Thèse présentée pour obtenir le grade universitaire de docteur

Discipline : MATHEMATIQUES ET INFORMATIQUE
Spécialité : Automatique

Abdel-Razzak MERHEB

Diagnostic and Fault-Tolerant Control Applied to an Unmanned Aerial Vehicle

Diagnostic et Tolérance aux Fautes Appliqués à un Drone

Soutenue le 05/12/2016 devant le jury :

Jean-Christophe PONSART Professeur Rapporteur
Mohammed CHADLI Maître de Conférences Rapporteur
Mustapha OULADSINE Professeur Examinateur
Ahmed EL HAJJAJI Professeur Examinateur
Thomas RAKOTOMAMONJY Docteur Examinateur
Hassan NOURA Professeur Directeur de thèse
François BATEMAN Docteur Co-Directeur de thèse
Cette œuvre est mise à disposition selon les termes de la Licence Creative Commons Attribution - Pas d’Utilisation Commerciale - Pas de Modification 3.0 France.
Acquire knowledge and impart it to the people

Prophet Muhammad (PBUH)

Al-Tirmidhi, Hadith 107
This work is dedicated to mom and dad,

IBTIHAJ and ABDALLAH

my wife,

MAYA

and my children,

ABDOUSSALAM and MARYAM
Abstract

In this thesis, the design of Fault Tolerant Controllers for rotating wing Unmanned Aerial Vehicles is interpreted.

Unmanned Aerial Vehicles (UAV) are more and more popular for their civil and military applications. They are designed to easily move in different environments to achieve specific tasks autonomously. For many years, several researchers have developed nominal control laws that allow the autonomous flight of the UAV. However, classical control laws usually show weaknesses in the presence of parameter uncertainties, environmental disturbances (e.g. wind and ground effect), and if a fault occurs on the embedded sensors or on the actuators. Actually, such a fault may lead to the crash of the UAV causing financial losses and putting people at risk if the UAV crashes in populated areas. Therefore, it is judicious to design a control law capable of stabilizing the UAV not only in the fault-free nominal cases, but also in the presence of disturbances and faults.

In this thesis, a new bio-inspired search algorithm called Ecological Systems Algorithm (ESA) suitable for engineering optimization problems is developed. ESA imitates natural ecological rules to find iteratively the optimum of a given function, and is shown to be suitable for both static and dynamic search environments. ESA is used over the thesis to find the optimal gains for the fault tolerant controllers designed for UAVs.

Sliding Mode Control theory is used to develop two Passive Fault Tolerant Controllers (PFTC) for quadrotor UAVs. The first controller is a regular Sliding Mode Controller (SMC) augmented with an integrator, and the second type is a Cascaded Sliding Mode Controller with an inner and outer SMC loops. The controllers have demonstrated their effectiveness by tolerating high amount of partial loss of different motor speeds. Moreover, it is shown that tuning controllers using a faulty quadrotor system improves further their ability to afford more severe faults.

Because PFTCs have restricted robustness and can handle a few number of faults, an Active Fault Tolerant Controller (AFTC) using Kalman Filter (KF) based Fault Detection and Isolation (FDI) unit and Sliding Mode Control (SMC) is developed for quadrotor UAVs. The fault here is estimated online using the FDI unit, and the reconfigurable sliding mode controller is updated accordingly.

Active and Passive Fault Tolerant Controllers play an important role in under-actuated systems’ safety, but they are useless in case of severe faults or complete
failure of actuators. To overcome harsh situations, an emergency controller is
developed for quadrotor UAVs suffering a severe fault or a failure in one of their
motors. The emergency controller is based on the Quadrotor-to-Trirotor conver-
sion maneuver, where the damaged motor of a quadrotor is stopped, and the
quadrotor continues its path as a trirotor. The Fault Diagnosis and Identification
(FDI) unit that estimates online severe faults and failures is based on Kalman
Filter estimations. With the emergency controller, it becomes possible for the
quadrotor to continue following its path despite the total failure of one motor.

The PFTC, AFTC, and Trirotor conversion maneuver (emergency controller)
developed for quadrotor UAVs are then integrated to form a powerful fault to-
lerant controller that is able to handle a wide number of faults. The Integrated
Fault Tolerant Controller (IFTC) possesses a fault severity based decision station
that activates the suitable controller based on fault information provided by The
FDI unit. The new controller not only ensures fault tolerance, but also saves
actuator resources, and processor computational effort.

The lack of actuator redundancy in quadrotors restricts the potential of fault
tolerant controllers. This yields to the development of hexarotors and octorotors
that have more payload and are able to handle multiple severe faults and actu-
ator failures. In this thesis, Passive and Active Fault Tolerant Controllers are de-
dsigned for octorotor UAVs based on First Order and Second Order Sliding Mode
Control (FOSMC, SOSMC). Super Twisting Algorithm (STA) is applied to form
the discontinuous part of the SOSMC. The AFTC uses Control Allocation methods
to redistribute the control effort among healthy actuators reducing the effect of
fault. Two control allocation methods are used and compared, Dynamic Control
Allocation and Pseudo-Inverse Control Allocation. Dynamic Control Allocation is
seen to be able to compensate for the effect of multiple severe faults while taking
into account actuator and time constraints. On the other hand, Pseudo-Inverse
Control Allocation is straightforward and effective, but it reduces the applicabi-

ty and capabilities of the controller.

**Keywords**: Fault Tolerant Control, UAV control, Sliding Mode Control, Kalman
Filter, Fault Detection and Identification, Ecological Systems Algorithm, Emergency
Controller, Control Allocation, Quadrotor, Octorotor.
Résumé

Les travaux de recherches sur la commande, le diagnostic et la tolérance aux défauts appliqués aux drones deviennent de plus en plus populaires ; et ce du fait du large domaine d'applications civiles et militaires de ces aéronefs. Ces avions sans pilote sont conçus afin de pouvoir voler dans des conditions opérationnelles très variées tout en accomplissant leurs tâches de manière autonome.

Les commandes nominales généralement implémentées sur ce type de systèmes ne sont pas performantes en présence de fortes perturbations ou de défaillances affectant les capteurs ou les actionneurs embarqués sur le drone. En particulier, l'apparition de défauts peut conduire à la perte du drone ; cela met en danger les personnes et les biens et engendre des pertes d'argent. Il paraît donc judicieux de concevoir des lois de commande qui garantissent la stabilité et les performances du drone, non seulement dans le cas nominal, mais également en présence de défauts.

Dans cette thèse, un nouvel algorithme bio-inspiré adapté pour la recherche de solutions dans des problèmes d'optimisation est développé. L'Algorithme des Systèmes Ecologiques (ESA) imite les règles écologiques pour trouver, de manière itérative, le maximum d'une fonction. Cet algorithme peut être utilisé dans des environnements de recherches statiques et dynamiques. Il est utilisé dans notre thèse pour trouver les gains des différents contrôleurs tolérants aux défauts conçus pour les drones.

La commande par mode glissant est ensuite utilisée pour développer deux contrôleurs passifs tolérants aux défauts (PFTC) pour les quadrirotors. Le premier contrôleur est une commande par mode glissant augmentée avec un intégrateur, tandis que le second est un contrôleur par mode glissant implémenté en cascade comprenant une boucle interne et une boucle externe.

Parce que les commandes passives tolérantes aux défauts ont une robustesse réduite et parce qu'elles ne peuvent gérer qu'un petit nombre de défauts, une commande active par mode glissant utilisant un filtre de Kalman est développée pour les quadrirotors. Le défaut est estimé en temps réel et la commande est reconfigurée en conséquence.

Les contrôleurs tolérants aux défauts, actifs et passifs, jouent un rôle important dans la sécurité des systèmes sous-actionnés, mais ils sont inutiles en cas de défauts graves ou de perte complète des actionneurs. Pour traiter ces situations extrêmes, un contrôleur d'urgence est développé pour les quadrirotors soumis à
de tels défauts. Le dispositif de commande d’urgence est basé sur la conversion du quadrirotor en trirotor, le moteur endommagé du quadrirotor est stoppé, et le quadrirotor poursuit son vol configuré en trirotor. L’unité de diagnostic et d’identification des défauts qui estime en ligne les défauts et les défaillances graves est basée sur les estimations d’un filtre de Kalman. Avec ce dispositif, il devient possible pour le quadrirotor de suivre sa trajectoire même dans le cas de la perte totale d’un moteur.

Les commandes actives, passives, et la conversion quadrirotor-trirotor (contrôleur d’urgence) développées pour les quadrirotors sont ensuite intégrées pour former un contrôleur tolérant aux défauts puissant qui est capable de gérer un grand nombre de défaillances. Le contrôleur tolérant aux défauts intégré possède un module de décision basée sur la gravité du défaut. Le nouveau contrôleur assure l’accommodation aux défauts tout en garantissant les ressources actionneur et en limitant la charge de calcul du processeur.

Le manque de redondance d’actionneurs dans les quadrirotors restreint l’efficacité des contrôleurs tolérants aux défauts. Pour pallier ces insuffisances, les drones à voilure tournante tels que les hexarotors et les octorotors sont capables de gérer des situations de défaut plus sévères tout en embarquant des charges utiles plus lourdes. Des contrôleurs tolérants aux défauts, actifs et passifs, basés sur des méthodes par mode glissant du premier ordre et du deuxième ordre sont développées pour les octorotors. L’algorithme dit de Super Twisting est utilisé pour générer la partie discontinue de la commande du deuxième ordre par mode glissant. La commande tolérante aux défauts active utilise des méthodes d’allocation de contrôles pour redistribuer les efforts sur les actionneurs sains, réduisant ainsi l’effet du défaut. Deux méthodes d’allocation de commandes sont utilisées et comparées, l’allocation de contrôle dynamique et une méthode de redistribution basée sur la commande pseudo-inverse. L’allocation de contrôle dynamique est en mesure de compenser les effets de plusieurs défauts graves tout en tenant compte des contraintes de temps et celles sur les actionneurs.

**Mots clés :** Commandes tolérantes aux défauts, Contrôle des drones, Commande par mode glissant, Filtre de Kalman, Diagnostic des défauts, Algorithme des Systèmes Écologiques, Contrôleur d’urgence, Allocation des commandes, Quadrirotor, Octorotor.
Acknowledgments

The author believes that this part is much more difficult than writing the whole thesis itself. It is here where one stands at acknowledging the people with whom he spent most of his days during his work. I find it really a kind of an impossible mission to thank individually all the people who contributed in the preparation of this thesis, either directly or indirectly.

It is a great honor for me to express my sincere appreciation to my thesis supervisors Prof. Dr. Hassan Noura and Dr. François Bateman for their support, guidance, and helpful suggestions during my research work. I would like also to thank my previous advisors, Prof. Dr. Osman Parlaktuna and Prof. Dr. Veysel Gazi for their guidance during my Bachelor and Masters degrees.

I wish to express my sincere appreciation to all the members of my family for their infinite encouragement, support and help during my study. Mom, dad, my brother, my sister, and my dear wife have done their best. They have sacrificed a lot of their comfort and time supporting and encouraging me. I would like to thank them for their infinite moral support and care. Never forgot my lovely children for their nice existence in my life.

