$u_i = u + S_{u_i} L_{fr} r \sin\theta_{fr} \tag{3.9}$$

$$v_i = v + S_{v_i} L_{fr} r \cos\theta_{fr} \tag{3.10}$$

$$w_i = w + R_{x_i} q + R_{y_i} p \tag{3.11}$$

where L_{fr} and θ_{fr} are the linear and angular distances from the quadcopter centre of gravity to the frame rotor hub, respectively. $\bar{S}_u = [S_{u_1}, ..., S_{u_4}]$ is the distance scale factor for each ith rotor. R_x and R_y are level arms resulting from the quadcopter angular motion along the body x-axis and y-axis respectively.

The computation of the ith rotor torque Q_r is defined as:

$$Q_{r_i} = C_Q \rho (\Omega_i R_r)^2 \pi R_r^3 \tag{3.12}$$

where the torque coefficient is given by:

$$C_Q = C_T(\lambda_{0i} - \mu_z) + \frac{C_{D_0}\sigma_r}{8}\left(1 + \frac{7}{3}\mu_{ri}^2\right) \tag{3.13}$$

where C_{D_0} is the zero incidence angle drag coefficient. To avoid the computationally expensive numerical method of solving for propeller aerodynamics, wind tunnel data was used instead as shown in Figure 3.5. By combining each ith rotor flapping dynamics into a simplified tip-path-plane (TPP) model (excluding the dynamics of a feathering hinge motion), the following can be described [78]:

$$\dot{b}_{1i} = -p - \frac{b_{1i}}{\tau_e} - \frac{1}{\tau_e}\frac{\delta b_{1i}}{\delta \mu_v}\frac{v_i - v_{wind}}{\Omega_i R_r} \tag{3.14}$$

$$\dot{a}_{1i} = -q - \frac{a_{1i}}{\tau_e} - \frac{1}{\tau_e}\left(\frac{\delta a_{1i}}{\delta \mu}\frac{u_i - u_{wind}}{\Omega_i R_r} + \frac{\delta a_{1i}}{\delta \mu_z}\frac{w_i - w_{wind}}{\Omega_i R_r}\right) \tag{3.15}$$

the effective rotor time constant is defined as τ_e is. The effect of TPP dihedral on the longitudinal dynamics is given as:

$$\frac{\delta a_{1i}}{\delta \mu} = 2K_\mu\left(\frac{4\theta_0}{3} - \lambda_{0i}\right) \tag{3.16}$$

K_μ is a constant coefficient that introduces a restoring moment. The dihedral coefficients are defined as:

$$\frac{\delta b_{1i}}{\delta \mu_v} = -\frac{\delta a_{1i}}{\delta \mu} \tag{3.17}$$

$\frac{\delta a_{1i}}{\delta \mu_z}$ and $\frac{\delta b_{1i}}{\delta \mu_v}$ are the dihedral derivatives in the longitudinal and lateral axes respectively. For model simplification, they are of equal magnitude and have a destabilising effect.

The advancing blade produces more thrust during a heave movement resulting in a rotor hub moment. This effect can be described as:

$$\frac{\delta a_{1i}}{\delta \mu_z} = K_\mu \frac{16\mu_{ri}^2}{(1-\mu_{ri}^2/2)(8\,|\mu_{ri}| + a_r \sigma_r)} \tag{3.18}$$

By considering the rotor flapping motion as a contributing factor in the generation of rotor

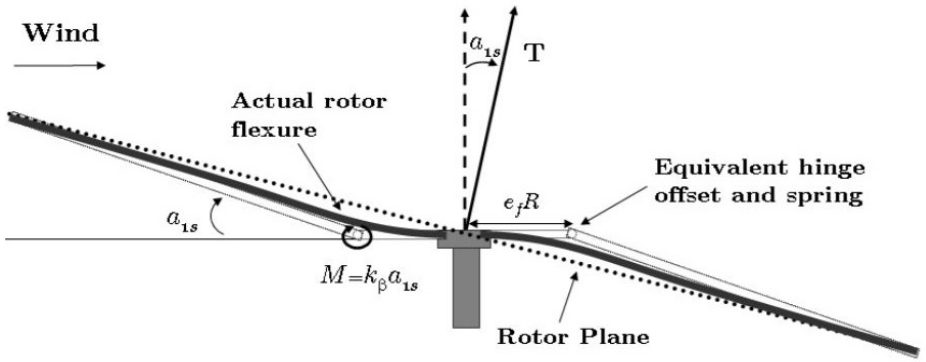


Figure 3.10: Rotor flapping approximation [89]

moments, aerodynamic moments generated by the quadcopter rotors can be represented by a linear model with a constant torsional stiffness value, k_β. This is shown in Figure 3.10 and defined as:

$$M_{k,lon} = K_\beta a_{1i} \tag{3.19}$$

$$L_{k,lat} = K_\beta b_{1i} \tag{3.20}$$

where $M_{k,lon}$ and $L_{k,lat}$ are the restoring moments due to the rotor flapping motion in the longitudinal and lateral axes respectively. Additional moments are generated through the tilting of the ith rotor due to the rotor flapping motion. Assuming the rotor flapping angles are small, a linear approximation of the total moments about the quadcopter centre of gravity can be defined (N is the number of rotors):

$$L_r = \sum_{i=1}^{N} (K_\beta + Th_r)\, b_{1i} \tag{3.21}$$

$$M_r = \sum_{i=1}^{N} (K_\beta + Th_r)\, a_{1i} \tag{3.22}$$

$$N_r = \sum_{i=1}^{N} Q_i \tag{3.23}$$

where the distance along z-axis between the quadcopter centre of gravity and the rotor head is defined as h_r. Similarly, the rotor forces can be described as:

$$X_r = -\sum_{i=1}^{N} T_i sin(a_{1i}) \tag{3.24}$$

$$Y_r = \sum_{i=1}^{N} T_i sin(b_{1i}) \tag{3.25}$$

$$Z_r = -\sum_{i=1}^{N} T_i cos(a_{1i}) cos(b_{1i}) \tag{3.26}$$

Given the above equations of motion, the additional dynamics of the incipient fault condition is defined through the model estimation of propeller angular speed Ω_i for the ith affected rotor as function of faulty motor angular speed. The system identification of the propeller slippage faulty condition is described in Section 3.3.5. Such an integrated model is also known as a grey-box model which is used for the development of a nonlinear system identification to predict faulty rotor dynamics in Chapter 4.

3.3.3 Flight dynamic model verification

Using the MATLAB software, a 6-DOF flight dynamic model was developed based on the quadcopter dynamics described in Section 3.3.2. This was used as a verification step to minimise the integration effort with the ArduPilot software suite. This model is shown in Figure 3.11.

Unlike the typical use of Simulink block objects to create a simulation model, the simulation code was directly written in the C++ language and compiled into a MATLAB executable (MEX) file. The advantage of this approach is that debugging on the compiled code can still be performed by introducing conditional breakpoints within an IDE and integration into the Ardupilot codebase, also written in C++, is achieved.

The verification process included the assessment of the trim and linearisation outputs of the simulation model for various flight conditions. Specific trim and linearisation routines had to be developed to ensure linear models were generated around steady-state trim conditions. The objective was to analyse the effect of the flapping dynamics on the behaviour of the quadrotor at various flight conditions. This is shown in Figure 3.12. As expected for the impulse output, the flapping angle increases in magnitude as forward speed increases. Moreover, the bandwidth of the response is independent of forward speed which enabled the optimisation of the system identification manoeuvres described in Section 3.3.6.

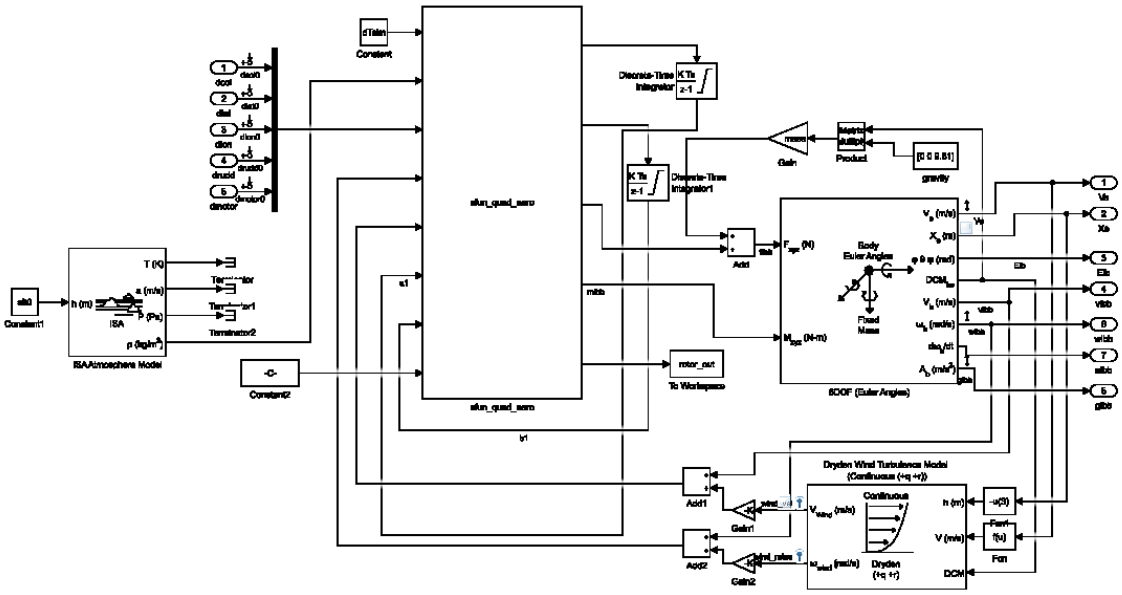


Figure 3.11: Quadrotor airframe model

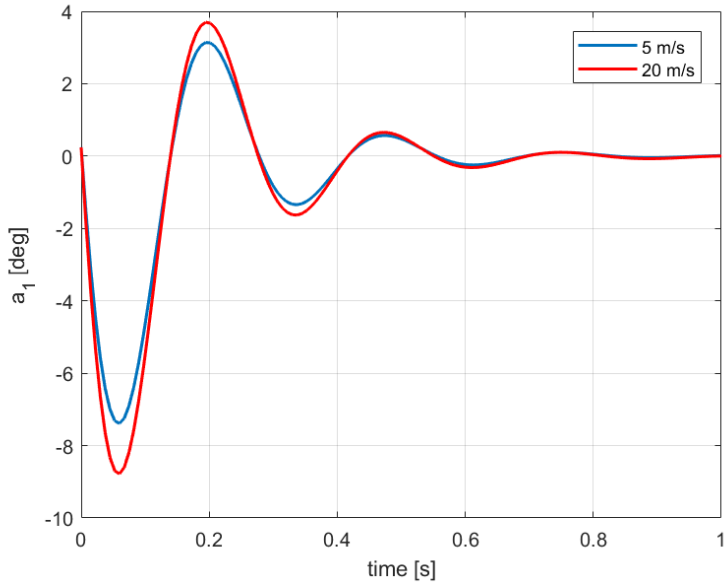


Figure 3.12: Model verification - flapping angles linear response.

The development of the nonlinear identification algorithms described in Chapter 4 was done using the generated data from this simulation model. Given that the quadcopter dynamics are unstable without closed-loop control, a PID controller was designed such that the quadcopter could be simulated. Algorithm robustness to noise was also analysed prior to their integration into a real-time environment. The results were presented in [3].

3.3.4 Embedding simulation into ArduPilot

The ArduPilot software makes use of a software-in-the-loop (Software-in-the-Loop (SITL)) to verify new algorithms or updates software routines by executing a 6-DOF model that emulates a particular platform such as: helicopter or airplane. This model assumes non-physical properties in order to minimise the compiling time.

This model was subsequently replaced with the developed flight dynamic model as described in Section 3.3.3 to ensure the system identification routine would capture the physical behaviour of the system. A telemetry interface for post-processing was modified and execution of test-bench was monitored and analysed using the MavProxy Ground station, shown in Figure 3.13.

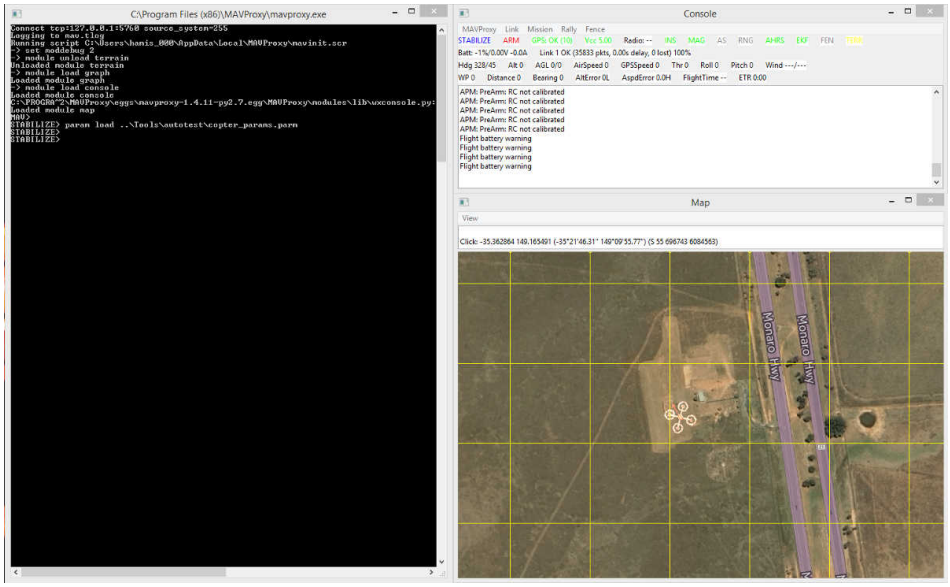


Figure 3.13: MavProxy ground station console.

3.3.5 Fault emulation implementation

The fault emulation routine as shown in Figure 3.8 consisted of integrating the fault filters, as identified in Section 3.4 as part of the Ardupilot software suite. In order to minimise the amount of computational overhead due to this functionality, the fault injector was modelled as second-order transfer function with an initialisation flag based on user inputs.

Secondly, given that further validation will be performed using the avionics platform as described in Section 3.2, the injection filters were applied on the motor output commands from the attitude PID controllers, as shown in Figure 3.14. This enabled the faulty emulation filters to execute at the same rate as the motor output routine ensuring the rest of the autopilot structure remains unchanged.

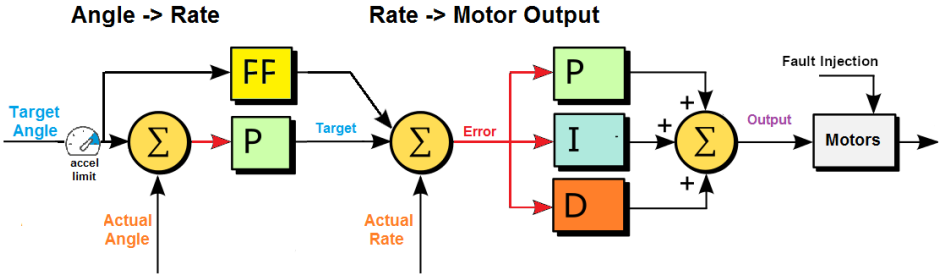


Figure 3.14: Attitude PID controller with fault filter injection.

3.3.6 Automated manoeuvres design

The design of input manoeuvres to generate data for real-time system identification is a major objective of optimising the information content during the dynamic motion of the test-bench platform. Using the system linear response analysis described in Section 3.3.3 and the nonlinear validation of the embedded simulation described in Section 3.3.4, the power spectral function of both doublet and 3211 manoeuvres was used to develop software routines to stimulate sequentially all four rotors. This is defined as [84, pp. 38]:

$$E(\omega) = 2\Delta t^2 \left[\frac{1-\cos(\Omega)}{\Omega^2}\right] \times \left[\sum_{i=1}^{M} V_i^2 + 2\sum_{j=1}^{M}\cos(j\Omega)\sum_{i=1}^{M} V_i V_{i+j}\right] \quad (3.27)$$

where M is the number of impulses with a time duration Δt and amplitude V and normalised frequency $\Omega = \omega\Delta t$. Figure 3.15 shows a superposition of the doublet manoeuvre

on the command for each rotor. Figure 3.16 shows the power spectral energy of an manoeuvre design step-size as a function of forward speed.

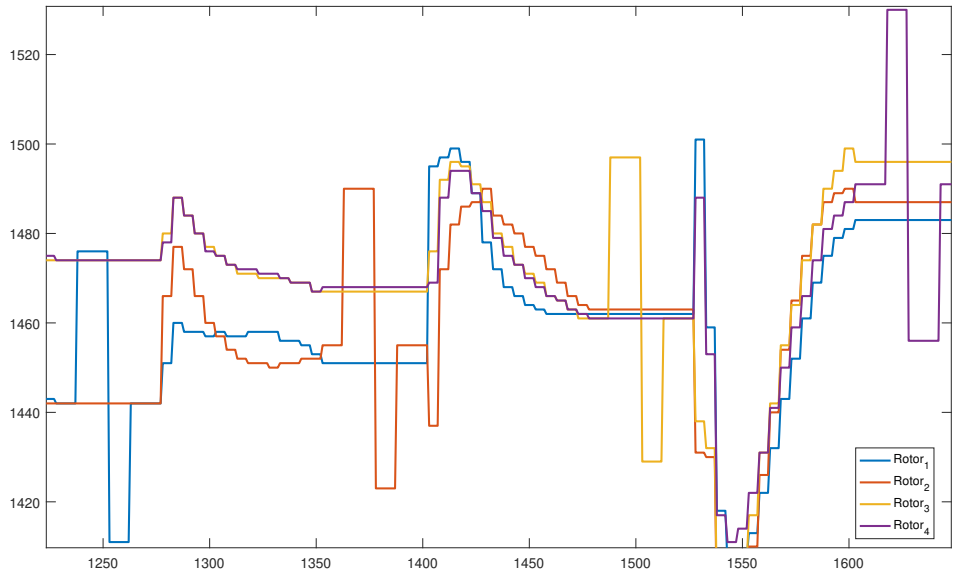


Figure 3.15: Illustration of superimposed identification manoeuvres on commanded rotor signals.

Once an optimal manoeuvre step-size was chosen (in this case 135 ms), the manoeuvres were then superimposed on each motor output in sequential order or for user-defined motor. Similar to the implementation of the fault emulation filters, the system identification manoeuvres were activated based on user-input and could be deactivated once the FDD process was complete.

3.3.7 Software-in-the-loop simulation

The complete simulation procedure, known as software-in-the-loop (SITL), included the compiling of the ArduPilot software suite such that it can execute on the kernel layer of an operating system ensuring real-time constraints are honoured. In addition, this enabled the testing of the full functionality of the ArduPilot software through logging the system dynamic behaviour generated from the flight dynamic model as described in Section 3.3.3.

The typical mission profile consisted of various waypoints to ensure the designed trajectory tested adequately the implemented algorithms. This is shown in Figure 3.17. Mission-critical data transmission between the SITL simulation and the Mission Planner was done using the UDP (User Datagram Protocol) link. The identification manoeuvres were triggered at a pre-

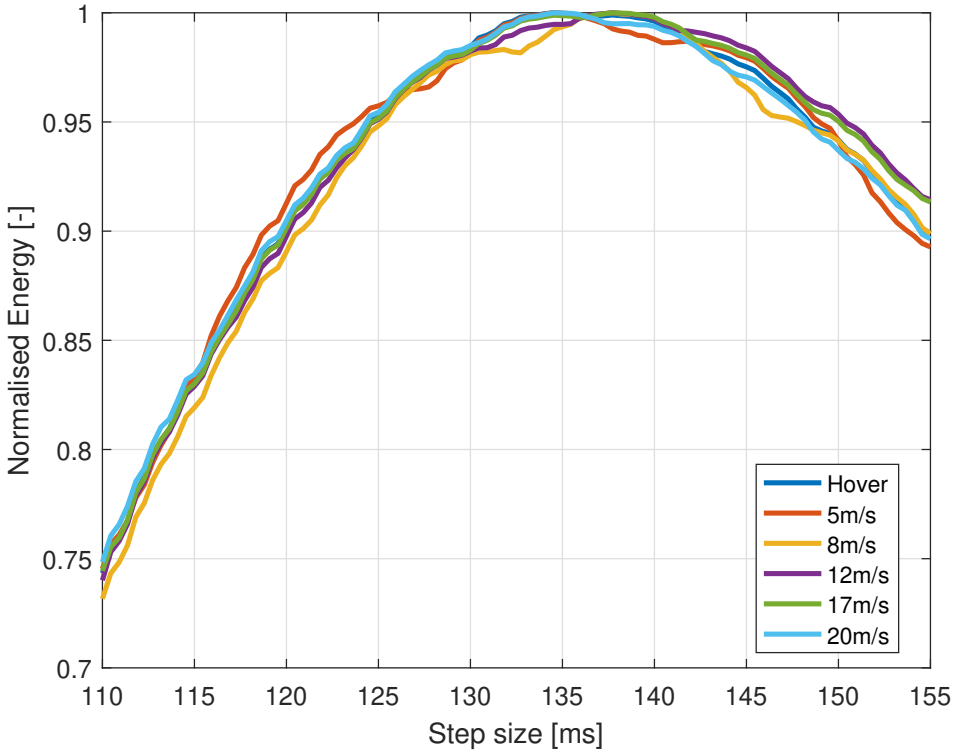


Figure 3.16: Normalised power spectral function as function of forward speed

defined forward speed while the quadcopter performed waypoint navigation.

Verification of the SITL simulation was done for both healthy and faulty conditions where the slippage condition for rotor #1 resulted in a thrust loss of 85%. This faulty condition was triggered when the quadcopter forward speed was higher than 3 m/s. This is shown in Figures 3.18 and 3.19. As objective of online application of the system identification manoeuvres was achieved through the complete execution of a programmed mission.

The performance of the PID flight controllers was also analysed through the tracking performance of the system angular rate tracking. This is shown in Figures 3.20 and 3.21 respectively. As expected, there is an increase in proportional (P) error after every manoeuvres with faulty rotor case. Furthermore, integrator (I) windup can be noticed especially during a turn around a waypoint resulting in an unstable behaviour.

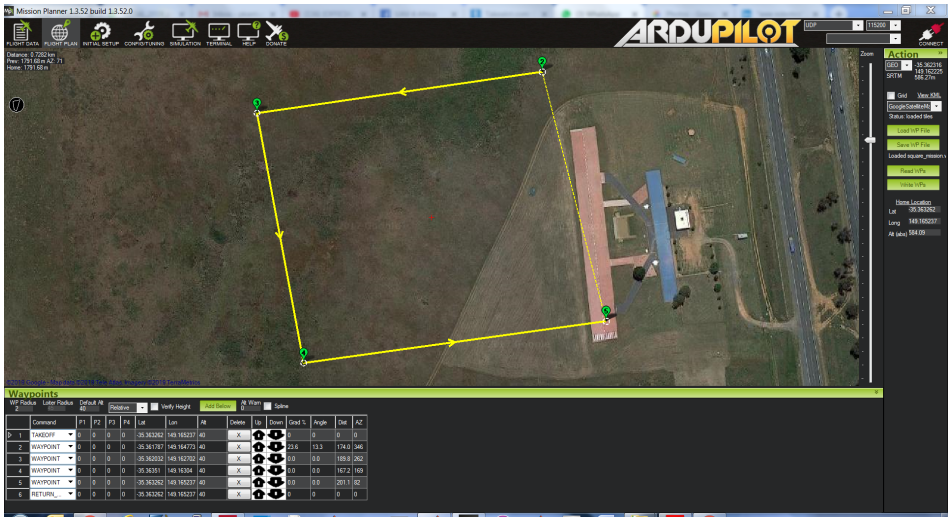


Figure 3.17: Programmed waypoints trajectory in mission planner.

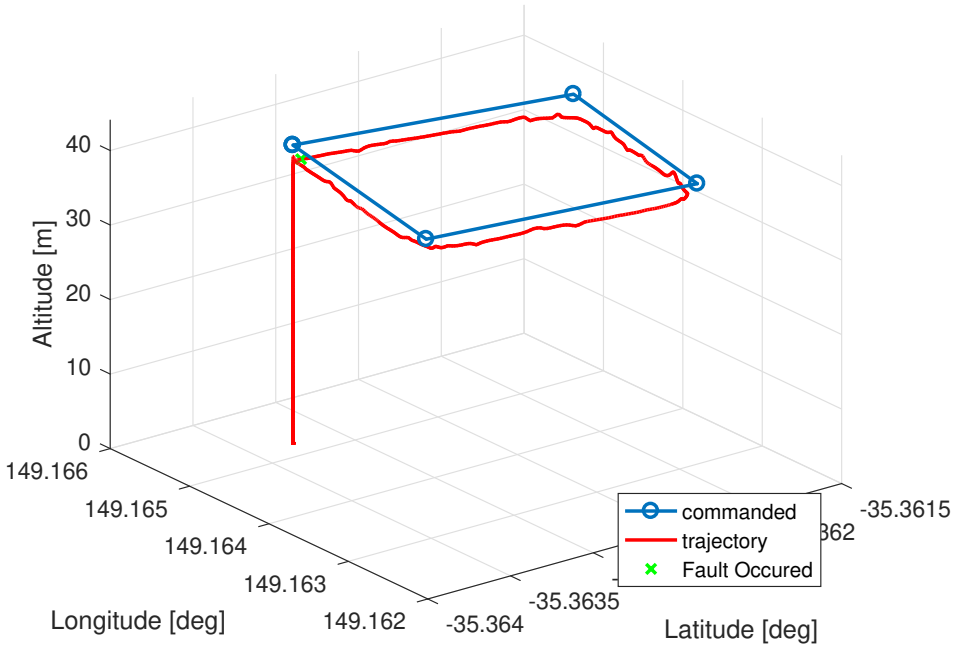


Figure 3.18: Mission flight path with no thrust loss. SITL simulation

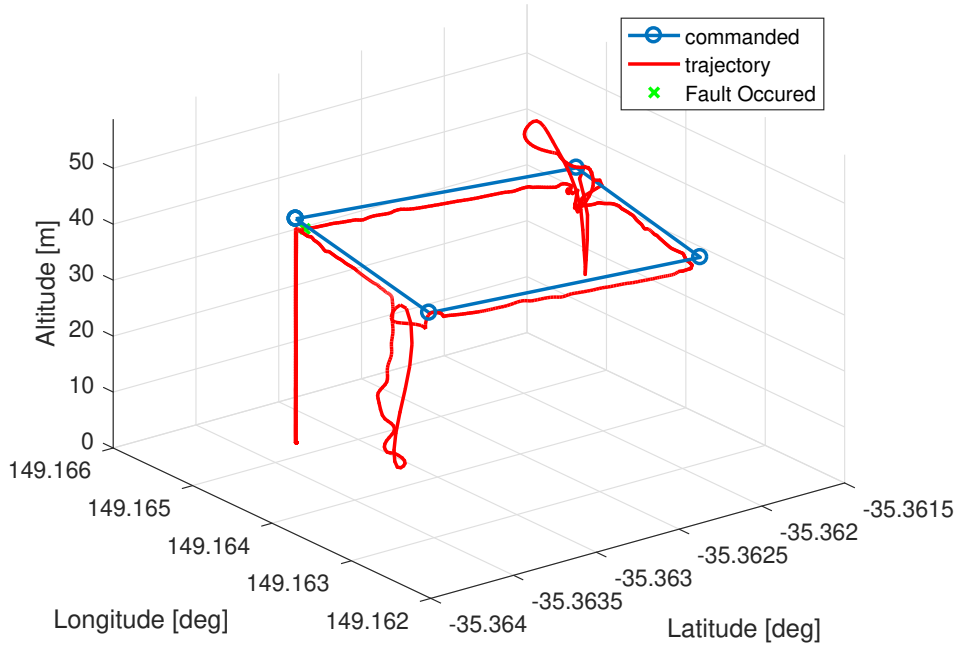


Figure 3.19: Mission flight path with 80% thrust loss on rotor #1. SITL simulation

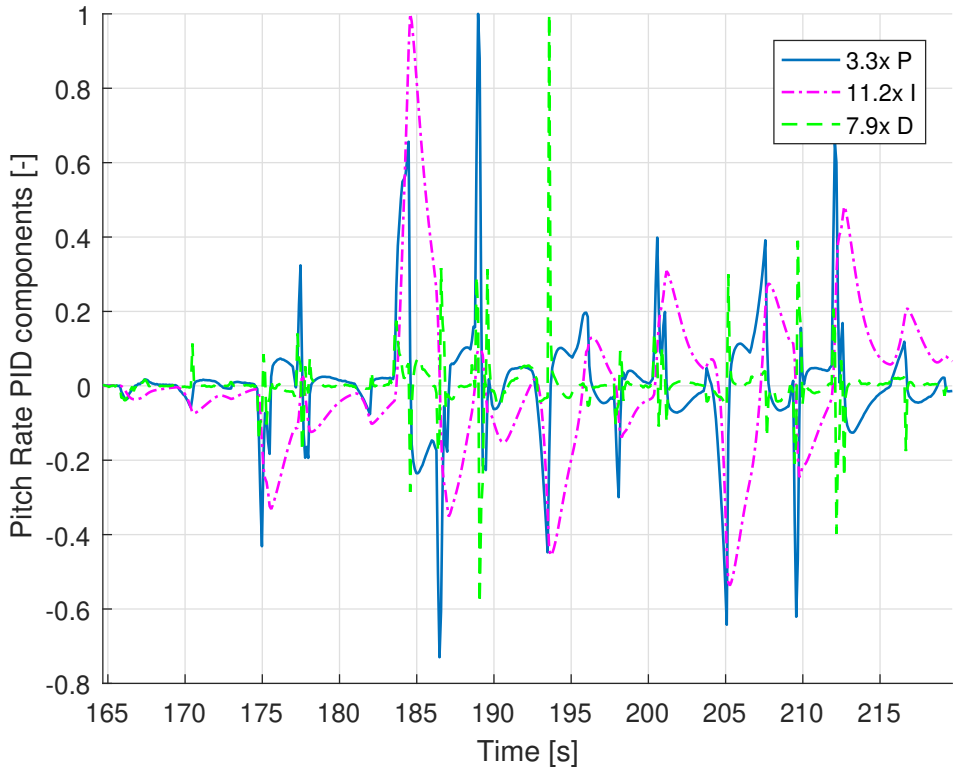


Figure 3.20: Pitch controller performance with healthy rotor during identification manoeuvres. SITL simulation

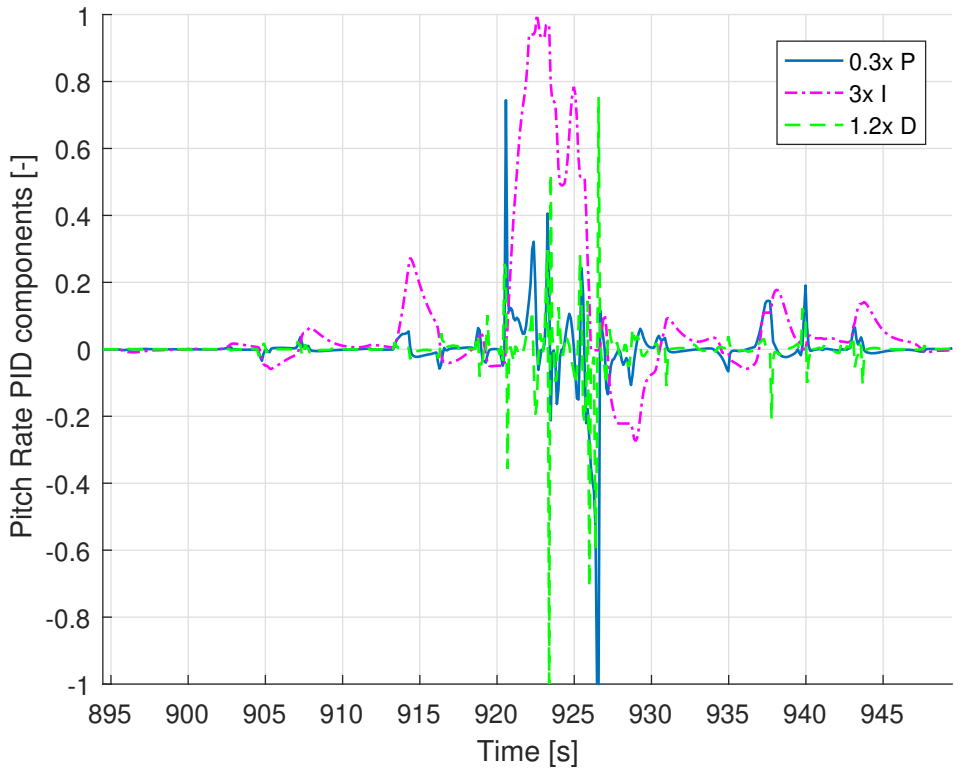


Figure 3.21: Pitch PID controller performance with faulty rotor during identification manoeuvres. SITL simulation

3.4 Identification of Rotor-Slip Incipient Fault Dynamics

3.4.1 Incipient fault model

The most important part of a quadcopter is the power electronics. Besides the power distribution board, the high-amperage battery and the electronic speed controller (Electronic Speed Controller (ESC)), the rotor system comprises of a propeller tightened to the shaft of a three-phase brushless DC (BLDC) motor. This is shown in Figure 3.22. Even though some rotors have a self-locking system, the application of dust, grease, uneven mating surface and wear-and-tear can decrease the friction joint effectiveness causing a faulty quadcopter behaviour. If such a fault has a dynamic component or its amplitude increases with time, this is known as an incipient fault condition.