I would also like to thank deeply the dean of the Engineering school at the Lebanese International University (LIU) Dr. Amin Hajj-Ali, the chairman of the Electrical and Electronics Engineering department Dr. Hassan Bazzi, Engineering lab chief MEng. Nahida Abdullah and all my colleagues for their help and support. Not forgetting LIU Tripoli campus managers Dr. Georges Matta and Dr. Ahmad Ahdab for their support and understand.
## Contents

Abstract 6

Résumé 8

Acknowledgments 9

Figure list 13

Table list 18

Introduction 19

0.1 Unmanned Aerial Vehicles 19

0.2 Fault, Failures, and Fault Tolerant Control 22

0.2.1 Fault Tolerant Control (FTC) 23

0.2.2 Fault Tolerant Control Applied to UAVs 23

0.3 Structure of the Thesis 30

1 Ecological Systems Algorithm With Application on Quadrotor Controller Tuning 32

1.1 Ecological Systems Algorithm 32

1.2 Quadrotor Model and Controller Design 38

1.3 Tuning of Controller Gains 42

1.4 Genetic Algorithms Based Tuning 44

1.5 Bacterial Foraging Algorithm 49

1.6 ESA Based Tuning Algorithm 57

1.7 Result Analysis 61

2 Passive and Active Fault Tolerant Control of Quadrotor UAVs 63

2.1 State Space Representation of Pelican Quadrotor 63

2.2 Passive FTC Sliding Mode Controller Design 65

2.2.1 Regular SMC Design 66

2.2.2 Cascaded SMC Design 67

2.3 Active FTC Sliding Mode Controller Design 68

2.3.1 State Estimation Using Extended Kalman Filter 69

2.3.2 Fault Estimation 72
2.3.3 Design of Adaptive SMC

2.4 Tuning The Controllers
   2.4.1 Tuning Process
   2.4.2 ESA Parameters
   2.4.3 Tuning the Controllers Under Fault

2.5 Stability Test Of The Proposed Controllers

2.6 Simulations And Results
   2.6.1 Simulations
   2.6.2 Results

3 Emergency Controller for Quadrotor UAVs
   3.1 Trirotor Dynamics
   3.2 Quad-Tri Conversion
      3.2.1 Different Trirotor Configurations
      3.2.2 Study of the Trirotor Static Equilibrium
      3.2.3 Study of the Trirotor Dynamic Equilibrium
      3.2.4 Trirotor Moment of Inertia Calculation
   3.3 Emergency Control of Quadrotor UAVs
   3.4 Simulation and Results
   3.5 Experimental Work : Emergency Controller Application on AR Drone 2 Platform
      3.5.1 Technical Specifications of AR Drone 2
      3.5.2 Dynamics of AR Drone 2
      3.5.3 Using AR Drone 2 in Plus Configuration
      3.5.4 Emergency Control of AR Drone 2 Quadrotor

4 Integrated FTC of Quadrotor UAVs in Noisy Environments Based on Fault Severity
   4.1 Integrated FTC Controller design
      4.1.1 Passive FTC Sliding Mode Controller
      4.1.2 Active FTC Sliding Mode Controller
      4.1.3 Emergency Controller of Quadrotor UAVs
   4.2 Simulation and Results

5 Fault Tolerant Control of Octorotor UAVs
   5.1 Octorotor Model
   5.2 Control Allocation Problem
      5.2.1 Pseudo-Inverse Control Allocation
      5.2.2 Dynamic Control Allocation
   5.3 First and Second Order Sliding Mode FTC
      5.3.1 First Order Sliding Mode Controller Design
      5.3.2 Second Order Sliding Mode Controller
   5.4 Design of the FDI Unit
5.5 Simulations and Results 154
  5.5.1 Fault Tolerant Control of Octorotor UAVs 154
  5.5.2 Results 155

6 Conclusion 162

Conclusion 162
  6.1 Thesis Outcome 162
  6.2 Future Work 163

A Bibliography 165

Bibliography 165

Appendix 177
  A Publications associated with this thesis 177
  B Detailed calculations 178
  C Quadrotor Data 181
Figure list

0.1 Different types of UAVs [4]. 21
0.2 Possible future civil uses of UAVs (public domain pictures). 22
0.3 General structure of active fault tolerant control systems (AFTCS) [7]. 23

1.1 Actions taken by zebra and lion individuals. 35
1.2 The function used to check ESA. 36
1.3 ESA result (black dots are agents, red crosses are predators). 37
1.4 Results of ESA searching a dynamic environment. 38
1.5 Quadrotor schematic and Asctec Pelican quadrotor in UAEU. 39
1.6 Flowchart of a bioinspired search algorithm. 43
1.7 Crossover operation (single point crossover). 45
1.8 Mutation operation (binary mutation). 45
1.9 Response of quadrotor using gains of the first GA experiment (with MSE fitness). 46
1.10 Controls applied in GA experiment-1. 47
1.11 Fitness change of GA algorithm for experiment-1. 47
1.12 Response of quadrotor using gains of the second GA experiment (with complex fitness). 48
1.13 Controls applied in GA experiment-2. 48
1.14 Escherichia coli bacterium (http://www.howstuffworks.com). 49
1.15 Flowchart of E.Coli Search Algorithm. 52
1.16 Response of quadrotor using gains of the first E.Coli experiment. 54
1.17 Controls applied in E.Coli experiment-1. 54
1.18 Fitness change of E.Coli algorithm experiment-1. 55
1.19 Response of quadrotor using gains of the second E.Coli experiment. 55
1.20 Controls applied in E.Coli experiment-2. 56
1.21 Response of quadrotor using gains of the first ESA experiment. 59
1.22 Controls applied in ESA experiment-1. 59
1.23 Fitness change of ESA experiment-1. 60
1.24 Response of quadrotor using gains of the second ESA experiment. 60
1.25 Controls applied in ESA experiment-2. 61

2.1 Cascaded SMC control of the height. 67
2.2 FDI unit configuration. 73
2.3 Tuning the controllers using ESA. 76
2.4 Search space of the height SMC controller.
2.5 Phase portrait of Regular SMC controlled quadrotor with no faults.
2.6 Phase portrait of Cascaded SMC controlled quadrotor with no faults.
2.7 Phase portrait of Regular SMC controlled faulty quadrotor (M2 fault of 54% at t = 20s and M1 fault of 55% at t = 40s).
2.8 Phase portrait of Cascaded SMC controlled faulty quadrotor (M1 fault of 55% at t = 40s, and M2 fault of 54% at t = 20s).
2.9 Bode diagram of the x-y position controller.
2.10 Path of Regular SMC controlled Quadrotor (2 faults of 55% at 40s and 40% at 20s).
2.11 Attitude tracking error of Regular SMC controlled Quadrotor (2 faults of 55% at 40s and 40% at 20s).
2.12 Altitude and heading tracking error of Regular SMC controlled Quadrotor (2 faults of 55% at 40s and 40% at 20s).
2.13 Controls and Rotor speeds of Regular SMC experiment (tuned without fault).
2.14 Path of Regular SMC controlled Quadrotor tuned under fault (2 faults of 57% at 40s and 56% at 20s).
2.15 Attitude tracking error of Regular SMC controlled Quadrotor tuned under fault (2 faults of 57% at 40s and 56% at 20s).
2.16 Altitude and heading tracking error of Regular SMC controlled Quadrotor tuned under fault (2 faults of 57% at 40s and 56% at 20s).
2.17 Path of Cascaded SMC controlled Quadrotor tuned under no fault (2 faults of 55% at 40s and 54% at 20s).
2.18 Attitude tracking error of Cascaded SMC controlled Quadrotor tuned under no fault (2 faults of 55% at 40s and 54% at 20s).
2.19 Altitude and heading tracking error of Cascaded SMC controlled Quadrotor tuned under no fault (2 faults of 55% at 40s and 54% at 20s).
2.20 Controls and Rotor speeds of Cascaded SMC experiment (tuned without fault).
2.21 Path of Quadrotor controlled with Cascaded SMC tuned under fault (2 faults of 56% at 40s and 56% at 20s).
2.22 Attitude tracking error of Quadrotor controlled with Cascaded SMC tuned under fault (2 faults of 56% at 40s and 56% at 20s).
2.23 Altitude and heading tracking error of Quadrotor controlled with Cascaded SMC tuned under fault (2 faults of 56% at 40s and 56% at 20s).
2.24 Paths of the Active SMC controlled Quadrotor.
2.25 Attitude response of the Active SMC controlled Quadrotor with 55% fault (helical path, 55% fault to M1 at t = 40s).
2.26 Heading and altitude response of the Active SMC controlled Quadrotor with 55% fault (helical path, 55% fault to M1 at t = 40s).
2.27 Controls and rotor speeds of the Active SMC controlled Quadrotor (helical path, 55% fault to M1 at t = 40s).

2.28 Attitude response of the Active SMC controlled infected Quadrotor (square path, two faults at t = 40s, and t = 50s).

2.29 Heading and altitude response of the Active SMC controlled infected Quadrotor (square path, two faults at t = 40s, and t = 50s).

2.30 Controls and rotor speeds of the Active SMC controlled infected Quadrotor (square path, two faults at t = 40s, and t = 50s).

3.1 Trirotor schematic.

3.2 Shifting the CoG of the quadrotor towards the tail rotor.

3.3 Different configurations of the quadrotor used as trirotor.

3.4 Shifting the CoG of the quadrotor towards the tail rotor.

3.5 Relation between distance shifted and added mass.

3.6 Relations between $l_2$ and rotor speeds.

3.7 Quadrotor path (Failure of the second rotor).

3.8 Attitude of the quadrotor (Failure of the second rotor).

3.9 Heading and height of the quadrotor (Failure of the second rotor).

3.10 Controls and rotor speeds of the quadrotor (Failure of the second rotor).

3.11 AR Drone 2 with indoor and outdoor hulls [88].

3.12 AR Drone 2 body frame.

3.13 AR Drone 2 configuration conversion (plus-rotor1 conversion).

3.14 Shifting of the AR Drone 2 CoG towards rotor4.

3.15 Experimental set-up for AR Drone 2.

3.16 The quadrotor/trirotor control scheme.

3.17 Quadrotor to trirotor control conversion.

3.18 Path of the quadrotor model without emergency controller with and without fault (following a circular path).

3.19 Path of the quadrotor model without emergency controller with and without fault (following a square path).

3.20 Path of the faulty quadrotor model controlled with emergency controller (following a circular path).

3.21 Response of the faulty quadrotor model controlled with emergency controller (following a circular path).

3.22 Path of the faulty quadrotor model controlled with emergency controller (following a square path).

3.23 Response of the faulty quadrotor model controlled with emergency controller (following a square path).

3.24 Response of the real fault-free quadrotor experiment (following a circular path).

3.25 Response of the real fault-free quadrotor experiment (following a square path).
3.26 Response of the real faulty quadrotor without emergency controller (following a circular path).
3.27 Response of the real faulty quadrotor without emergency controller (following a square path).
3.28 Path of the real faulty quadrotor controlled with emergency controller (following a circular path).
3.29 Response of the real faulty quadrotor controlled with emergency controller (following a circular path).
3.30 Path of the real faulty quadrotor controlled with emergency controller (following a square path).
3.31 Response of the real faulty quadrotor controlled with emergency controller (following a square path).
3.32 Path of the real faulty quadrotor with emergency controller applied without weight redistribution (following a circular path).
3.33 Response of the real faulty quadrotor with emergency controller applied without weight redistribution (following a circular path).

4.1 Structure of the IFTC
4.2 Helical path and decision unit output - first fault scenario.
4.3 Controls and rotor speeds - first fault scenario.
4.4 Response of the quadrotor with the IntFTC - first fault scenario.
4.5 Helical path and decision unit output - second fault scenario.
4.6 Controls and rotor speeds - second fault scenario.
4.7 Response of the quadrotor with the IntFTC - second fault scenario.
4.8 Square path and decision unit output.
4.9 Controls and rotor speeds - Square path.
4.10 Response of the quadrotor with the IntFTC - Square path.

5.1 Octorotor schematic [97].
5.2 Octorotor control configuration with control allocation.
5.3 2-D and 3-D Path of the octorotor controlled with FO DCA FTC (total loss of 4 motors).
5.4 X and Y response of the octorotor controlled with FO DCA FTC (total loss of 4 motors).
5.5 Height and attitude response of the octorotor controlled with FO DCA FTC (total loss of 4 motors).
5.6 Controls and Rotor speeds of the octorotor controlled with FO DCA FTC (total loss of 4 motors).
5.7 2-D and 3-D path of the faulty octorotor UAV controlled with Second Order SMC (4 total failures and 1 partial loss of effectiveness faults).
5.8 X and y response of the faulty octorotor UAV controlled with Second Order SMC (4 total failures and 1 partial loss of effectiveness faults).

16
5.9 Altitude and attitude response of the faulty octocopter UAV controlled with Second Order SMC (4 total failures and 1 partial loss of effectiveness faults).

5.10 Controls and Rotor speeds of the faulty octocopter UAV controlled with Second Order SMC (4 total failures and 1 partial loss of effectiveness faults).