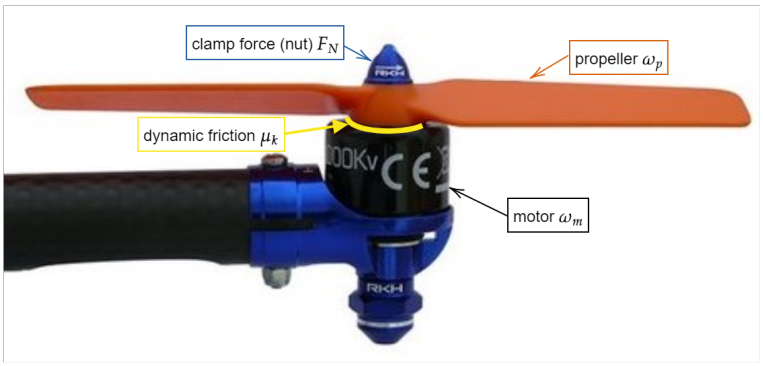


Figure 3.22: Configuration of a quadcopter powertrain with an incipient fault condition.

Given a motor torque on the ith rotor defined as T_{m_i}, a fault condition can become incipient if both conditions are true:

$$(\omega_{m_i} - \omega_{p_i}) > 0 \tag{3.28}$$

$$T_{m_i} \geq \mu_k(\xi_i) d_h (F_{N_i} - T_{p_i}) \tag{3.29}$$

where $\omega_{m_i}, \omega_{p_i}$ are the motor and propeller rotational speeds for the ith rotor respectively. d_h and F_{N_i} are the friction joint diameter and the friction joint clamping force respectively. The propeller thrust is defined as T_{p_i}. $\mu_k(\xi_i)$ and ξ_i are the parameters defining the dynamic friction model for a Stribeck friction coefficient. [90]. Without access to the motor electrical parameters, such an incipient dynamic model can be identified through experimental methods as described in [1].

3.4.2 Experimental setup

To the author's knowledge, the investigation of partial propeller failure through the modelling of its change in rotational speed has not been explored. This is particularly relevant given that numerous custom-made quadcopters have propellers assembled unto their motors through the tightening of a nut, relying solely on contact friction. Given the complex nature of the dynamics involved and the unknown model parameters correlation the motor RPM (revolutions per minute) and the propeller RPM, this thesis investigates an experimental approach in the modelling of propeller RPM dynamics by analysing the images of a high-speed camera.

This approach has been used to capture the deformation of a flexible rotor blade [91], and estimate the rotation angles of moving objects [92]. The processing of RPM dynamics is achieved by analysing each still frame of the video and determining the rotational position of the high-contrast blade of the propeller through computing the RGB (red-green-blue) index values. Although common methods for measuring RPM dynamics are available such as optical sensors [93]; this method has an advantage of capturing the dynamics of the propeller under various faulty conditions through a simple hardware implementation.

The main objective of the experiment was to capture the rotational speed of the propeller at normal (high-contact friction) and faulty (low-contact friction) conditions within the bandwidth of the dynamics in question. A GoPro ©camera was chosen because of its versatility and ease of application. A GoPro camera is a fixed focus, 11 mega-pixel, still frame (or 1080p high-definition video) camera measuring just 42 mm x 60 mm x 30 mm. The Hero 4 Silver version has a viewfinder or LCD screen, which allows the framing of the image to be done without post-processing the video data on a computer. This type of cameras can be used with various accessories for both outdoor and indoor application.

To enhance the quality of the video and keep camera in focus, each propeller blade was spray-painted with black and white colours respectively and a black background was chosen to add contrast. This enabled the imaging processing algorithm (described in Section 3.4.3) to accurately track the position of the white-painted blade. Additionally, the GoPro camera was set to capture the video at 240 fps (frames per second), at the sacrifice of the video resolution which was set at WVGA (Wide Video Graphics Array) with an image size of 800 pixels (width) by 480 pixels (height). The high frame rate enabled the measuring of rotational

speed up to approximately 18300 RPM (assuming a maximum capture of 45 deg increment given possible blade blurring effects).

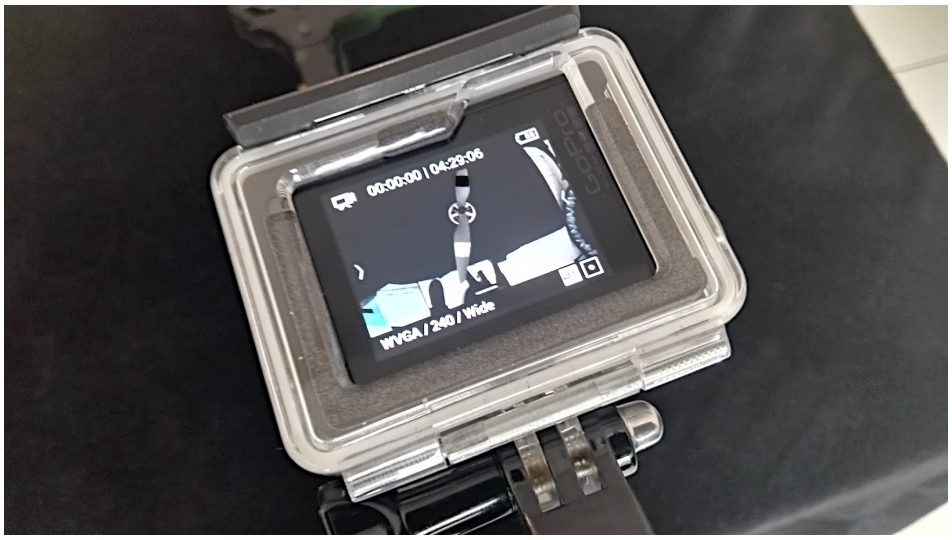


Figure 3.23: GoPro camera with WVGA setting and drone rotor setup

It was decided not to modify the Pixhawk autopilot code for this experiment but rather use a separate method to test the motor. An Arduino Uno was chosen and was programmed to provide PWM (Pulse Width Modulation) commands to the ESC (Electronic Speed Controller). This was done in order to maintain input repeatability between the various rotor test conditions. The ESC was then connected to the motor to produce rotational speed. A separate 4S Lithium-Polymer (Li-Po) battery was used to power the drone rotor while the Arduino was powered while connected to the laptop computer. The experiment layout is illustrated in Figure 3.24.

Using an iterative approach, four various PWM commands were generated which would represent the RPM range during a typical flying mission (hover and slow forward speed). The step changes between the commands were also considered in order to establish a detailed analysis of the rotor dynamics in both negative and positive step changes. Each step changes lasted two seconds to allow the transients of the dynamics to be captured by the imaging algorithm. The following steps (in microseconds) were used: [1240 1340 1270 1400 0].

A delay of 2000 milliseconds was used to create a step command effect. The tests were performed using an Arduino microcontroller. The difference in tests between a healthy rotor and a faulty rotor was through the loosening of the nut on the propeller, enough for a slip

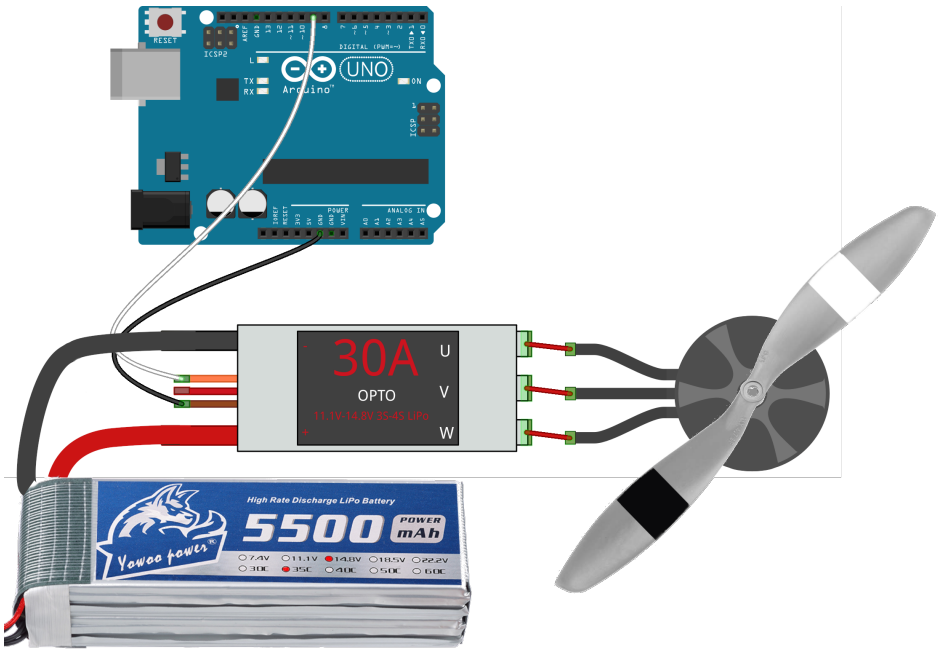


Figure 3.24: Rotor RPM experimental setup

to occur under small amount of torque. The GoPro camera was mounted on a stand and the recording was stored on a Micro-SD card and transferred to a laptop for further post-processing.

3.4.3 Image processing algorithm

An image processing algorithm was developed using MATLAB software. Other measurement methods were investigated such as: LED/laser pointer method, Laser tachometer, Photo resistor. Such methods are great effective methods to determine the steady state frequency. Given all the above alternative methods work with a fixed position of the sensor and the propeller used in this research only has two blades, the dynamic response to a step command will not be accurately tracked. So a method that captures the entire rotation of the propeller blades was required.

The chosen image processing method has a obvious drawback given the inherent noise generated in the video processing which post-processing could introduce artificial delays. This was deemed acceptable given the transient behaviour should be well captured given the higher signal-to-noise ratio during the transient response to a step command.

The `VideoReader` function was used as it allows the processing of video frames within a specific time range. The centre of rotation was chosen using a cursor over the image. This enabled the generation of a circular array of data points that would determine the location of the blade by analysing the RGB values along the circumference of drawn circle. This circle can be seen in Figures 3.25 and 3.26 at different times of a video stream.

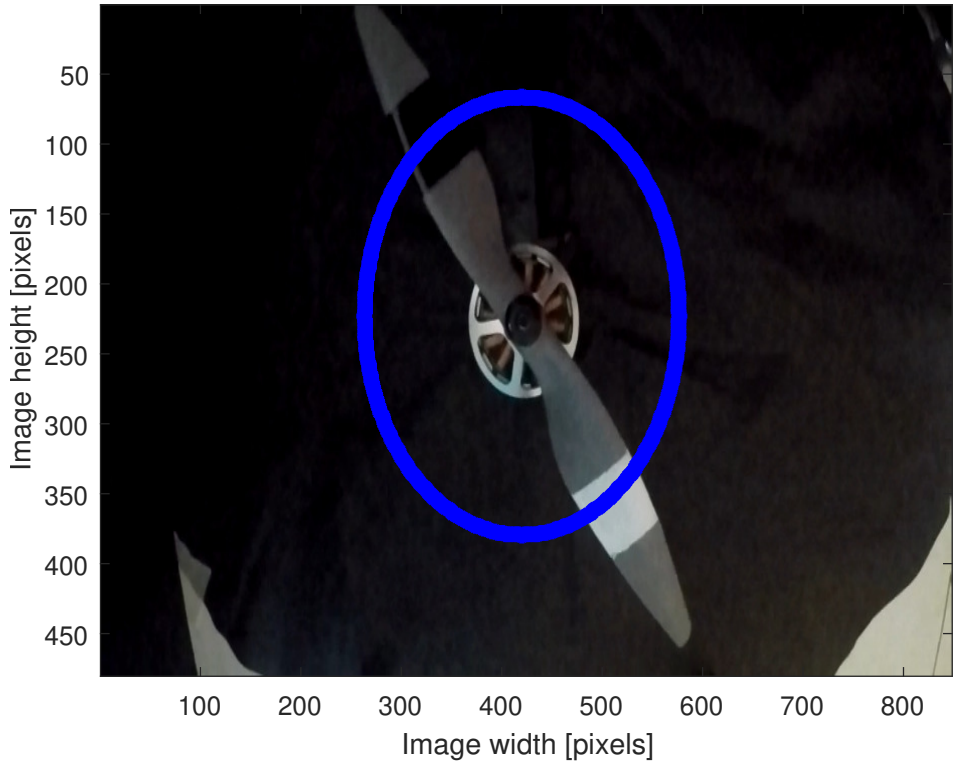


Figure 3.25: Image capture at 0.5s (stationary)

A flowchart of the logic on how the imaging algorithm processes the raw image data is shown in Figure 3.27. An adaptive RGB threshold has been implemented to prevent the computation of the blade angular position from a discontinuous dataset, which is evidence of RGB values from glares or reflected light on the paint. The RGB threshold is then adjusted accordingly and the analysis of the RGB values along the circumference of the drawn circle is repeated. An illustration of the RGB threshold controller eliminating glares is shown in Figure 3.28. There is also a check on whether some RGB values have crossed the zero angle line and show false discontinuity due to angle wrapping (0 to 360 degrees). The angular rates, and subsequently the RPM values, are computed based on time difference between each frame (time interval between frames is 1/240th of a second).

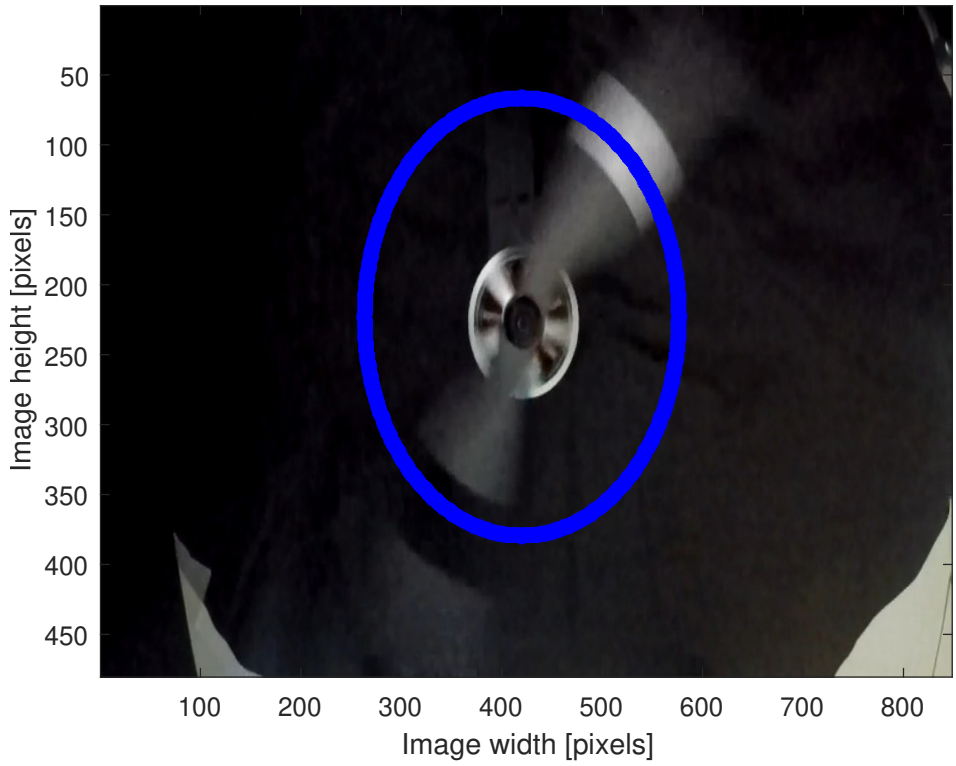


Figure 3.26: Image capture at 3.5s

3.4.4 Kalman filter application

A Kalman filter, also known as a linear quadratic estimator, was adopted for signal conditioning prior to model estimation [94]. A high measurement noise covariance was chosen, given the process noise (generation of the PWM commands through the Arduino) is known to be pretty small in comparison. The measurement noise standard deviation σ_v was set at 0.015 and the process noise standard deviation σ_w was set at 0.15 such that the discrete covariance matrices can be defined:

$$Q(k) = \sigma_w^2 \tag{3.30}$$

$$R(k) = \sigma_v^2 \tag{3.31}$$

where Q and R are the process and measurement covariances respectively. The discrete Kalman filter can be easily defined as two-stage state estimator with time update and measurement update. Given one state, the Kalman filter reduces to near-zero phase shift low-pass filter.

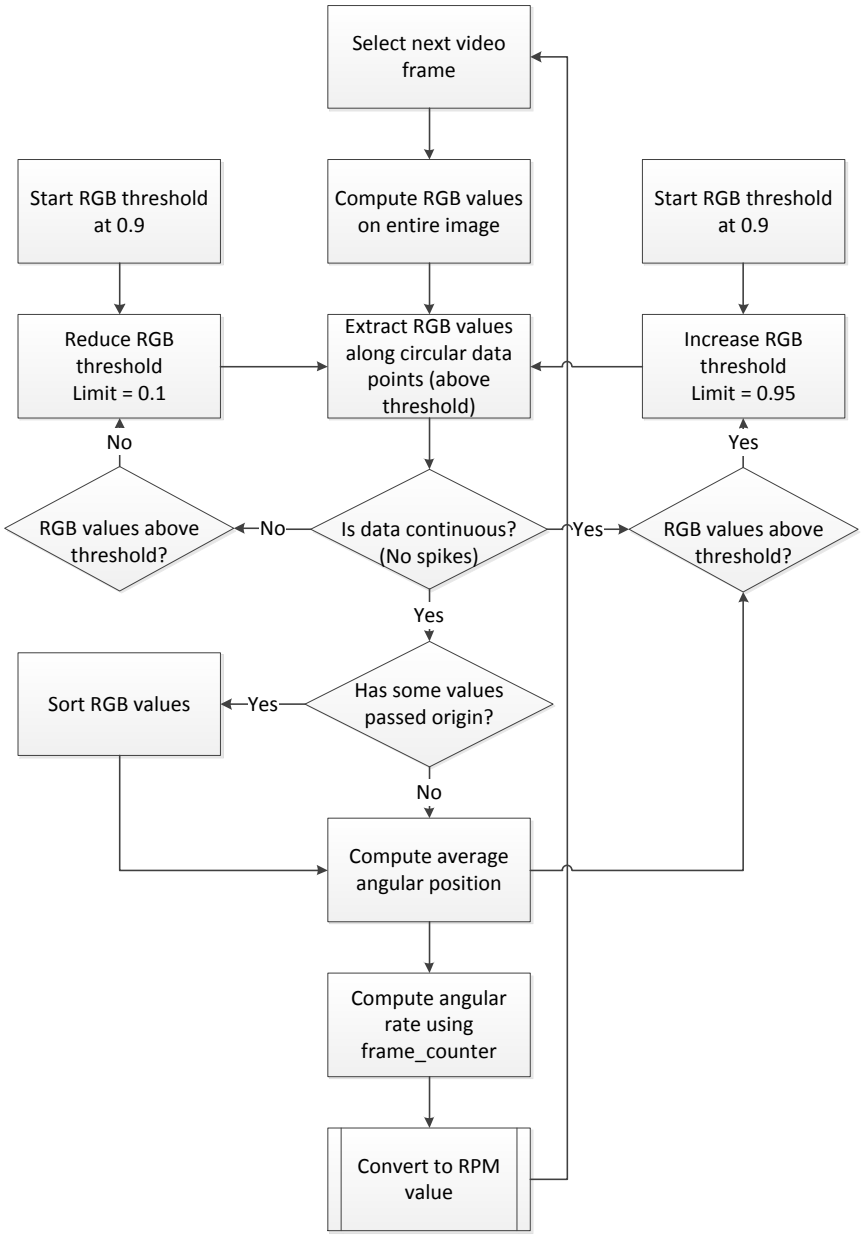


Figure 3.27: Imaging algorithm flowchart

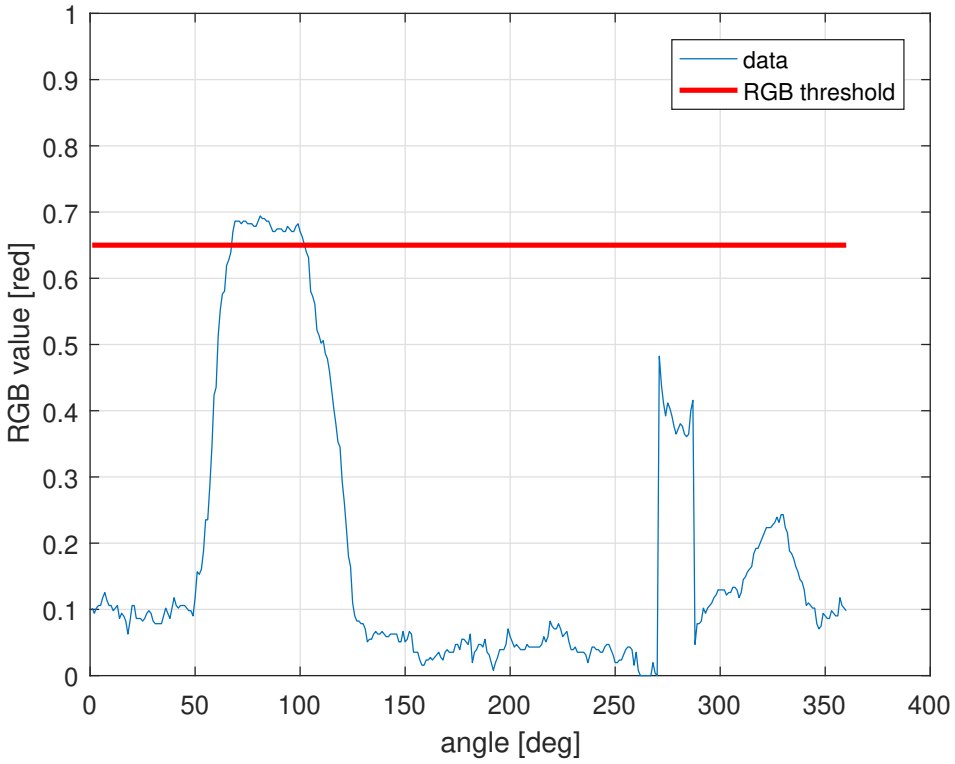


Figure 3.28: RGB threshold monitor for angular estimation and glare rejection

Time Update

Given the unity state transition $F(k)$ and observation $H(k)$ matrices, the predicted state and covariance can be computed as follows:

$$x(k)^- = F(k)x(k-1)^+ \tag{3.32}$$

$$P(k)^- = F(k)P(k-1)^+F(k)^T + Q(k) \tag{3.33}$$

where the $x(k)^-$ and $P(k)^-$ are the state and covariance matrices respectively prior to measurement update.

Measurement Update

The computation of the Kalman gain and update of the state and covariance can then be performed:

$$K(k) = P(k)^-H(k)^T\left(H(k)P(k)^-H(k)^T + R(k)\right)^{-1} \tag{3.34}$$

$$x(k)^+ = x(k)^- + K(k)(z(k) - H(k)x(k)^-) \tag{3.35}$$

$$P(k)^+ = (1 - K(k)H(k))P(k)^- \tag{3.36}$$

where $z(k)$ is the raw input (in this problem the RPM values). $x(k)^+$ and $P(k)^+$ are the state and covariance matrices respectively corrected with a measurement update.

3.4.5 Model identification and estimation

Given that the RPM dynamics for both a healthy rotor and a faulty rotor can be measured repeatedly and accurately through the use of the imaging processing algorithm, the objective is to establish a fault emulation process that could be invoked in software. This requires the identification and estimation of a RPM dynamic model through system identification. This estimated model would then enable a relationship between the healthy and faulty rotor RPM models to be determined. This result can then act as pre-filter of the PWM commands of a healthy rotor to $mimic$ a faulty rotor behaviour.

Identification of continuous time models

Given a sampled dataset $(u_k(t), y_k(t))$ which is assumed to be from a continuous function [95] the problem of model estimation is to derive $G(s)$ such that:

$$Y(s) = G(s)U(s) \tag{3.37}$$

$$G(s) = \frac{B(s)}{A(s)} \tag{3.38}$$

$$B(s) = b_0 s^m + b_1 s^{m-1} + ... + b_m \tag{3.39}$$

$$A(s) = s^n + a_1 s^{n-1} + ... + a_n \tag{3.40}$$

where n, m are the number of poles and zeros and a, b are the estimated coefficients. If $Y(s)$ is the Laplace transform of $y(t)$ and provided these derivatives were accessible, a linear regression method extended from a discrete time parameter estimation could apply [96]. Given the time-derivative problem, a pre-filtering process is done and defined as:

$$z_k(t) = \mathcal{L}^{-1}[L(s)]y_k(t) \tag{3.41}$$

$$w_k(t) = \mathcal{L}^{-1}[L(s)]u_k(t) \tag{3.42}$$

where $\mathcal{L}^{-1}[L(s)]$ is the inverse Laplace transform of the applied filter $L(s)$. The choice of filter depends on the inter-sample behaviour of the input. Common ideas are:

- The SVF (State Variable Filter)

$$L(s) = \left(\frac{\lambda}{s+\lambda}\right)^n \tag{3.43}$$

- GPMF (Generalised Poisson Moment Function)

$$L(s) = \left(\frac{\lambda}{s+\lambda}\right)^{n+1} \tag{3.44}$$

- Refined choice of filter

$$L(s) = \frac{1}{A(s)} \tag{3.45}$$

where the value of λ reflects the dynamics of the system and n is the system order. The last option is an iterative approach initialised with guessed estimates. Given the quantities of $z_k(t)$ and $w_k(t)$ are defined at each sampling interval $t = t_j$, the parameters $A(s)$ and $B(s)$, bundled here as θ, can be easily computed by a least-squares method:

$$\hat{\theta} = \left(\sum_j \phi(t_j)\phi^T(t_j)\right)^{-1} \sum_j \phi(t_j) z_n(t_j) \tag{3.46}$$

where ϕ is a built-up notation from z_k and w_k [95]. In the event of a white noise being present, the removal of the bias is achieved through using the Instrumental Variable (IV) method. The noise effected outputs are replaced by instruments $\hat{z_k}$ to form an instrument vector $\eta(t_j)$ of the same form as $\phi(t_j)$ resulting in a new estimate [95]:

$$\hat{\theta} = \left(\sum_j \eta(t_j)\phi^T(t_j)\right)^{-1} \sum_j \eta(t_j) z_n(t_j) \tag{3.47}$$

The MATLAB System Identification toolbox makes use of the theory above with specific focus on the IV method. This is implemented through the function `tfest`. This function applies a pre-filter that is the denominator of the current model, initialised with SVF. The pre-filter is iterated using various Gauss-Newton least-squares methods up to default maximum of 20 iterations or tolerance of 1e-3.

Estimation of a Fault Emulation Model

Fault emulation through software is an inexpensive method of testing system behaviour within a safe and repeatable environment. However, the main issue has been to assure the

modelling of the faults are representative of actual faults (mainly mechanical or hardware related). Given that continuous time models of the RPM dynamics can be estimated, a faulty model filter on input data (PWM commands) can be deduced. Given the estimated models $G(s)$ and $H(s)$ defined as:

$$y(s) = G(s)u(s) \tag{3.48}$$

$$y_f(s) = H(s)u(s) \tag{3.49}$$

where $y_f(s)$ is the rotor faulty response to the same RPM commands $u(s)$. A fault injection filter $G_f(s)$ can be formulated such as:

$$y_f(s) = G(s)G_f(s)u(s) \tag{3.50}$$

$$G_f(s) = G(s)^{-1}H(s) \tag{3.51}$$

The inversion of the $G(s)$ implies that the estimation process cannot result in an ill-conditioned matrix. A large condition number implies that the inversion of a matrix will lead to a singularity. For the objective of implementing $G_f(s)$ for real-time execution, a model reduction process was followed with the maximum number of poles being one (equivalent to a first-order filter) using the `balred` function. This function decomposes $G(s)$ into two parts:

$$G(s) = G_s(s) + G_u(s) \tag{3.52}$$

where $G_s(s)$ and $G_u(s)$ are the stable and the unstable parts respectively. $G_s(s)$ is then reduced by applying the balanced truncation method, which computes the Hankel singular values σ_j and ensures the absolute error $||G_s(s) - G_r(s)||_\infty$ is within the bounds $2\sum_{j=r+1}^{n}\sigma_j$. $G_r(s)$ is the reduced matrix, r is the model order and n is the number of states in $G_s(s)$.

3.4.6 Results

The recording of the healthy and faulty rotor tests was done in a well-lit room away from windows to prevent glares on the rotor blades. The synchronisation of the video recording and the start of the PWM commands was done through trial-and-error given the GoPro camera does not have a timer mechanism to start recording.

The results of the imaging algorithm, described in Section 3.4.3, and the subsequent filtered data is shown in Figures 3.29 (healthy rotor) and 3.30 (faulty rotor). The Kalman filter was

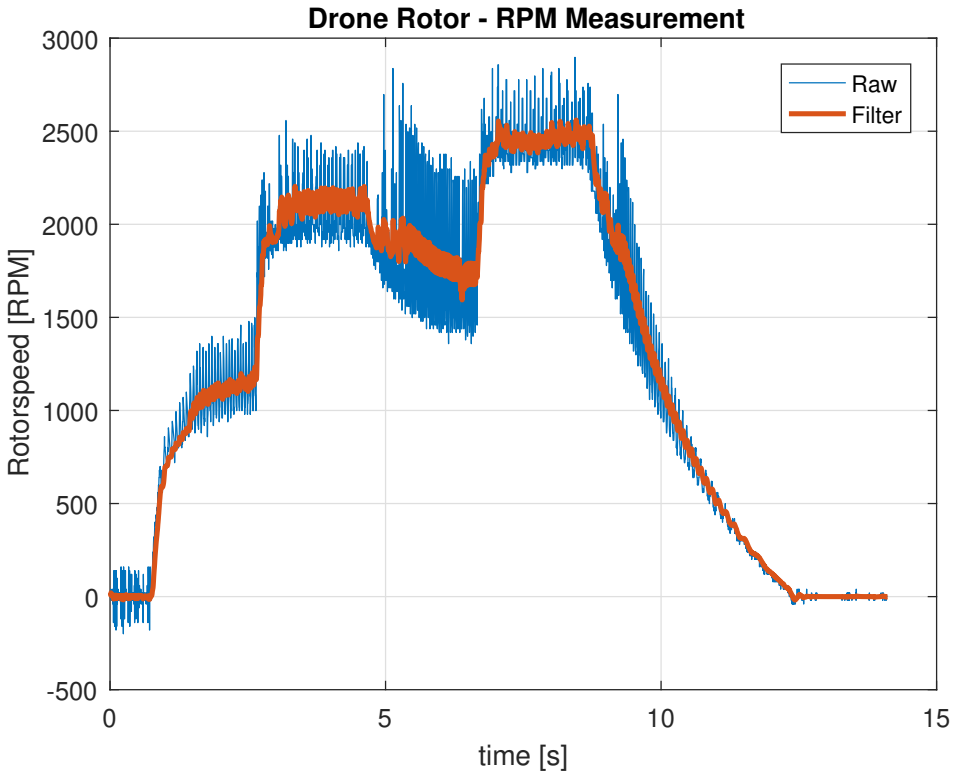


Figure 3.29: RPM dynamics - healthy rotor

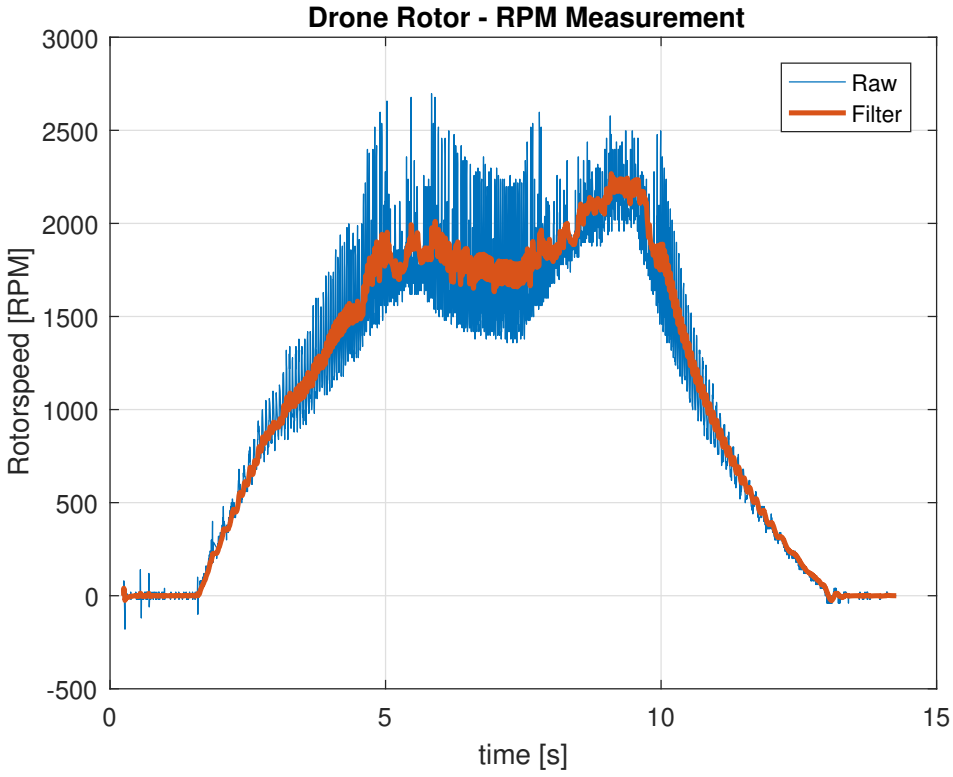


Figure 3.30: RPM dynamics - faulty rotor

tuned, based on analysis of the raw data, with a process noise standard deviation σ_w of 0.015 and measurement noise standard deviation of σ_v of 0.15. The complete degradation of the tracking performance is evident in the faulty rotor test which was accompanied during the test by rattling noise of the propeller slipping against the motor shaft. For both faulty rotor and healthy rotor tests, the step command to zero results in a large settling time. This is due to the fact that the brushless motor magnetic field effectively collapsed effectively reduced the braking system based on the back-emf, resulting in the propeller angular momentum being reduced by only aerodynamic drag and mechanical friction of the rotor. Modern ESCs are able to provide this functionality provided some voltage is still applied, but this was not used in this research.