.1 Pelican quadrotor data.
.2 Pelican quadrotor model using Solidworks.
Table list

1.1 Variables of Astec Pelican Quadrotor 40
1.2 Parameters of GA Used to Tune the Quadrotor PD Controller 46
1.3 Values Of The Gains Found By Two GA Experiments 46
1.4 Comparison Values of GA Experiments 49
1.5 Parameters of Ecoli Algorithm Used to Tune the Quadrotor PD Controller 51
1.6 Values Of The Gains Found By Two E.Coli Experiments 53
1.7 Comparison Values of E. Coli Experiments 56
1.8 Variables Used in ESA to Tune the Quadrotor PD Controller. 58
1.9 Values of The Gains Found By Two ESA Experiments 58
1.10 Comparison Values of ESA Experiments 58
2.1 Look-Up Tables Used To Estimate Fault Magnitude 74
2.2 Best Gain Vector Values Found Using ESA 78
2.3 Variables Used in ESA to Tune Regular SMC Gains. 79
2.4 Variables Used in ESA to Tune Cascaded SMC Angles Gains. 79
2.5 Variables Used in ESA to Tune Cascaded SMC Height Gains. 79
2.6 Gain Vector Values with Regular and Cascaded Controllers Tuned Under Fault 80
3.1 Variables of AR Drone 2 Quadrotor 114
5.1 Octorotor Variables 143
5.2 Variables Used In ESA To Tune the Octorotor SMC. 149
5.3 Best FOSMC Gain Values Found Using ESA 149
5.4 Best SOSMC Gain Values Found Using ESA 153
5.5 Fault Effects On Motor Speeds 153
5.6 Look-Up Table Used To Estimate Fault Magnitude 153
1. Introduction

During the last decade, Unmanned Aerial Vehicles (UAVs) have started leaving their military bases and clandestine life to participate more in the civil society. From packet delivery to weather inspection, forest fire mapping to crop dusting, UAVs use their experience and extraordinary capabilities to conduct dull, dirty, and dangerous missions. The increase use of UAVs in civil missions hardens the cost, safety, reliability, and performance requirements. The loss of a UAV equipped with expensive sensors and equipment for weather inspection for example could stop the whole scientific research project. Similarly, the loss of control of a logistics UAV used for packet or mail delivery in an urban area may result in injury or death incidents. This emphasizes that the more UAVs are taking part in our lives, the more they need to be equipped with powerful and reliable Fault Tolerant Controllers.

1.1. Unmanned Aerial Vehicles

An Unmanned Aerial Vehicle (UAV) (AKA Uninhabited Aerial Vehicle, Remotely Operated Aircraft (ROA), and Remotely Piloted Vehicle (RPV)) is defined by the American Institute of Aeronautics and Astronautics (AIAA) as to be "An aircraft which is designed or modified, not to carry a human pilot and is operated through electronic input initiated by the flight controller or by an onboard autonomous flight management control system that does not require flight controller intervention" [1].

If size and endurance is used to arrange their types, UAVs can be classified into four main groups: Micro/Mini UAVs, Tactical UAVs, Strategic UAVs, and Special Task UAVs. According to the US Defense Advanced Research Projects Agency (DARPA), a Micro UAV has a size of less than 15 cm, a maximum weight of 100g, a payload less than 20 g, a range varying between 1 and 10 Km, an endurance of 60 min or less, a maximum altitude of 150 m, and a maximum speed of 15 m/s. Tactical UAVs are mainly used in military applications, they have a maximum range of 500 Km and can reach altitudes of 8000 m. Tactical UAVs are divided into six sub-categories, Close, Short, Medium, and Long Range (CR, SR, MR, LR) UAVs, as well as Endurance (EN), and Medium Altitude Long Range (MALE) UAVs. Strategic UAVs, also known as High Altitude Long Range
UAVs, have very big platforms designed to take heavy payloads up to 2000Km range and 20000 m altitude [2].

UAVs can also be classified according to their wing-shape into one of the following classes: fixed-wing UAVs, rotary-wing UAVs, hybrid-wing UAVs, flapping-wing UAVs, and airship UAVs (Fig. 0.1). Fixed-wing UAVs are unmanned airplanes that are able to fly with high cruising speeds, but which need a long runway to take-off and land, or require auxiliary means, such as catapults, to be launched. Moreover, they have restricted maneuverability in urban areas, and they lack hovering capability used to monitor fixed points. Compared with fixed-wing UAVs, Rotary-wing UAVs (also known as Vertical Take-Off Landing or VTOL UAVs) have less cruising speeds, which reduces the mission area, but they possess very high maneuverability capabilities, and the ability to hover in a fixed point. Moreover, they need no prepared landing zones since they can take-off and land vertically in a very restricted area. Hybrid-wing, or tilting-wing, UAVs have convertible configuration and they can take-off and land vertically. Once the take-off is realized, hybrid-wing UAVs tilt their wings, rotors, or bodies and fly like airplanes. While using advantages of both fixed and rotary wing UAVs advantages, the main disadvantage of hybrid-wing UAVs is their complexity in manufacturing, modeling, and control phases. Flapping-wing UAVs generate aerodynamic forces and moments using a complex multi degree of freedom wing motion such as flapping, pitching, twisting, and lagging. This is a new trend in the UAV world, where scientists are imitating birds and insects motion to design effective platforms. UAVs of this type are expected to have higher maneuverability and better energy efficiency compared with classical UAV types. Airship or Lighter-Than-Air (LTA) UAVs use helium-filled balloons to generate their lift force. Despite their slow speed and restricted maneuverability, LTA UAVs are very cost effective, and have very long mission endurance. In addition, this type of UAVs is less complex, and very restrictive concerning energy saving. Moreover, airships by their nature have reduced radar and infrared signatures [3], which make them ideal in military operations where endurance not speed is of primary importance.
The main use of UAVs is in military intelligence, defense surveillance, and aerial strikes. Recent years have witnessed the use of UAVs in many civil and commercial applications. The first civil application of UAVs is in aerial reconnaissance. This application includes the inspection of terrain, oil and gas pipelines, utilities and buildings, the surveillance of forest fires, coastal borders, waterways, and road traffic, along with environmental and meteorological monitoring, disaster management, maritime and mountain search, as well as rescue operations. Another civil application of UAVs is in civil logistics and transportation. This application varies from mail and packet delivery to crop dusting and realizing communication relays. UAVs are also used in scientific research, where UAVs are equipped with sensors and sent to collect information and data from pre-defined or hazardous locations. UAVs are used as reliable and safe agents in remote sensing, aerial mapping, pollution monitoring, air sampling, hurricane hunting, and geophysical surveying, along with control systems development, robotic systems research, and path planning algorithm testing.

Multi-rotor rotary-wing UAVs are chosen for study in this thesis because they have higher maneuverability and hovering capability compared with fixed-wing and single-rotor UAVs [4]. These properties make rotary-wing UAVs ideal for civil uses in urban areas and cities.
1.2. Fault, Failures, and Fault Tolerant Control

With UAVs involved more and more in our lives, it becomes essential to equip UAV platforms with Fault Tolerant Controllers (FTCs). Because UAVs are used in harsh and hazardous environments, they are subjected to severe weather conditions, uncertain wind gusts, and turbulences which increase the risk of mechanical or structural faults and failures. Moreover, UAVs lack the fast smart decision making of human beings, which emphasizes the importance of using FTCs integrated in the UAV controllers for abnormal conditions. FTCs ensure safety, reliability, and high performance of the platforms, along with cost reduction of the missions by reducing the cost associated with UAV loss [5].

A fault is defined as an "unpermitted deviation of at least one characteristic property or parameter of the system from the acceptable/usual/standard condition". Compared with faults, failures are more severe, and result in the total loss of the infected actuators. A failure is "a permanent interruption of a system's ability to perform a required function under specified operating conditions" [6]. While some faults and failures affect the UAV actuators (i.e. control surfaces, rotors, etc.) and sensors, other faults, known as structural damage, may affect the structure of the UAV system. This means that new operating conditions of the system emerge as result of the change in the UAV dynamics (e.g. new aerodynamic coefficients or new center of gravity location as a result of collision).
1.2.1. Fault Tolerant Control (FTC)

According to Zhang and Jiang [7], a Fault Tolerant Controller is a closed-loop controller which can tolerate component malfunctions while maintaining desirable performance and stability properties. One can classify FTCs into two general groups, Active, and Passive FTCs. Passive FTCs are designed using robust controllers, e.g. Sliding Mode Controllers (SMCs) and H-infinity Controllers, such that the system gains immunity against disturbances and design uncertainties. Any partial fault affecting the system is assumed as disturbance, and the controller maintains the system stability with an acceptable performance degradation. Fault tolerance capability of Passive FTCs is limited, only presumed faults can be tolerated, and handling failures or severe faults cannot be achieved without reconfiguration of the controller. Active FTCs are more reliable and can handle critical failures and severe faults, but they need to detect and estimate the system faults in short time. Active FTCs are based on adaptive control theory, and they use the fault information provided on-line by the Fault Detection and Identification (FDI) unit to reconfigure the control law and achieve a real-time fault/failure accommodation. The general structure of an AFTC is shown in Figure 0.3.

![Figure 0.3: General structure of active fault tolerant control systems (AFTCS) [7].](image)

1.2.2. Fault Tolerant Control Applied to UAVs

Designing a Fault Tolerant Controller for Unmanned Aerial Vehicles is a challenging research topic since the controller has a very short time to apply corrective moves and recover from fault/failure effect. However, many recent researches accept the challenge, and scientists are almost competing in designing reliable, fast, and safe Fault Tolerant Controllers for UAV applications.
Both Active and Passive Fault Tolerant Controllers are developed based on Sliding Mode (SMC) theory. In [8], authors develop passive and active FTCs for quadrotor UAVs based on the SMC theory. The Passive FTC utilizes the inherent robustness property of the SMC for fault tolerance. Active SMC, on the other hand, activates different pre-designed controllers based on the fault information. Simulation and experimental results on the Qball-X4 platform show that both controllers succeeded in fault tolerance, and that the active FTC is more robust and has better tracking performance in presence of faults. The faults injected to the platform are small actuator faults / propeller damage. Finally, a detailed comparison showing the advantages, disadvantages, and limitations of each type is given. In [9], different algorithms for quadrotor fault tolerant control are presented and compared. Authors examined Sliding Mode Control (SMC), Backstepping Control (BSC) (both are passive FTC type), as well as Gain-Scheduled PID, Model Reference Adaptive Control, and Model Predictive Control (active FTC type) applied in simulation and experimentally on a Qball-X4 experimental testbed. The quadrotor suffers a partial loss in the control effectiveness of one or more actuators. SMC based controller is shown to exhibit good performance by compensating rapidly the fault in rotor 4. In [10], Hao et al. design a new active FTC using SMC theory. Matrix full-rank factorization technique is used to design a new Sliding Mode surface. The gain of the discontinuous part of the SMC are updated online using flexible design parameters. This gives the new controller ability to compensate system faults resulting from model uncertainties and disturbances, as well as input faults resulting from actuator outage, loss of effectiveness, and stuck. The new design does not require a Fault Detection and Isolation (FDI) unit, and can, under a special actuator redundancy assumption, compensate some total actuator failures. A different FTC based on SMC for a class of uncertain systems is developed in [11]. The controller has a time-varying state-delay that is included in the condition of existence of the sliding surface. The controller shows good fault tolerant performance on a linear model of an aircraft subjected to parametric uncertainties, actuator faults, and external disturbances. A robust SMC is developed in [12] for fault tolerant control of the highly nonlinear Machan UAV. The controller uses online feedback linearization technique to linearize and decouple the nonlinear system into three SISO second order sub-systems. Three SMCs are then developed using a simple and effective phase modulation method to control the three sub-systems. Phase modulation rectifies the structure and parameters of the controller, and guarantees robust performance and robust stability. Simulations show that the new controller has improved the system response against disturbances and chattering, and has promising fault tolerant control scheme for UAV systems. Authors in [13] developed an adaptive SMC based on neural networks. The controller has shown robustness against chattering and dynamical model uncertainties problems, with no prior knowledge required about dynamic model and external disturbances. Simulation results emphasized the efficiency of the proposed controller that might
have excellent FTC properties.

Cascaded SMCs using multi-layer interlaced sliding surfaces are developed in [14] and [15]. The first controller is used to control a 2DOF under-actuated manipulator, while the second controller is designed for bicycle robot control. Simulation results emphasize the robustness and effectiveness of the two controllers. In [16], a cascaded SMC is developed for quadrotor control. The inner loop responsible of stabilizing altitude and attitude variables is based on feedback linearization control, while the outer loop that controls the position of the quadrotor is based on SMC. Simulation results show good performance of the proposed controller despite the presence of external disturbances.

The total failure of single rotor of a quadrotor is interpreted in [17], [18], [19], [20] and [21]. In these researches, unconventional FTCs are developed. Authors in [20] proposed a new periodic solution that maintains the height of a hovering quadrotor despite the loss of one, two opposing, or three of its rotors. Authors first make the infected quadrotor spins around a fixed axis, then they use the remaining rotors to rotate this axis and apply the translational motion for quadrotor. A cascaded controller with an inner loop for the roll and pitch angle control, and an outer loop for the UAV translational control is used. Authors provide experimental results for the one and two opposite rotors cases, and simulation results for the three rotor case. Lanzon et al. proposed an FTC for the quadrotor suffering a total failure of one rotor in [21]. The new method has a model-based double control loop scheme, it uses robust feedback linearization to control all the UAV states except the yaw angle, and utilizes an $H_\infty$ loop shaping technique to control the roll and pitch angles. Whenever a rotor fails, the quadrotor enters a spinning motion around its $z$-axis, while the controller provides the quadrotor with cyclic desired roll and pitch values, ensuring its translation motion to the desired destination. However, and to be able to apply the new method, quadrotor motors should be able to provide high thrust, and should have high saturation levels. Moreover, the motor opposing the infected rotor should be able to perform a pulse-like thrust. Results based on simulations illustrate the idea of controlling an infected quadrotor with only three rotors. A new method for controlling quadrotors with one disabled rotor is presented in [22]. The novel algorithm stops the rotor opposite to the infected rotor, and uses backstepping control to drive the resulting birotor anywhere in the cartesian space. Applicability of the proposed method is proved using simulation results. In [18], a path following controller using a smooth dynamic feedback control law is proposed. The new controller helps quadrotor UAVs to keep path tracking of closed and open embedded curves when one rotor fails. The uncontrolled yaw dynamics of the quadrotor are also shown to be bounded. Simulations emphasize the applicability of the new controller. In his thesis, Freddi developed a Thau observer-based diagnostic scheme that uses adaptive thresholds for the detection of sensor and actuator faults affecting quadrotor UAVs in presence of disturbance and model uncertainty [17]. Freddi then proposes a cascaded controller for the quadrotor
with inner loop based on feedback linearization and outer loop that manipulates the values of the desired roll and pitch angles. The proposed controller is used for fault-free case, and is updated to be applied in case of the failure of one actuator. The controller makes the quadrotor to hover using only two rotors, and then changes the speed of the used rotors slowly to make the quadrotor reach a desired point on the ground despite the loss of yaw angle control. Finally, an alternative control fault tolerant control law based on Robust Feedback Linearization (RFL) and $H_{\infty}$ loop shaping is also proposed and tested in simulation.

The new trend in UAV Fault Tolerant Control is to develop integrated controllers that combine the properties of Active and Passive FTCs to accommodate for various types of faults. This type of FTCs activates the relevant controller essential to drive the UAV safely based on fault magnitude and severity. These FTCs ensure fault tolerance and also save actuator resources and processor computational effort as shown in [23], [24], [25], [26], [27], and [28]. In [28], the FTC presented for aircraft systems ensures both effectiveness and reliability. The new controller combines the properties of active and passive FTCs to tolerate faults, and is able to compensate simultaneously for actuator faults, model mismatch, and parameter variations. The passive part of the controller is based on SMC, while the active part is based on model reference adaptive approach. The new FTC activates corrective actions only when necessary, which results in ameliorated flight effectiveness and better system reliability. Simulation results using an aircraft system subjected to various types of faults emphasize the importance of the new scheme. A systematic FTC strategy for quadrotor UAVs that can coordinate various FTC methods is proposed in [27]. The new method can compensate faults and failures depending on the fault type and severity level. The controller uses an FDI unit based on Adaptive Thau Observer that estimates online fault magnitudes. If the fault is small, a passive FTC is used to achieve the quadrotor control. For higher fault values, an active FTC controller is used to recover losses in control laws. If the fault is critical or an actuator failure occurs, an emergency fault parking strategy is activated to land the quadrotor vertically. Simulation results carried with different faults injected show the robustness and effectiveness of the new strategy. In [26], a new FTC is proposed for flight systems. The controller utilizes adaptive model reference control method incorporated with control allocation to compensate for simultaneous actuator failures, input saturation, and model mismatch. Simulation results prove the performance of the new scheme. Authors in [25] propose an intelligent self-repairing control scheme for a class of nonlinear MIMO system. The new scheme is formed of four parts, the nonlinear regulator, the equal controller, the first compensator, and the second compensator. Simulations of a helicopter controlled with the new scheme show that the controller was able to accommodate for loss-in-effectiveness faults. In his thesis, Avram develops, analyzes, and implements experimentally fault diagnosis, fault-tolerant control, and controller verification methods to achieve safety assurance and trusted autonomy of quadrotor UAVs [29]. First, quadrotor
sensors fault diagnosis is interpreted under two situations: when all the attitude angles are available, and when roll and pitch angles are not measurable and need to be estimated. The fault detection, isolation, and bias estimation of accelerometer and gyroscope sensors are achieved by the design of nonlinear adaptive estimators. Second, the fault-tolerant control of quadrotor UAVs under actuator faults is investigated using two different methods. As a first method, author represents the design of an integrated actuator fault diagnosis and accommodation scheme. This method is an active fault tolerant controller that uses the fault diagnostic information to accommodate the effect of the faults. The second method is a back-stepping method based passive nonlinear adaptive fault-tolerant altitude and attitude controller that recovers tracking performance of the quadrotor automatically without the need of a fault diagnosis unit. The controller is shown to guarantee asymptotic convergence of the altitude and attitude tracking error even in the presence of multiple simultaneous actuator faults and modeling uncertainties. Finally, a Run-Time Assurance (RTA) architecture is investigated to verify and validate the adaptive fault-tolerant controller developed in the thesis. This is essential to detect the presence of controller software faults and unstable adaptation behaviors that emerge when faults are injected. Avram represents real-time flight results using indoor quadrotor test environment to evaluate all the methods developed in his thesis. In [24], a class of robust adaptive state feedback controllers are used to compensate automatically mismatched parameter uncertainties, external disturbances, and actuator faults in linear systems. Actuator faults are caused by loss of effectiveness, outage and stuck. Simulations illustrate the efficiency of the proposed fault-tolerant design. In [30], authors use model predictive controller and reference trajectory management techniques to develop a novel reconfigurable active FTC. Whenever an actuator is infected, an optimization algorithm generates a new degraded reference trajectory, and the controller calculates new admissible controls. To reduce the control energy spent in closed loop dynamics and avoid actuator saturation, a constraint set and a cost function are established. Simulations conducted on a hydrothermal system subjected to actuator faults and actuator constraints show the effectiveness of the new design.

Fault tolerant control of quadrotors were the subject of many recent fault tolerant control studies [29], [27], [31], [21], [32], [8], [33], [34], [35], [9], and [36]. Despite the evolution made in this field, lack of redundancy still the main problem of quadrotors that prevents any fault tolerant control to handle high fault percentages, severe damages, and total failure of actuators. Underactuated systems have less actuators than controlled variables, and any total loss or stuck of one of their actuator results in a catastrophic end. Control Allocation methods are used to develop Fault Tolerant Controllers for over-actuated systems such as vehicles, vessels, aircrafts, and UAVs in [37], [38], [39], [40], [41], [42], [43], [44], [5], [45], [36], [46], [47], [48], and [49]. Necessary and sufficient conditions to analyze the controllability of multirotor systems subject to rotor
failure and wear based on the Available Control Authority Index (ACAI) is presented in [50].

In [37], weighted pseudo-inverse control allocation method is presented, and shown to be capable of exploiting a larger domain in the virtual control than the simple pseudo-inverse method. The new method has low computational load, which makes it more suitable for microcontroller controlled UAVs. Authors provided a detailed mathematical analysis, but the paper lacks simulation and experimental results that support the idea. Control allocation algorithm is coupled with an ad-hoc online parameter estimator to form an FTC for Autonomous Overactuated Vehicles in [38]. Dynamic Inversion and Control Allocation are used to develop an FTC for a fully coupled 6 DOF fixed-wing aircraft in [42]. Simulations show that the aircraft was able to recover its stability after the occurrence of a control surface degradation fault. Partial loss of control effectiveness fault affecting a realistic nonlinear aircraft is accommodated using reconfigurable control allocation method in [41]. The controller uses Pseudo-Inverse control allocation method, and shows good performance based on simulation results. Weighted Least Squares (WLS) reallocation algorithm is used to develop an FTC for UAVs in [43]. The controller was able to handle stuck faults, as well as partial loss of effectiveness of control surfaces of ALTAV UAV in presence of Gaussian noise. Authors in [44] showed that Integral Sliding Mode Controller is robust to disturbances and model uncertainty but cannot handle actuator faults. The controller is augmented with fixed and online control allocation methods to control a passenger aircraft in presence of severe actuator faults. Simulations have shown that the augmentation allows the controller to handle actuator faults, and that online control allocation have better fault tolerant performance. In his thesis, [5], Bason studies in details the application of control allocation methods in the design of fault tolerant controllers for UAVs. Author provided detailed information on the choice of the cost function and the optimization methods, and gave wide explanation on the practical implementation of control allocation systems. Different control allocation methods are applied on a highly overactuated fixed wing UAV, and on a slightly overactuated blended-wing body UAV under various types of faults. In order to handle both fast and slow actuators, as well as fast and slow control commands, the method is updated to be frequency-based. Simulation results on the SLADE ducted-fan UAV are promising. Alwi and Edwards present two fault tolerant control schemes for octorotors using LPV based sliding mode control in [46] and [47]. The first approach uses an FDI unit to provide the control allocator with the rotor effectiveness levels. These levels are used by a Pseudo-Inverse control allocation method to set the amount of control effort distributed on each motor. In the second approach, no FDI unit is used, and all faults and disturbances are assumed to be matched uncertainty that can be tolerated using SMC. Rotor effectiveness levels of the control allocation are always fixed in this method. To ensure the stability of the controller in presence of disturbance and faults, authors suggest a restriction on the SMC gains. Simulations
show that the proposed methods can handle multiple rotor failures in presence of disturbance and gust conditions. In [48], the dynamic model of a 4Y octorotor is derived, and a suitable feedback controller is designed. Moreover, a simple control reconfiguration scheme is proposed to handle the failure of one rotor. Simulation results show the feasibility of the proposed controller. An active FTC is developed for octorotor UAVs using PD controller and Redistributed Pseudo Inverse (RPI) control allocation method in [45]. Despite that no FDI unit is proposed, authors include their design a 0.5s delay for the FDI response. Numerical results show that the octorotor can handle the total loss of four rotors while performing a hover flight. Saied et al. present an actuator fault tolerant controller for a coaxial octorotor in [51]. The controller is based on a simple PID controller, and uses a strategy that benefits from the redundancy of the octorotor to compensate for motor faults. Once the faulty motor is detected using a nonlinear sliding mode observer, the speed of its dual motor is decreased to compensate the missing amount of torque, and the speeds of the remaining motors are increased to generate sufficient thrust that maintains the UAV at the desired altitude. The paper contains valuable experimental results, but the measurement/estimation process of the fault magnitude is unclear. In [52], authors develop a Fault Detection and Isolation (FDI) strategy based on Second Order Sliding Mode Observer (SOSMO) to detect and estimate actuator faults of a coaxial octorotor using the modified super-twisting algorithm. After the observer converges giving the octorotor real states, its equivalent output is used to isolate and estimate the actuator fault magnitudes analytically. Both MATLAB/SIMULINK and experimental results emphasize the effectiveness of the proposed FDI strategy. In her thesis, Majd developed two fault detection and isolation modules to diagnose actuator faults of octorotor UAVs [53]. The first module is constructed based on a nonlinear observer and on the outputs of the inertial measurement unit. The second module uses motor speeds and currents provided by the on board electronic speed controllers to detect actuator failures and distinguish between motor failure and propeller damage. Next, an online rule-based reconfigurable control mixing is developed to accommodate for actuator faults. The controller redistributes the control effort on the healthy actuators and can handle multiple successive faults in one or more actuators. Author provides valuable experimental results of a complete architecture including fault detection and isolation followed by system recovery applied on a coaxial octorotor. The results of the proposed architecture are also compared with the results of two architectures based on pseudo-inverse control allocation and a robust controller using second order sliding mode.