The response of the faulty rotor didn't reach a steady-state condition. The missing data could have introduced some estimation errors. Unfortunately, this short-coming could not have been alleviated at the time of conducting this experiment. The given the modelling choice for the fault injection filters $G_f(s)$, the impact on FTCS framework is minimal. Table 3.3 shows the resulting model estimates for both test cases. The validation of the model estimates fit accuracy for both test cases is shown in Figures 3.33 and 3.32. Based on the results, it can be seen that the zero location of the faulty rotor estimated model given an initial step change of 1240 microseconds is positive resulting in non-minimum phase behaviour of the model, which when implemented in software will be just a constant delay on the input signal.

This result ensure that the estimation models give an adequate representation of changing rotor dynamics between a healthy and a faulty rotor. Figure 3.31 shows the bandwidth across each step change between the test cases. The mapping of the step changes to the PWM command is shown in Table 3.2. It is quite evident that the higher the PWM command, the

Table 3.2: Mapping of the PWM commands to step changes

Step No.	Step size	PWM command [microseconds]
#1	1240	1240
#2	100	1340
#3	-70	1270
#4	130	1400
#5	1400	0

more the faulty rotor struggles to track the command. This is seen in step No. #4 where the

increased noise on the propeller was due to the slippage caused by an increase in opposing drag overcoming the low-contact friction. This results in the faulty model having a slow pole (small bandwidth).

Table 3.3: Transfer function estimated models

PWM command	Healthy Rotor	Faulty Rotor
1240	$\frac{5.76s + 32.5}{s^2 + 11.8s + 35.7}$	$\frac{-1.79s + 6.25}{s^2 + 8.43s + 3.08}$
1340	$\frac{83.9s + 871}{s^2 + 36.4s + 119}$	$\frac{6.2}{s + 0.365}$
1270	$\frac{2.06}{s + 1.45e - 06}$	$\frac{0.998}{s + 4.12e - 06}$
1400	$\frac{58.1}{s + 12.5}$	$\frac{1.35}{s + 1.96e - 09}$
0	$\frac{0.58s + 48.9}{s^2 + 51.7s + 21.6}$	$\frac{0.00676s + 3.71}{s^2 + 4.38s + 1.99}$

The above estimation method has revealed that rotor dynamics after a step change require a certain amount of settling time to estimate the correct RPM value. Moreover, the magnitude and direction of the step change plays a crucial role in establishing an accurate relationship between the commanded input and the measured output. This applies for both healthy and faulty rotors. This can be seen in the Figure 3.31 whereby a step change smaller than 100 microseconds shows an estimation model with a large settling time. For the computation of the fault injection filters, such data was discarded.

Table 3.4 shows the result of computing the fault injection filters $G_f(s)$. The fault emulation PWM commands $u_f(t)$ is compared with the original command and it is shown in Figure 3.34. It can be seen that as the rotor speed increases beyond 1340 RPM, the filter pole sharply decreases to zero. This relationship can be modelled to study various levels of contact friction for the purpose of analysing the rotor dynamic behaviour.

The above method has proven to have both advantages and disadvantages. Extracting time series data from an imaging processing algorithm produces quite noisy data which needs to be processed and in itself could introduce artificial dynamics. However, the above results does show that dynamic behaviour of slippage condition has been captured through both small and large step commands ensuring that the various nonlinear effects are captured.

Comparing the above method with a light sensing approach (such as laser) could have resulted in less noisy data (especially combined with an oscilloscope), but the same parameters affecting the estimation of fault injection filters would have been identified.

Table 3.4: Fault injection filter estimated models

PWM step	Model	pole
1240	$\frac{-0.23s + 0.86}{s + 0.38}$	-0.384
1340	$\frac{0.16s + 0.85}{s + 0.37}$	-0.367
1270	$\frac{0.48s + 7.04e - 7}{s + 4.12e - 06}$	-4.12e-06
1400	$\frac{0.02s + 0.29}{s + 1.96e - 09}$	-1.96e-09
0	$\frac{-0.10s + 7.46}{s + 7.46}$	-7.46

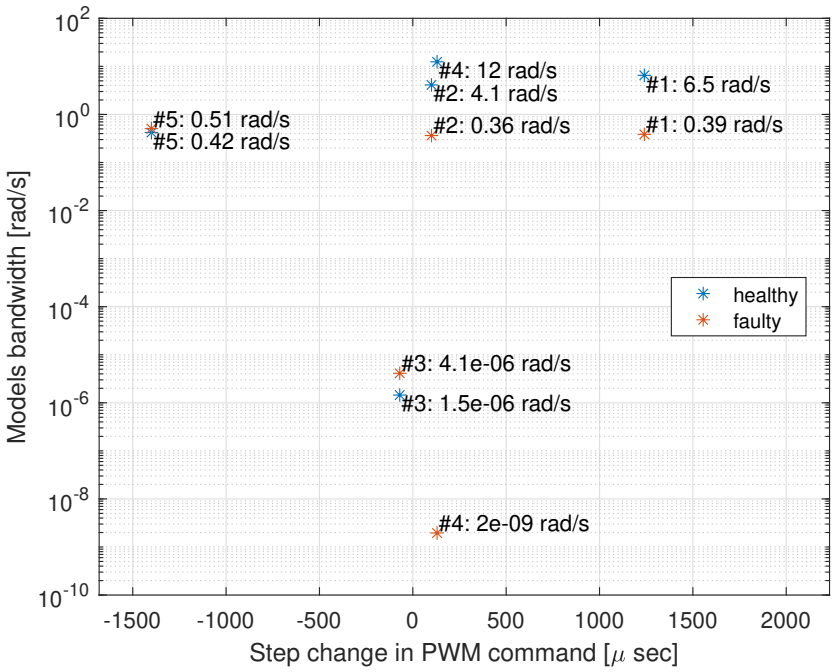


Figure 3.31: Estimated transfer functions bandwidth comparison

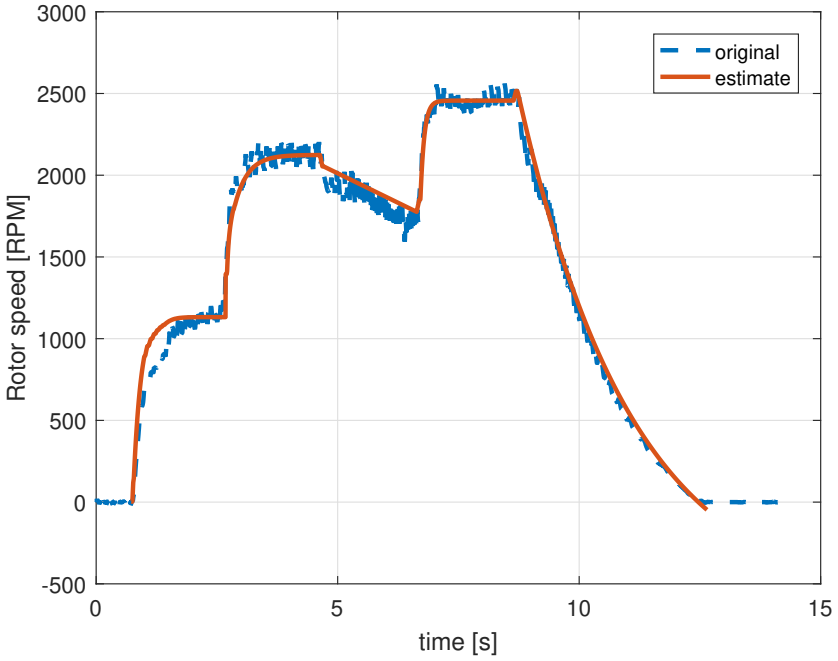


Figure 3.32: Validation of the estimation models fit accuracy - healthy rotor

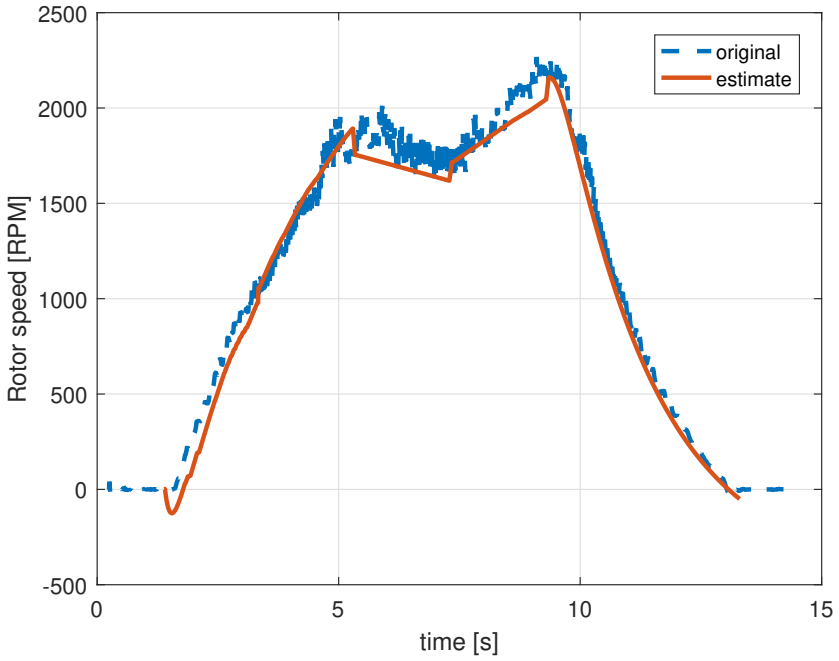


Figure 3.33: Validation of the estimation models fit accuracy - faulty rotor

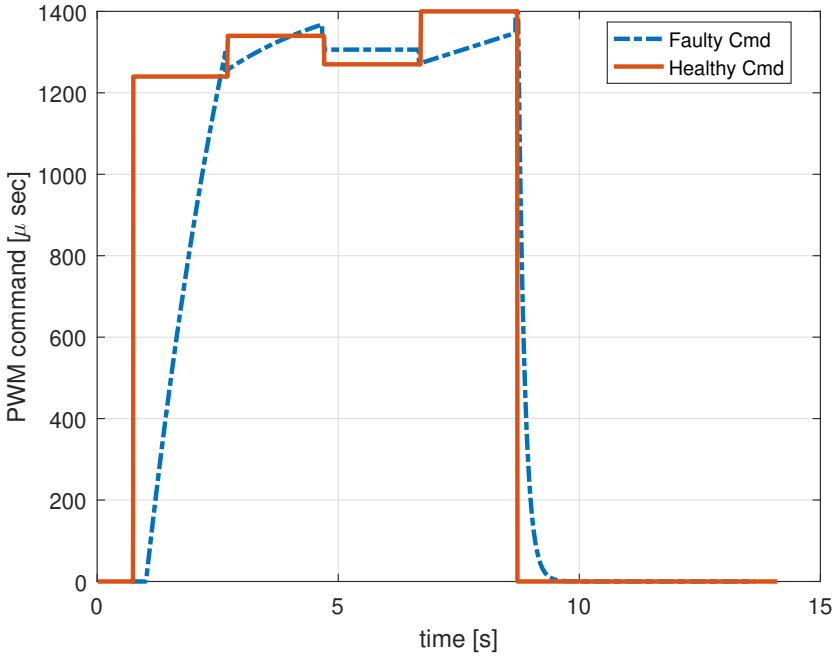


Figure 3.34: Comparison of PWM input commands that would result in the rotor speed response shown in 3.32 for healthy motor and 3.33 for faulty motor

Chapter 4

Nonlinear Identification of Unmanned Rotorcraft Systems

4.1 Introduction

In Chapter 3, the system identification problem was introduced through the description of the system-under-test (defined as the test-bench platform) and the experimental approach that would achieve repeatable results without compromising safety and system integrity. With reference to Figure 3.1, the part of nonlinear (system) identification discussed in this chapter focuses on the *Estimation* and *Evidence* stages applicable to nonlinear systems. A detailed description of model structures and their associated estimation algorithms are given while taking into consideration the nonlinear dynamics of a faulty unmanned quadcopter system.

Given the difficulty of achieving accurate and practical models which can be used in controller synthesis applications, a direct approach in system identification has been used to represent unknown system dynamics [97]. This makes use of a nonlinear model structure which enables the mapping of observed data (input-output) to a cost function minimisation regressor matrix to be achieved [98, 80]. Within closed-loop systems, such as rotorcraft in general, the direct approach applies a prediction method by identifying the open-loop system using the measurements of the inputs u and the outputs y. This method is well-suited for closed-loop systems as it requires no knowledge of the state of the feedback signal and

prediction consistency and accuracy is obtained only through the choice of model structure [99].

The development of a nonlinear system identification approach is based on the generation of input-output data from a grey-box simulation model which incorporates estimation models of a rotor with an incipient fault condition, described in 3.3.2 into a quadcopter equations of motion. Such an approach to modelling is a well established discipline, whereby the dynamic behaviour is described based on the physical model and the parameters in that model structure are estimated (in part or in whole) with input-output measurements [100]. Grey-box models use a-priori knowledge of the system dynamics and estimate the unknown free parameters through system identification techniques. Such an approach has been used to estimate quadcopter motor dynamics and propeller aerodynamic coefficients [101], estimating the longitudinal/lateral coefficients of small-scale fixed-wing UAV [102] and the estimation of a quadcopter heave and rotational aerodynamic coefficients using closed-loop identification techniques [103].

The two primary objectives in nonlinear system identification is to choose a model structure that is able to capture the underlying nonlinearity and predict the unseen data accurately [104]. The second objective is particularly difficult in the case where the underlying dynamics are time-varying based on the type of inputs used to excite the system. This is the case as seen in Section 3.4 whereby incipient fault conditions, excited by the magnitude of the actuation input, have a direct impact on the attitude dynamics of the quadcopter. This extra level of complexity in the identification problem puts more emphasis on the generalisation performance of the prediction model and the computational complexity of its training methods [105].

4.1.1 Nonlinear system identification

A control-affine nonlinear system can be expressed as:

$$\dot{x} = f(x) + g(x)u \tag{4.1}$$

where $u \in \Re^m$ and $x \in \Re^n$ are the input and state vector respectively. In real-world applications, functions $f(x)$ and $g(x)$ have unknown nonlinearities which can be parametrised through their respective approximations $\hat{f}(x, \theta_f)$ and $\hat{g}(x, \theta_g)$ and based on adjustable weights

$\theta_f \in R^{n_f}$, $\theta_g \in R^{n_g}$. The complexity level of the parametrised model can be controlled through the number of weight factors n_f, n_g.

Expressing Equation 4.1 as an identification problem is as follows:

$$\dot{x} = \hat{f}(x, \theta_f^*) + \hat{g}(x, \theta_g^*)u + \nu(t) \tag{4.2}$$

where θ_f^* and θ_g^* denote the optimal weights values in the approximations of $f(x)$ and $g(x)$ respectively such that

$$\theta_f^* := arg_{\theta_f} min \left[f(x) - \hat{f}(x, \theta_f) \right] \tag{4.3}$$

$$\theta_g^* := arg_{\theta_g} min \left[g(x) - \hat{g}(x, \theta_g) \right] \tag{4.4}$$

and $\nu(t)$ is the *modelling error* defined as:

$$\nu(t) = \left[f(x(t)) - \hat{f}(x, \theta_f^*) \right] + \left[g(x(t)) - \hat{g}(x, \theta_g^*) \right] \tag{4.5}$$

The identification problem using classical methods such as spectral analysis and prediction error methods are well suited for open-loop systems resulting in good estimation models which have captured the underlying model dynamics given the observed input-output data [99]. However for unstable open-loop systems such as a quadcopter, the identification of the rotor dynamics can only be achieved with the flight controller in the loop called *closed-loop identification*. Such a method is well-suited provided (1) the observed data is rich in information, (2) persistent excitation of the input signals contains the true system dynamics within the chosen model structure [80, p. 435]. Both factors can be achieved through the optimal design of identification manoeuvres as described in Section 3.3.6.

4.1.2 Nonlinear black-box system identification

A nonlinear model structure can be applied either by using linear-in-the-parameters approach (polynomial structure) which can be solved analytically or nonlinear-in-the-parameters approach (neural network structure) which involves the mapping from the observed data (input-output) to a cost function minimisation regressor matrix [80]. Given that most industrial system identification problems involves dynamical systems with unknown nonlinearities, neural network-based model architectures are often used to describe complex input-output mappings when no physical insight about the system dynamics is available or has

been previously quantified [98, 106]. This is known as nonlinear black-box system identification. Such an approach has been used to identify quadcopter dynamics using radial basis function neural networks (RBFNN) which were trained with a minimal resource allocating network (MRAN) algorithm [82].

A typical RBFNN is a model structure which consists of three layers: an input layer, a hidden node layer and an output layer which are connected through a network of weighted parameters. Each node in the hidden layer computes the nonlinear transformation (defined by an activation function) based on the Euclidean distance from all the weighted inputs to the node neuron *centre*. The weighted summation of each node is used to compute the output layer. The RBFNN mapping of a function $\hat{f}_r : \mathbf{R}^n \rightarrow \mathbf{R}$ can be expressed as follows:

$$\hat{f}_r(\mathbf{x}) = \eta + \sum_{j=1}^{n_h} \eta_{ij} \phi \left(\|\mathbf{x} - \mathbf{c}_j\|, \rho_j \right), \quad 1 \leq i \leq m \tag{4.6}$$

where the input vector is denoted by $\mathbf{x} \in \mathbf{R}^n$, $\phi(\cdot)$ is the activation function mapping from $\mathbf{R}^n$ to $\mathbf{R}$, the Euclidean distance is represented by $\|\cdot\|$, the network weights from the input layer are given as η, $0 \leq i \leq n_h$, the RBF centre are denoted by $\mathbf{c}_j \in \mathbf{R}^n$, $0 \leq i \leq n_h$, with n_h being the number of nodes. ρ_j denotes the RBF centre width. A Gaussian activation function can be described as:

$$\phi(z, \rho) = exp(-z^2/\rho^2) \tag{4.7}$$

where ρ is a design parameter. To maximize network learning and ensure network convergence time is minimised, various methods such as the orthogonal least squares (OLS) method have been used [107]. The implementation of an RBF learning algorithm in a real-time application needs to consider memory and computing time which will be discussed further in later sections in this chapter.

4.1.3 Outline

This chapter is organised as follows. Section 4.1.2 introduces the concept of identification of a nonlinear system using neural network framework. Section 4.2 discusses various learning algorithms applicable for a RBFNN structure. The verification process of the continuous forward algorithm (CFA) using a 6-DOF quadcopter model, is discussed in Section 4.3. Finally, the validation process of CFA algorithm within a real-time environment of the test-bench platform described in Chapter 3, is discussed in Section 4.4.

4.2 Neural Network Modelling

4.2.1 Orthogonal least-squares algorithm

A regression model is based on establishing a relationship between model basis functions and the dataset defining the parameter space. Although this model has been applied in numerous applications, the algorithm computational cost is proportional to the number of samples in the dataset. The orthogonal least-squares (Orthogonal Least Squares (OLS)) algorithm has been developed to minimise computational complexity through adopting a selection process of the radial basis centres [107, 78]. This is linear regression model that aims at computing the weights of the pre-chosen radial basis centres while minimising an objective function. Considering a linear regression model defined as:

$$d(t) = \sum_{i=1}^{M} p_i(t)\theta_i + e(t) \tag{4.8}$$

where M is the number of regressors. The desired output is defined by $d(t)$, θ_i and $p_i(t)$ are the estimated parameters and the regressors which are weighted functions of $\mathbf{x}(t)$, respectively. $e(t)$ is the assumed estimation error. Such regressors can be defined as:

$$p_i(t) = p_i(\mathbf{x}(t)) \tag{4.9}$$

The selection of the subset of regressors is based on achieving a low correlation factor with the assumed model error $e(t)$ which coincides with the selection problem of RBF centres $\mathbf{c}_i$ for a given nonlinearity $\phi(\cdot)$. A linear regression matrix system can be defined as:

$$\mathbf{d} = \mathbf{P}\Theta + \mathbf{E} \tag{4.10}$$

where

$$\mathbf{P} = [\mathbf{p}_1 \cdots \mathbf{p}_M], \quad \mathbf{p}_i = [p_i(1) \cdots p_i(N)]^T, \quad 1 \leq i \leq M \tag{4.11}$$

$$\Theta = [\theta_1 \cdots \theta_M]^T \tag{4.12}$$

$$\mathbf{E} = [e(t) \cdots e(N)]^T \tag{4.13}$$

The projection of $\mathbf{d}$ on the basis vectors consisting of regressors $\mathbf{p}_i$, defined as $\mathbf{P}\hat{\Theta}$, is a condition to be satisfied by the computation of $\hat{\Theta}$ through the least-squares (LS) method. The

desired output energy produced by the chosen regressors is computed by the square of $\mathbf{P}\hat{\Theta}$. The transformation of the $\mathbf{p}_i$ into orthogonal basis functions in order to compute the contribution of an individual regressor to the desired output energy, is the basis of the orthogonal least-squares method. Given a regression matrix $\mathbf{P}$ described as:

$$\mathbf{P} = \mathbf{WA} \tag{4.14}$$

where $\mathbf{A}$ is a triangular matrix of size $M \times M$ consisting of:

$$A = \begin{bmatrix} 1 & \alpha_{12} & \alpha_{13} & \cdots & \alpha_{1M} \\ 0 & 1 & \alpha_{23} & \cdots & \alpha_{2M} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & \cdots & 0 & 1 & \alpha_{M-1M} \\ 0 & \cdots & 0 & 0 & 1 \end{bmatrix}$$

and $\mathbf{W}$ is a matrix with orthogonal columns $\mathbf{w}_i$ of size $N \times M$ such that

$$\mathbf{W}^T\mathbf{W} = \mathbf{H} \tag{4.15}$$

where H is a matrix with diagonal elements h_i:

$$\mathbf{h}_i = \mathbf{w}_i^T\mathbf{w}_i = \sum_{t=1}^{N} w_i(t)w_i(t), \quad 1 \leq i \leq M \tag{4.16}$$

Given that the regressors $\mathbf{p}_i$ and the orthogonal vector $\mathbf{w}_i$ spanned the same space, Eq.4.10 can be rewritten as:

$$\mathbf{d} = \mathbf{Wg} + \mathbf{E} \tag{4.17}$$

The orthogonal least-squares solution $\hat{g}$ is given as:

$$\hat{\mathbf{g}} = \mathbf{H}^{-1}\mathbf{W}^T\mathbf{d} \tag{4.18}$$

and expanded as:

$$\hat{\mathbf{g}}_i = \mathbf{w}_i^T\mathbf{d}/(\mathbf{w}_i^T\mathbf{w}_i), \quad 1 \leq i \leq M \tag{4.19}$$

The Gram-Schmidt algorithm has been used to obtain the orthogonal decomposition of $\mathbf{P}$ [107]. This is achieved by computing each column of matrix $\mathbf{A}$ as follows: at the kth stage

make the kth column orthogonal to each of the $k-1$ previously orthogonalised columns and repeat the operation for $k = 2, \cdots, M$. By initialising a vector such that:

$$\mathbf{w}_1 = \mathbf{p}_1$$

$$\alpha_{ik} = \mathbf{w}_i^T \mathbf{p}_k / (\mathbf{w}_i^T \mathbf{w}_i), \quad 1 \leq i \leq k$$

$$\mathbf{w}_k = \mathbf{p}_k - \sum_{i=1}^{k-1} \alpha_{ik} \mathbf{w}_i$$

Given that the orthogonal nature between $\mathbf{w}_i$ and $\mathbf{w}_j$ for $i \neq j$, the sum squares or energy of $d(t)$ is

$$\mathbf{d}^T \mathbf{d} = \sum_{i=1}^{M} g_i^2 \mathbf{w}_i^T \mathbf{w}_i + \mathbf{E}^T \mathbf{E} \tag{4.20}$$

The variance of $d(t)$ can be computed once the mean has been removed from the desired output vector $\mathbf{d}$. Such variance is defined:

$$N^{-1} \mathbf{d}^T \mathbf{d} = N^{-1} \sum_{i=1}^{M} g_i^2 \mathbf{w}_i^T \mathbf{w}_i + N^{-1} \mathbf{E}^T \mathbf{E} \tag{4.21}$$

The desired output variance consists of the explained desired output defined as $g_i^2 \mathbf{w}_i^T \mathbf{w}_i / N$ computed by $\mathbf{w}_i$ and an unexplained variance of *d(t)*. Dividing Eq. 4.21 by $N^{-1} \mathbf{d}^T \mathbf{d}$, the explained error can be computed:

$$\mathbf{E}^T \mathbf{E} / \mathbf{d}^T \mathbf{d} = 1 - \frac{1}{\mathbf{d}^T \mathbf{d}} g_i^2 \mathbf{w}_i^T \mathbf{w}_i \tag{4.22}$$

Based on Eq. 4.22 the maximum value of the ratio $g_i^2 \mathbf{w}_i^T \mathbf{w}_i / \mathbf{d}^T \mathbf{d}$ will result the unexplained error being minimised. The above approach enables the selection of a subset of salient regressors $\mathbf{w}_i$ such that the desired output energy unexplained error will be minimised. An error function can be defined:

$$[err]_i = g_i^2 \mathbf{w}_i^T \mathbf{w}_i / \mathbf{d}^T \mathbf{d} \tag{4.23}$$

The selection of regressors is such that:

$$1 - \sum_{j=1}^{M_s} [err]_j < \rho \tag{4.24}$$

where M_s denotes size of the significant regressors subset and $0 < \rho < 1$ denotes a design tolerance. The above can be formulated into a procedure: At the first step, for $1 \leq i \leq M$

$$\mathbf{w}_1 = \mathbf{p}_i$$

$$g_1^{(i)} = (\mathbf{w}_1^{(i)})^T \mathbf{d} / ((\mathbf{w}_1^{(i)})^T \mathbf{w}_1^{(i)})$$

$$[err]_1^{(i)} = (g_1^{(i)})^2 (\mathbf{w}_1^{(i)})^T \mathbf{w}_1^{(i)} / (\mathbf{d}^T \mathbf{d})$$

Find

$$[err]_1^{(i_1)} = max[err]_1^{(i)}$$

and make the selection

$$\mathbf{w}_1 = \mathbf{w}_1^{(i_1)} = \mathbf{p}_{i_1}$$

At the *k*th step, where $k \geq 2$

$$\alpha_{jk}^{(i)} = \mathbf{w}_j^T \mathbf{p}_i / (\mathbf{w}_j^T \mathbf{w}_j)$$

$$\mathbf{w}_k^{(i)} = p_i - \sum_{j=1}^{k-1} \alpha_{jk}^{(i)} \mathbf{w}_j$$

$$g_k^{(i)} = (\mathbf{w}_k^{(i)})^T \mathbf{d} / ((\mathbf{w}_k^{(i)})^T \mathbf{w}_k^{(i)})$$

$$[err]_k^{(i)} = (g_k^{(i)})^2 (\mathbf{w}_k^{(i)})^T \mathbf{w}_k^{(i)} / (\mathbf{d}^T \mathbf{d})$$

find

$$[err]_k^{(i_1)} = max[err]_k^{(i)} \quad 1 \leq i \leq M,\ i \neq i_1$$

and make the selection

$$\mathbf{w}_k = \mathbf{w}_k^{(i_k)} = p_{i_k} - \sum_{j=1}^{k-1} \alpha_{jk}^{(i_k)} \mathbf{w}_j$$

The procedure is completed at the M_s-th step based on the following condition:

$$1 - \sum_{j=1}^{M_s} [err]_j < \rho$$

The prediction of the desired output is achieved by capturing the underlying system dynamics through a matrix of $\mathbf{w}_k^{(i_k)}$. The size of such a matrix is also representative of the number of neuron centres $\mathbf{c}_i$ within a neural network architecture.

4.2.2 Continuous forward algorithm

The continuous forward algorithm (Continuous Forward Algorithm (CFA)) was introduced by Peng et al., [108], as a framework with analytical capability for both parameter optimisation and neural network construction. The CFA, unlike feedforward selection algorithms, has the ability to optimise over the parameter space, the nonlinear parameters as the network architecture is adapted [3, 4]. The main advantage of such a method includes an improved

method for neural network modelling while achieving a low memory footprint per each network learning iteration. An input-output dataset for a nonlinear RBFNN system is defined as:

$$\hat{\mathbf{y}} = \sum_{i=1}^{m} w_i \phi_i(\mathbf{x}, \boldsymbol{\sigma}_i, \mathbf{c}_i) \tag{4.25}$$

where m is the number of hidden nodes, $\hat{y}$ and $\mathbf{x}$ are the RBFNN output and the input vector respectively. $\phi_i(\mathbf{x}, \boldsymbol{\sigma}_i, \mathbf{c}_i)$ is the activation function of the ith node of the hidden layer. $\mathbf{c}_i$, $\boldsymbol{\sigma}_i$ and w_i denotes the node centre, width and linear output weight. The optimal parameter set is obtained by minimising a sum squared error (Steady-State Error (SSE)) defined as:

$$\mathbf{J}(\mathbf{w}, \boldsymbol{\sigma}, \mathbf{c}) = (\mathbf{y} - \hat{\mathbf{y}})^T (\mathbf{y} - \hat{\mathbf{y}}) \tag{4.26}$$

The output vector is denoted as $\mathbf{y}$. A subset of k basis vectors exists from M candidates such that the cost function can be minimised by computing network weights defined as:

$$\mathbf{w} = (\boldsymbol{\Phi}_k^T \boldsymbol{\Phi}_k)^{-1} \boldsymbol{\Phi}_k^T \mathbf{y} \tag{4.27}$$

where $\boldsymbol{\Phi}_k = [\phi_1, ..., \phi_k]$ is a regressor subset to ensure the minimisation of the cost function defined in (4.26) can be expressed as a function of regressor vector:

$$\mathbf{J}(\boldsymbol{\Phi}_k) = \mathbf{y}^T \left[\mathbf{I} - \boldsymbol{\Phi}_k \left(\boldsymbol{\Phi}_k^T \boldsymbol{\Phi}_k \right)^{-1} \boldsymbol{\Phi}_k^T \right] \mathbf{y} \tag{4.28}$$

Further minimisation of the cost function can be achieved by specifying a basis vector $\forall \phi \in (\phi_{k+1}, ..., \phi_M)$ as part of the regression matrix resulting in $\boldsymbol{\Phi}_{k+1} = [\boldsymbol{\Phi}_k, \phi]$. This cost function net reduction based on the updated regression matrix is defined as:

$$\Delta \mathbf{J}_{k+1}(\phi) = \mathbf{J}(\boldsymbol{\Phi}_k) - \mathbf{J}([\boldsymbol{\Phi}_k, \phi]) \tag{4.29}$$

Based on the above cost function net reduction, a residual matrix exists such that [109]:

$$\mathbf{R}_k = \begin{cases} \mathbf{I} - \boldsymbol{\Phi}_k \left(\boldsymbol{\Phi}_k^T \boldsymbol{\Phi}_k \right)^{-1} \boldsymbol{\Phi}_k^T, & 0 < k < M \\ \mathbf{I}, & k = 0 \end{cases} \tag{4.30}$$

such that:

$$\phi^{(k)} \triangleq \mathbf{R}_k \phi \quad \mathbf{y}^{(k)} \triangleq \mathbf{R}_k \mathbf{y} \tag{4.31}$$

where ϕ and $\mathbf{y}$ are a column vector and output vector respectively such that: $\phi^{(0)} = \phi$ and $\mathbf{y}^{(0)} = \mathbf{y}$. A recursive update of such vectors can be achieved as follows:

$$\phi^{(k)} = \phi^{(k-1)} - \frac{(\phi_k^{(k-1)})^T(\phi^{(k-1)})}{(\phi_k^{(k-1)})^T(\phi_k^{(k-1)})}\phi_k^{(k-1)} \tag{4.32}$$

Without a detail analysis of the residual matrix properties defined in [109], the recursive computation of the node regressors can be applied for the kth basis vector, defined as:

$$\mathbf{y}^{(k)} = \mathbf{y}^{(k-1)} - \frac{(\phi_k^{(k-1)})^T(\mathbf{y}^{(k-1)})}{(\phi_k^{(k-1)})^T(\phi_k^{(k-1)})}\phi_k^{(k-1)} \tag{4.33}$$

Similarly, the cost function net contribution is defined as:

$$\Delta\mathbf{J}_{k+1}(\phi) = \frac{[(\mathbf{y}^{(k)})^T\phi^{(k)}]^2}{(\phi^{(k)})^T\phi^{(k)}} \tag{4.34}$$