Harkegard first presented Dynamic Control Allocation (DCA) in [54]. This method is based on the distribution of the control effort over the available actuators according to the current control demands and the previous control allocation results. Over-actuated systems using DCA are able to deal with different actuator dynamics, and have an extra degree of freedom based on actuator frequency characteristics. Authors in [55] improve the DCA by adding a two-step dynamic
control allocation term. The new method achieves smoother profile control command by relaxing the strict equality constraint between the virtual and actual control inputs. In [56], authors proposed a new DCA that takes into account the actuator rate and magnitude limits in addition to their dynamics. Simulations using a fixed-wing airplane show that the controls are allocated according to the actuators bandwidths. Lower frequencies are allocated to slow actuators, while higher frequencies are allocated to fast actuators.

1.3. Structure of the Thesis

The objective of this thesis is to develop different Passive and Active Fault Tolerant Control methods to increase the reliability and survivability of civil UAVs against actuator faults using bioinspired search algorithms, Sliding Mode theory, and Control Allocation Methods.

In chapter 1, a new bio-inspired search tool based on ecological systems equilibrium is developed. The new search algorithm, called Ecological Systems Algorithm (ESA), can be used in static and dynamic environments to find optimal solutions for real-life engineering problems. The algorithm is shown to be able to find the optimal gains of a PD controller for a quadrotor Unmanned Aerial Vehicle.

In chapter 2, Sliding Mode theory is used to develop Passive and Active fault tolerant controllers for quadrotor UAVs. Passive FTCs are based on regular and cascaded SMCs, and use the inherent robustness of Sliding Mode Control to compensate partial loss of effectiveness in the UAV actuators. Active FTCs use the fault magnitude estimated by a Fault Detection and Identification (FDI) unit to reconfigure a Sliding Mode based controller. Using estimates of the EKF in the FDI design ensures the precision of the fault estimation despite the presence of measurement noise, model uncertainties, and disturbances. ESA algorithm presented in the previous chapter is used to tune the controllers of this chapter.

In chapter 3, an emergency controller for quadrotor UAVs is presented. Quadrotors are under-actuated systems, their actuators are less than their Degrees Of Freedom (DOF). This means that any severe fault or failure of an actuator (rotor/motor) results in the UAV crash. The emergency controller presented in this chapter is based on the quadrotor to trirotor conversion maneuver. Whenever a rotor/motor is affected with a severe fault (or fails), the emergency controller turns the infected motor off, and controls the UAV as a trirotor. Despite theory shows that this maneuver is applicable only if a weight re-distribution is applied, experimental results show that the controller is able to fly the infected quadrotor even thought without the application of weight redistribution maneuver.

Chapter 4 presents an Integrated Fault Tolerant Controller for quadrotor UAVs based on Sliding Mode theory. In no fault situations or when actuator faults are small, Passive SMC FTC is responsible for the quadrotor control. When the
amount of fault exceeds a given threshold, the control unit activates the Active FTC SMC, and the fault estimation is used to compensate for the emerging situation. When the fault magnitude rises to unacceptable value, or when a motor fails, the emergency controller is activated, weight re-distribution is applied, and the quadrotor continues its path as a trirotor. Similar to chapter 2, the Fault Detection and Identification (FDI) unit uses the controls based on the output of Extended Kalman Filters (EKF).

In chapter 5, the Fault Tolerant capabilities of over-actuated UAVs is studied. Sliding Mode Controllers are designed for an octorotor UAV using First and Second Order Sliding Mode theory. The octorotor is then equipped with a control allocator which re-distributes the control effort among the actuators according to the fault magnitude that affect each motor/rotor. Two control allocation methods are used and compared, Pseudo-Inverse and Dynamic Control Allocation. The Fault Detection and Identification (FDI) unit is based indirectly on EKF design, and estimates online the faults that effect each rotor.
2. Ecological Systems Algorithm With Application on Quadrotor Controller Tuning

The new trend in engineering is to use Artificial Intelligence (AI) algorithms to find the desired solution of a problem, rather than assembling complicated formula and solving them analytically. In AI, "intelligent" algorithms are developed by imitating human reasoning, animal behavior, and bacterial action/reactions. Research conducted in the field of animal behavior showed that a group (swarm) of simple individuals, each one with intuitive action/reaction behavior, is able to solve daily foraging and safety problems, and thus can be considered "intelligent". This gives birth to the "Swarm Intelligence" field, where swarms of naive individuals are used to solve complex engineering optimization problems that have very little information provided about the domain. Swarm intelligence algorithms start with random assumption of the solution, and iterate until the optimal solution of the problem is found.

In this chapter, a new swarm intelligence algorithm called Ecological Systems Algorithm (ESA) is developed and tested. ESA imitates the ecological balance in nature to find the optimal solution of a given problem. A comparison between ESA, Genetic Algorithm (GA), and E. Coli Bacterial Foraging Algorithm (BFA) based on their ability to handle real engineering problems is also provided.

2.1. Ecological Systems Algorithm

Biologically inspired algorithms mimic animal, insect, and bacterial systems to solve complex engineering problems using the trivial decision making of elementary creatures. Ecological Systems Algorithm is a biologically inspired algorithm that imitates natural selection, which assures the ecological equilibrium, to optimize a given function. ESA uses two individual groups in its search, a herbivores group (e.g. zebras) that benefits from high fitness values of the optimized function, and a carnivores group (e.g. lions) which travels across the environment looking for herbivores preys (i.e. zebras). The environment is a search space that contains possible solutions for the optimized function. Health of zebra indivi-
duals increase in an iteration when they reach a location which contains better solutions for the optimized function. They are assumed to forage on the result of a fitness function measuring how much the solutions of their location are close to the optimal solutions. Zebras move randomly in the search space looking for the location giving the highest fitness values. When zebras reach a place with high fitness value, they become healthier, and breed new babies which increases the population at this location. When a zebra flock is attacked by lions, it disperses and zebras escape to various locations. This means that new, and previously undiscovered locations could be reached. Lions in ESA assure the role of dispersal and elimination processes in Bacterial Foraging Algorithm and mutation process in Genetic Algorithm. Lions species adds randomness to the search process, which guarantees the diversity of individuals, and strengthens their chance to avoid local optimums. The algorithm developed here differs from the ECO algorithm shown in [57] in that a complete new algorithm based on species interaction is developed. In [57], the Artificial Bee Colony (ABC) is improved by introducing the intra-habitat relations between the individuals of the same location, and the inter-habitat interactions between the individuals of different locations. Moreover, this algorithm didn't introduce the interaction between the populations of different species, which has a very important role in ecosystem equilibrium.

ESA starts by creating zebra and lion individuals in random locations and with random health values. Zebras move randomly in the search space looking for high fitness locations, while lions search randomly the environment looking for their preys. When a lion and a zebra individual confront, their health values are compared, and the individual with the higher health value survive. If the predator is healthier, it eats the zebra and increases its health. On the other hand, if the zebra is healthier it manages to escape but suffering some injuries which decrease its health. This means that the lion was not able to forage in this iteration, resulting in a decrease in its health. Lion and zebra individuals have a maximum iteration age and health values that are defined at the beginning of the run. After running the program for a while, a balance between the two species is established, and the majority of zebras will be gathered in the location with the highest fitness value.

ESA simulates the motion of real animals in nature, where individuals perform naive actions that require no calculation or reasoning. The actions/reactions of zebra and lion individuals are shown in Figure 1.1. A zebra individual moves with random steps in the search space, and calculates the nutrient density (using fitness function) in the new location. If the new location has more nutrient density than the previous one, health of the zebra individual increases, and a new zebra baby is born. Zebra individuals are attacked whenever they are found close to a lion individual. When a zebra is attacked, its health is compared with that of the predator. If the prey’s health is greater, it manages to escape but while suffering from injuries that decrease its health. On the other hand, if the preda-
tor is healthier, it eats the zebra increasing its own health. Similar to the zebras’ life cycle, lions move randomly in the space searching for their preys, they attack any close prey, eat it if they are healthier, increase their health, and breed new babies. The health of a lion individual decreases if it could not eat in an iteration. After some iterations, zebra individuals found in low nutrient locations extinct, while zebras found in high nutrient locations resist extinction and breed. At the end, most of the zebra individuals will be gathered in the high nutrient locations, thus optimal values will be found. Stop criteria of the algorithm is evaluated by calculating the average of the fitness values of all zebra individuals. If this average exceeds a predefined value, it means that the majority of zebras have successfully reach a high nutrient location, and the algorithm stops.
As an example, ESA algorithm is used to search for maxima of the following
complex mathematical function with multiple peaks and valleys (Figure 1.2)

\[ z = 2e^{-0.1((x-15)^2+(y-20)^2)} - 4e^{-0.08((x-20)^2+(y-15)^2)} + \]
\[ 7e^{-0.08((x-25)^2+(y-10)^2)} + e^{-0.11((x-10)^2+(y-10)^2)} - \]
\[ e^{-0.5((x-5)^2+(y-10)^2)} - 6e^{-0.11((x-15)^2+(y-5)^2)} - \]
\[ 3e^{-0.5((x-8)^2+(y-25)^2)} - 2e^{-0.5((x-21)^2+(y-25)^2)} + \]
\[ 3e^{-0.5((x-25)^2+(y-16)^2)} + 5e^{-0.5((x-5)^2+(y-14)^2)} \]

Using ESA to perform the search above requires the definition of many constants. AugA is the factor by which a zebra health increases if it moves to a better location, and it is taken as 1.8 in the above search. DecA, taken above as 1.3, is the factor that multiplies the zebra health when it moves to a worse location. The zebra individual is still finding nutritive resources, but less than before. Dec, taken in the above search as 0.8, is the factor by which the zebra health is multiplied when it manages to escape a lion attack. This is a result of injuries taken after its fight with the lion. DecP, taken as 0.9, is the factor that multiplies a predator’s health when its prey escapes. Aug, taken above as 1.5, is the factor by which a
predator health increases after eating a prey. $Eps_H$, equal to 0.8, is the health boundary, and any individual with health value less than $Eps_H$ dies. If a zebra individual reaches a location with nutrient density less than $Eps_F$, taken here as 1.5, its health decreases regardless of its previous location. On the other hand, if a zebra individual reaches a location with nutrient density more than $Fit_G$, taken as 2 here, its health increases regardless of its previous location.

The above search is started by defining the search space dimensions to be within $x = [0 \ 30]$ and $y = [0 \ 30]$ intervals, and creating a zebra population of 50 individuals, and a lion population of 5 individuals, each one with maximum life of 10 iterations. The maximum step that an individual can make in $x$ and $y$ dimensions is equal to 5, which is also the maximum distance for a lion to detect and attack a zebra. The fitness function used to find the nutrient density in a location is the magnitude of the mathematical function $z$ at that location. The average of the fitness values of all zebra individuals is calculated at each iteration, and whenever this average is equal to 4 the algorithm stops. Multiple runs of the algorithm showed that ESA was able to find the maximum of the proposed function in an average time of 0.35 seconds. Figure 1.3 shows the result of ESA found in only 7 iterations (0.236650 seconds). As many other search algorithms, ESA search could reach a dead-end. In some scarce situations, all zebra individuals extinct, and ESA is unable to converge. These experiments are done on an Intel dual core PC with i3 CPU, speed of 2.4 GHz, 32 bit operating system, and 3 GB of usable RAM.

![Fitness function (contour map)](image)

*Figure 2.3 – ESA result (black dots are agents, red crosses are predators).*

37
After showing promising results in static environments, ESA is used to search dynamic environments. The above mathematical function is set to move in random speed and direction, while ESA is used to find its maximum. Under these conditions, ESA takes more time to find the maximum, but still able to converge. Figure 1.4 shows the result of ESA for two different experiments of dynamic environments.

![Fitness function (contour map)](image)

(a) Exp.1- 3 seconds, 12 iterations  
(b) Exp.2- 5.35 seconds, 24 iterations

Figure 2.4. – Results of ESA searching a dynamic environment.

2.2. Quadrotor Model and Controller Design

Quadrotor UAV’s are under-actuated rotary wing aircrafts that use four rotors to perform their motion in space. Using its four actuators, a quadrotor can climb or descend (along z-axis), can make roll movement (rotation around x-axis), pitch movement (rotation around y-axis), and yaw movement (rotation around z-axis). Figure 1.5 shows the schematic of the quadrotor along with its coordinate system.