The implementation of (4.32) and (4.33) can be simplified by introducing an upper triangular matrix $\mathbf{A}$ of size $k \times M$ which consists of the cost function k basis vectors. Such a matrix is defined as:

$$\mathbf{A} \triangleq [a_{i,j}]_{k\times M} \tag{4.35}$$

$$a_{i,j} = \begin{cases} 0, & j < i \\ \phi_i^T\mathbf{R}_{i-1}\phi_i = (\phi_i^{(i-1)})^T(\phi_i^{(i-1)}), & j = i \\ \phi_i^T\mathbf{R}_{i-1}\phi_j = (\phi_i^{(i-1)})^T(\phi_j^{(i-1)}), & i < j < M \end{cases} \tag{4.36}$$

shown here in matrix format:

$$A = \begin{bmatrix} a_{1,1} & a_{1,2} & a_{1,3} & \cdots & a_{1,M} \\ 0 & a_{2,2} & a_{2,3} & \cdots & a_{2,M} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & \cdots & 0 & a_{k-1,M-k} & a_{k-1,M} \\ 0 & \cdots & 0 & 0 & a_{k,M} \end{bmatrix}$$

whereby:

$$\mathbf{a}_y \triangleq [a_{i,y}]_{M\times 1}, \quad a_{i,y} = \mathbf{y}^T\mathbf{R}_k\phi_i = (\mathbf{y}^{(k)})^T\phi_i^{(k)} \tag{4.37}$$

$$\mathbf{d} \triangleq [d_i]_{M\times 1}, \quad d_i = \phi_i^T\mathbf{R}_k\phi_i = (\phi_i^{(k)})^T\phi_i^{(k)} \tag{4.38}$$

the parameter set of each hidden node (width and centre) can be expressed as:

$$\phi(\mathbf{x}(t), \boldsymbol{\sigma}, \mathbf{c}) = \Phi(\mathbf{x}(t), \omega) \tag{4.39}$$

where

$$\omega = [\omega_0, \omega_1, ..., \omega_n] = [\boldsymbol{\sigma}, \mathbf{c}] \tag{4.40}$$

The cost function net contribution can be defined as:

$$\Delta \mathbf{J}_{k+1}(\omega) = C^2(\omega)/D(\omega) \tag{4.41}$$

where

$$C(\omega) = (\mathbf{y}^{(k)})^T \phi^{(k)}(\omega) = \sum_{t=1}^{N} y^{(k)} \phi^{(k)}(\mathbf{x}(t), \omega) \tag{4.42}$$

$$D(\omega) = (\phi^{(k)}(\omega))^T \phi^{(k)}(\omega) = \sum_{t=1}^{N} (\phi^{(k)}(\mathbf{x}(t), \omega))^2 \tag{4.43}$$

where N denotes the number of training samples. The optimisation of each $k+1$-th added hidden node parameter set added to the regression matrix ensures that the cost function net contribution is maximised at each training iteration. This is achieved through a conjugate gradient approach by computing the derivative of cost function net contribution defined as [3]:

$$\begin{aligned} &\frac{\partial(\Delta \mathbf{J}_{k+1}(\omega)}{\partial \omega_i} = \nabla \Delta \mathbf{J}_{k+1}(\omega) \\ =&\frac{2C(\omega)}{D(\omega)} \left(\mathbf{y}^{(k)} - \frac{C(\omega)}{D(\omega)} \phi^{(k)}(\omega) \right)^T \phi_{\omega_i}^{(k)}(\omega) \\ &i = 0, 1, ..., n \end{aligned} \tag{4.44}$$

where $\phi_{\omega_i}^{(k)}$ is the impact factor of the k-th hidden node regressor based on the computed parameter set ω_i for a given i-th input vector, defined as:

$$\begin{aligned} \phi_i^{(s)}(\omega) = \phi_i^{(s-1)}(\omega) - \frac{\partial a_{s,k+1}(\omega)}{\partial \omega_i} \frac{\phi_s^{(s-1)}}{a_{s,s}} \\ s = 1, ...k \quad i = 0, 1, ..., n \end{aligned} \tag{4.45}$$

where

$$\frac{\partial a_{s,k+1}(\omega)}{\partial \omega_i} = (\phi_s^{(s-1)}(\omega))^T \phi_i^{(s-1)}(\omega) \tag{4.46}$$

The algorithm can be initialised at $k = 0$ such that:

$$\begin{aligned}
&\phi^{(0)}_{\omega_0}(\mathbf{x}(t), \omega) = -2\omega_0 \sum_{i=1}^{n} (x_i(t) - \omega_i)^2 \phi(\mathbf{x}(t), \omega) \\
&\phi^{(0)}_{\omega_i}(\mathbf{x}(t), \omega) = 2\omega_0^2 (x_i(t) - \omega_i) \phi(\mathbf{x}(t), \omega) \\
&i = 1, ..., n
\end{aligned} \tag{4.47}$$

with node parameter set can be initialised as:

$$\omega_0^{(0)} = \left[\sum_{i=1}^{n} \sum_{t=1}^{N} (x_i(t) - \omega_i^{(0)})^2 / N \right]^{-1/2} \tag{4.48}$$

This conjugate gradient method makes use of a line search iterative procedure such that gradient of the contribution of $\nabla \Delta \mathbf{J}_{k+1}(\omega^{(p)}_{k+1})$ results in a parameter set $\omega^{(p)}_{k+1}$ that maximises $\Delta \mathbf{J}_{k+1}(\omega^{(p)}_{k+1})$ and is considered an optimal value. $\phi(\omega^{(0)}_{k+1})$ is then computed along with the triangular matrix A resulting in the $k+1$th hidden node being updated with $\phi(\omega^{(k)}_{k+1})$ as defined in (4.32) and its output vector can be computed as defined in (4.33). This results in the reduction in SSE defined as:

$$SSE^{(k+1)} = SSE^{(k)} - \Delta \mathbf{J}_{k+1}(\omega^{(k)}_{k+1}) \tag{4.49}$$

The above procedure is iterated until the net contribution of the cost function meets the condition of being below a defined threshold η (0.1 has been used) or the SSE is below $1e-3$.

4.2.3 Computational complexity

Both the OLS and CFAs described above attempt at computing the location of the regressor which maximises the projection of the sum squares error along the stated regressor vector such that each RBFNN node configuration captures the underlying dynamics of the input-output mapping. In order to achieve this recursive computation, the normalised input-output space needs to be stored to memory prior to the learning process to commence. Moreover, the computation of the OLS-trained RBFNN output is based on the pseudo-inverse of the hidden layer network node outputs summation which is prone to numerical errors and results in higher CPU computation effort.

In contrast to the CFA which provides an analytical framework for computing the RBFNN parameters as part of the regressor location, the OLS training algorithm logic is simpler although the user-defined values of RBFNN spread σ and centre output $\mathbf{c}$ require that the

entire cross-correlation regression matrix $\mathbf{P}$ be stored in memory. In addition, in order for the network trained by the OLS algorithm to predict an output using a validation dataset, the Euclidean distance between the orthogonal vectors $\mathbf{w}$ and regressors $\mathbf{p}$ needs to be computed. This is required to compute the weighted output of each network node.

4.2.4 Model prediction performance

As stated through the description of the OLS algorithm in Section 4.2.1, parameter estimation of the network weights and biases is based on user-defined network node centre and spread. However, this poses some numerical difficulties in the OLS algorithm prediction/validation capabilities compared to the CFA. The validation step is satisfied through the condition that the prediction model captures the true system dynamics. A common method is to simulate the predicted output $\hat{y}$ from measured inputs and compare it with the measured output $y(t)$. The model fit is given as:

$$P_f = \left\| 1 - \frac{\|\nu(t)\|}{\|y(t) - \bar{y}\|} \right\| \tag{4.50}$$

$$P_i = \left\| 1 - \frac{\|y(t) - \bar{y}\|}{\|\nu(t)\|} \right\| \tag{4.51}$$

where $\|.\|$ is the L2 norm and $\bar{y}$ is the mean value of $y(t)$. $\|\nu(t)\|$ is the training SSE. To ensure valid model fit, the data used to validate must be different from the data used during the estimation/training. This process is known as cross-validation [80, p. 501]. The P_f value of above 0.5 with the low number of neurons (less than a third of the sample size) ensures that a good estimated model has been achieved. An inverse of this function given as P_i which results in number close to 0, can also be used to verify the estimated model is free from overfitting. The fraction of these two numbers, falling in the range of 1.0 and 3.0, confirms the cross-validation process has been successful. A comparison in training fit, prediction accuracy and sensitivity to signal noise between the OLS and the CFA algorithm is shown in Figures 4.1 to 4.4.

The input-output mappings from the linear model described in Section 4.3.1 is used to evaluate the network performance and robustness to noise. The input vector used for the training was $[\phi, \theta, \psi, u, v, w, a_x, a_y, a_z, p, r]$ and the output vector used for training was q. For the no-noise case both the OLS and CFA algorithms produced a network structure with 2 and 4 neurons respectively. This resulted in an SSE of 3.44 and 1.07 for the OLS and CFA outputs.

This can be seen clearly given the poor representation of the training dataset by the OLS algorithm. It is evident that the CFA has greater robustness to noise than the OLS algorithm through its ability to prediction the output. This is because the OLS model structure increase to 3 neurons with the SSE of 2.51. The CFA algorithm performed also better with a model structure of 5 neurons and a SSE 1.97. This ensures that there is an adequate balance between data over-fitting and not capturing the underlying system dynamics.

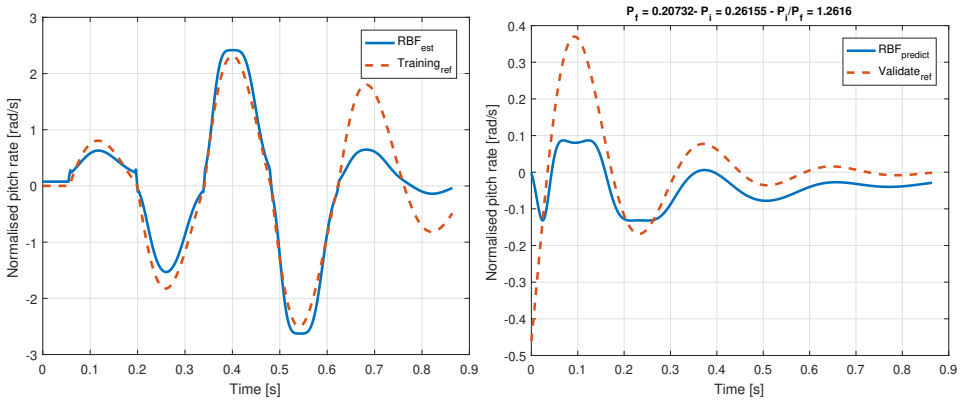


Figure 4.1: OLS algorithm (1) network estimation/training output (2) network prediction output for 5m/s forward flight and no noise.

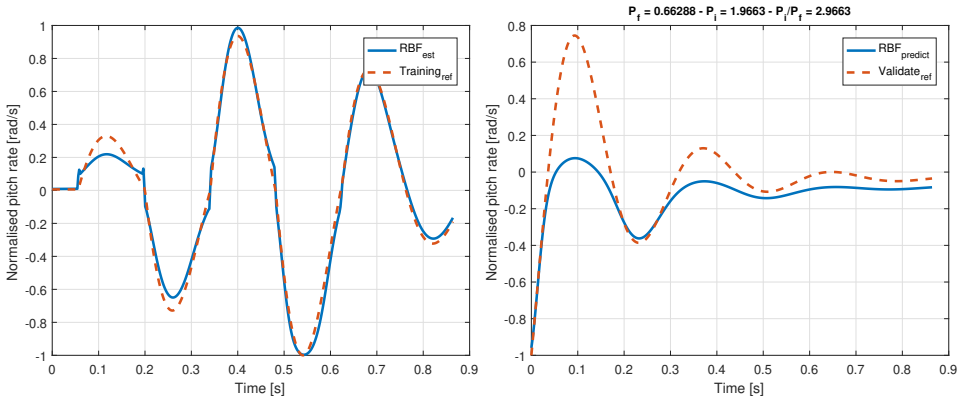


Figure 4.2: CFA algorithm (1) network estimation/training output (2) network prediction output for 5m/s forward flight and no noise.

4.3 CFA Verification

In order to verify the computational effectiveness of the CFA as described in Section 4.2.2, the algorithm was coded in the MATLAB environment and structured in a way for error debugging to be possible. Moreover, the prediction performance of the CFA for the purpose

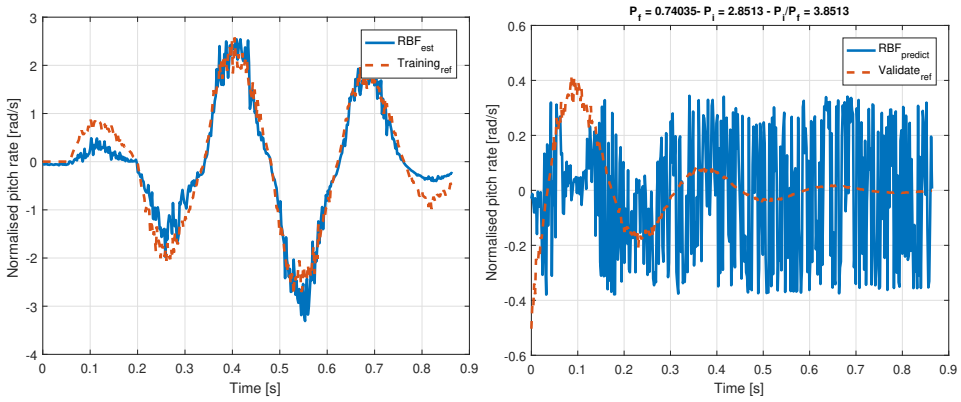


Figure 4.3: OLS algorithm (1) network estimation/training output (2) network prediction output for 5m/s forward flight and low noise.

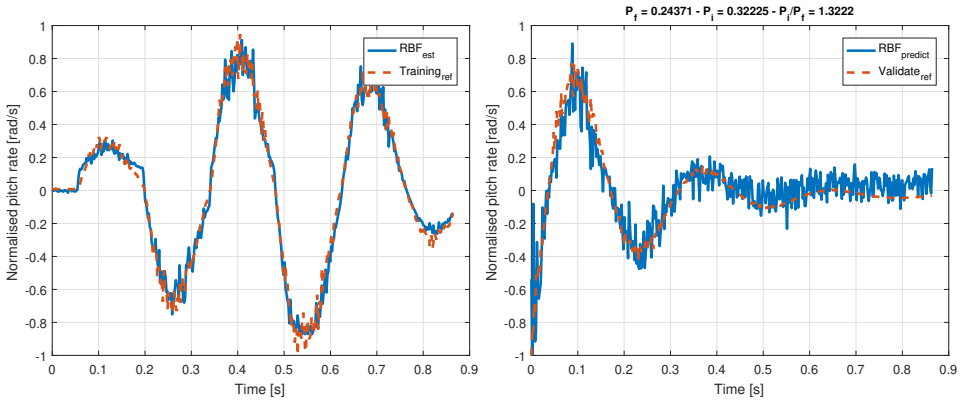


Figure 4.4: CFA algorithm (1) network estimation/training output (2) network prediction output for 5m/s forward flight and low noise.

of the aerodynamic parameter estimation was adopted. This was an essential step towards the real-time validation of the CFA as part of the neural network identification of faulty nonlinear rotor dynamics of a quadcopter.

4.3.1 Linear model structure

As described above, one of the objectives of nonlinear system identification was to predict the underlying dynamics of the estimated system. Extracting linearised aerodynamic coefficients has proven in the past research to be an efficient way of quantifying if that objective was met [110, 111]. The chosen linear model is based on the analytical linearisation of the nonlinear quadcopter dynamics described in Section 3.3.2, with respect to forward speed. Such simple linear models can be used for (1) the model-based design of a linear controller, (2) the assessment of various RBFNN training algorithms while ensuring the training and validation data based on excitation around the trim condition.

The linear model of the quadcopter was obtained through trimming the model described in Section 3.3.3 (including the rigid-body dynamics) at five trim conditions: hover, 5m/s, 8m/s, 12m/s, 17m/s and 20m/s. This resulted in the reduced linearised model given as:

$$\dot{u} = X_\theta \theta + X_q q + X_u u + \sum_{i=1}^{N} X_{a_{1i}} a_{1i} + X_{\delta_{lon}} \delta_{lon} \tag{4.52}$$

$$\dot{v} = Y_\theta \phi + Y_p p + + Y_v v \sum_{i=1}^{N} Y_{b_{1i}} b_{1i} + Y_{\delta_{lat}} \delta_{lat} \tag{4.53}$$

$$\dot{w} = Z_\theta \theta + Z_q q + Z_w w + Z_{\delta_{mot}} \delta_{mot} \tag{4.54}$$

$$\dot{p} = L_p p + \sum_{i=1}^{N} L_{b_{1i}} b_{1i} + L_{\delta_{lat}} \delta_{lat} + L_{\delta_{rud}} \delta_{rud} \tag{4.55}$$

$$\dot{q} = M_q q + \sum_{i=1}^{N} M_{a_{1i}} a_{1i} + M_{\delta_{lon}} \delta_{lon} + M_{\delta_{mot}} \delta_{mot} \tag{4.56}$$

$$\dot{r} = N_r r + N_{\delta_{lat}} \delta_{lat} + N_{\delta_{rud}} \delta_{rud} \tag{4.57}$$

$$\dot{a_{1i}} = A_q^{1i} q + A_{a_{1i}}^{1i} a_{1i} + A_{\delta_{lon}}^{1i} \delta_{lon} \tag{4.58}$$

$$\dot{b_{1i}} = B_p^{1i} p + B_{b_{1i}}^{1i} b_{1i} + B_{\delta_{lat}}^{1i} \delta_{lat} \tag{4.59}$$

where N represents the number of rotors. The above equations form part of the model to generate input-output mappings for neural network identification and algorithm verification. Given that the longitudinal short-period mode of a quadcopter is dominated by the rotor flapping dynamics (as the moments generated are not constant around a trim condi-

tion but exhibit a dynamic behaviour as a function of flapping angle), its natural frequency can be described as [112, pp. 302]:

$$\omega_n = \sqrt{-\sum_{i=1}^{N} M_{a_{1i}}} \tag{4.60}$$

Unlike helicopters, quadcopters flapping motion dynamics is of a higher-frequency nature which is a result of high flap stiffness (propeller blades have no hinge of rotation) and rotational speed (large helicopter blades rotate slower than smaller quadcopters). A resonant frequency that would maximise the spectral density of the system input-output mapping (defined in Section 4.3.3), such that the natural frequency can be identified and can be defined as: $\omega_R = \omega_n\sqrt{1-2\zeta^2}$.

This parameter estimation technique of the longitudinal rotor flapping short-period M_{a_1} is used to verify that the CFA has been correctly implemented and its prediction capabilities of the underlying system dynamics are being tested on linear model prior to the identification of the nonlinear dynamics.

4.3.2 Model structure determination

To improve further parameter estimation inaccuracies and reduce the network structure, collinearity between dependent and independent variables was used as the preferred approach. The correlations between independent variables x_i and output variable y is computed as [84, pp. 200]:

$$\rho_i = \frac{\sum_{k=1}^{N}(y(k)-\bar{y})(x_i(k)-\bar{x_i})}{\sqrt{\sum_{k=1}^{N}(y(k)-\bar{y})^2\sum_{k=1}^{N}(x_i(k)-\bar{x_i})^2}} \tag{4.61}$$

A signal voting mechanism which uses the signal mean and its standard deviation applied such that signals with both statistics below 1e-3 are removed from the dataset. Thereafter, the input vector is reduced further by selecting the four input signals with the strongest correlation to the output signal for RBFNN training using the CFA. This approach was also useful as quadcopter model dynamics change with velocity (such as the transition from hover to forward flight).

4.3.3 Optimal input design

In order to implement system identification framework with real-time constraints, the formulation of the system excitation inputs needs to minimise time and peak response while maximising information content. This can be achieved through the a-priori knowledge of system dynamic response to various system identification manoeuvres [112, pp. 85]. Time domain manoeuvres such as doublet, step and 3211 inputs can be analysed using the power spectral function defined as [84, pp. 38]:

$$E(\omega) = 2\Delta t^2 \frac{1 - cos\Omega}{\Omega^2} \times \left[\sum_{i=1}^{N} V_i^2 + 2 \sum_{j=1}^{N} cosj\Omega \sum_{i=1}^{N} V_i V_{i+j} \right] \quad (4.62)$$

where N is the number of impulses with a time duration Δt and amplitude V and normalised frequency $\Omega = \omega \Delta t$. A typical 3211 manoeuvre resulting in an input-output mapping used for the RBFNN training and validation is shown in Figure 4.5. Figure 4.6 shows the normalised power spectral output as a function of step-size length. The optimal step size, based on Figure 4.5, was 135 milliseconds, was then used during the implementation of the identification manoeuvres as part of the FDD framework as described in Chapter 5. It can be observed that the power spectral energy is not affected by forward speed although the overall forces and moments increased with airspeed given that this measured is normalised. The fact that the spectral energy decreases with smaller step sizes is due to the airframe inertial effects which are overcome by sustained energy. The decrease of the spectral energy by too large step size is due to the closed-loop response in regulating the measured variable (in this case pitch rate) and reducing the overall energy over the same time period.

4.3.4 Results

Using the linearised simulation model as described in the Section 4.3.1 the model structure was determined using the computed correlations as described in Section 4.3.2. Based on studying the underlying dynamics of the quadrotor under nominal and faulty conditions, the yaw rate r was chosen single-output variable for network training. The highest correlation coefficients to the dependent input variable were chosen for the RBFNN training based on the following set available from the IMU sensor: $[\phi, \theta, \psi, p, q, u, v, w, a_x]$. Table 4.1 shows the independent variables chosen as a function of forward speed. The effect of noise was

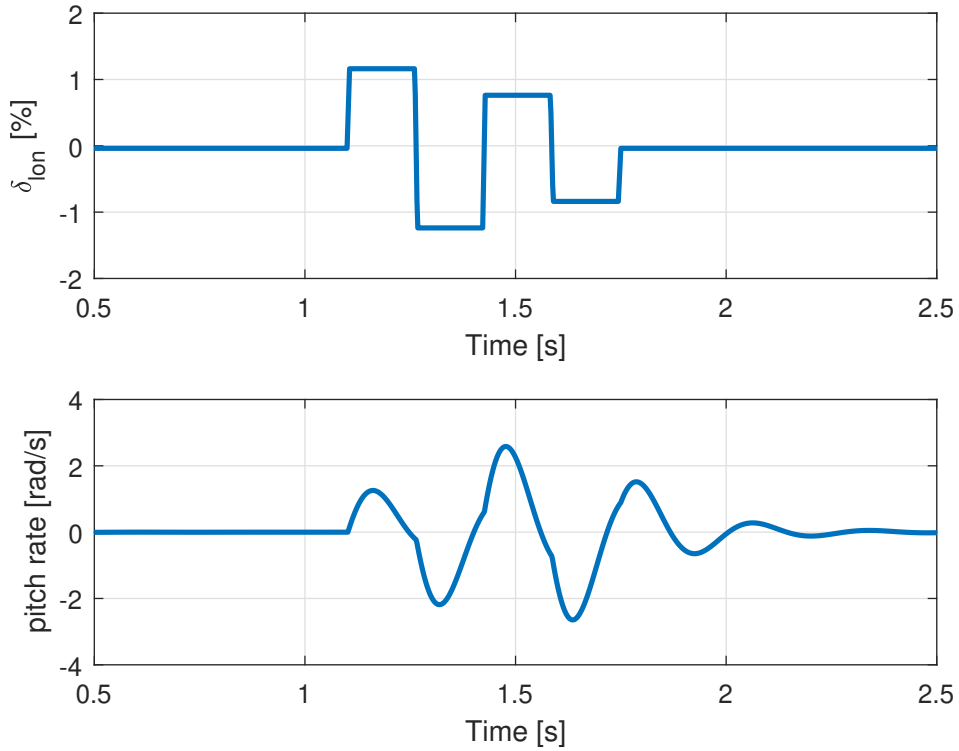


Figure 4.5: Input/Output design for rotor dynamics identification

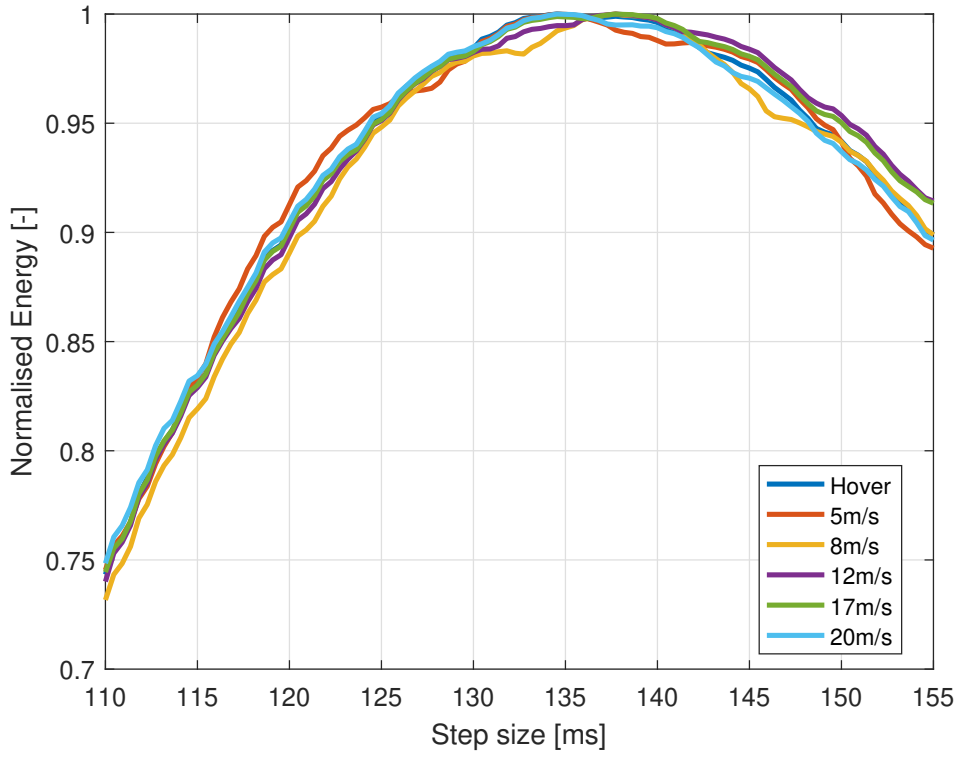


Figure 4.6: Normalised power spectral function sensitivity with speed

also investigated to evaluate its effect on the model structure determination. It shows that the model structure is not sensitive to noise which can be introduced with the digitisation of analog signals and the numerical computation errors. The limited input-output mapping size at hover shows that the RBFNN identification at low speeds yields poor model prediction capability.

Table 4.1: RBFNN model structure selection

Noise	Hover	5 m/s	12 m/s	20 m/s
No noise	δ_{lon}	δ_{lon}, a_z	δ_{lon}, a_x, a_z	δ_{lon}, a_x, a_z
Low	δ_{lon}	δ_{lon}, a_z	δ_{lon}, a_x, a_z	δ_{lon}, a_x, a_z
High	δ_{lon}, w	δ_{lon}, a_z	δ_{lon}, a_x, a_z	δ_{lon}, a_x, a_z

The CFA as described in Section 4.2.2 was used for training the RBF model structure based on simulated data. For each flight condition and noise level, the number of neurons and the resultant SSE were computed once ΔJ_{k+1} had reached zero. This is shown in Table 4.2. Unlike the OLS algorithm which needs a user-defined goal value, CFA is capable of stopping training automatically ensuring minimal RBFNN model overfitting. The CFA's good robustness to noise through its ability to complete the training while keeping the number of neurons in the prediction model consistently below 5.

Table 4.2: RBFNN training results - no. of neurons (SSE)

Noise	Hover	5 m/s	8 m/s	12 m/s	17 m/s	20 m/s
No noise	5 (7.43)	3 (0.59)	3 (0.47)	3 (0.56)	3 (0.36)	3 (0.55)
Low	3 (7.41)	3 (1.63)	4 (1.21)	3 (1.09)	3 (0.98)	3 (0.99)
High	2 (7.89)	3 (3.42)	3 (2.78)	3 (2.31)	3 (2.63)	4 (1.59)

The CFA's robustness to noise is further illustrated in Figure 4.7 which shows the prediction accuracy of the RBFNN model as a function of forward speed and noise levels. As expected, poor prediction capability (given an input-output mapping with low system con-

tent) in hover gets worse with increase in noise levels. However, as the information content improves with high-flight speeds, the prediction error decreases.

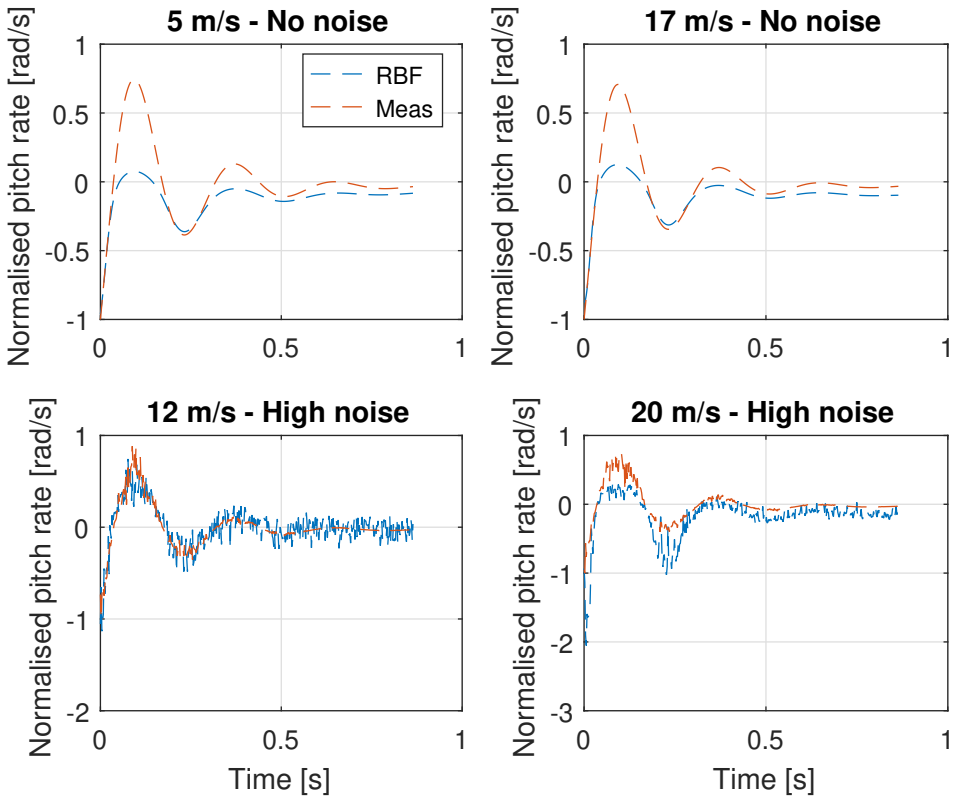


Figure 4.7: RBFNN prediction accuracy with various noise levels/forward speeds

The final CFA verification step was to compute the longitudinal rotor flapping dynamic coefficient M_{a_1} as described in Section 4.3.1. The predicted output from the CFA-trained RBFNN model was used to compute the resonant frequency ω_R by computing the damping ratio ζ defined as:

$$\zeta = \frac{1}{\sqrt{1 + \left(\frac{2\pi}{\delta}\right)^2}} \tag{4.63}$$

where δ is the logarithmic parameter defined as:

$$\delta = \frac{x_1}{x_2} \tag{4.64}$$

where x_1, x_2 are the amplitudes of two consecutive peaks of a decaying oscillating signal [113]. The true value of the M_{a_1} parameter was extracted from the linear model. The parameter estimation of M_{a_1} as a function of the forward speed is shown in Figure 4.8. It shows poor estimation of the M_{a_1} at low speeds due to the increases network having not captured the underlying dynamics. It has been shown above, that this is a function of network over-fitting (with number of neurons) and limited iterations in optimising networks

weights. The computational accuracy at higher speeds is largely dependent on signal noise, proving the estimation accuracy of the CFA algorithm. This shows that the CFA should be used in conjunction with an input-output mapping that contains enough information content to be insensitive to noise. This approach is expected to produce similar results with an input-output mapping obtained from a nonlinear system.

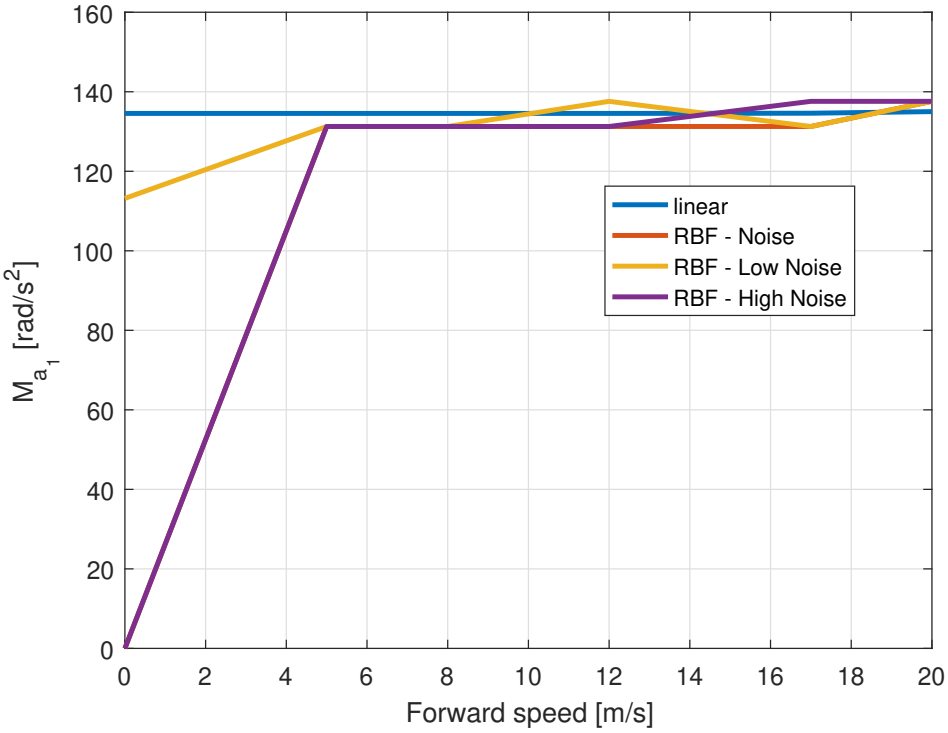


Figure 4.8: Longitudinal rotor flapping dynamic coefficient with forward speed

4.4 CFA Real-Time Validation

The validation of the CFA required that complete implementation and execution in the real-time environment. Achieving this objective was part of the development of an FDD process capable of integrating with real-time controller. Moreover, the CFA line search algorithm needed to be compatible with real-time constraints, so Golden Section Search with multi-modal optimisation capability was developed. This is described in Section 5.3.1. The Pixhawk controller as described in Section 3.2.2 is a Cortex-M4 with Floating Point Unit, an equivalent microcontroller with the same specifications, called the Teensy 3.6, was used to develop the embedded version of the CFA. A hardware-in-the-loop (HILS) process was built using a windows PC to send pre-simulated training data to the microcontroller via USB and

monitor the algorithm performance through serial outputs. This is shown in Figure 4.9.