The rotors of a quadrotor rotate in opposite pair directions. This makes the torques produced by the rotors to cancel each others at equilibrium. To hover, all the rotors are made to turn with the same angular velocity. By increasing and decreasing this speed, the quadrotor gains or looses altitude. To perform rolling, the velocity of the left and right rotors are made different. By increasing the left rotor speed and decreasing the right rotor speed, the quadrotor performs a positive roll angle. Similarly, pitching is performed by creating a difference between velocities of front and rear rotors. Finally, yawing is ensured by setting a difference between the velocities of the front-rear and right-left rotor pairs. The
dynamic model of the quadrotor is found by applying Newton’s second law on forces and torques [58] and adding the translation drag and the friction aerody-

![Quadrotor schematic and Asctec Pelican quadrotor in UAEU.](image)

Figure 2.5. – Quadrotor schematic and Asctec Pelican quadrotor in UAEU.
The variables of Astec Pelican Quadrotor are shown in Table 2.1. The table includes the inertia moments, b and d, as well as the mass, m.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$I_x$</td>
<td>$8.1e^{-3}$ N.m.s$^2$</td>
</tr>
<tr>
<td>$I_y$</td>
<td>$8.1e^{-3}$ N.m.s$^2$</td>
</tr>
<tr>
<td>$I_z$</td>
<td>$14.2e^{-3}$ N.m.s$^2$</td>
</tr>
<tr>
<td>$I_{rotor}$</td>
<td>$104e^{-6}$ N.m.s$^2$</td>
</tr>
<tr>
<td>m</td>
<td>$1Kg$</td>
</tr>
</tbody>
</table>

where $x, y, z$ are the coordinates of the quadrotor with respect to the base frame, $U = [U_1 \ U_2 \ U_3 \ U_4]$ is the control vector, and $\gamma = \Omega_1 - \Omega_2 + \Omega_3 - \Omega_4$ is the rotor speed imbalance assumed as disturbance if the rotor speeds are not accessible during flight [58]. $I_x, I_y, I_z$ are the body inertia with respect to the corresponding axis, $g$ is the gravitational constant, and $m$ is the mass of the quadrotor. The translation drag coefficients are $K_{ftx} = K_{fty} = 5.5670e^{-4}$ N/m/s, and $K_{ftz} = 6.3540e^{-4}$ N/m/s, and the friction aerodynamic coefficients are $K_{fax} = K_{fay} = 5.5670e^{-4}$ N/rad/s, and $K_{fax} = 6.3540e^{-4}$ N/rad/s.

Table 2.1 – Variables of Astec Pelican Quadrotor

<table>
<thead>
<tr>
<th>Variable</th>
<th>Value (Unit)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$I_x$</td>
<td>$8.1e^{-3}$ N.m.s$^2$</td>
</tr>
<tr>
<td>$I_y$</td>
<td>$8.1e^{-3}$ N.m.s$^2$</td>
</tr>
<tr>
<td>$I_z$</td>
<td>$14.2e^{-3}$ N.m.s$^2$</td>
</tr>
<tr>
<td>$I_{rotor}$</td>
<td>$104e^{-6}$ N.m.s$^2$</td>
</tr>
<tr>
<td>m</td>
<td>$1Kg$</td>
</tr>
</tbody>
</table>

The maximum thrust of the quadrotor is $15.7$ N. This value is used to bound $U_1$ control. The quadrotor’s variables, where $b$ and $d$ are respectively the thrust and the drag factors.
following equations (control effectiveness equations)

\[ U_1 = b(\Omega_1^2 + \Omega_2^2 + \Omega_3^2 + \Omega_4^2) \] (2.7)
\[ U_2 = bl(-\Omega_2^2 + \Omega_4^2) \] (2.8)
\[ U_3 = bl(-\Omega_1^2 + \Omega_3^2) \] (2.9)
\[ U_4 = d(\Omega_1^2 - \Omega_2^2 + \Omega_3^2 - \Omega_4^2) \] (2.10)

The inverse control effectiveness equations are

\[ \Omega_1^2 = \frac{1}{4b} U_1 - \frac{1}{2bl} U_3 + \frac{1}{4d} U_4 \] (2.11)
\[ \Omega_2^2 = \frac{1}{4b} U_1 - \frac{1}{2bl} U_2 - \frac{1}{4d} U_4 \] (2.12)
\[ \Omega_3^2 = \frac{1}{4b} U_1 + \frac{1}{2bl} U_3 + \frac{1}{4d} U_4 \] (2.13)
\[ \Omega_4^2 = \frac{1}{4b} U_1 + \frac{1}{2bl} U_2 - \frac{1}{4d} U_4 \] (2.14)

To perform the control of the quadrotor, a PD controller is designed. Figure 1.5 shows that a rotation of the quadrotor around its \( x \)-axis generates a displacement along its \( y \)-axis. Similarly, a rotation around \( y \)-axis generates a displacement along \( x \)-axis of the quadrotor. The orientation of the quadrotor is assured by the rotation around its \( z \)-axis. An efficient PD controller for the roll, pitch, and yaw angles, while assuming that the quadrotor axes are uncoupled, is given in the following equations

\[ U_2 = K_{rp}(\phi_d - \phi) - K_{rd}(\dot{\phi}) \] (2.15)
\[ U_3 = K_{pp}(\theta_d - \theta) - K_{pd}(\dot{\theta}) \] (2.16)
\[ U_4 = K_{yp}(\psi_d - \psi) - K_{yd}(\dot{\psi}) \] (2.17)

\( U_1 \) is the altitude control input, which is used to cancel the gravitational effect and the nonlinearities in the \( \ddot{z} \) dynamics, and is chosen as

\[ U_1 = g + \frac{K_{zp}(z_d - z) - K_{zd}(\dot{z})}{(\cos(\phi) \cos(\theta))} \] (2.18)

\( K_{rp} \) and \( K_{rd} \) are PD gains of the roll axis controller, \( K_{pp} \) and \( K_{pd} \) are PD gains of the pitch axis controller, \( K_{yp} \) and \( K_{yd} \) are PD gains of the yaw axis controller, and \( K_{zp} \) and \( K_{zd} \) are PD gains of the altitude controller.

In the following, three optimization algorithms are used to tune offline the PD controller of the AscTec Pelican quadrotor UAV. PD controller is chosen for its simplicity and adequacy [59], [60]. In addition, PD and PID controllers are still the most used controllers in industrial applications. Finally, the goal of the experiment is to show the efficiency of ESA algorithm in finding the gains of
the controller. The experiments are assumed to be performed under no complex aerodynamic effects or disturbances, in a fault and disturbance free environment. Despite these assumptions, experiments remain sufficient enough to compare the search algorithms based on results of a real complex engineering problem.

2.3. Tuning of Controller Gains

The number of control variables of a quadrotor is four, the height, and the roll, pitch, and yaw angles. With the PD controller designed previously, each control variable has a proportional gain $K_p$, and a differential gain $K_d$ that need to be tuned. This means that the dimension of the search space is eight, and optimization algorithms will be searching for eight variables at once. The flowchart of an optimization algorithm used to tune the quadrotor controllers is shown in Figure 1.6. The algorithm starts by generating a random solution cluster called population. For tuning the quadrotor PD controller, the population is formed by proposing randomly many $K_p$ and $K_d$ values. Each gain set in the population is then used to control the quadrotor, and a fitness value for this gain set is formed using the quadrotor response. The more a gain set is suitable (or optimal), the more it gives high fitness value. At this point, the search algorithm applies its random search processes, where new individuals are created and the worst individuals are terminated. The final two steps are repeated until a stop condition (e.g. gains with maximum fitness value) is reached. At each iteration, the gain values will be approaching the optimal gains. The search individuals (created initially as proposed solution population) are genes in GA and bacteria in BFA. In ESA, the search is performed by different animal species that interact to find the optimal foraging location.

The fitness function is a function used to weight a proposed set of PD gains (gains for the altitude and attitude controllers of the quadrotor). If these gains are close to their optimal values, their fitness function returns a high value. Note that the fitness function is the only thing related to the quadrotor system when optimization algorithms are used. To find the fitness value of a set of gains, these gains are used with the PD controllers to control the quadrotor performing a clearly defined motion (step functions in our case), and the error of each variable is calculated. These errors are then used to evaluate the gains used. The fitness value of a given set of gains using the Mean Squared Error (MSE) is

$$\text{Fitness}_i = \frac{1}{MSE_i + 0.1}$$  \hspace{1cm} (2.19)

$$MSE_i = \frac{\sum_{t=0}^{tf} e_{it}^2}{\text{size}(E_i)}$$  \hspace{1cm} (2.20)

where $y_i$ is the actual control variable (z, roll, pitch or yaw angles), $y_{di}$ is the set
point of variable $y_i$, $E_i = y_{di} - y_i$ is the error vector of $y_i$ at all instants, and $e_t$ is the value of error at a specific instant $t$. To ensure that the fitness value will not tend to infinity when MSE becomes zero, a small value (0.1) is added at the denominator.

Another fitness function based on the Integral of Time multiplied by Absolute Error (ITAE) can also be used. ITAE weights each error along with its time, and has an increasing high effect of the errors in later moments. The fitness function based on ITAE is formed by replacing $MSE_i$ term in equation (1.19) by

$$ITAE_i = \sum_{t=0}^{tf} (t\|e_t\|)$$

(2.21)

To have more realistic fitness values that reflect the time domain response of the quadrotor, steady state error, settling time, and overshoot values of the response can be used to form the fitness function [61].

The PD controller tuning process in this chapter uses two types of fitness functions. The first function is the MSE based function, and the second function uses
the overshoot, the rise time, and the steady-state error

$$Fitness(K_p, K_d) = \frac{1}{\alpha E_{ss} + \beta M_p + \gamma t_r}$$

(2.22)

where $\alpha$ is chosen as 0.1, $\beta$ as 0.4, and $\gamma$ as 0.5 ($\gamma$ here is a constant, different than the disturbance in quadrotor equations).

### 2.4. Genetic Algorithms Based Tuning

Genetic Algorithm (GA) is an Artificial Intelligence algorithm that imitates natural selection, and is used to solve optimization problems. GA uses genes to search the environment, and mimics gene operations such as crossover and mutation to find iteratively the optimal solution. GA is practical and easy to use with nonlinear spaces, and is not subjected to local minimum problem. Moreover, GA can be used safely to search noisy and discontinuous environments. Roughly speaking, GA assigns random candidate solutions (called population) for the problem, and checks how close each solution is to the optimal solution using a fitness function. The best solutions are then used to generate new solutions by applying genetic operations. The last step is repeated until the optimal solutions are found.

Similar to other heuristic algorithms, GA doesn’t require any mathematical model of the optimized problem, and doesn’t acquire any calculations. It treats the problem as a black-box, and uses a fitness function to evaluate the candidate solutions. After creating the first population as digital strings, with each bit called a gene, GA uses the individuals of this population to find more fitted solutions by applying crossover, mutation, and selection operations. The best individuals in the population survive to the next iteration (called also generation); this is the natural selection operation. Crossover operation consists of choosing randomly two individuals with high fitness value (called parents), breaking each individual randomly into two parts, and swapping one part of each individual with the remaining part of the other (See Figure 1.7). This operation gives birth to two new individuals that inherit some characteristics from both parents, and good characteristics remain in the new generation. Mutation operation, shown in Figure 1.8, consists of choosing a random individual from the population, and toggling a random bit in this individual. This operation helps in recovering an important gene that might be lost in the population. Moreover, mutation creates probabilistic diversity, and helps preventing the local minimum problem which is common in search algorithms.

GA has been used for controller tuning for a long time [62], [63], [64], [65], [66]. Tuning the PD controller of the quadrotor is an optimization problem, where the proportional and derivative gains that give the minimum response error are the optimal solution. The search space has eight dimensions, where PD gains for the
four control variables height, roll, pitch and yaw are needed. The population size, the maximum number of generations, the environment intervals, and the crossover and mutation probabilities are defined at the start. Table 1.2 shows the GA parameters used in tuning the PD controller of the AscTec Pelican quadrotor. The optimal gains of the controller found by GA using fitness functions 1.19 and 1.22 are shown in table 1.3.