Given that the microcontroller only has 256 Kbytes of memory, the CFA needed to be split into subroutines such that the CPU stack (local variables) and heap (global variables) memory allocations could be utilised. The MATLAB Coder workflow enabled the generation of embedded C-language code suitable for the microcontroller environment while verifying variable compatibility and possible memory leaks. Computing performance was further improved by typecasting each variable to its minimum size resulting in the entire training process not exceeding 2ms (approx 500Hz) with spikes of up to 4ms (250Hz) when using a dataset of 352 samples per signal and an input vector of 15 signals. Monitoring of the CFA training process is shown in Figure 4.10. The initial process of signal pre-conditioning and model structure determination, as described in Section 4.3.2, was achieved in 18 ms. This is expected as all the data arrays are being pre-populated prior to network training. The periodic spikes is due to some complex operation of storing of data from one parameter to another. This achievement formed the basis for the nonlinear system identification of faulty dynamics within an integrated environment.

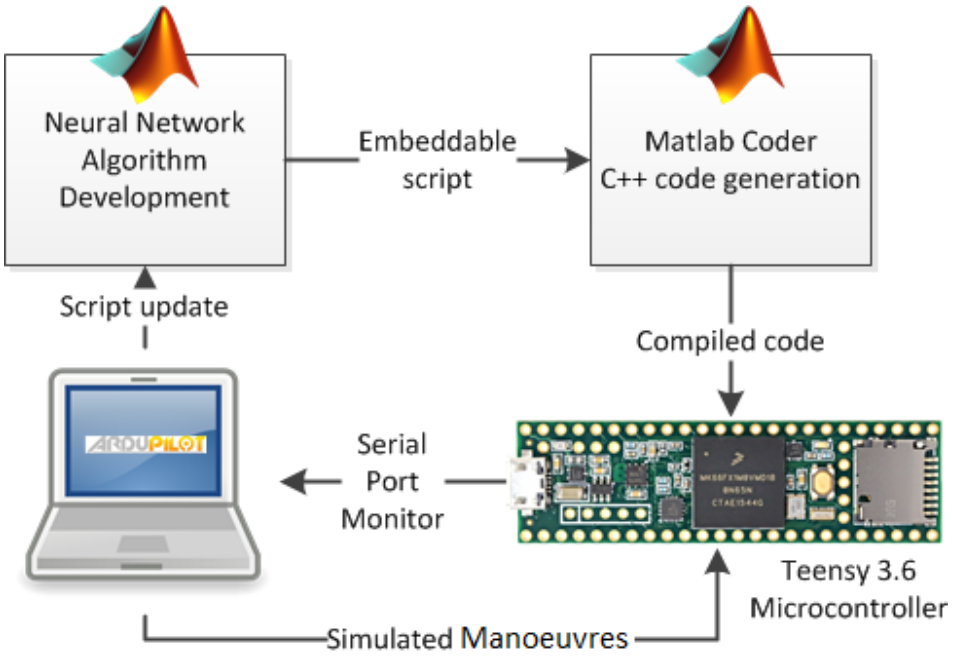


Figure 4.9: HILS framework for the real-time testing of FDD algorithms.

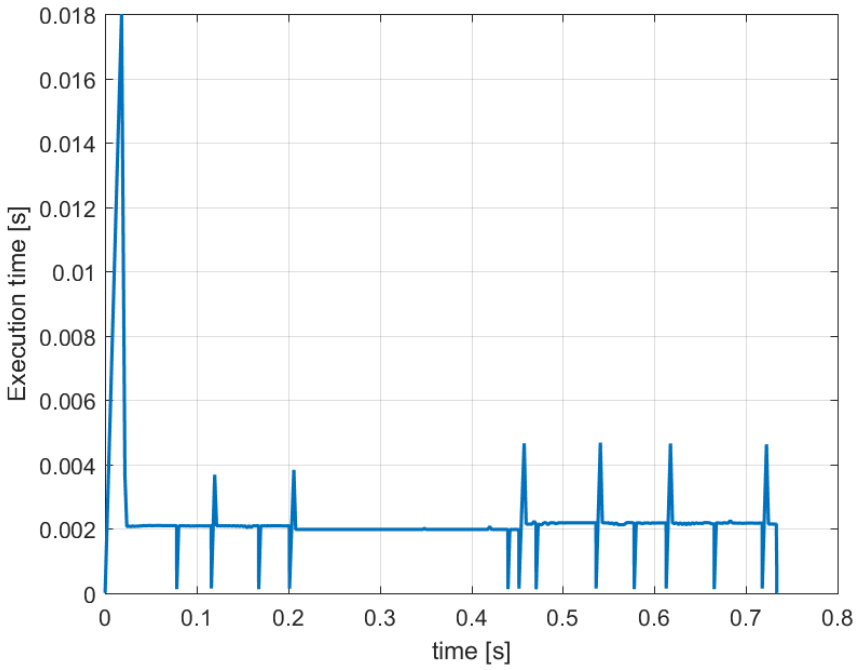


Figure 4.10: RBFNN - layer training performance in HILS.

Chapter 5

Integrated Fault-Tolerant Controller Synthesis

5.1 Introduction

From a system engineering perspective, the process of system integration requires the implementation of individual elements to be done in such a way that the overall system design properties and characteristics are achieved within its operating environment. Given that small unmanned rotorcraft are required to operate in increasingly condensed environments, concerns regarding such systems' reliability and operational safety in the event of both detected and undetected faults, are the primary focus for aviation regulation authorities. In addition, the level of effort in implementing fault-tolerant controllers within such unmanned systems has shifted towards incorporating a concurrent design approach of the individual elements and their interfaces [114].

The interface management of a FTCS main elements consists of defining the requirements to be imposed on the FDD and the RC designs such that their verification will result in an acceptable level of robustness to the list of possible faults associated with such a FTCS. Even though the behaviour of an unmanned rotorcraft system such as quadcopter can be poorly predicted mathematically once a severe fault has occurred, the definition of such requirements should be based on the FTCS desired response described by a set of design parameters. This will then ensure that the expected FTCS performance can be achieved through the

monitoring of such design parameters within the operational environment [115].

The importance and difficulty of developing such design parameters within an integrated FTC framework has been a discussion topic with associated research effort for the past few years [72, 21, 116]. Some researchers have focused more on providing a suitable FDD method compatible with adaptable RC with minimal consideration on real-time FTC performance [20], while others have ensured a practical solution is adapted through considering a subset of faults that an integrated FTCS can accommodate [74]. Integration issues such as: controller robustness of the poorly predicted post-fault model, quantification of system nonlinearities such as actuator saturation and speed and accuracy of overall FTC response have been addressed through indirectly imposed design constraints of both FDD and RC developments.

5.1.1 FDD integration requirements

Real-time fault estimation capability

The main advantage of an active FTCS is its ability to maintain overall system stability and performance by reacting to the occurrence of a fault once it has been detected and analysed. In order to achieve this, some form of online FDD and fault identification is required. This imposes real-time constraints on the development and implementation of the FDD mechanism such that its capability is fully utilised by the RC. However, the degree of accuracy that such an FDD framework can provide needs to be considered during the selection process of potential FDD methods such as observer-based, Kalman filter-based or neural networks-based [114]. This is especially crucial for fault localisation errors which has a direct impact on the adjustment of controller gains and overall system performance. The FDD mechanism needs to be adaptive to variations in system real-time behaviour in pre/post fault state, while ensuring the fault information and its associated accuracy can be incorporated in the controller reconfiguration.

Consideration of actuator limits

Most controllers' software implementation have some form of control signal rate saturation or saturation limits to prevent integration error windup and actuator electrical burn-out.

Provided that the chosen FDD mechanism requires some form of system excitation through superimposed system identification manoeuvres, considerations on reaching actuation or actuation rate limits must be taken into account. If ignored, such an event will not only deteriorate the tracking performance of the controller (even before controller reconfiguration is triggered), but will also cause the fault estimation error to diverge rendering the entire FTC mechanism ineffective towards system faults. Moreover, if the system actuator constraints are unknown to the FDD mechanism, some form of adaptive excitation signals needs to be incorporated, which might have a negative impact on the speed and accuracy of the fault information to be supplied to the controller reconfiguration mechanism [117]. The choice of the FDD mechanism will determine the level of system robustness to such control constraints being violated.

5.1.2 Reconfigurable controller integration requirements

Incorporate fault estimation uncertainties

As stated above, the real-time capability of an FDD mechanism will be accompanied with a degree of accuracy on the fault information. Under classical linear controller design, this can be incorporated in the gain and phase margin requirements. However, fault information from an FDD is often accompanied by a nonlinear system behaviour which is not easily mapped to the linear design thereby introducing reconfiguration errors in the RC mechanism. However, a nonlinear controller approach could minimise such problems but with an increase of computational requirements for a real-time implementation [114]. Moreover, fault uncertainties are time-varying and correlated to the type of FDD mechanism such that the chosen RC mechanism must be able to incorporate fault estimation uncertainties without negatively affecting the stability and performance of the baseline controller. This implies that some form of time-scale separation between the incorporation of the fault estimation uncertainties and the update controller gains must be established within real-time constraints [77].

Minimal reconfiguration time

Most FDD mechanisms are prone to estimation delays and false alarms. To ensure the accuracy of the fault information, some time is required between the fault occurring and the fault information being available to the controller reconfiguration mechanism. The minimisation of the controller reconfiguration time could imply three approaches: (1) the reconfiguration mechanism has the ability to initialise with fault information that contains large uncertainties without affecting the controller performance, (2) the controller reconfiguration time is relatively quick compared to the FDD detection time even under worse-case scenarios or (3) a combination of the first two approaches. However, system transient behaviour post the occurrence of fault needs to be taken into account which has an impact on both the FDD detection time and the RC reconfiguration time [114]. Assuming that a steady condition can be attained once a fault has occurred (albeit slowly diverging from that condition), the choice of reconfiguration mechanism should ensure the controller gains are updated and the system divergent behaviour is arrested.

5.1.3 Outline

This chapter is organised as follows: Section 5.1 introduces the concept of the integration within the context of FTCSs. Section 5.2 introduces an approach for quantifying the degree of system degradation due to incipient faults, known as the time-difference-of-arrival method. Section 5.3 describes the neural network approach for the development of an FDD framework with integrated RC requirements. The associated results of such a method is given in Section 5.3.2. Section 5.4 describes the reconfigurable controller development using the extremum seeking control technique and integrated FDD requirements. The associated results are presented in Section 5.4.6.

5.2 Time-Difference-of-Arrival Fault Detection

5.2.1 Application of the time-difference-of-arrival process

Given that the underlying describing factor for incipient faults are a function of time [118], an FDD framework compatible for such fault types, should be formulated based on the analysis of time-series data. Time-difference-of-arrival (Time Difference of Arrival (TDOA)) measurements, involve the comparison of two or more signal sources through the assumption that their position and the transmission signal speed is known to the receiver. Alternatively, if the signal has an associated timestamp, an analytical expression of their differences can be described [119].

TDOA can be classified as a data-driven method which can be applied by using the emission of signals from multiple satellites to perform geo-location. Moreover, TDOA is a frequently used method in locating signal sources without *a-priori* knowledge of the state of the source [120]. The effectiveness of TDOA has also been tested as part of investigating the indoor navigation of unmanned systems without the reliance on GPS receivers [121]. A stocktaking warehouse was installed with transceivers which would communicate with the ones on-board thereby enabling the precise navigation of the flight system while performing object avoidance.

By injecting a pulse width modulated (PWM) signal $\Delta\omega_{m_i}$ in a sequential manner and with an associated timestamp in each motor of a quadcopter configuration, as shown in Figure 5.1, a localisation scheme can be developed using the TDOA method. Each motor acts as an emitter (where motor commanded PWM is used) and the receiver of each emission is the inertial measurement unit (IMU) (where angular rates, body-fixed velocities are used) such that:

$$S_i = [s_0, s_1, ..., s_n] \tag{5.1}$$

where n represents the number of samples collected during the excitation of the ith rotor with a PWM manoeuvre $\Delta\omega_m$. Ensuring that such a manoeuvre results in an optimal energy content and the description of its implementation in the Ardupilot software is described in [3].

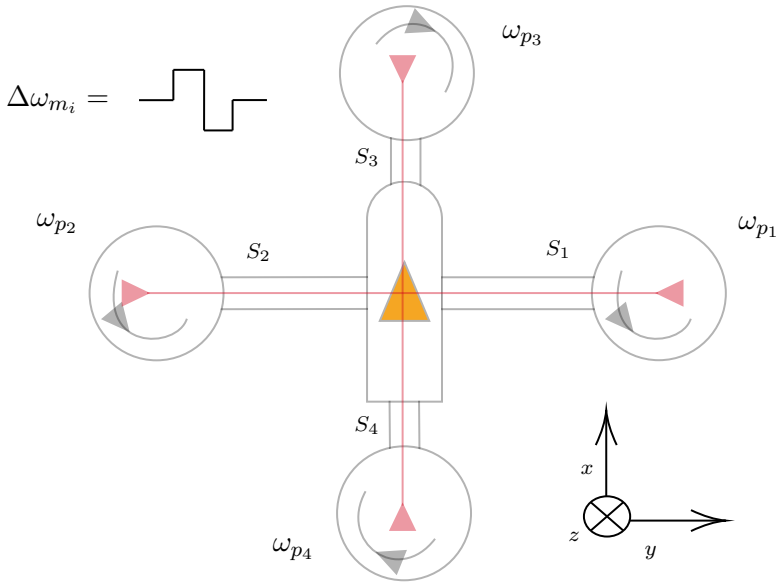


Figure 5.1: TDOA concept for a quadcopter configuration.

The TDOA relies on the premise that two emitting sources have been recorded and their respective timestamps (time of arrival or time of departure) are available and the time delay is the difference between the two timestamps. A TDOA square full-rank matrix $\overline{Q}$ (given $m = n$), can be defined as:

$$\overline{Q} = \begin{bmatrix} S_{11} & S_{12} & \dots & S_{1n} \\ S_{21} & S_{22} & \dots & S_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ S_{m1} & S_{m2} & \dots & S_{mn} \end{bmatrix} \tag{5.2}$$

where S_{mn} is the time difference cross-correlation between two signals as a function of the number of samples lagging or leading the reference signal defined as:

$$S_{mn} = \begin{cases} Z(P_m, E_n) & m = n \\ Z(P_m, P_n) & m \neq n \end{cases} \tag{5.3}$$

where $Z(P_m, E_n)$ represents the cross-correlation time delay output between the predicted NN output of the m^{th} motor P_m and the estimated NN output of the n^{th} motor E_n. The

diagonal of matrix $\overline{Q}$ represents the time delay cross-correlation when $m = n$. This is defined as:

$$Z(P_m, E_n) = \underset{t \in \Re}{\arg\max} \int_0^T P_m(t) E_n(t+\tau) dt \tag{5.4}$$

For the condition where $m \neq n$ can also be deduced:

$$Z(P_m, P_n) = \underset{t \in \Re}{\arg\max} \int_0^T P_m(t) P_n(t+\tau) dt \tag{5.5}$$

where the time sample delay factor τ ranges between $[-T; T]$ based on the following constraint:

$$P_n(t+\tau) = \begin{cases} 0 & \tau \leqq t \\ P_n(t+\tau) & \tau > t \end{cases} \tag{5.6}$$

At each timestep τ and for the $m = n$ condition, the cross-correlation is defined as:

$$P_m(t)E_n(t+\tau) = \frac{\sum_{t=0}^{T} \left(P_m(t) - \overline{P_m(t)}\right)\left(E_n(t+\tau) - \overline{E_n(t)}\right)}{\sum_{t=0}^{T} \left(P_m(t) - \overline{P_m(t)}\right)^2 \sum_{t=0}^{T} \left(E_n(t) - \overline{E_n(t)}\right)^2} \tag{5.7}$$

where $\overline{P_m(t)}$ and $\overline{E_n(t)}$ are the mean values. The localisation of the fault can be achieved through the index of $\overline{Q}$ column with the highest variance, defined as α, once the rank of $\overline{Q}$ is at least $n - 2$. Such an index is written as:

$$\alpha = \arg\max_{i \in n} \sigma_S^2(i) \tag{5.8}$$

$$\sigma_S^2(i) = \frac{1}{m} \sum_{z=1}^{m} \left(S_{zi} - \overline{S_i}\right) \tag{5.9}$$

where the variance of the ith column for the $\overline{Q}$ matrix is denoted as $\sigma_S^2(i)$. The mean value and variance of the ith column is defined as $\overline{S_i}$ and $\sigma_S^2(i)$. To ensure the reconfiguration time is maximised, initial FDD information can be provided through fault localisation without the $\overline{Q}$ matrix being full rank. In addition, a level of uncertainty can be defined and associated with the estimated fault location. The impact of such information uncertainties will be analysed in section 5.2.2.

5.2.2 Requirements from a RC

As introduced in Section 5.1.2, the augmented control of the remaining healthy rotors of a post-fault control system while the impact on system stability and performance is minimised,

is the primary objective of the RC [122]. Given the increased transient behaviour of a post-fault model with multiple actuators, a control allocation-based reconfiguration scheme has the following integration requirements:

- Maximise the reconfiguration time to ensure robustness against fault location and fault magnitude uncertainties.
- Minimise the fault detection errors for optimal post-fault system performance.

To comply with the above requirements, the following mechanism has been developed. Given a faulty nonlinear system:

$$\dot{\mathbf{x}} = H_f(\mathbf{x}, \mathbf{u}) \tag{5.10}$$

$$\mathbf{y} = G_f(\mathbf{x}) \tag{5.11}$$

where $\mathbf{x} \in \mathbb{R}^n$ is the state vector, $\mathbf{y} \in \mathbb{R}^n$ is the output vector, $\mathbf{u} \in \mathbb{R}^n$ is the input vector and G_f, and H_f are unknown nonlinear functions describing the system faulty dynamics. A fault magnitude uncertainty factor $\mathbf{F}$ associated with the fault localisation index α can be defined as:

$$\mathbf{F} = \begin{cases} 0 & \operatorname{rank} \overline{Q} < n-2 \\ 1 - e^{-\left(\eta \frac{F_\alpha}{\operatorname{rank} \overline{Q}}\right)} & \operatorname{rank} \overline{Q} \geqq n-2 \end{cases} \tag{5.12}$$

where

$$F_\alpha = \overline{S}_\alpha \tag{5.13}$$

$\overline{S}_\alpha$ is the mean value of the αth column $\overline{Q}$. Provided the minimum rank of $\overline{Q}$ matrix has been achieved, a user-defined parameter $0 < \eta < 1$ can be defined such that $\mathbf{F}$ is close to zero when the $\overline{Q}$ matrix is close to full rank or the maximum variance column α is close to zero (no fault has occurred). With the correct selection of a suitable controller reconfiguration method, the above integration requirements can be achieved.

5.3 Integrated Neural Network-Based Fault Detection and Diagnosis Technique

5.3.1 Modified continuous forward algorithm

The integration of the TDOA method described in Section 5.2.1 with a RC requires that the cross-correlation time delay to be computed in real-time. This enforces a constraint on the cross-correlated estimated signal (E_m) and cross-correlated predicted signal (P_m) to be computed without *a-priori* knowledge of the pre-fault system behaviour. The effectiveness of the CFA combined with a RBFNN architecture has been shown in Section 4.3. Given that real-time actuator dynamics data can be generated through system identification manoeuvres as described in Section 3.3.6, the function approximation capability of the neural networks can be used as a basis for real-time FDD using the TDOA approach.

However, the CFA makes use of a conjugate gradient descent approach by computing the gradient of the cost function net contribution $\nabla\Delta\mathbf{J}_{k+1}(\omega)$ for each adjustable parameter at a particular RBFNN node. Moreover, this is based on the maximum value of the cost function net contribution $\Delta\mathbf{J}_{k+1}(\omega)$ which is obtained by a line search procedure. In order to minimise memory storage and computational complexity within a real-time discrete signal environment, the CFA was modified by incorporating a line search function suitable for real-time computation.

Golden section search with multimodal capability

As described in the previous section, the maximisation of the cost function net contribution, defined as $\Delta\mathbf{J}_{k+1}(\omega)$, will be achieved by finding the optimal value of $\omega_{k+1}^{(p)}$ through a line search procedure [4]. If a function $f(x)$ is defined as unimodal, the maximum value of such a function $\texttt{max}\ f(x*)$ exists between a specified interval $[a, b]$. Such a maximisation problem can be solved by the Golden section search (GSS) algorithm by finding a subset interval $[x_1, x_2]$. If $f(x_1) < f(x_2)$ the bracketing of triplet points is (x_1, x_2, b) with the following update step:

$$b = x_2 \quad x_2 = x_1 \quad x_1 = a + (1 - \tau)(b - a) \tag{5.14}$$

if $f(x_1) > f(x_2)$, then the new bracketing triplet is (a, x_1, x_2) [123] such that:

$$a = x_1 \quad x_1 = x_2 \quad x_2 = a + \tau(b - a) \tag{5.15}$$

where τ is the *Golden ratio* defined as:

$$\tau = \frac{\sqrt{5} - 1}{2} = .618 \tag{5.16}$$

However, nonlinearities are introduced during the CFA implementation given the following division protection:

$$\Delta \mathbf{J}_{k+1}(\omega) = \begin{cases} 0 & D(\omega) < 1e-5 \\ C^2(\omega)/D(\omega) & D(\omega) > 1e-5 \end{cases} \tag{5.17}$$

These results in a multimodal function with the Golden search algorithm alone will fail to consistently find the global maximum. This issue is mitigated by combining the above with a meta-heuristic search algorithm where $\omega = f(\lambda)$ such that:

$$\lambda = \begin{cases} \pm \frac{Sp_H - Sp_L}{8Sp_i} e^{\frac{Sp_c - 1}{6}} & k > 0 \\ \pm 4^k \frac{Sp_H - Sp_L}{8Sp_i} Sp_c & k < 1 \end{cases} \tag{5.18}$$

where Sp_H, Sp_L, and Sp_i are the search space upper limit, lower limit and number of allowed iterations respectively. Sp_c is the search counter, k is the hidden layer node. The update of the GSS algorithm limits $[a, b]$ can then be computed:

$$a(\lambda) = \begin{cases} -\frac{Sp_H - Sp_L}{8Sp_i} e^{\frac{Sp_c + 3}{6}} & k > 0 \\ \lambda - 3 \cdot 4^k \frac{Sp_H - Sp_L}{8Sp_i} & k < 1 \end{cases} \tag{5.19}$$

$$b(\lambda) = \begin{cases} -\frac{Sp_H - Sp_L}{8Sp_i} e^{\frac{Sp_c - 5}{6}} & k > 0 \\ \lambda + 3 \cdot 4^k \frac{Sp_H - Sp_L}{8Sp_i} & k < 1 \end{cases} \tag{5.20}$$

A peak finder algorithm is also implemented by monitoring the line gradient:

$$\Delta f(x_1)_c = f(x_1)_c - f(x_1)_{c-1} \tag{5.21}$$

A peak is found once:

$$\Delta f(x_1)_{c-1} > 0 \quad \Delta f(x_1)_c < 0 \tag{5.22}$$

once the equivalent $\Delta f(x_2)_c$ is computed and more than one peak is found, the unimodal GSS algorithm is executed on the highest peak to compute the maximum of the function $\Delta \mathbf{J}_{k+1}(\omega)$.

Case study

The multimodal capability introduced in the GSS algorithm ensures that adequate RBFNN model estimation and prediction accuracy is achieved to ensure the TDOA cross-correlation matrix $\overline{Q}$ has minimal estimation error. Given that the computation of the cost function net contribution gradient $\nabla \Delta \mathbf{J}_{k+1}$ is not bound to a few iterations but requires the search of unknown values of the GSS interval $[a, b]$. Such computational overhead, can prevent the above method from being implementable for real-time execution. A case study of the modified CFA is analysed to determine its suitability for the proposed FDD technique.

Given the input-output mapping for CFA training shown in Figure 5.2, the initial cost function net contribution $\Delta \mathbf{J}_{k+1}$ for $k = 0$ is 1.02. This is based on the parameter set $\omega_{k+1} = [1.174, 0, 0]$. At each iteration, the peak finder algorithm searches local maximum (peak) by

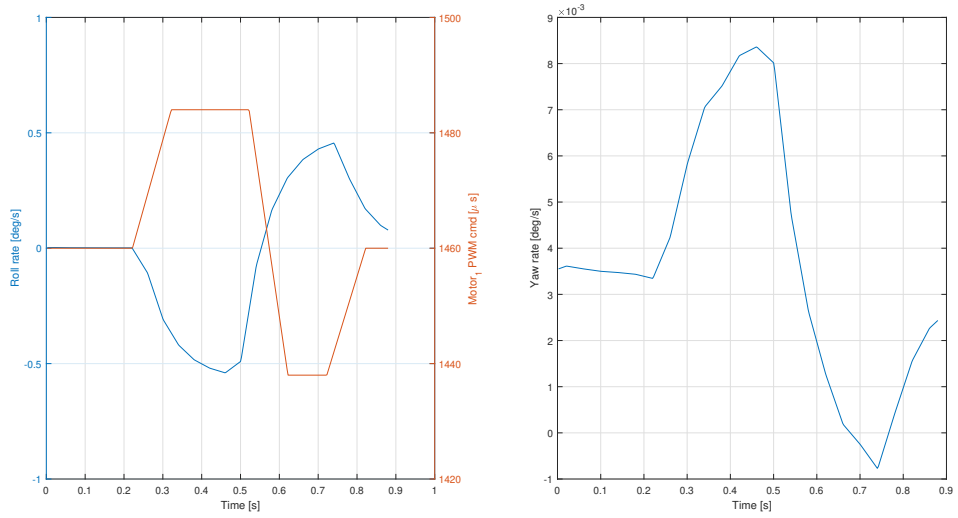


Figure 5.2: ModifiedCFA case study: Input (left) and output (right) mapping.

line gradient until a flat line is detected at both side of net contribution $\Delta \mathbf{J}_{k+1}(\lambda)$. This shown in Figure 5.3 after 43 iterations. The λ limits of $[-0.71, 1.79]$ and the highest peak of 64.56 can then used to initialise the GSS algorithm with $a = 0.21, b = 0.46$. The GSS algorithm is executed with initial conditions $x_1 = 0.3038, x_2 = 0.3628$. After 19 iterations, the $f(x_1)$ converges to 64.58 as expected. The updated regressor matrix ϕ_{k+1} results in the cost function

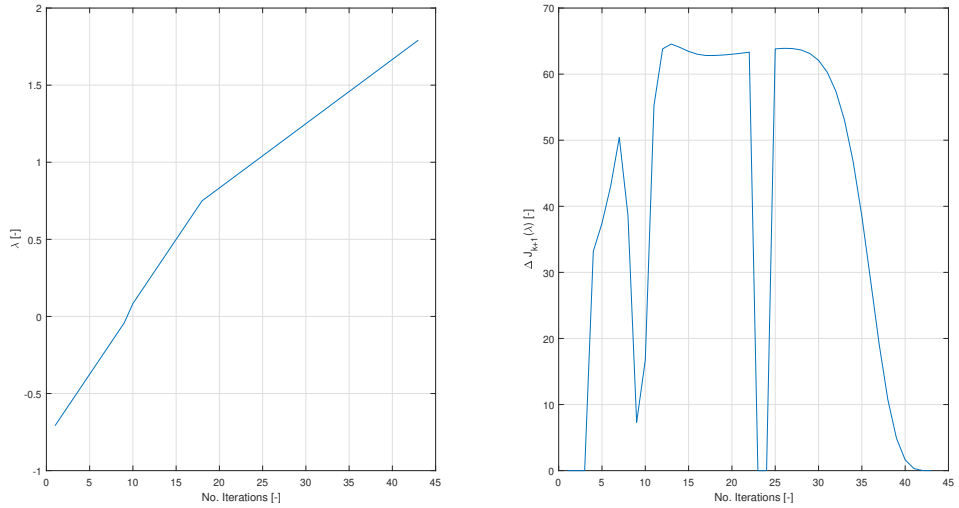


Figure 5.3: ModifiedCFA case study: Meta-heuristic search of maxima of $\Delta\mathbf{J}_{k+1}(\lambda)$ at $k = 0$.

net contribution $\Delta\mathbf{J}_{k+1}(\lambda)$ of 64.6. A second pass of the GSS algorithm is initialised with $a = -0.081, b = 0.081$. After 28 iteration, the GSS verified outputs are used to update the parameter set $\omega_{k+1} = [0.989, -1.520, 1.464]$ and trigger a hidden layer update.

For node $k = 1$, the net contribution $\Delta\mathbf{J}_{k+1}(\lambda)$ of 51.09 is found within 27 iterations after the second pass of the GSS algorithm and the meta-heuristic algorithm completing after only 7 iterations and 28 iterations for GSS outputs verification. This is shown in Figure 5.4. This resulted in parameter set $\omega_{k+1} = [-0.634, 1.981, -1.768]$ For node $k = 2$, the net contribution

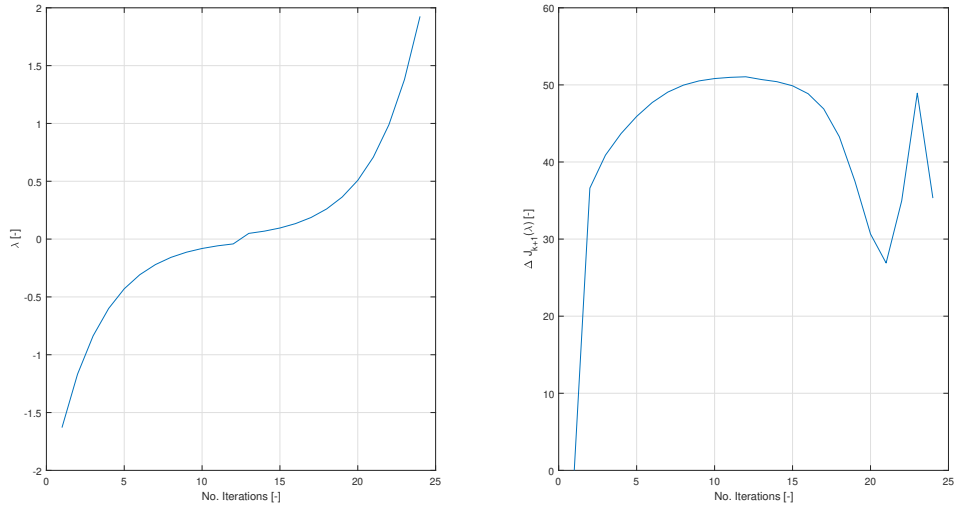


Figure 5.4: ModifiedCFA case study: Meta-heuristic search of the maxima of $\Delta\mathbf{J}_{k+1}(\lambda)$ at $k = 1$.

$\Delta\mathbf{J}_{k+1}(\lambda)$ of 0.22 is found within 31 iterations after the second pass of the GSS algorithm and

the meta-heuristic algorithm completing after only 17 iterations and 29 iterations for GSS outputs verification. This is shown in Figure 5.5. This resulted in parameter set $\omega_{k+1} = [-0.091, 1.930, -1.767]$

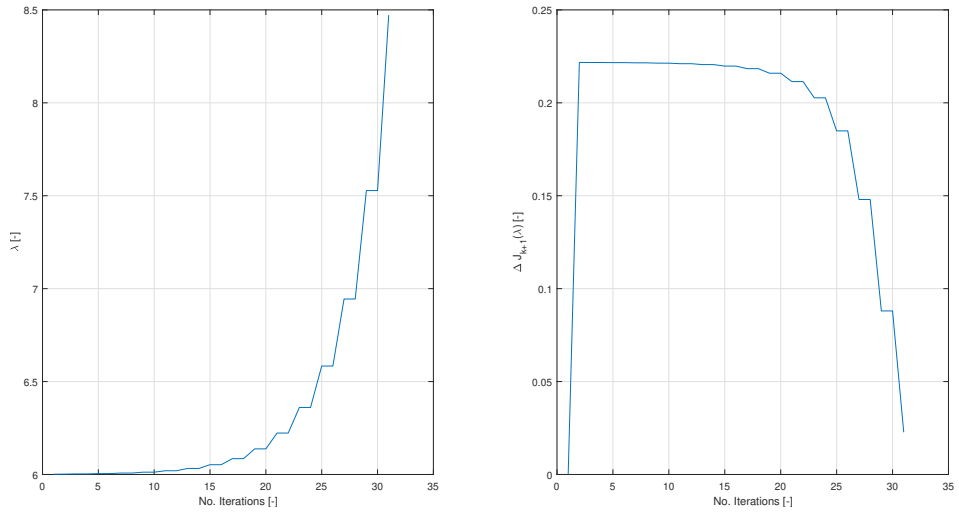


Figure 5.5: ModifiedCFA case study: Meta-heuristic search of the maxima of $\Delta\mathbf{J}_{k+1}(\lambda)$ at $k = 2$.