Figures 1.9 through 1.13 show the quadrotor response when controlled with gain values shown in table 1.3. Figure 1.11 shows the changes of the highest fitness and the average fitness values during 115 generations. Overshoot, rise time and steady-state errors of the experiments are shown in table 1.4.
Table 2.2. – Parameters of GA Used to Tune the Quadrotor PD Controller

<table>
<thead>
<tr>
<th>Bounds</th>
<th>Population size</th>
<th>20</th>
</tr>
</thead>
<tbody>
<tr>
<td>Height</td>
<td>$K_p$ [20 90]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$K_d$ [10 40]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Crossover probability</td>
<td>0.8</td>
</tr>
<tr>
<td>Roll</td>
<td>$K_p$ [10 50]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$K_d$ [8 30]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Mutation probability</td>
<td>0.08</td>
</tr>
<tr>
<td>Pitch</td>
<td>$K_p$ [10 50]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$K_d$ [8 30]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Stop criterion</td>
<td>$Fit_{ave} &gt; 3.8$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$&gt;0.6$</td>
</tr>
<tr>
<td>Yaw</td>
<td>$K_p$ [10 50]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$K_d$ [8 30]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Max generation</td>
<td>100</td>
</tr>
</tbody>
</table>

Table 2.3. – Values Of The Gains Found By Two GA Experiments

<table>
<thead>
<tr>
<th>Gain</th>
<th>Height</th>
<th>Roll</th>
<th>Pitch</th>
<th>Yaw</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exp-1</td>
<td>$K_{zp}$ 90</td>
<td>$K_{zd}$ 17.3</td>
<td>$K_{rp}$ 50</td>
<td>$K_{rd}$ 8</td>
</tr>
<tr>
<td>Exp-2</td>
<td>$K_{zp}$ 90</td>
<td>$K_{zd}$ 13.79</td>
<td>$K_{rp}$ 50</td>
<td>$K_{rd}$ 8</td>
</tr>
</tbody>
</table>

Figure 2.9. – Response of quadrotor using gains of the first GA experiment (with MSE fitness).
Figure 2.10. – Controls applied in GA experiment-1.

Figure 2.11. – Fitness change of GA algorithm for experiment-1.
Figure 2.12. – Response of quadrotor using gains of the second GA experiment (with complex fitness).

Figure 2.13. – Controls applied in GA experiment-2.

The best fitness values shown in figure 1.11 emphasize an important property of GA: the best fitness value is always increasing. This property is because the best individual chromosomes are always kept for crossover and mutation and never extinct. The average fitness values shown in the same figure oscillate because at each generation, individuals are subjected to mutation and crossover operations randomly. This generates a variety of fitness values, resulting in ran-
Comparison Values of GA Experiments

<table>
<thead>
<tr>
<th></th>
<th>Height</th>
<th>Roll</th>
<th>Pitch</th>
<th>Yaw</th>
</tr>
</thead>
<tbody>
<tr>
<td>$M_p$</td>
<td>Exp-1</td>
<td>0.098</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>Exp-2</td>
<td>0.3662</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$t_r$</td>
<td>Exp-1</td>
<td>0.45</td>
<td>0.31</td>
<td>0.38</td>
</tr>
<tr>
<td></td>
<td>Exp-2</td>
<td>0.33</td>
<td>0.31</td>
<td>0.32</td>
</tr>
<tr>
<td>$E_{ss}$</td>
<td>Exp-1</td>
<td>0</td>
<td>0.1046e^{-5}</td>
<td>-0.0073e^{-5}</td>
</tr>
<tr>
<td></td>
<td>Exp-2</td>
<td>0</td>
<td>0.1051e^{-5}</td>
<td>-0.0002e^{-5}</td>
</tr>
</tbody>
</table>

Figure 2.14. – Escherichia coli bacterium (http://www.howstuffworks.com).

domly changing average fitness. Table 1.4 show the overshoot, the rise time, and the steady-state error values of the quadrotor variables for the two experiments. These values will be used to compare GA with BFA and ESA algorithms.

2.5. Bacterial Foraging Algorithm

Similar to GA, Bacterial Foraging Algorithm (BFA) is a biologically inspired search algorithm that uses natural selection to solve optimization problems. In BFA, the basic individual that searches the environment is a bacterium that moves randomly to locate global optimum. The algorithm mimics the foraging behavior of Escherichia coli bacteria to perform the search. Similar to ESA and GA, BFA is a swarm intelligence algorithm where the decision of each naive individual looking for locations with high nutrient density affects other individuals in the colony resulting in an intelligent behavior of the whole swarm. An E. Coli bacterium has no intelligent action, it detects the density of the nutrient or toxic in its location using receptor proteins, and uses flagella to alternate between two types of motion (Figure 1.14).

E. coli bacteria colony uses four operations during its search process: che-
motaxis, swarming, reproduction, and elimination/dispersal. E. Coli bacterium individual applies chemotaxis (or chemotactic steps) to move in its environment using its set of tensile flagella to approach regions with high nutrient density, and avoid regions with high toxic density. The bacterium has two basic moves, forward move realized by turning the flagella counter clockwise, and tumble move realized by turning the flagella clockwise. Swarming is a process that aggregates the bacteria individuals in groups. High nutrient density positions stimulate the individuals to release a material that attracts bacteria individuals to each others. Swarming has the effect of decreasing the search time with search individuals moving in the wrong direction are attracted by individuals moving towards regions with high nutrient density. Reproduction and elimination processes are very important processes in E. coli bacteria colony. These processes simulate natural selection, and ensure that weak search individuals that have been in toxic or poor nutrient density locations for the past few steps expire. On the other hand, the healthiest search individuals that have been in high nutrient locations (thus searching in the right direction) split in two. The reproduction and elimination processes increase the number of bacteria in regions with high nutrient density, and decreases the number of bacteria in toxic regions, or in regions with low nutrient density. The final process of E. coli bacteria is the dispersal process. Sudden changes in the conditions of a region with high nutrient concentration (e.g. temperature), affect the bacteria colony resulting in the death of the individuals or dispersal of the swarm. Dispersal process is important for the search algorithm, it strengthens the algorithm chances to avoid local minima and discover new regions.

E. coli bacterium individual found for a period of time in a neutral region searches randomly for high nutrient density regions. The bacterium assures its random moves by alternating between tumbling and running forward. The same random search is applied when the individual is found in a toxic region or a negative nutrient gradient region. When the search individual detects a homogeneous concentration of nutrient, it alternates between tumbling and running but with lower tumble time and higher run distance compared with the previous situations. If the search individual is detecting a positive nutrient gradient, a reaction inside the bacterium decreases the tumbling time and increases the forward running time. This assures the motion of the bacterium in the direction of increasing nutrient gradient. Detecting nutrient and noxious gradients requires memory to save the present concentration, and compare it with previous nutrient/noxious concentrations. E. coli bacterium saves the nutrient/noxious concentrations for the past 3 seconds, compares them with the concentration of its present location, and decides the next move based on this result. Swarming process includes attraction and repletion forces among the colony individuals, where far individuals are attracted, and close individuals are repelled to avoid collision. Swarming is very important for the search, it assures the investigation of nutrient density in a wide region in spite that each individual has limited receptor range [67].
Table 2.5. – Parameters of Ecoli Algorithm Used to Tune the Quadrotor PD Controller

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dimensions</td>
<td>8</td>
</tr>
<tr>
<td>Stepsize</td>
<td>2</td>
</tr>
<tr>
<td>Population $S$</td>
<td>20</td>
</tr>
<tr>
<td>$P_{ed}$</td>
<td>0.25</td>
</tr>
<tr>
<td>$d_{att}$</td>
<td>0.1</td>
</tr>
<tr>
<td>$w_{att}$</td>
<td>0.2</td>
</tr>
<tr>
<td>$h_{rep}$</td>
<td>0.1</td>
</tr>
<tr>
<td>$w_{rep}$</td>
<td>10</td>
</tr>
<tr>
<td>$N_{c}$</td>
<td>10</td>
</tr>
<tr>
<td>$N_{s}$</td>
<td>6</td>
</tr>
<tr>
<td>$N_{re}$</td>
<td>6</td>
</tr>
<tr>
<td>$N_{ed}$</td>
<td>4</td>
</tr>
</tbody>
</table>

BFA starts by generating a colony of candidate solutions, and then applies iteratively the four E. coli processes. After each step taken by a search individual, the medium nutrient density is measured using a fitness function $J(\theta)$, and the individual variables, such as its health, along with the next step variables are changed accordingly. By representing environments rich with nutrient by $J(\theta) < 0$, neutral environments by $J(\theta) = 0$, and noxious environments by $J(\theta) > 0$, the algorithm is transformed to a search algorithm that iterates to minimize the function $J(\theta)$, where $\theta$ is the location of each bacterium. The position of each bacterium can be represented by the function

$$P(j, k, l) = \{\theta^i(j, k, l)|i = 1, 2, ..., S\}$$

(2.23)

here $S$ is the number of bacteria in the population, and $j$, $k$, and $l$ are respectively the chemotactic step, the reproduction step, and the elimination-dispersal event indexes. The function $J(i, j, k, l)$ is the cost value at the location of the $i^{th}$ bacterium. The next chemotactic step of a bacterium individual $\theta^i$ is

$$\theta^i(j + 1, k, l) = \theta^i(j, k, l) + C(i)\phi(j)$$

(2.24)

with $C(i) > 0$ the basic chemotactic step size of bacterium $i$, and $\phi(j)$ is a unit length random direction defining the direction after a tumble. Note that no information about the gradient $\nabla J(\theta)$ is provided. By representing the position and the next location of bacteria as shown above, the BFA is simplified to the following: If at step $\theta^i(j + 1, k, l)$, the cost function $J(i, j + 1, k, l)$ of bacterium $\theta^i$ is less than its previous value, a step of length $C(i)$ is taken at the same direction. As long as the cost function is reduced, and the number of steps taken in this direction is less than a number $N_s$, the bacterium keeps going in this direction. $N_s$, along with the maximum lifetime of a bacterium $N_c$, and other search variables used to tune the quadrotor controllers are shown in table 1.5. The simplified BFA is illustrated in figure 1.15. Swarming process between the colony individuals is ensured by using a potential function $J_{cc}(\theta, P)$ which provides the
attraction-repulsion forces among the bacteria. The potential function that drive each individual becomes

\[ J(\theta) = J(i, j, k, l) + J_{cc}(\theta, P) \]  

(2.25)
Table 2.6. – Values Of The Gains Found By Two E.Coli Experiments

<table>
<thead>
<tr>
<th></th>
<th>Height</th>
<th>Roll</th>
<th>Pitch</th>
<th>Yaw</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gain</td>
<td>$K_{zp}$</td>
<td>$K_{zd}$</td>
<td>$K_{rp}$</td>
<td>$K_{rd}$</td>
</tr>
<tr>
<td>Exp-1</td>
<td>37.76</td>
<td>9.55</td>
<td>33.97</td>
<td>7.96</td>
</tr>
</tbody>
</table>

with

$$J_{cc} (\theta, P) = \sum_{i=1}^{S} J_{cc}^i (\theta, \theta^i (j,k,l)) =$$

$$\sum_{i=1}^{S} \left[ -d_{att} \exp (-w_{att} \sum_{m=1}^{P} (\theta_m - \theta^i_m)^2) \right]$$

$$+ \sum_{i=1}^{S} \left( h_{rep} \exp (-w_{rep} \sum_{m=1}^{P} (\theta_m - \theta^i_m)^2) \right)$$  (2.26)

here $d_{att}$ and $h_{rep}$ are respectively the attraction and repulsion regions depth of a bacterium (set to be equal), $w_{att}$ is the diffusion rate of the attractant chemical, and $w_{rep}$ is the magnitude of the repellent effect. $P(j,k,l) = \theta^i (j,k,l) \mid i = 1, 2, \ldots, S$ is the set of bacteria locations, and $S$ is the total number of bacteria. By using the potential function shown above, the search individual moves in the direction of high nutrient distribution and avoids noxious regions, while trying to move closer towards the other cells but not too close so nutrient is spent rapidly.

To apply the reproduction step, all the individuals are sorted based on their health values, and the colony is divided in two groups. The individuals of the group with least health are eliminated, and replaced with a copy of the individuals found in the healthiest group. The reproduction step is taken each $N_{re}$ number of chemotactic steps. Dispersal process is applied by generating a random number for each individual and comparing this number with the dispersal probability of the algorithm. The search individual disperses if its dispersal number exceeds the dispersal probability of the algorithm. Dispersal process is repeated each $N_{ed}$ generations.