Based on the above case study, it can be stated that although a large amount of iterations are required, the accuracy of the model fit is increased through verification steps of the estimated CFA parameter set which ensures the RBFNN training is complete without over-fitting from the dataset and minimal residual net contribution $\Delta\mathbf{J}_{k+1}$.

5.3.2 Results

The meta-heuristic algorithm was tuned to maximise the speed of training in mind but also the accuracy of the predicted RBFNN output. A dataset of 352 points was used to generalise the identification manoeuvres. $Sp_H = 20, Sp_L = -20, Sp_i = 120$ were the chosen values most suitable for this dataset. If the search counter Sp_c reached the Sp_i, then the search was terminated irrespective of whether a maximum of the $\Delta\mathbf{J}_{k+1}(\omega)$ was found or not.

In order to validate the above FDD architecture, amending the ArduPilot software structure was required in order to implement the TDOA algorithm. Given that CFA was already validated as described in Section 4.4, an emulated version of CFA was developed and implemented in the ArduPilot code-base. This CFA emulation function has two objectives: (1)

it prevented the real-time desktop simulation from having memory overload preventing the evaluation of the TDOA algorithm; (2) it enabled the execution of the TDOA algorithm with the same frequency as the reconfiguration mechanism and control system at 400 Hz.

The modelling of the incipient fault as described in Section 3.4 was achieved within the ArduPilot servo library with user-defined activation. The system identification manoeuvres as described in Section 3.3.6 were implemented as mission configuration parameters. These were then superimposed on the servo PWM signals and injected into the flight dynamic model. For a typical mission, the fault detection manoeuvres started when the GPS speed reached 3 m/s (around 73s). An incipient fault is activated at 5 m/s (around 75s). These time events are illustrated on the PWM signals shown in Figure 5.6. In order to ensure the quadcopter was able to follow a set trajectory, after each rotor was excited, a stabilisation/recovery period followed.

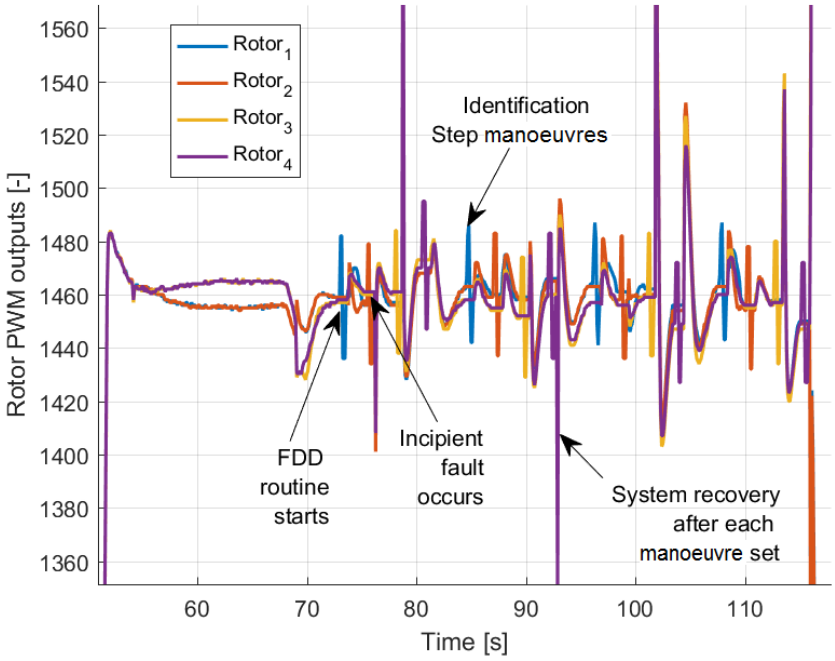
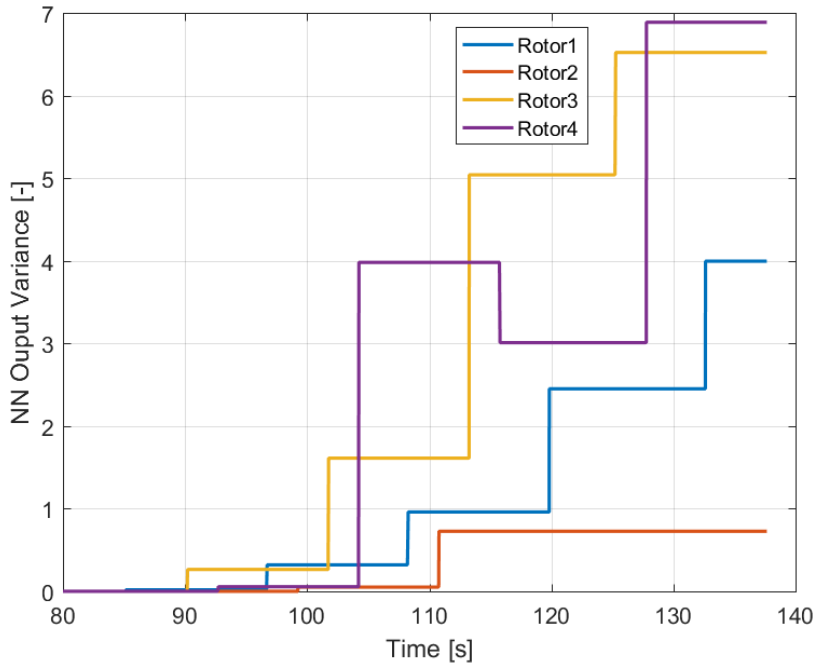


Figure 5.6: FDD algorithm - motor identification manoeuvres.

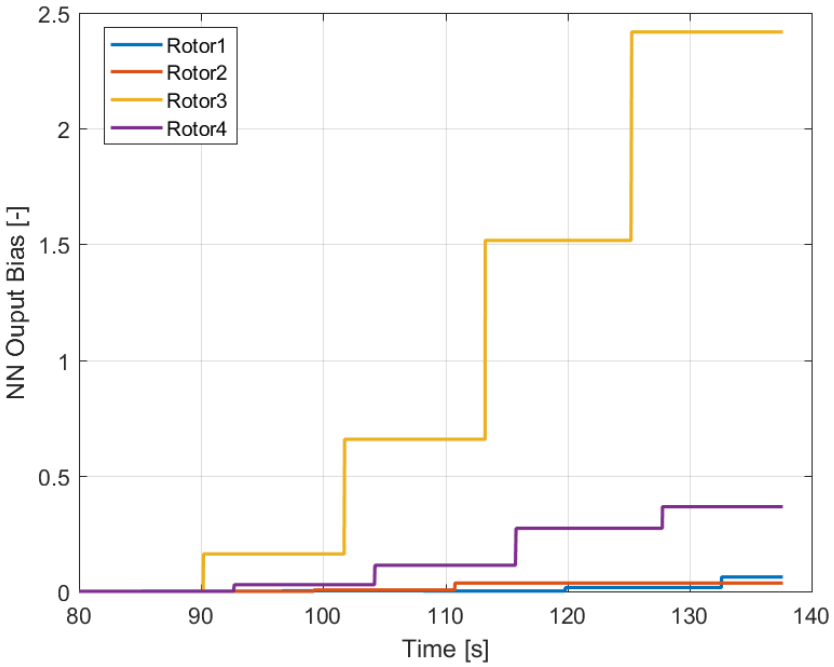
The RBFNN input was formed through the model selection technique described in Section 4.3.2. The yaw rate r was chosen as the output set, given it showed the lowest correlation with the other channels (this was analysed off-line). The predicted output of each rotor RBFNN was analysed for computing their respective variance and bias for each subsequent testing data introduced. This can be seen in Figure 5.7. The increase in variance in all the

signals is expected as the RBFNN prediction, based on current data deviates from RBFNN training/estimation data. This is due to the fact that the quadcopter flight dynamics changes with an increase in speed. This effect was verified in Section 4.3.4. The robustness of the TDOA to such RBFNN prediction uncertainties is based on analysing the time delay of predicted signals and not their magnitude.

Computation of the TDOA $\bar{Q}$ matrix and the associated column variance used for fault detection is shown in Figure 5.8. A variance of 600 for rotor 1 is reached at 90s of flight. This is as a result of the 1st column getting populated with TDOA estimates as the RBFNN for each rotor completes its training. Confirmation of the robustness of the method is shown through rotor 2 to rotor 4 having a similar variance proving that no fault has occurred. The associated uncertainty and its impact on controller reconfiguration is shown in Figure 5.9. It clearly shows that there is a balance between early detection occurring at 85s (with a low rank of $\overline{Q}$ and large uncertainty) and late detection at 50% less uncertainty but at the cost of controller re-configuring only from 92s.



(a)



(b)

Figure 5.7: FDD algorithm - RBFNN predicted output (a) variance. (b) bias.

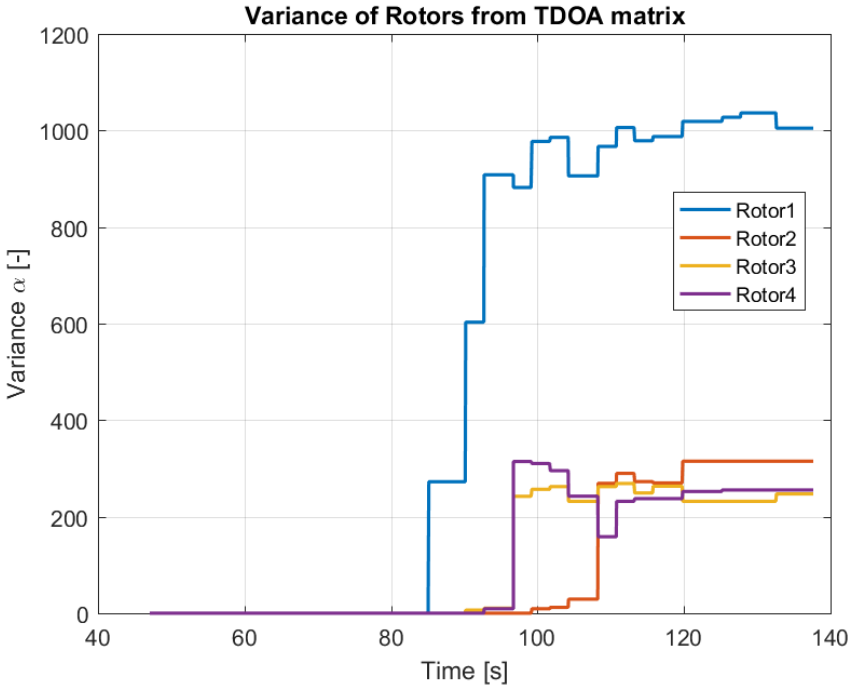
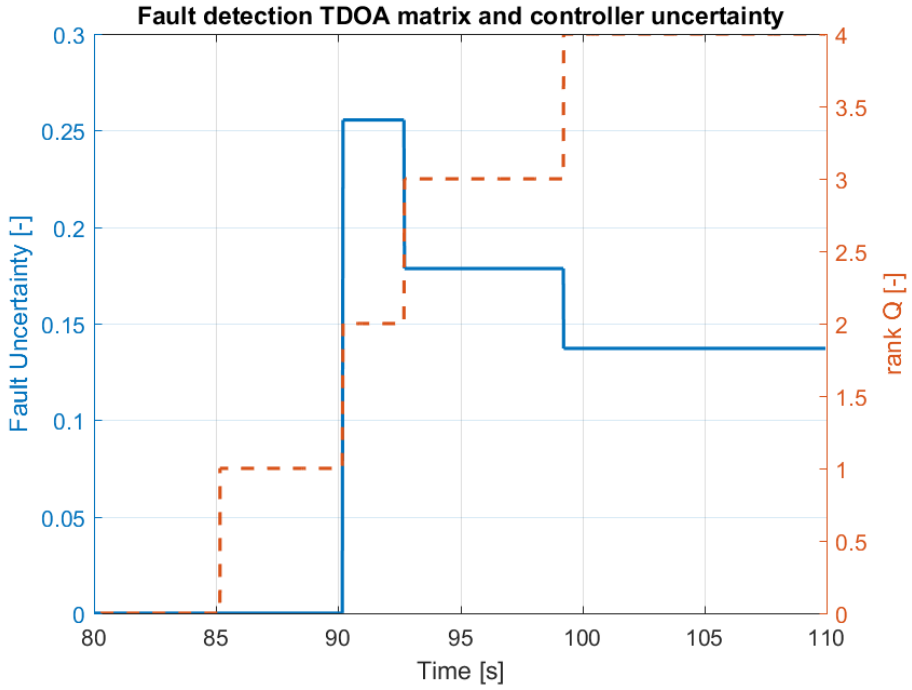
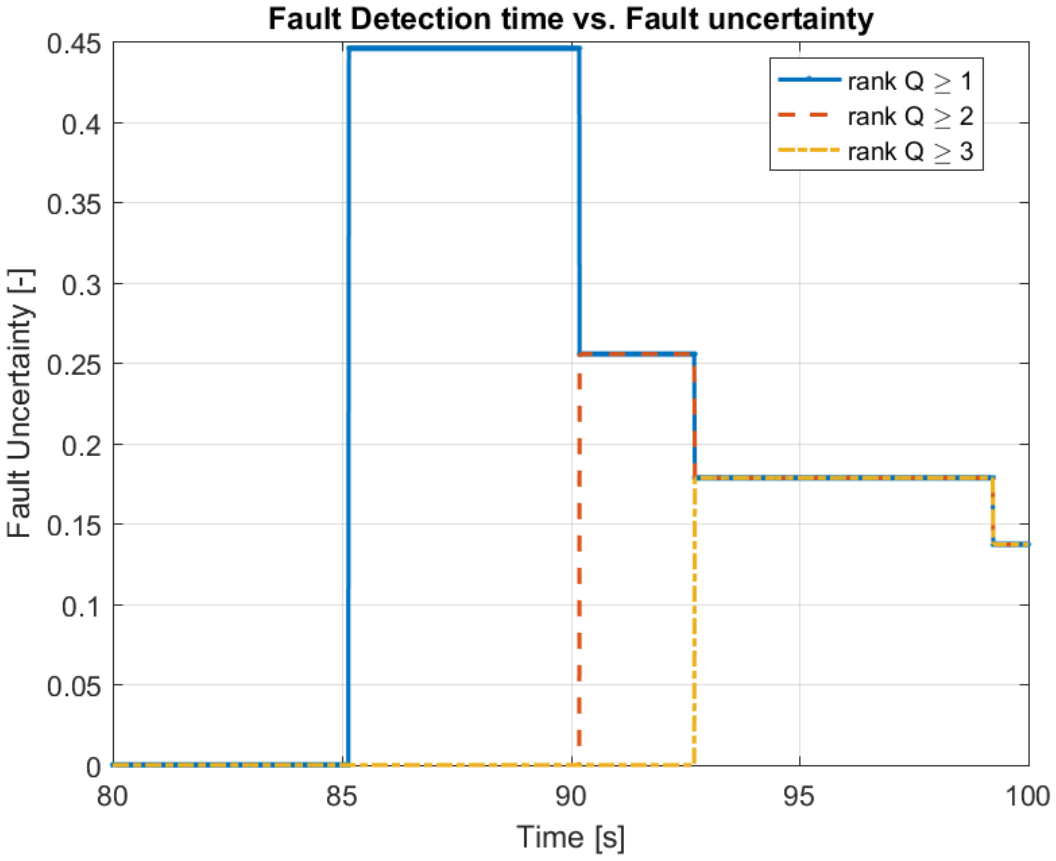


Figure 5.8: FDD algorithm performance with the $\overline{Q}$ matrix α value.



(a)



(b)

Figure 5.9: FDD algorithm - (a) fault uncertainty. (b) uncertainty sensitivity.

5.4 Extremum Seeking Algorithm for Control Reconfiguration

5.4.1 Application of an extremum seeking controller

The integration of a data-based FDD architecture, as described in Section 5.3 is well-suited provided that the controller reconfiguration can be executed without *a priori* knowledge of the post-fault model system stability and performance degradation. Popular control reconfiguration techniques such as: linear quadratic, gain scheduling, model reference adaptive control, eigenstructure assignment or dynamic inversion rely on estimating the system dynamics as a preliminary step to computing the required control action. Such an approach is sensitive to fault uncertainties and often requires high computational resources for real-time implementation [14].

Extremum seeking control is a type of adaptive control approach which drives both the input and output to their respective extrema (minimum or maximum). Its implementation and execution is model independent and relies on computing the gradient of an optimisation function based on the persistent excitation of the system dynamics [124]. The extremum seeking controller has two attractive properties: (1) it can be implemented as a control allocation which can accommodate real-time constraints and; (2) it has very low computational requirements which make it well-suited for online system reconfiguration.

Over the past decade, the stability and performance of the extremum seeking control algorithm have been investigated. The optimisation of a PID (Proportional-Integral-Derivative) controller step response of closed-loop system with an unknown plant was achieved using a discrete-time extremum seeking algorithm. It was found that through an iterative approach the extremum seeking controller could achieve both robustness and performance goals as compared to well-known methods such as Ziegler-Nichols or internal model control. It was found that such method can accommodate actuator saturation which is common in incipient faulty systems [125].

More recently, the extremum seeking controller speed of convergence was studied [126]. A major drawback of extremum seeking controller is that the reconfiguration time is directly linked to the integrator gain of the optimisation signal and the persistent signal amplitude. This can lead to large vibrations and in the context of an FDD mechanism, has a direct impact on the fault estimation accuracy. Yin et al., therefore investigated the use of the Newton-

based method instead of the gradient-based method to maximise the convergence speed while keeping the control signal oscillations small [126]. The mechanism of the extremum seeking algorithm is detailed in Section 5.4.2.

5.4.2 Extremum seeking algorithm

Given the state-space representation of a linear system:

$$\dot{\mathbf{x}} = A\mathbf{x} + B\mathbf{u} \tag{5.23}$$

$$\mathbf{y} = C\mathbf{x} + D\mathbf{u} \tag{5.24}$$

where $\mathbf{x} \in \mathbb{R}^n$ is the state vector, $\mathbf{y} \in \mathbb{R}^q$ is the output vector, $\mathbf{u} \in \mathbb{R}^p$ is the input vector and $A_{n\times n}$, $B_{n\times p}$, $C_{q\times n}$ and $D_{q\times p}$ are state, input, output and feedthrough matrices respectively. Assuming there exists a control law, parameterised by $\theta = [\theta_1, \ldots, \theta_p]$, given as:

$$\mathbf{u}_\theta = -KH(\theta, \mathbf{F})\mathbf{x} \tag{5.25}$$

The above state-space equation can be re-written as:

$$\dot{\mathbf{x}} = A\mathbf{x} + B\mathbf{u}_\theta \tag{5.26}$$

$$\mathbf{y}_\theta = C\mathbf{x} + D\mathbf{u}_\theta \tag{5.27}$$

such that the closed-loop system:

$$(A - BKH(\theta, \mathbf{F}))\mathbf{x} = 0 \iff \mathbf{x} = L(\theta) \tag{5.28}$$

where K is the compensator, $H(\theta, \mathbf{F})$ is a control allocation based on an FDD scheme and fault uncertainty gain $\mathbf{F}$, which is unity when no fault has been detected. $L(\theta)$ is the system state equilibrium which is achieved through finding the extremum (in this case minimum) value:

$$\theta^* = \underset{\theta\in\mathbb{R}^p}{\arg\min} J(\mathbf{y}_\theta) \tag{5.29}$$

where $J(\mathbf{y}_\theta)$ is the objective function comprising of the output vector $\mathbf{y}_\theta$, such that the steady-state value of θ^* is obtained without knowledge of either system functions A, B, C and D. This minimisation feedback mechanism is known as extremum seeking [127, 124]. The extremum seeking (Extremum Seeking (ES)) framework and its effect on a closed-loop system is shown in Figure 5.10.

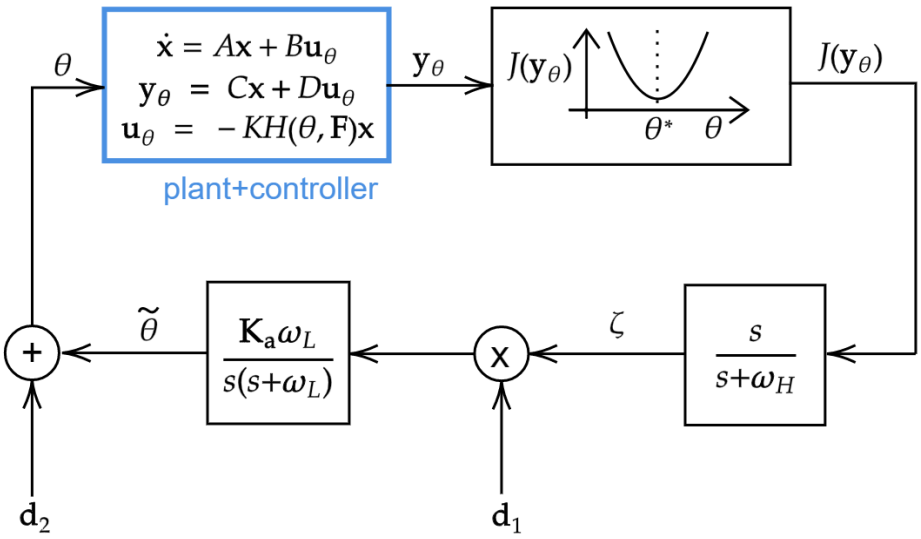


Figure 5.10: Extremum seeking control framework [127]

The ES process is to modulate the estimate $\tilde{\theta}$ with a periodic signal $\mathbf{d}_2$ such that a periodic response is created in $J(\theta)$. This is passed through a high-pass filter $s/(s+\omega_H)$ to remove the mean value. Another periodic signal $\mathbf{d}_1$, demodulates the signal ζ and get passed through a low-pass filter with integrator $\mathbf{K_a}\omega_L/(s(s+\omega_L))$ to update the estimate $\tilde{\theta}$. Given that the search is for minimisation, $\mathbf{K_a} < 0$. It should be noted that the parameter gradient, can be defined as [128]:

$$\dot{\tilde{\theta}} = \mathbf{K_a}\left(\frac{\mathbf{d}_1\mathbf{d}_2}{2}\right)\frac{\partial J}{\partial \tilde{\theta}} \tag{5.30}$$

the user-defined choice of the periodic signal vectors is as follows:

$$\mathbf{d}_1 = [D_1 \sin(\omega_1 t), ..., D_n \sin(\omega_n t)] \tag{5.31}$$

$$\mathbf{d}_2 = [E_1 \sin(\omega_1 t + \phi_1), ..., E_n \sin(\omega_n t + \phi_n)] \tag{5.32}$$

where ω_i and ϕ_i are the frequency and phase angle respectively. The magnitude for signal vectors $\mathbf{d}_1$ and $\mathbf{d}_2$ is given as $[D_1, ..., D_n]$ and $[E_1, ..., E_n]$ respectively. The adaptation gain vector $\mathbf{K_a} = [K_{a_1}, ..., K_{a_n}]$, the low-pass filter frequency vector $\omega_L = [\omega_{L_1}, ..., \omega_{L_n}]$ and the high-pass filter frequency vector $\omega_H = [\omega_{H_1}, ..., \omega_{H_n}]$, determines the speed of convergence of $\theta \rightarrow \theta^*$ and stability of the closed-loop system. Detailed analysis into the stability of the ES algorithm has been done in [128]. Incorporating the fault emulation filter G_f defined in Equation 3.50, in the system of equations results in the stability criterion:

$$(A - BG_fKH(\theta, \mathbf{F}))\mathbf{x} \leq \xi \iff \mathbf{x} = L_f(\theta) \tag{5.33}$$

where ξ is a design parameter representing the new equilibrium state described by L_f. Given $\theta \in \mathbb{R}^p$ and a real-time implementation of the ES algorithm is required for a quadcopter system, a reduction of the optimisation parameter θ search space $p \times p$, is introduced in the next section.

5.4.3 Extremum seeking simplification using geometric representation

Quadcopter control input vector $\mathbf{u}$ can be defined as:

$$\mathbf{u}_\theta = [\delta_{mot}, \delta_{lat}, \delta_{lon}, \delta_{rud}] \tag{5.34}$$

where $[\delta_{mot}, \delta_{lat}, \delta_{lon}, \delta_{rud}]$ is the heave, lateral cyclic, longitudinal cyclic and yawing command inputs respectively. Given a quadcopter '+' configuration as shown in Figure 5.11 and a weight factor based on the ES optimisation vector $P(\theta, \mathbf{F})$, the mixing to each rotor command δ_{c_i} can be defined using the right-hand rule (short-handed as P):

$$\begin{bmatrix} \delta_{mot} \\ \delta_{lat} \\ \delta_{lon} \\ \delta_{rud} \end{bmatrix} = \frac{1}{4} \begin{bmatrix} P_1 & P_2 & P_3 & P_4 \\ -2P_1 & 2P_2 & 0 & 0 \\ 0 & 0 & -2P_3 & 2P_4 \\ 2P_1 & 2P_2 & -2P_3 & -2P_4 \end{bmatrix} \begin{bmatrix} \delta_{c_1} \\ \delta_{c_2} \\ \delta_{c_3} \\ \delta_{c_4} \end{bmatrix}$$

$$\mathbf{u}_\theta = H(\theta, \mathbf{F})\mathbf{u}_c \tag{5.35}$$

where $P_i(\theta, \mathbf{F})$ is a weight factor on a motor control command δ_{c_i} and based on an FDD fault uncertainty factor $\mathbf{F}$. The optimisation parameter vector θ can be defined as:

$$\theta \triangleq [\beta, \gamma] \tag{5.36}$$

where β is the incircle radius of a geometric representation of quadrotor actuator configuration and γ is the offset of the incircle along a line of symmetry connecting the vertex location of the faulty rotor $\omega_{p_i}^{LCF}$ as shown in Figure 5.11.

Let us consider two pairs of adjacent sides of equal length (a, b). Under normal condition, the geometry is a square $[a = b]$ and once a fault is detected, the geometry is a right kite $[a_f > b_f]$ (this is a convex quadrilateral and has two opposite right angles). Considering the

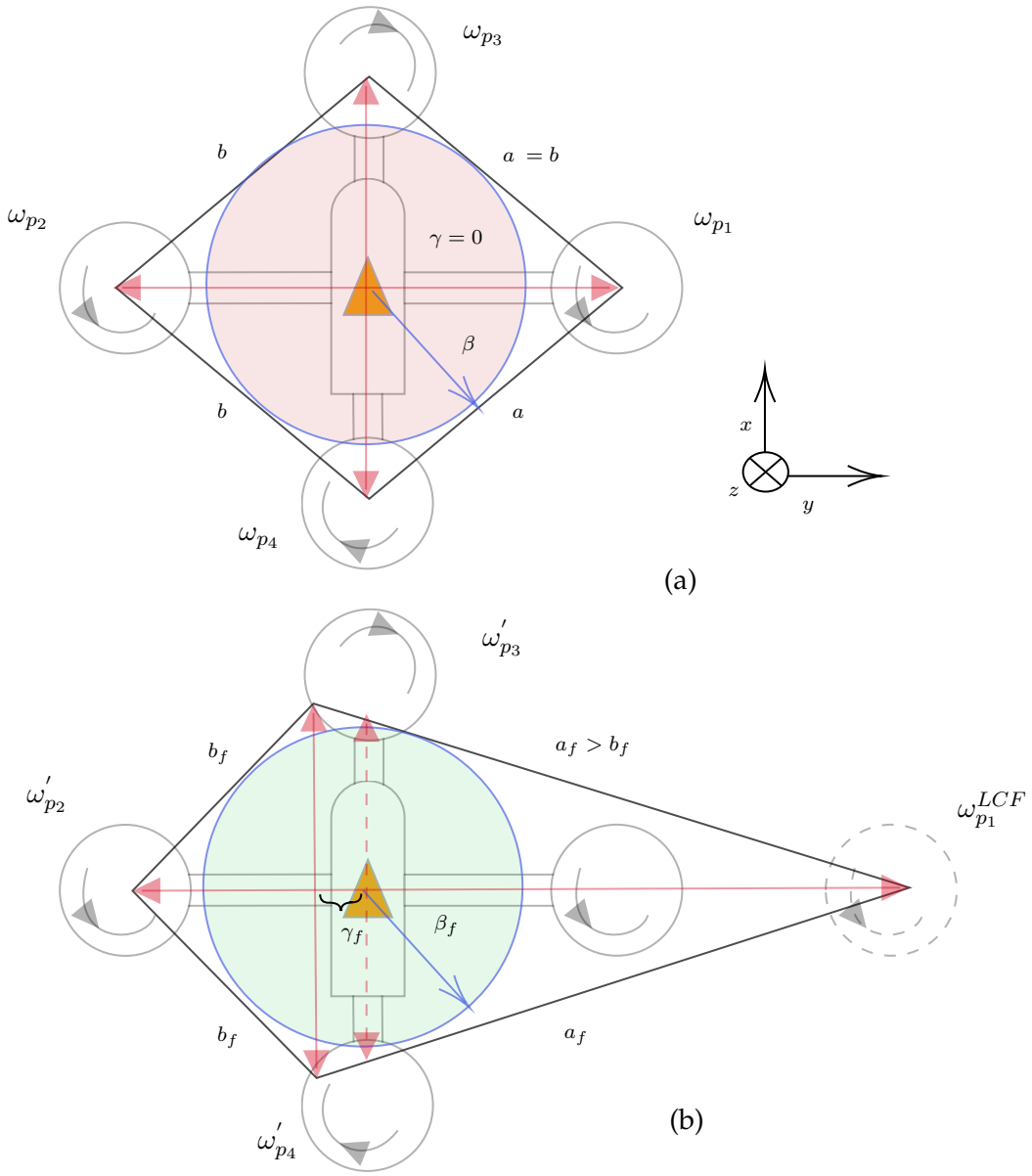


Figure 5.11: Geometric representation of normal and faulty dynamics: (a) normal case, (b) faulty case.

properties of a right kite and its circumscribed circle [129], the incircle radius and incircle offset can be described as:

$$\beta = \frac{ab}{a+b} \tag{5.37}$$

$$\gamma = \sqrt{\frac{2a^2b^2}{(a+b)^2} - \frac{a^2b^2}{(a^2+b^2)}} \tag{5.38}$$

Each of the equal-length pairs can be described in terms of the incircle parameters:

$$a = \frac{\beta\left(2\beta^2 - \gamma^2 + \gamma\sqrt{2\beta^2 - \gamma^2}\right)}{\beta^2 - \gamma^2} \tag{5.39}$$

$$b = \frac{\beta\left(2\beta^2 - \gamma^2 - \gamma\sqrt{2\beta^2 - \gamma^2}\right)}{\beta^2 - \gamma^2} \tag{5.40}$$

The weight factor on motor control command can be defined as:

$$P_i(\theta, \mathbf{F}) = \begin{cases} 1 & t < T_{detect} \\ a_f/a_0 & t \geq T_{detect} \\ b_0/b_f & t \geq T_{detect} \text{ and } \omega_{p_i} = \omega_{p_i}^{LCF} \end{cases}$$

where b_0, a_0 are the initial values of the equal-length pairs based on $\beta_0 = 0$ and $\gamma_0 = \mathbf{F}$, when $t = T_{detect}$. $\mathbf{F}$ is the fault magnitude computed by an FDD module described in Section 5.4.5. The modulation of the incircle parameters can be constructed as:

$$\beta_f = a_\beta\left(\sin\left(\frac{2\pi t}{T_\beta}\right)\right) + \overline{u}_\beta \tag{5.41}$$

$$\gamma_f = a_\gamma\left(\sin\left(\frac{2\pi t}{T_\gamma}\right)\right) + \overline{u}_\gamma \tag{5.42}$$

which makes use of a square wave modulation with period T. $\overline{u}$ is the demodulated, low-pass filtered integral of the objective function $J(\theta)$ described as:

$$\overline{u}_\beta = \beta + A_\beta\left(\sin\left(\frac{2\pi t}{T_\beta}\right)\right) \times \frac{K_{a_\beta}\omega_{F_\beta}}{s(s+\omega_{F_\beta})}\frac{s}{s+\omega_{H_\beta}}J(\theta) \tag{5.43}$$

$$\overline{u}_\gamma = \gamma + A_\gamma\left(\sin\left(\frac{2\pi t}{T_\gamma}\right)\right) \times \frac{K_{a_\gamma}\omega_{F_\gamma}}{s(s+\omega_{F_\gamma})}\frac{s}{s+\omega_{H_\gamma}}J(\theta) \tag{5.44}$$

where the amplitude of the modulation signals are defined as:

$$A_\beta = \begin{cases} 0 & t < T_{detect} \\ \sqrt{K_{a_\beta}}\overline{u}_\beta & t \geq T_{detect} \end{cases} \tag{5.45}$$

$$A_\gamma = \begin{cases} 0 & t < T_{detect} \\ \sqrt{K_{a_\gamma}}\overline{u}_\gamma & t \geq T_{detect} \end{cases} \tag{5.46}$$

Considering the rate of change of the equal-length pairs (excluding the proof):

$$\begin{aligned} \lim_{\gamma\to 0} \frac{\delta a}{\delta \beta} = \frac{\delta b}{\delta \beta} = 2 \\ \lim_{\gamma\to 0} \frac{\delta a}{\delta \gamma} = \frac{\sqrt{2\beta^2}}{\beta} \\ \lim_{\gamma\to 0} \frac{\delta b}{\delta \gamma} = -\frac{\sqrt{2\beta^2}}{\beta} \end{aligned} \tag{5.47}$$

It can be noticed that for small values of γ and a constant value of β, the effect on reducing control allocation to a faulty rotor while increasing control allocation to healthy rotors, is achieved through the tuning of the parameter γ. Consequently, this enables the ES optimisation parameter space to be further reduced to:

$$\theta \triangleq [\beta_0, \gamma] \tag{5.48}$$

where β_0 is a user-defined constant.

Formulation of $J(\mathbf{y}_\theta)$

Given the real-time computation requirement on the ES objective function $J(\mathbf{y}_\theta)$, consideration of the nominal autopilot architecture is required. The ArduCopter software was modified by implementing the ES controller and interfacing it to the main navigation routine. The ES controller was allowed to override the trajectory commands $\mathbf{T}_c$ with step functions commands $\mathbf{S}_c$ as inputs to the routine attitude controller. This is shown in Figure 5.12.

Given the attitude controller updating the motor outputs through a motor mixing routine

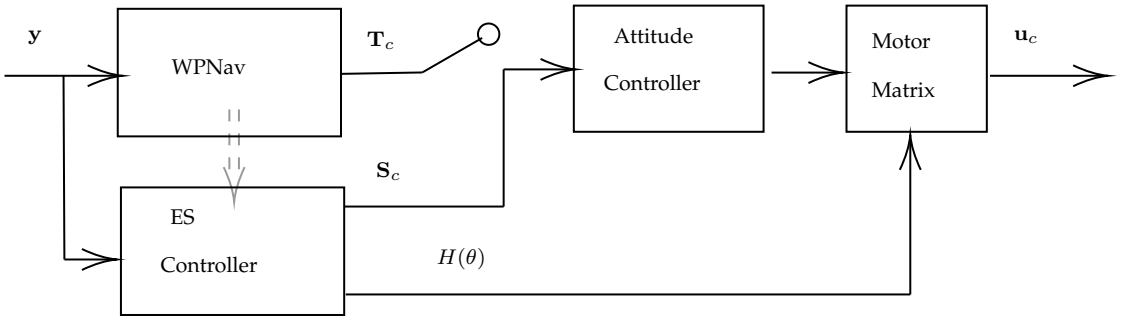


Figure 5.12: ES controller within the ArduCopter software.

every k time-steps, the objective function is defined as:

$$J(\mathbf{y}_\theta)^k = \underbrace{\int_{k-2}^{k-1} \varepsilon_S(t)dt}_{J(\mathbf{y}_\theta)_S} + \underbrace{\int_{k-1}^{k} \varepsilon_T(t)dt}_{J(\mathbf{y}_\theta)_T} \tag{5.49}$$

where,

$$\begin{aligned} \varepsilon_S(t) &= \frac{V_0}{V_k} \sum (\mathbf{S}_c - \mathbf{E}_m)^2 \\ \varepsilon_T(t) &= \frac{V_0}{V_k} \sum (\mathbf{T}_c - \mathbf{E}_m)^2 \end{aligned} \tag{5.50}$$

where V_0 and V_k are the flight speeds at $k = 0$ and k steps respectively. $\mathbf{S}_c$ is the vector of attitude step commands in Euler angles. $\mathbf{T}_c$ is the vector of trajectory commands in Euler angles. $\mathbf{E}_m$ is the vector of Euler angles measurements. At every k step, the motor weight factor matrix $H(\theta)$ was updated.

5.4.4 First-pass variance computation

Given the real-time constraint on the reconfiguration of the ES controller, a first-pass variance and mean computation scheme, based on Welford online algorithm, was implemented. Given n samples in a dataset, the sample mean $\overline{x}_n$ is given as:

$$\overline{x}_n = \overline{x}_{n-1} + \frac{x_n - \overline{x}_{n-1}}{n} \tag{5.51}$$

The sum square (also known as non-normalised variance) $S(x)$ can also be computed:

$$S(x)_n = S(x)_{n-1} \ + (x_n - \overline{x}_{n-1})\,(x_n - \overline{x}_n) \tag{5.52}$$

where the variance is given as:

$$\sigma^2 = \frac{S(x)_n}{n-1} \tag{5.53}$$

In the event a moving window variance of $m \geqslant n$ samples is required, $S(x)$ is computed as:

$$S(x)_m = S(x)_{m-1} \ + \mu\left((x_m - \overline{x}_m) + (x_{m-1} - \overline{x}_{m-1})\right) \tag{5.54}$$

$$\mu = (x_m - \overline{x}_{m-1}) \tag{5.55}$$

The variance σ^2 for the measured Euler roll, pitch and yaw angles, was used to stabilise the quadcopter controller. Passing a stability condition was required prior to start computing the step cost function $J(\mathbf{y}_\theta)_S$ ($t = k - 2$) and after the computation of the trajectory cost function $J(\mathbf{y}_\theta)_T$ ($t = k$).

5.4.5 Requirements from an FDD framework

As explained in detail in Section 5.1.1, an FDD framework has three main objectives: (1) to detect that a fault has occurred, (2) to isolate the location of the occurred fault and (3) to identify the type of fault (classification and/or magnitude) [130]. In order to reduce the time between a fault been detected and the control action being reconfigured, the reconfiguration controller needs to accommodate for fault uncertainties in the estimation of the post-fault model and should adapt to such uncertainties without further degrading system stability and performance.

Assuming the FDD mechanism is constructing a square matrix $\overline{Q}_{m\times n, m=n}$, which has the same size as the motor weight factor matrix $H(\theta)$. At every time-step it can be defined as:

$$\overline{Q} \ = \ \begin{bmatrix} S_{11} & S_{12} & \dots & S_{1m} \\ S_{21} & S_{22} & \dots & S_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ S_{n1} & S_{n2} & \dots & S_{nm} \end{bmatrix} \tag{5.56}$$

where $S_{mn} = f(\omega_{p_m}, \omega_{p_n})$ is the cross-correlated time difference response between two motors tracking a system identification δ_{c_i} input (e.g., a doublet). Relative to the objective function $J(\mathbf{y}_\theta)$ computation, S_{mn} is populated at every $p = N \cdot k$ time-steps. S_{mn} is defined as the number of samples lagging or leading the cross-correlated time difference. A fault can

be isolated by indexing the maximum variance value α of the $\overline{Q}$ matrix along its n columns, once its rank is at least $n-2$. This is defined as:

$$\alpha \ = \arg\max_{i \in n} \sigma_S^2(i) \tag{5.57}$$

where $\sigma_S^2(i)$ is the variance of the i^{th} column (representing the i^{th} faulty motor) of the $\overline{Q}$ matrix which is a square matrix containing motor fault estimations and its `rank` determines the quality of information. The fault uncertainty control allocation can then be described as:

$$\mathbf{F} = \begin{cases} 0 & \text{rank}\,\overline{Q} \ > n-2 \\ 1 - e^{-\left(\eta \frac{F_\alpha}{\text{rank}\,\overline{Q}}\right)} & \text{rank}\,\overline{Q} \ \leqq n-2 \end{cases} \tag{5.58}$$

where F_α is the average value of the α^{th} column of the absolute value of the $\overline{Q}$ matrix. η with values between 0 and 1, is the user-defined parameter such that $\mathbf{F}$ is close to zero when the $\overline{Q}$ matrix is close to full rank or the maximum variance column α is close to zero (no fault has occurred). Once the initial value of $\mathbf{F}$ is passed to the ES algorithm, the FDD is updated once the stability of the closed-loop system is achieved, through optimisation parameter $\theta \ \rightarrow \theta^{'}$ where $\theta^{'}$ is a local minimum results towards the global minimum θ^{*} as described in Section 5.4.2.

5.4.6 RFC results

The ES controller framework was implemented within the ArduPilot software-in-the-loop (SITL) framework as described in Section 3.3. Two main functions `FDD_exe` and `RFC_exe` were executed alternatively at 400 Hz. The MavProxy GUI was used for flight simulation execution and data logging. The Mission Planner software was used to do flight trajectory planning and data validation prior to MATLAB data processing and plotting.

The integration requirements imposed on the RC to incorporate fault uncertainties was validated through activating the incipient fault dynamics and executing the FDD emulation algorithm as described in 5.3.2. The requirement to execute in minimal time was achieved by implementing the ES algorithm with the online computation of the geometric representation control allocation parameters along the attitude controller, which is executed at 400Hz. The maximum sampling of 100 data points was used to compute the first-pass variance and mean values which acted as gates enabling the next objective function manoeuvre to be ex-

ecuted. In addition, in order to ensure the system waypoint navigation commands are not corrupted, the commands were inhibited during each objective function manoeuvre.

Figure 5.13 illustrates the angles tracking performance between the two phases. In order to maximise stability during the controller reconfiguration, a low-pass filter was implemented for control allocation commands of $\omega_{cmd} = 0.2rad/s$. This ensured that the update of the control allocation gains was slow enough between manoeuvres while ensuring the minimisation of the objective function. The feedback-loop filters were designed as follows: high-pass filter bandwidth $\omega_H = 2rad/s$ and low-pass filter bandwidth $\omega_F = 4rad/s$. This ratio was chosen such that high-frequency noise contained in the modulated objective function was attenuated while the steady-state value $\tilde{\theta}$ for control allocation was achieved prior to the next manoeuvre. The adaption gain $\mathbf{K_a} = 0.01$ was chosen. Given a control loop frequency of 400 Hz and the chosen value for ω_H, this value ensures that the stabilisation period was minimised prior to the subsequent manoeuvre. The demodulation and modulation signals amplitude of 0.05 and 0.01 respectively were chosen with the same frequency. Combined with the values for ω_F and ω_H, this ensured that the signal-to-noise ratio remained high enough for the minimisation of the objective function. The performance of those feedback filters is shown in Figure 5.14.

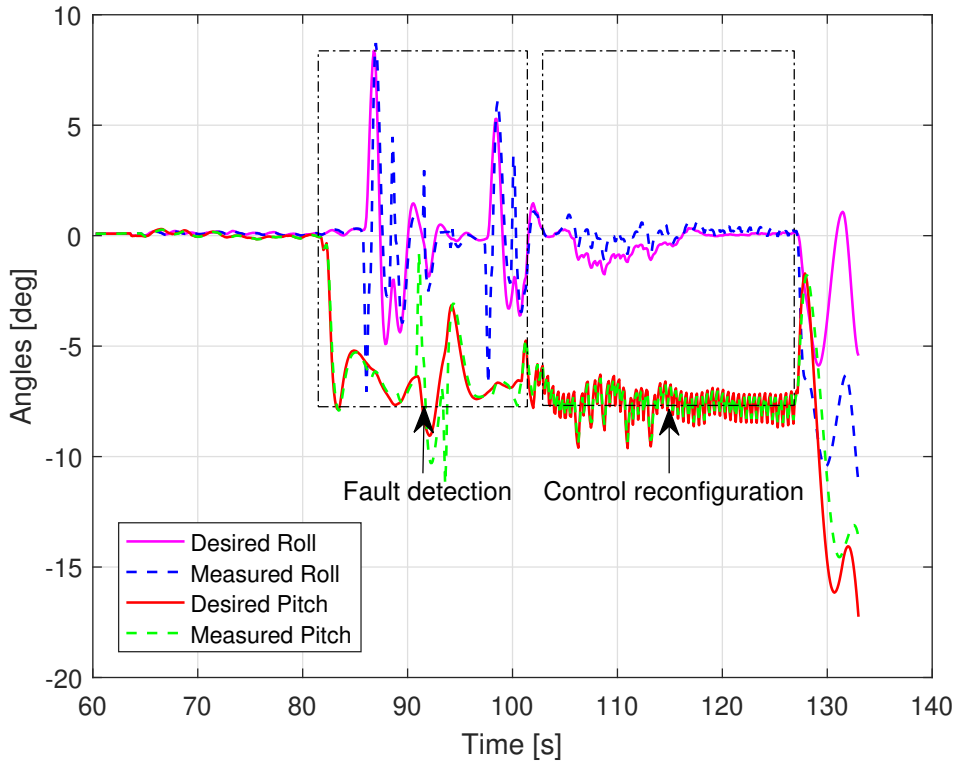


Figure 5.13: Angles tracking during fault-tolerant process.

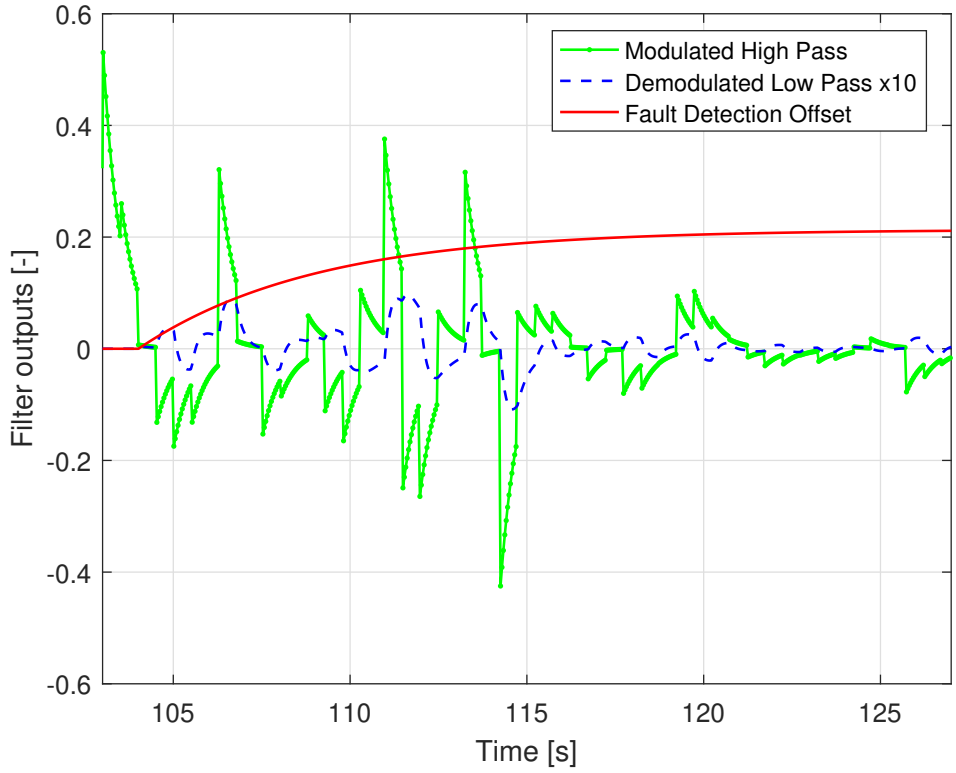


Figure 5.14: ES controller filters output.

The objective function $J(\mathbf{y}_\theta)$ and its tracking error components $J(\mathbf{y}_\theta)_S$ and $J(\mathbf{y}_\theta)_T$ are shown in Figure 5.15. It can be seen that a minimum is reached with the controller remaining stable. Given that the geometric incircle radius β set constant at 1.5, the geometric offset γ and the computed controller allocation reach an equilibrium state, which is shown in Figure 5.16(a) and (b) respectively. The contribution from the integrated modulated signal reaches a stable state which reflects robustness in the feedback loop. This stability is then proven through the waypoint navigation of the 500m by 500m mission profile. The system without an active RC, shown in Figure 5.17, becomes unstable after each sharp turn due to the incipient rotor fault degrading the performance of the default controller. This degradation is completely removed with an active FTCS shown in Figure 5.18.

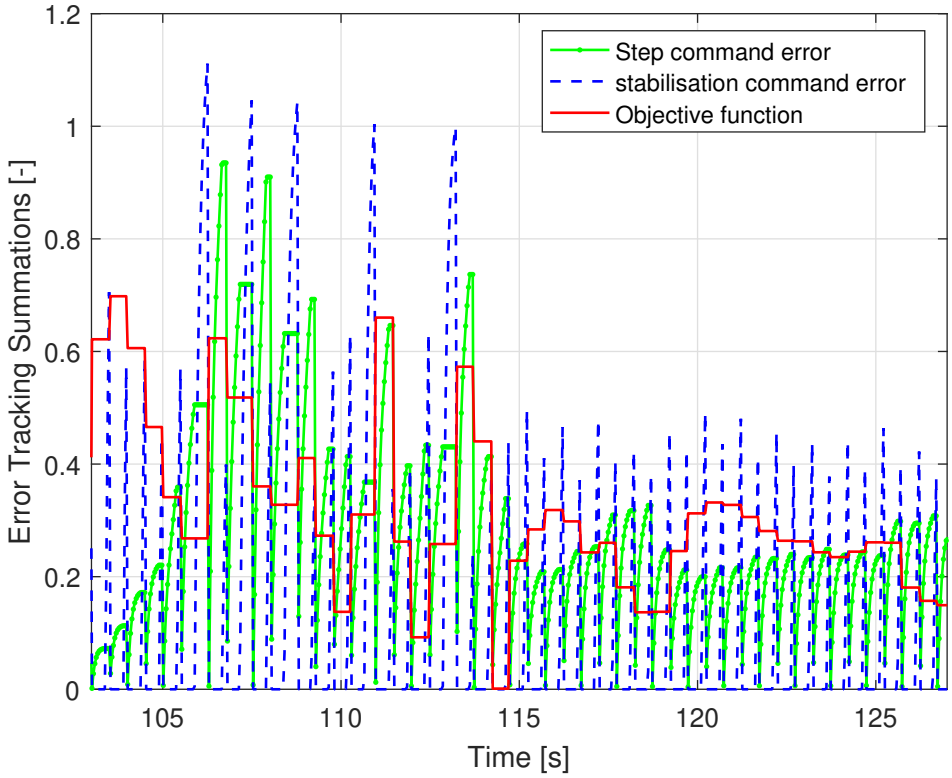
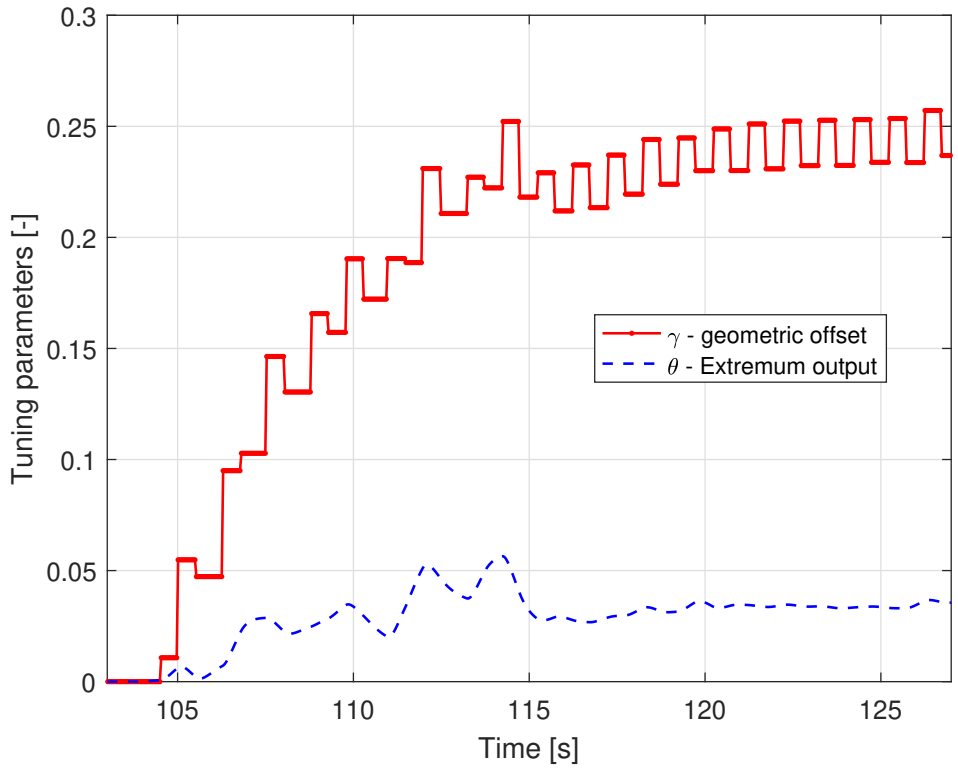
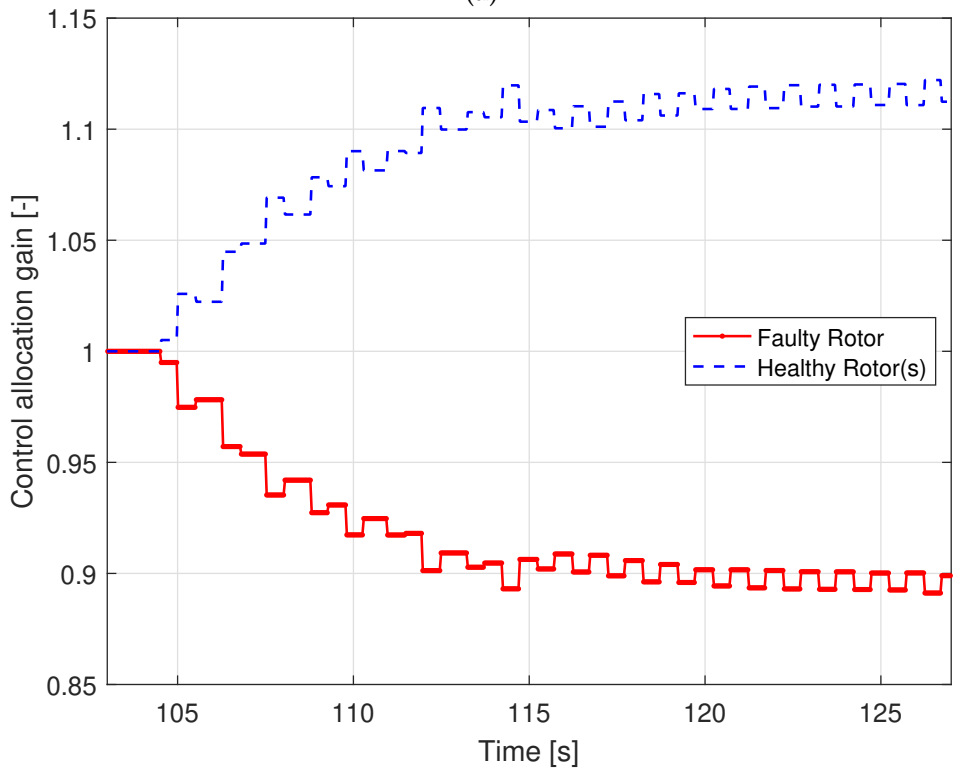


Figure 5.15: ES controller objective function.



(a)



(b)

Figure 5.16: (a) ES controller geometric offset. (b) ES controller motor allocation.

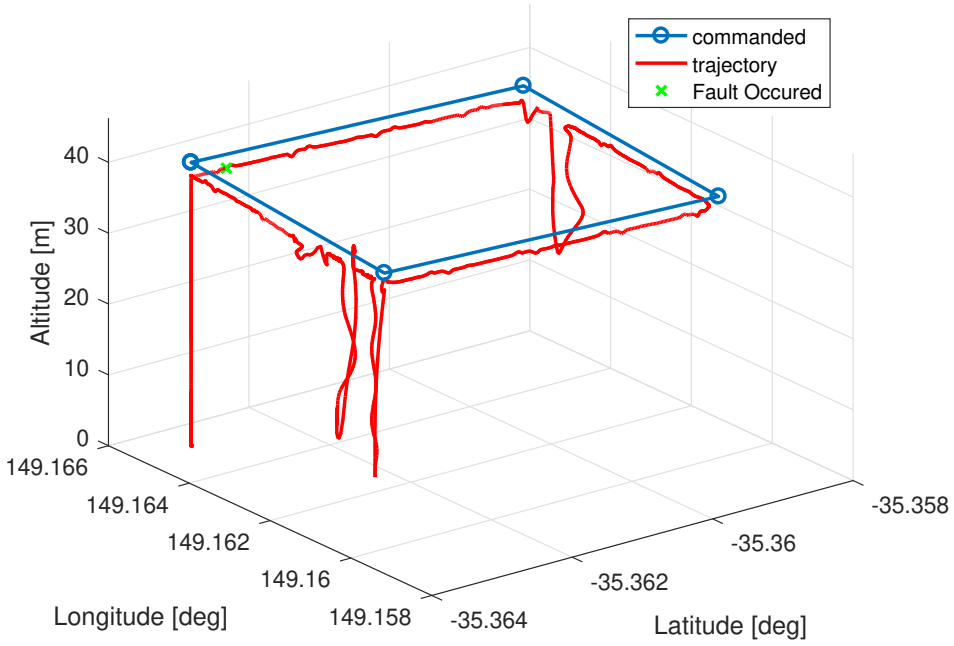


Figure 5.17: Trajectory tracking - no active FTCS.

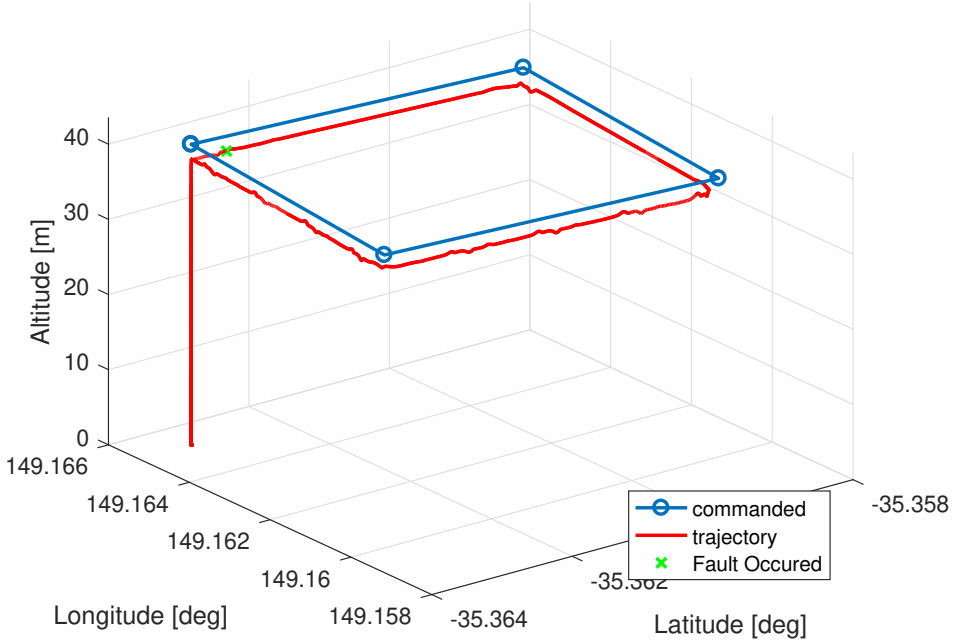


Figure 5.18: Trajectory tracking - active FTCS.

Chapter 6

Conclusions and Recommendations for Future Work

6.1 Summary

In this thesis, we have developed and experimentally tested a set of methods and tools which enabled the design and evaluation of an integrated FTCS for an unmanned rotorcraft without *a priori* knowledge of its dynamics. In Chapter 3, the development and construction of a quadcopter avionics system with the ability of performing system identification manoeuvres was achieved by modifying the flight controller software. This formed the basis of a test-bench platform capable of software-in-the-loop (SITL) simulation and the incorporation of a fault emulation framework. This framework was able to incorporate estimated models of an incipient fault condition triggered by the low-contact friction between a propeller and its motor. This approach proved invaluable in ensuring the investigation for system behaviour subject to a critical fault within a safe and repeatable environment.

Given the complex nature of a highly-coupled system behaviour under fault, a nonlinear system identification method using radial basis function neural networks (RBFNN) was investigated in Chapter 4. A comparison of learning algorithms between the orthogonal least-squares (OLS) and the continuous forward algorithm (CFA) was investigated with specific focus on their predicted robustness to signal noise and computational complexity for the implementation in a real-time environment. Although the CFA was a more elaborate method to

implement, it is based on an analytical framework for computing the RBFNN parameters as part of the regressor location which ensures its generalisation for the system dynamics while remaining robust to signal uncertainties. The verification and validation of the CFA was then achieved through the prediction of an aerodynamic parameter and successful learning and prediction of an input-output dataset within a hardware-in-the-loop (HIL) simulation environment respectively.

The development effort of any system is often underpinned by the level of emphasis on integration of its major sub-systems. Investigation of the requirements imposed on a fault detection and diagnosis (FDD) and reconfigurable controller (RC) methods is described in Chapter 5. The FDD requirement for real-time fault estimation capability was verified through the modification of the CFA line search algorithm by incorporating a Golden section search (GSS) with meta-heuristic algorithm for multimodal function minimisation. The FDD requirement of considering actuation limits was achieved with the time difference of arrival (TDOA) method. This was based on signal correlation of the actuator stimulated inputs which was the basis for developing a fault localisation scheme with associated uncertainty. The RC requirement to incorporate fault uncertainties was achieved through the development of extremum seeking (ES) algorithm with an objective function based on the geometric representation of the provided fault magnitude and fault uncertainty. Given the low-memory requirement of such a method, the RC requirement to minimise reconfiguration time was verified through SITL simulation. This integrated framework provided a method of improving system robustness through the re-allocation of the control input.

6.2 Conclusions

The development of a software-in-the-loop (SITL) dynamic model of an unmanned rotorcraft with flapping dynamics and emulated faults, for pseudo real-time desktop simulation using the ArduPilot baseline was achieved. This simulation platform was used to study the dynamic behaviour of highly nonlinear system in the presence of faults unstable from the exact data format as flight test data, ensuring that inferences from this research would be hardware-compatible. Moreover, the development of a fault-emulation framework through the system identification of a dynamic model representing the faulty behaviour of a propeller-motor slippage condition through video image processing was investigated. This

method enabled the implementation of a grey-box simulation model, by integrating empirical data obtained from an actuator subject to an incipient fault condition into a quadcopter equations of motion for the purpose of closed-loop system identification.

A meta-heuristic hybrid line search algorithm to enable the real-time implementation of the CFA in a low-memory microcontroller, for online FDD was developed. This formed part of a nonlinear system identification framework including the combination of the multi-lateration of actuator responses and the real-time learning of artificial neural networks, providing an integrated framework for control augmentation. This approach was found to be well-suited for systems with multiple actuators, enabling a control allocation scheme within the boundaries of an objective function.

The development and real-time testing of a PID reconfiguration framework based on the extremum seeking (ES) algorithm improving the probability of system preservation under the presence of unforeseen rotor faults and external disturbances was achieved. The execution and verification of such a method within a HILS environment, validated the potential of a neural network-based integrated FTCS that can function without *a-priori* knowledge of either the nominal controller robustness margins or the system physical parameters. Finally, the development of an unmanned rotorcraft avionics system for the validation of an integrated fault-tolerant control (FTC) framework comprising of a fault-emulation, real-time FDD, and control reconfiguration in the presence of actuator faults, was achieved through the manufacturing and testing of the Uav4africa H1 quadcopter.

The application of the above FTC framework for other types of faults such as: additive or multiplicative faults within various types of the systems, as listed in 1.1 can be considered. It should be noted that the TDOA method assumes a functioning sensor module which eliminates some sensor faults from being able to result in an appropriate controller reconfiguration, thereby limiting the robustness recovery for a post-fault system. The user of the above framework will have to ensure the type of fault:

- Both FDD and RC modules work in isolation from each other and can be verified within a real-time environment. This means the type of fault needs to be observable by the FDD module and system behaviour can be controllable by the RC module despite the occurred fault.
- The method of feedback for both modules to complete their tasks should be iterative

and ideally while the system is operating, thereby increasing the confidence that the system can continue to operate until such a time it can either be fixed or replaced.

6.3 Recommendations for Future Work

Given the complexity and financial budget of this research, the above conclusions excluded some investigations which have academic merit and could be part of future research activities. The list below also includes some recommendations that could enhance the impact of this research in non-aerospace applications.

- Incorporate active actuator saturation monitoring and control in a neural network-based FDD. This is to ensure that system identification estimation errors do not diverge in the event of actuator saturation. Although the actuator excitation signals are optimised for maximum energy content while ensuring small amplitude of the superimposed excitation signals, no active monitoring of actuation limits is performed. In the event the hover thrust of the actuators are close to their limits, the probability of saturation during system identification is high leading to poor NN training and consequently inactive or poor controller reconfiguration. A method of reconstructing the excitation signals in relation of the actuation limits should mitigate the occurrence of such inadvertent behaviour.

- Investigate the use of current NN-based FDD for the simultaneous estimation of the state signals used in the controller feedback and parameter values for the controller reconfiguration. This could lead the FTC framework to be able to accommodate both sensor and actuator faults. This implies that the NN architecture predicted output will also be used within a signal health monitoring system and switch signals source from measured to predicted signals in the controller feedback. The effect of NN bias and NN variance on the controller performance with and without reconfiguration will have to be quantified. The real-time implementation of multiple NN for each monitored measured signal should be investigated.

- Investigate the applicability of the FTC framework under divergent system behaviour once a fault has occurred. This will have an impact on the FDD detection time and RC reconfiguration time. An analysis of the system recovery will be able to quantify

some of the limitations of the proposed FTC framework. The premise in this thesis was that the closed-loop behaviour of the post-fault system is stable within certain bounds of the control inputs. In the event such bounds become smaller than the amplitude of the excitation signals, the system will enter an unstable state. The investigation of the above FTC behaviour in such a system state should provide valuable insight in the prediction performance of the FDD mechanism in terms of fault uncertainties and the additional reconfiguration effort needed to recover the system within bounded actuation inputs.

- Application of the above integrated FTC framework could have an impact on the structural assessment of bridges in relation to locating growing hairline cracks by analysing the measured signal correlation based on the predicted system behaviour under certain load conditions. The signal emitters could be classified in terms of traffic patterns and sensors could be placed in locations with the higher excitation content. A bridge reconfiguration controller which is able to change the stiffness of the load-bearing units based on the located fault could reduce traffic downtime due to maintenance. Moreover, a similar concept can be introduced for wind turbines whereby the excitation of each turbine blade can be used in FDD fault localisation and a reduction of its angle of attack through a reconfigurable controller could improve the overall operational life of each wind turbine station.

Bibliography

[1] Paulin Kantue and Jimoh O. Pedro. "Real-Time Identification of Faulty Systems : Development of an Aerial Platform with Emulated Rotor Faults". In: *2019 4th Conference on Control and Fault Tolerant Systems (SysTol2019)*. Casablanca, Morocco, 2019, pp. 20–25.

[2] Paulin Kantue and Jimoh O. Pedro. "Grey-box Modelling of an Unmanned Quadcopter during Aggressive Maneuvers". In: *International Conference on System Theory, Control and Computing (ICSTCC)*. Sinaia, Romania: IEEE, 2018, pp. 640–645.

[3] Paulin Kantue and Jimoh O. Pedro. "Nonlinear Identification of an Unmanned Quadcopter Rotor Dynamics using RBF Neural Networks". In: *International Conference on System Theory, Control and Computing (ICSTCC)*. Sinaia, Romania: IEEE, 2018, pp. 292–298.

[4] Paulin Kantue and Jimoh O. Pedro. "Integrated Fault Detection and Diagnosis of an Unmanned Aerial Vehicle using Time Difference of Arrival". In: *International Conference on System Theory, Control and Computing (ICSTCC)*. Sinaia, Romania: IEEE, 2020, pp. 336–342.

[5] Jean Carlo Salazar, Adrián Sanjuan, Fatiha Nejjari, and Ramon Sarrate. "Health–Aware and Fault–Tolerant Control of an Octorotor UAV System Based on Actuator Reliability". In: *International Journal of Applied Mathematics and Computer Science* 30.1 (2020), pp. 47–59.

[6] Matthew Osborne, Jennifer Lantair, Zain Shafiq, Xingyu Zhao, Valentin Robu, David Flynn, and John Perry. "UAS Operators Safety and Reliability Survey: Emerging Technologies towards the Certification of Autonomous UAS". In: *2019 4th International Conference on System Reliability and Safety, ICSRS 2019* (2019), pp. 203–212.

[7] Shruti Lahoti, Ashish Lahoti, and Osamu Saito. "Application of Unmanned Aerial Vehicle (UAV) for Urban Green Space Mapping in Urbanizing Indian Cities". In: *Unmanned Aerial Vehicle: Applications in Agriculture and Environment*. Springer International Publishing, 2020, pp. 177–188.

[8] Claudio Rosales, Carlos Miguel Soria, and Francisco G. Rossomando. "Identification and Adaptive PID Control of a Hexacopter UAV Based on Neural Networks". In: *International Journal of Adaptive Control and Signal Processing* 33.1 (2019), pp. 74–91.

[9] Ayman A El-Badawy and Mohamed A Bakr. "Quadcopter Aggressive Maneuvers along Singular Configurations: An Energy-Quaternion Based Approach". In: *Journal of Control Science and Engineering* 4 (2016).

[10] Peng Lu and Erik Jan Van Kampen. "Active Fault-Tolerant Control for Quadrotors Subjected to a Complete Rotor Failure". In: *IEEE International Conference on Intelligent Robots and Systems*. 2015, pp. 4698–4703.

[11] Douglas P Looze, Jerold L Weiss, John S Eterno, and Nancy M Barrett. "An Automatic Redesign Approach for Restructurable Control Systems". In: *IEEE Control System Magazine* 5.2 (1985), pp. 16–22.

[12] Seema Mallavalli and Afef Fekih. "A Fault Tolerant Tracking Control for a Quadrotor UAV Subject to Simultaneous Actuator Faults and Exogenous Disturbances". In: *International Journal of Control* 93.3 (2020), pp. 655–668.

[13] Arslan Ahmed Amin and Khalid Mahmood Hasan. "A Review of Fault Tolerant Control Systems: Advancements and Applications". In: *Measurement* 143.1 (2019), pp. 58–68.

[14] Youmin Zhang, A. Chamseddine, C. A. Rabbath, B. W. Gordon, C. Y. Su, S. Rakheja, C. Fulford, J. Apkarian, and P. Gosselin. "Development of Advanced FDD and FTC Techniques with Application to an Unmanned Quadrotor Helicopter Testbed". In: *Journal of the Franklin Institute* 350.9 (2013), pp. 2396–2422.

[15] Ronald C Kramer. "The Space Shuttle Challenger Explosion: A Case Study of State Corporate Crime". In: *White-collar crime reconsidered* (1992), pp. 214–243.

[16] Zhangming He, Juhui Wei, and Bowen Hou. "Detecting Incipient Faults in Quadrotor Unmanned Aerial Vehicle Based on Detrending and Denoising Techniques". In: *Proceedings of 2018 IEEE 7th Data Driven Control and Learning Systems Conference, DDCLS 2018*. IEEE, 2018, pp. 959–964.

[17] Zhangming He, Yuri A. W. Shardt, Dayi Wang, Bowen Hou, Haiyin Zhou, and Jiongqi Wang. "An Incipient Fault Detection Approach via Detrending and Denoising". In: *Control Engineering Practice* 74.1 (2018), pp. 1–12.

[18] Youmin Zhang and Jin Jiang. "Bibliographical Review on Reconfigurable Fault-Tolerant Control Systems". In: *Annual Reviews in Control* 32.2 (2008), pp. 229–252.

[19] Yishi Liu, Xiwang Dong, Qingdong Li, and Zhang Ren. "Integrated Strategy for Fault-Tolerant Flight Control System Design with Rate Saturating Actuators". In: *IEEE International Conference on Control and Automation, ICCA* 2019-July (2019), pp. 136–141.

[20] Yishi Liu, Fei Liu, Xiwang Dong, Qingdong Li, and Zhang Ren. "Integrated Fault-Tolerant Flight Control System with Actuator Amplitude Saturation". In: *Chinese Control Conference, CCC* 2019-July (2019), pp. 5118–5123.

[21] Yan Liu and Guang Hong Yang. "Integrated Design of Fault Estimation and Fault-Tolerant Control for Linear Multi-Agent Systems Using Relative Outputs". In: *Neurocomputing* 329 (2019), pp. 468–475.

[22] Xiang Yu and Jin Jiang. "A Survey of Fault-Tolerant Controllers based on Safety-Related Issues". In: *Annual Reviews in Control* 39.April (2015), pp. 46–57.

[23] Damiano Rotondo. "Advances in Gain-Scheduling and Fault Tolerant Control Techniques". PhD thesis. Universitat Politecnica de Catalunya, 2016.

[24] Rudaba Khan. "Fault Tolerant Flight Control System for Unmanned Aerial Vehicles". PhD thesis. RMIT University, 2016.

[25] Gino Iannace, Giuseppe Ciaburro, and Amelia Trematerra. "Fault Diagnosis for UAV Blades Using Artificial Neural Network". In: *Robotics* 8.3 (2019).

[26] Tong Li, Youmin Zhang, and Brandon W Gordon. "Passive and Active Nonlinear Fault-Tolerant Control of a Quadrotor Unmanned Aerial Vehicle Based on the Sliding Mode Control Technique". In: *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering* 227.1 (Jan. 2013), pp. 12–23.

[27] Chong Wu, Juntong Qi, Dalei Song, Xin Qi, and Jianda Han. "Simultaneous State and Parameter Estimation Based Actuator Fault Detection and Diagnosis for an Unmanned Helicopter". In: *International Journal of Applied Mathematics and Computer Science* 25.1 (2015), pp. 175–187.

[28] Xin Qi, Juntong Qi, Didier Theilliol, Youmin Zhang, Jianda Han, Dalei Song, and ChunSheng Hua. "A Review on Fault Diagnosis and Fault Tolerant Control Methods for Single-rotor Aerial Vehicles". In: *Journal of Intelligent & Robotic Systems* 73.1-4 (2014), pp. 535–555.

[29] Zhiwei Gao, Senior Member, Carlo Cecati, Fellow Ieee, and Steven X Ding. "A Survey of Fault Diagnosis and Fault - Tolerant Techniques Part II : Fault Diagnosis with Knowledge - Based and Hybrid / Active Approaches". In: *IEEE Transactions on Industrial Electronics* 62.6 (2015), pp. 3768–3774.

[30] J. Marzat, H. Piet-Lahanier, F. Damongeot, and E. Walter. "Model-Based Fault Diagnosis for Aerospace Systems: A Survey". In: *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering* 226.10 (2012), pp. 1329–1360.

[31] M. Hadi Amoozgar, Abbas Chamseddine, and Youmin Zhang. "Experimental Test of a Two-Stage Kalman Filter for Actuator Fault Detection and Diagnosis of an Unmanned Quadrotor Helicopter". In: *Journal of Intelligent & Robotic Systems* 70.1-4 (Apr. 2013), pp. 107–117.

[32] Xin Qi, Didier Theilliol, Juntong Qi, Youmin Zhang, Jianda Han, Dalei Song, Ling Wang, and Yong Xia. "Fault Diagnosis and Fault Tolerant Control Methods for Manned and Unmanned Helicopters: A Literature Review". In: *Conference on Control and Fault-Tolerant Systems, SysTol* (2013), pp. 132–139.

[33] T. Lombaerts. "Model-Based Fault Detection and Identification with Online Aerodynamic Model Structure Selection". In: *Progress in Flight Dynamics, GNC, and Avionics* 6.1 (2013), pp. 299–316.

[34] Lijun Liu, Yi Shen, and Earl H. Dowell. "Integrated Adaptive Fault-Tolerant H-infinity Output Feedback Control with Adaptive Fault Identification". In: *Journal of Guidance, Control, and Dynamics* 35.3 (2012), pp. 881–889.

[35] Kee-Sang Lee and Tae-Geon Park. "An Actuator Fault Reconstruction Scheme for Linear Systems". In: *Journal of Process Control* 44 (2016), pp. 106–119.

[36] H Alwi and C Edwards. "Robust Fault Reconstruction for Linear Parameter Varying Systems Using Sliding Mode Observers". In: *International Journal of Robust and Nonlinear Control* 24.14 (2014), pp. 1947–1968.

[37] Ahmad Drak, Hassan Noura, Mohammad Hejase, and Younes Al Younes. "Sensor Fault Diagnostic and Fault-Tolerant Control for the Altitude Control of a Quadrotor UAV". In: *2015 IEEE 8th GCC Conference and Exhibition, GCCCE 2015* February (2015).

[38] Jun Ma, Shi-hong Ni, Wu-jie Xie, and Wen-han Dong. "An Improved Strong Tracking Multiple-Model Adaptive Estimation: A Fast Diagnosis Algorithm for Aircraft Actuator Fault". In: *Transactions of the Institute of Measurement and Control* 38.7 (June 2015), pp. 846–854.

[39] Damiano Rotondo, Andrea Cristofaro, Vahid Hassani, and Tor Arne Johansen. "Icing Diagnosis in Unmanned Aerial Vehicles Using an LPV Multiple Model Estimator". In: *IFAC PapersOnLine* 50.1 (2017), pp. 5238–5243.

[40] Yujiang Zhong, Wei Zhang, and Youmin Zhang. "Sensor Fault Diagnosis for Unmanned Quadrotor Helicopter via Adaptive Two-Stage Extended Kalman Filter". In: *2017 International Conference on Sensing, Diagnostics, Prognostics, and Control (SDPC)* (2017), pp. 493–498.

[41] Alessandro Benini, Francesco Ferracuti, Andrea Monteriu, and Stephan Radensleben. "Fault Detection of a VTOL UAV Using Acceleration Measurements". In: *2019 18th European Control Conference, ECC 2019*. Institute of Electrical and Electronics Engineers Inc., June 2019, pp. 3990–3995.

[42] Guilhem Jouhet, Luis E. González-Jiménez, Marco A. Meza-Aguilar, Walter A. Mayorga-Macías, and Luis F. Luque-Vega. "Model-Based Fault Detection of Permanent Magnet Synchronous Motors of Drones Using Current Sensors". In: *New Trends in Robot Control. Studies in Systems, Decision and Control*. Vol. 270. Springer, 2020, pp. 301–318.

[43] Farzin Piltan, Hossein Rezaie, Bamdad Boroomand, and Arman Jahed. "Design Robust Backstepping on-line Tuning Feedback Linearization Control Applied to IC Engine". In: *International Journal of Advanced Science and Technology* 43 (2012).

[44] Bin Yu, Youmin Zhang, Jianguo Yan, Yaohong Qu, and Zhuang Liu. "Fault Tolerant Control using Linear Quadratic Technique against Actuator Faults in a UAV". In: *Control Conference (CCC), 2013 32nd Chinese*. IEEE, 2013, pp. 6294–6299.

[45] Moses Bangura and Robert Mahony. "Real-Time Model Predictive Control for Quadrotors". In: *IFAC Proceedings Volumes* 47.3 (2014), pp. 11773–11780.

[46] Lijun Liu, Zhen Yu, and Chunhui Zhu. "Integrated Active Fault-Tolerant H-infinity Control by Designing an Auxiliary Self-Examination Plant". In: *Proceedings - 2015 IEEE International Conference on Systems, Man, and Cybernetics, SMC 2015* (2015), pp. 234–239.

[47] Xiaohong Nian, Weiqiang Chen, Xiaoyan Chu, and Zhiwei Xu. "Robust Adaptive Fault Estimation and Fault Tolerant Control for Quadrotor Attitude Systems". In: *International Journal of Control* 93.3 (2020), pp. 725–737.

[48] David W Kun and Inseok Hwang. "Linear Matrix Inequality-Based Nonlinear Adaptive Robust Control of Quadrotor". In: *Journal of Guidance, Control, and Dynamics* 39.5 (2015), pp. 1–13.

[49] Abhinav Sinha and Rajiv Kumar Mishra. "Robust Altitude Tracking of a Miniature Helicopter UAV Based on Sliding Mode". In: *ICIIECS 2015 - 2015 IEEE International Conference on Innovations in Information, Embedded and Communication Systems*. 2015, pp. 1–6.

[50] Lei Cui, Hongyuan Liu, and Ying Yang. "Youla Parameterization Fault-Tolerant Controller Design for Aircraft". In: *Journal of Guidance, Control, and Dynamics* 36.5 (2013), pp. 1533–1538.

[51] Vilas K Jadhav. "Robust Controller Design using Quantitative Feedback Theory (QFT)". In: *International Journal of Computer Applications* 47.7 (2012), pp. 9–13.

[52] J. Lunze and J.H. Richter. "Reconfigurable Fault-Tolerant Control: A Tutorial Introduction". In: *European Journal of Control* 14.5 (2008), pp. 359–386.

[53] Marcello R Napolitano, Steve Naylor, Charles Neppach, and Van Casdorph. "On-Line Learning Nonlinear Direct Neurocontrollers for Restructurable Control Systems". In: *Journal of Guidance, Control, and Dynamics* 18.1 (1995), pp. 170–176.

[54] Taeyoung Lee and Youdan Kim. "Nonlinear Adaptive Flight Control Using Backstepping and Neural Networks Controller". In: *Journal of Guidance, Control, and Dynamics* 24.4 (2001), pp. 675–682.

[55] D. Shore and M. Bodson. "Flight Testing of a Reconfigurable Control System on an Unmanned Aircraft". In: *Proceedings of the 2004 American Control Conference* 4.4 (2004), pp. 3747–3752.

[56] Zhong-Ping Jiang and Yu Jiang. "Robust Adaptive Dynamic Programming for Linear and Nonlinear Systems: An Overview". In: *European Journal of Control* 19 (2013), pp. 417–425.

[57] Qing Lin, ZhiHao Cai, and YingXun Wang. "Design, Model and Attitude Control of a Model-scaled Gyroplane". In: *2014 IEEE Chinese Guidance, Navigation and Control Conference, CGNCC 2014*. IEEE, Aug. 2014, pp. 1282–1287.

[58] Fikret Caliskan and Chingiz Hajiyev. "Reconfigurable Control of an UAV against Sensor/Actuator Failures". In: *IFAC-PapersOnlin* 48.9 (2015), pp. 7–12.

[59] Rudaba Khan, Paul Williams, Robin Hill, and Cees Bil. "Fault Tolerant Flight Control System Design for UAV's using Nonlinear Model Predictive Control". In: *Australian Control Conference* November (2011), pp. 297–302.

[60] Xiang Yu, Zhixiang Liu, and Youmin Zhang. "Fault-Tolerant Flight Control Design with Finite-Time Adaptation under Actuator Stuck Failures". In: *IEEE Transactions on Control Systems Technology* 25.4 (2017), pp. 1431–1440.

[61] K Niki Maleki, Kaveh Ashenayi, Loyd R Hook, Justin G Fuller, and Nathan Hutchins. "A Reliable System Design for Nondeterministic Adaptive Controllers in Small UAV Autopilots". In: *Digital Avionics Systems Conference (DASC), 2016 IEEE/AIAA 35th* (2016), pp. 1–5.

[62] Duc-Tien Nguyen, David Saussié, and Lahcen Saydy. "Quaternion-based Robust Fault-Tolerant Control of a Quadrotor UAV". In: *International Conference on Unmanned Aircraft Systems*. 2017, pp. 1333–1342.

[63] X Qu, J Shi, H Zhou, L Zuo, and Y Lyu. "Reconfigurable Flight Control System Design for Blended Wing Body UAV Based on Control Allocation". In: *18th International Conference on Control, Automation and Systems (ICCAS)*. 2018, pp. 632–638.

[64] Li Yu, Guang He, Shulong Zhao, and Xiangke Wang. "Dynamic Inversion-Based Sliding Mode Control of a Tilt Tri-Rotor UAV". In: *12th Asian Control Conference (ASCC)*. 2019, pp. 1637–1642.

[65] C. N. Nett, C. A. Jacobson, and A. T. Miller. "An Integrated Approach to Controls and Diagnostics: The 4-Parameter Controller". In: *1988 American Control Conference*. 1988, pp. 824–835.

[66] Jakob Stoustrup, M.J. Grimble, and H.H. Niemann. "Design of Integrated Systems for the Control and Detection of Actuator/Sensor Faults". In: *Sensor Reviews* (1997), pp. 138–149.

[67] Youmin Zhang and Jin Jiang. "Design of Integrated Fault Detection, Diagnosis and Reconfigurable Control Systems". In: *Proceedings of the 38th IEEE Conference on Decision and Control (Cat. No.99CH36304)*. Vol. 4. IEEE, 1999, pp. 3587–3592.

[68] Youmin Zhang and Jin Jiang. "Integrated Active Fault-Tolerant Control using IMM Approach". In: *IEEE Transactions on Aerospace and Electronic Systems* 37.4 (2001), pp. 1221–1235.

[69] A Marcos and G J Balas. "A Robust Integrated Controller/Diagnosis Aircraft Application". In: *International Journal of Robust and Nonlinear Control* 15.12 (2005), pp. 531–551.

[70] Qikun Shen, Bin Jiang, Peng Shi, and Cheng Chew Lim. "Novel Neural Networks-Based Fault Tolerant Control Scheme with Fault Alarm". In: *IEEE Transactions on Cybernetics* 44.11 (2014), pp. 2190–2201.

[71] Qikun Shen, Bin Jiang, and Peng Shi. "Adaptive Fault Tolerant Control against Actuator Faults". In: *International Journal of Adaptive Control and Signal Processing* 31.2 (2017), pp. 147–162.

[72] Jianglin Lan and Ron J. Patton. "A New Strategy for Integration of Fault Estimation within Fault-Tolerant Control". In: *Automatica* 69.February (2016), pp. 48–59.

[73] Saif H Almutairi. "Optimal Fault-Tolerant Flight Control for Aircraft with Actuation Impairments". PhD thesis. Cranfield University, 2016.

[74] Jianglin Lan, Ron J. Patton, and Xiaoyuan Zhu. "Integrated Fault-Tolerant Control for a 3-DOF Helicopter with Actuator Faults and Saturation". In: *IET Control Theory and Applications* 11.14 (Sept. 2017), pp. 2232–2241.

[75] Chun Liu, Bin Jiang, and Ke Zhang. "Integrated Multiple-Model Adaptive Fault Identification and Reconfigurable Fault-Tolerant Control for Lead-Wing Close Formation Systems". In: *International Journal of Systems Science* 49.4 (Mar. 2018), pp. 701–717.

[76] Karl Frederik Prochazka, Hugo Eduardo, and Sebastian Rudolf Klein. "Integrated Fault-Tolerant Control of an Over-Actuated Aircraft Using Optimal Control Allocation and Robust Sliding Mode Observers". In: *2018 IEEE Conference on Control Technol-*

ogy and Applications. Institute of Electrical and Electronics Engineers Inc., Oct. 2018, pp. 171–178.

[77] Konrad Rudin, Guillaume J. J. Ducard, and Roland Y. Siegwart. "Active Fault Tolerant Control with Imperfect Fault Detection Information: Applications to UAVs". In: *IEEE Transactions on Aerospace and Electronic Systems* 56.4 (Aug. 2020), pp. 2792–2805.

[78] Jimoh O. Pedro and Paulin Kantue. "Online Aerodynamic Parameter Estimation of a Miniature Unmanned Helicopter Using Radial Basis Function Neural Networks". In: *ASCC 2011 - 8th Asian Control Conference*. Kaohsiung, Taiwan, 2011, pp. 1170–1175.

[79] Christopher Kurt Fourie. "The Autonomous Landing of an Unmanned Helicopter on a Moving Platform". PhD thesis. University of Stellenbosch, 2015.

[80] Lennart Ljung. *System Identification Theory for User*. Prentice Hall, 1999.

[81] Berenice Mettler, Zhaodan Kong, Chad Goerzen, and Matthew Whalley. "Benchmarking of Obstacle Field Navigation Algorithms for Autonomous Helicopters". In: *Proceedings of the 66th Annual Forum of the American Helicopter Society* (2010), pp. 1–18.

[82] M F Pairan and S S Shamsudin. "System Identification of an Unmanned Quadcopter System Using MRAN Neural". In: *IOP Conference Series: Materials Science and Engineering* 270 (2017), pp. 1–10.

[83] Bill Canis. *Unmanned Aircraft Systems (UAS): Commercial Outlook for a New Industry*. Washington: Congressional Research Service, 2015.

[84] Ravindra V Jategaonkar, Dietrich Fischenberg, and Wolfgang Gruenhagen. "Aerodynamic Modeling and System Identification From Flight Data – Recent Applications At DLR". In: *Journal of Aircraft* 41.4 (2008), pp. 681–691.

[85] Matt R. Jardin and Eric R. Mueller. "Optimized Measurements of Unmanned-Air-Vehicle Mass Moment of Inertia with a Bifilar Pendulum". In: *Journal of Aircraft* 46.3 (2009), pp. 763–775.

[86] J. B. Brandt and M. S. Selig. "Propeller Performance at Low Reynolds Numbers". In: *49th AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition*. 2011, p. 1255.

[87] Vladislav Gavrilets. "Autonomous Aerobatic Maneuvering of Miniature Helicopters". PhD thesis. Massachusetts Institute of Technology, 2003, p. 86.

[88] Vladislav Gavrilets, B Mettler, and E Feron. "Human-Inspired Control Logic for Automated Maneuvering of Miniature Helicopter". In: *Journal of Guidance, Control, and Dynamics* 27.5 (2004), pp. 752–759.

[89] Gabriel M Hoffmann, Haomiao Huang, Steven L Waslander, and Claire J Tomlin. "Precision Flight Control for A Multi-Vehicle Quadrotor Helicopter Testbed". In: *Control Engineering Practice* 19.9 (2011), pp. 1023–1036.

[90] Karl J Åström. "Control Systems with Friction". In: *Proceedings of the Fourth International Conference on Motion and Vibration Control*. 1998, pp. 25–32.

[91] S Bosnyakov, A Bykov, V Coulech, S Fonov, A Morozov, and V Moskalik. "Blade Deformation and PSP Measurements on the Large Scale Rotor by Videometric System". In: *Instrumentation in Aerospace Simulation Facilities, 1997. ICIASF '97 Record., International Congress on* (1997), pp. 95–104.

[92] Huan Ming Wen and Xiao Hong Ma. "Continuous Rotation Angles Measurement Using Video Frame Images". In: *2010 International Conference on Multimedia Technology, ICMT 2010* (2010).

[93] Peng Cheng, Mohamed Sobh Mohamed Mustafa, and Bengt Oelmann. "Contactless Rotor RPM Measurement Using Laser Mouse Sensors". In: *IEEE Transactions on Instrumentation and Measurement* (2012).

[94] Fredrik Orderud. "Comparison of Kalman Filter Estimation Approaches for State Space Models with Nonlinear Measurements". In: *Proceedings of Scandinavian Conference on Simulation and Modeling*. Vol. 194. 7491. 2005, pp. 157–162.

[95] Lennart Ljung. *Experiments with Identification of Continuous Time Models*. Vol. 15. Linköping University Electronic Press, 2009, pp. 1175–1180.

[96] H Garnier, M Mensler, and A Richard. "Continuous-Time Model Identification from Sampled Data : Implementation Issues and Performance Evaluation". In: *International Journal of Control* 76.13 (2003), pp. 1337–1357.

[97] Bernard Mettler. *Identification Modeling and Characteristics of Miniature Rotorcraft*. Ed. by Springer. Springer Science & Business Media, 2013.

[98] Jonas Sjoberg, Qinghua Zhang, Lennart Ljung, Albert Benveniste, Bernard Deylon, Pierre-yves Glorennec, Hakan Hjalmarsson, and Anatoli Juditsky. "Nonlinear Black-Box Modeling in System Identification : a Unified Overview". In: *Automatica* 31.12 (1995), pp. 1691–1724.

[99] Urban Forssell and Lennart Ljung. "Closed-loop Identification Revisited". In: *Automatica* 35.7 (1999), pp. 1215–1241.

[100] Reinout Romijn, Leyla Özkan, Siep Weiland, Jobert Ludlage, and Wolfgang Marquardt. "A grey-box modeling approach for the reduction of nonlinear systems". In: *Journal of Process Control* 18.9 (2008), pp. 906–914.

[101] Qianying Li. *Grey-Box System Identification of a Quadrotor Unmanned Aerial Vehicle (Msc. Dissertation)*. 2014.

[102] Jabran Ahsan, Mansoor Ahsan, Anis Jamil, and Ahsan Ali. "Grey Box Modeling of Lateral-Directional Dynamics of a UAV through System Identification". In: *Proceedings - 14th International Conference on Frontiers of Information Technology, FIT 2016* (2017), pp. 324–329.

[103] Marcus Back. "Grey-box Modelling of a Quadrotor Using Closed-loop Data". In: *Linkopings University* (2015).

[104] Long Zhang, Kang Li, and Er-Wei Bai. "A New Extension of Newton Algorithm for Nonlinear System Modelling Using RBF Neural Networks". In: *IEEE Transactions on Automatic Control* 58.11 (2013), pp. 2929–2933.

[105] Hong Gui Han and Jun Fei Qiao. "Adaptive Computation Algorithm for RBF Neural Network". In: *IEEE Transactions on Neural Networks and Learning Systems* 23.2 (2012), pp. 342–347.

[106] Songwu Lu and Tamer Basar. "Robust Nonlinear System Identification Using Neural Network Models". In: *IEEE Transactions on Neural Networks* 9.3 (1998), pp. 1–24.

[107] Sheng Chen, Colin F N Cowan, and Peter M Grant. "Orthogonal Least Squares Learning Algorithm for Radial Basis Function Networks". In: *IEEE Transactions on Neural Networks* 2 (1991), pp. 302–309.

[108] Jian-xun Peng, Kang Li, and George W Irwin. "A Novel Continuous Forward Algorithm for RBF Neural Modelling". In: *IEEE Transactions on Automatic Control* 52.1 (2007), pp. 117–122.

[109] Kang Li, Jian-xun Peng, and George W Irwin. "A Fast Nonlinear Model Identification Method". In: *IEEE Transactions on Automatic Control* 50.8 (2005), pp. 1211–1216.

[110] Paulin Kantue. "Online aerodynamic parameter estimation of a miniature unmanned helicopter using radial basis function neural networks (Msc. Dissertation)". In: *University of the Witwatersrand* (2011), pp. 1–196.

[111] S. Singh and A.K. Ghosh. "Improved Delta Method for Parameter Estimation From Real-Flight Data Using Neural Networks". In: *IFAC Proceedings Volumes*. Vol. 40. 7. IFAC, 2007, pp. 200–205.

[112] Mark B. Tischler. *System Identification Methods for Aircraft Flight Control Development and Validation*. Tech. rep. August. NASA, 1995.

[113] Daniel J Inman and Ramesh Chandra Singh. *Engineering Vibration*. 4th. Prentice Hall, 2013.

[114] Youmin Zhang and Jin Jiang. "Issues on Integration of Fault Diagnosis and Reconfigurable Control in Active Fault-Tolerant Control Systems". In: *IFAC Proceedings Volumes* 39.13 (2006), pp. 1437–1448.

[115] Ralph R. Young. *The Requirements Engineering Handbook*. First. Artech House, 2004.

[116] Mojtaba Hashemi, Ali Kamali Egoli, and Mahyar Naraghi. "Integrated Fault Tolerant Control for Saturated Systems with Additive Faults: A Comparative Study of Saturation Models". In: *International Journal of Control, Automation and Systems* 17.4 (2019), pp. 1019–1030.

[117] Jinhua Fan, Youmin Zhang, and Zhiqiang Zheng. "Adaptive Observer-Based Integrated Fault Diagnosis and Fault-Tolerant Control Systems Against Actuator Faults and Saturation". In: *Journal of Dynamic Systems, Measurement, and Control* 135.4 (2013), p. 041008.

[118] Van Mien, Hee Jun Kang, and Young Shick Ro. "A Robust Detection and Isolation Scheme for Incipient and Abrupt Faults in Robot Manipulator Using Neural Network". In: *Proceedings of the 6th International Forum on Strategic Technology, IFOST 2011* 1.2 (2011), pp. 313–316.

[119] Simon Shuster, Andrew J. Sinclair, and T. Alan Lovell. "Initial Relative-Orbit Determination Using Heterogeneous TDOA". In: *IEEE Aerospace Conference Proceedings* (2017), pp. 1–7.

[120] Xin Chen, Ding Wang, Jiexin Yin, and Changgui Jia. "Augmented Lagrange Geolocation Algorithm using TDOA Measurements and Calibration Sources in the Presence of Satellite Position Errors". In: *AEU - International Journal of Electronics and Communications* 111 (2019).

[121] Janis Tiemann and Christian Wietfeld. "Scalable and Precise Multi-UAV Indoor Navigation Using TDOA-based UWB Localization". In: *2017 International Conference on Indoor Positioning and Indoor Navigation, IPIN 2017*. 2017, pp. 1–7.

[122] Liu Xulin and Guo Yuying. "Fault Tolerant Control of a Quadrotor UAV Using Control Allocation". In: *Proceedings of the 30th Chinese Control and Decision Conference, CCDC 2018* 621000.1 (2018), pp. 1818–1824.

[123] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes -The Art of Scientific Computing*. Third. Cambridge University Press, 2007.

[124] Libin Wang, Songlin Chen, and Kemao Ma. "On Stability and Application of Extremum Seeking Control without Steady-State Oscillation". In: *Automatica* 68 (2016), pp. 18–26.

[125] Nick J. Killingsworth and Miroslav Krstic. "PID Tuning Using Extremum Seeking: Online, Model-Free Performance Optimization". In: *IEEE Control System Magazine* 26.1 (2006), pp. 70–79.

[126] Chun Yin, Shanshan Wu, Shiwei Zhou, Jiuwen Cao, Xuegang Huang, and Yuhua Cheng. "Design and Stability Analysis of Multivariate Extremum Seeking with Newton Method". In: *Journal of the Franklin Institute* 355 (2018), pp. 1559–1578.

[127] Chun Yin, Sara Dadras, Xuegang Huang, Yuhua Cheng, and Hadi Malek. "The Design and Performance Analysis of Multivariate Fractional-Order Gradient-Based Extremum Seeking Approach". In: *Applied Mathematical Modelling* 62 (2018), pp. 680–700.

[128] Miroslav Krstic and Hsin-Hsiung Wang. "Stability of Extremum Seeking Feedback for General Nonlinear Dynamic Systems". In: *Automatica* 36.1 (2000), pp. 595–601.

[129] Martin Josefsson. "More Characterizations of Tangential Quadrilaterals". In: *Forum Geometricorum* 11 (2011), pp. 65–82.

[130] Noura Mouhssine, M Nabil Kabbaj, Mohammed Benbrahim, and Chakib El Bekkali. "Quadrotor Fault Detection and Isolation Based on Nonlinear Analytical Redundancy Relations". In: *International Multi-Conference on Systems, Signals & Devices*. 2017, pp. 325–330.