Authors in [68], and [69] use BFA to tune simple PID controllers. In our case, the PD gain values of the quadrotor found using BFA and the proposed fitness functions are shown in table 1.6. These gains are used to control the quadrotor, which produces a response shown in figures 1.16 through 1.20. The average and the maximum fitness values during 160 chemotactic steps are shown in figure 1.18. This figure shows that the algorithm spreads the bacteria to search the whole environment, as a result of dispersal step.

Similar to GA results, gains produced using two experiments of BFA are close to each other but not the same. This is because the algorithm starts with different
initial solutions at each run, and stops whenever the solution respects the stop criteria.

(a) Height response  
(b) Angles response

Figure 2.16. – Response of quadrotor using gains of the first E.Coli experiment.

Figure 2.17. – Controls applied in E.Coli experiment-1.
Figure 2.18. – Fitness change of E.Coli algorithm experiment-1.

Figure 2.19. – Response of quadrotor using gains of the second E.Coli experiment.
The response of quadrotor controlled with BFA gains shows higher rise time and higher steady state error compared with GA gains. Figure 1.18 illustrates the negative effect of swarming process on the search algorithm. While some bacteria have reached successfully high nutrient density locations, a great number of bacteria remain in bad regions, and try to attract individuals in good locations. This reproduces the oscillation in the best fitness values shown in the figure. Oscillations in the average fitness values indicate that individuals found in good locations loose their places as a result of inter-bacterial attraction forces. Table 1.7 show the comparison values of the quadrotor variables for the two experiments.
2.6. ESA Based Tuning Algorithm

To tune the PD controller of Pelican quadrotor, a zebra-lion search space is generated for each control variable. The $x$ and $y$ axes of each search space represents respectively candidate P and D gains of the proposed control variable. Two populations are created in each space, a zebra population looking for regions with the highest fitness values, and a lion population looking for zebra individuals to forage. The coordinates of each zebra individual are the candidate P and D gain axes of the proposed control variable. By moving through the environment, a zebra individual changes the values of candidate solutions (P-D gains) while changing its own coordinates. The dimension of each search space along with the length of steps taken by the individuals, and the minimum confrontation distances between individuals are different in each search space. Table 1.8 shows the search variables of each space.

A total number of 30 zebras and 25 lions, each with a maximum life of 10 iterations and a health value of 100, are created in each search space. The health of a lion increases by 1.6 ($AugP$) if it eats in an iteration, and decreases by 0.8 ($DecP$) if it doesn’t eat. After a lion attacks a zebra, injuries decreases the prey’s health by 0.8 ($Dec$) if it manages to escape. A successful step taking a zebra to a location with higher nutrient density increases its health by 1.6 ($AugA$). If the step takes the zebra to a worse location, its health decreases by 0.5 ($DecA$). The algorithm stops without finding a solution if the number of zebras in a given space drops below 4 (extinction number). If the health of any individual drops below 20 ($Eps_H$), this individual dies directly. If the fitness value of a location is 3 (0.25 for the second experiment), the nutrient is not enough to feed the zebra individual in this location, resulting in a decrease in its health by a factor of 0.8 ($Dec$). If a zebra individual makes a bad step that takes it to a location with a high fitness value of $Fit_G$ (4.5 in the first experiment and 0.45 in the second experiment), its health increases as if it has made a successful step. Search variables shown in Table 1.8 were all found by trial and error, and the starting solution candidates proposed by ESA are chosen randomly. The gain values found using ESA with both fitness functions are shown in Table 1.9. Table 1.10 presents the response of the quadrotor controlled with the gains given in the previous table.
Table 2.8. – Variables Used in ESA to Tune the Quadrotor PD Controller.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Dimension</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Height</td>
</tr>
<tr>
<td>x interval</td>
<td>[20, 90]</td>
</tr>
<tr>
<td>y interval</td>
<td>[7, 30]</td>
</tr>
<tr>
<td>x step size</td>
<td>10</td>
</tr>
<tr>
<td>y step size</td>
<td>2</td>
</tr>
<tr>
<td>Minimum distance</td>
<td>2</td>
</tr>
</tbody>
</table>

Table 2.9. – Values of The Gains Found By Two ESA Experiments

<table>
<thead>
<tr>
<th>Gain</th>
<th>Height</th>
<th>Roll</th>
<th>Pitch</th>
<th>Yaw</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$K_{zp}$</td>
<td>$K_{zd}$</td>
<td>$K_{rp}$</td>
<td>$K_{rd}$</td>
</tr>
<tr>
<td>Exp-1</td>
<td>75.37</td>
<td>14.54</td>
<td>74.8</td>
<td>10.68</td>
</tr>
</tbody>
</table>

Table 2.10. – Comparison Values of ESA Experiments

<table>
<thead>
<tr>
<th>$M_p$</th>
<th>Height</th>
<th>Roll</th>
<th>Pitch</th>
<th>Yaw</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exp-1</td>
<td>0.0835</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Exp-2</td>
<td>0.0565</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$t_r$</th>
<th>Height</th>
<th>Roll</th>
<th>Pitch</th>
<th>Yaw</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exp-1</td>
<td>0.43</td>
<td>0.28</td>
<td>0.38</td>
<td>0.68</td>
</tr>
<tr>
<td>Exp-2</td>
<td>0.57</td>
<td>0.37</td>
<td>0.38</td>
<td>0.68</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$e_{ss}$</th>
<th>Height</th>
<th>Roll</th>
<th>Pitch</th>
<th>Yaw</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exp-1</td>
<td>0</td>
<td>0.2117$e^{-6}$</td>
<td>-0.0584$e^{-6}$</td>
<td>0.0045$e^{-6}$</td>
</tr>
<tr>
<td>Exp-2</td>
<td>0</td>
<td>0.754$e^{-5}$</td>
<td>-0.0061$e^{-6}$</td>
<td>0.0005$e^{-6}$</td>
</tr>
</tbody>
</table>
(a) Height response

(b) Angles response

Figure 2.21. – Response of quadrotor using gains of the first ESA experiment.

Figure 2.22. – Controls applied in ESA experiment-1.
Figure 2.23. – Fitness change of ESA experiment-1.

Figure 2.24. – Response of quadrotor using gains of the second ESA experiment.
Figures 1.21 to 1.25 illustrating the quadrotor response with ESA tuned controllers show small overshoot and rise time compared with GA and BFA gains. The algorithm can be summarized relying on figure 1.23 showing the best fitness values across iterations, where all the individuals are spread all over the space searching for better foraging region. This figure shows also that individuals in good locations may leave their places during the search, but always find a better place or return to their original location.

2.7. Result Analysis

Comparison value tables show that using a complicated fitness function has decreased the undesirable deflections in the response and reduced the rise time. The tables show also that the use of a simple fitness function also gives good results and with shorter tuning time. When ESA, GA, and BFA are compared according to their convergence time, GA is seen to outperform the other algorithms. This is a natural consequence of using digital candidate solutions, where changing one bit results in a huge jump in the number (changing the third bit of 1011101 (109) transforms the number to 1001101 (77)). This means that search individuals can apply very large steps, and steps are not bounded with a constant step size like in BFA and ESA. ESA is second concerning the convergence time. Moreover, ESA found gains which produce the overshoot and rise time values that compete with the values found using GA.

By comparing the fitness values of the three algorithms, one can realize that GA and ESA have high average fitness values. This means that the majority of
search individuals have found optimal regions. The fitness graph of each algorithm emphasizes its properties; elitism of the best chromosomes in GA, swarming between bacteria of BFA, and the permanent movement of zebras in ESA are realized clearly by reading the fitness graphs.

In order to improve ESA, swarming process between bacteria of the same region in BFA can be applied to zebras of the same herd. This is expected to make the search faster. Moreover, breeding of zebras could be improved by simulating the crossover process in GA. A more realistic breeding process can be made by introducing mating process. Mating can be accomplished when two healthy individuals meet, and a baby individual is born at their location. Finally, note that it is possible for ESA to continue the search with only zebra species, even when lion species has extinct. This however increases the search time, as the non-healthy individuals take 10 iterations (a complete life) to die.
3. Passive and Active Fault Tolerant Control of Quadrotor UAVs

In this chapter, Sliding Mode Control (SMC) theory will be used to develop Passive and Active Fault Tolerant Controllers for AscTec Pelican UAV quadrotor. Passive SMC FTCs use the inherent robustness of Sliding Mode theory to compensate partial loss of effectiveness in the drone actuators. Active FTC scheme uses a Fault Detection and Isolation (FDI) unit based on the Kalman Filter (KF) outputs to estimate actuator faults online. The estimated fault magnitudes are used to recover the amount of control effort lost through an adaptive SMC controller. The proposed controllers are tuned using Ecological Systems Algorithm (ESA) developed in the previous chapter. MATLAB/SIMULINK simulation environment is used to check the robustness of the proposed controllers in the presence of multiple partial loss of motor speed faults.

3.1. State Space Representation of Pelican Quadrotor

In this section, Pelican quadrotor dynamics are transformed to the state space form. By using equations (1.1) to (1.6), and by taking $\mathbf{x} = [x_1 \ x_2 \ x_3 \ x_4 \ x_5 \ x_6 \ x_7 \ x_8 \ x_9 \ x_{10} \ x_{11} \ x_{12}]^t$ as the state vector, $\mathbf{U} = [U_1 \ U_2 \ U_3 \ U_4]^t$ as the control input, and $\mathbf{y} = \mathbf{C}\mathbf{x} = [x \ y \ z \ \dot{\phi} \ \dot{\theta} \ \dot{\psi}]^t$ as the output vector. $\mathbf{f}(\mathbf{x})$ is a $12 \times 1$ column
vector, and matrix $\mathbf{g}(\mathbf{x})$ is $12 \times 4$.

\[
\mathbf{f}(\mathbf{x}) = \begin{bmatrix}
\frac{x_2}{m} \\
-K_{f_{tx}} x_2 \\
\frac{x_4}{m} \\
-K_{f_{ty}} x_4 \\
\frac{x_6}{m} \\
-K_{f_{tz}} x_6 - g \\
\frac{I_y - I_x}{I_z} x_{10} x_{12} + \frac{I_{rotor}}{I_z} x_{10} \gamma - \frac{K_{f_{ax}}}{I_z} x_8^2 \\
\frac{I_x - I_z}{I_y} x_8 x_{12} - \frac{I_{rotor}}{I_y} x_8 \gamma - \frac{K_{f_{ay}}}{I_y} x_{10}^2 \\
\frac{I_x - I_z}{I_y} x_8 x_{10} - \frac{K_{f_{az}}}{I_z} x_{12}^2
\end{bmatrix}
\] (3.1)

\[
\mathbf{g}(\mathbf{x}) = \begin{bmatrix}
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
\frac{\sin x_{11} \sin \gamma + \cos x_{11} \sin x_9 \cos \gamma}{m} & 0 & 0 & 0 \\
\frac{- \cos x_{11} \sin \gamma + \sin x_{11} \sin x_9 \cos \gamma}{m} & 0 & 0 & 0 \\
\frac{\cos x_9 \cos \gamma}{m} & 0 & 0 & 0 \\
0 & 0 & 0 & 1 \frac{1}{I_z} \\
0 & 0 & 0 & \frac{1}{I_y} \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0
\end{bmatrix}
\] (3.2)

\[
\mathbf{C} = \begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0
\end{bmatrix}
\] (3.3)

The state space form of quadrotor dynamics becomes

\[
\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}) + \mathbf{g}(\mathbf{x})\mathbf{u}
\] (3.4)

Note here that the translation drag $K_{f_{ti}}$ and the friction aerodynamic forces $K_{f_{ai}}$ for $i = x, y, z$ are discarded in this equation. Values of the real Asctec Pelican quadrotor testbed available at UAEU university are used to restrict the controller design in SIMULINK. The velocities of the quadrotor motors are saturated.
between 1200 and 5000 rpm (125-523 rad/sec), \(U_1\) control is bounded within the maximum thrust of the quadrotor (15.7 N), and the quadrotor angles are between \(-18^\circ\) and \(+18^\circ\) (-0.31 - +0.31 rad). The maximum motor voltages are 12 volts. The quadrotor motor speeds and its controls are related by the control effectiveness matrix \(B\) as

\[
\begin{bmatrix}
U_1 \\
U_2 \\
U_3 \\
U_4
\end{bmatrix} = B \begin{bmatrix}
\Omega_1^2 \\
\Omega_2^2 \\
\Omega_3^2 \\
\Omega_4^2
\end{bmatrix}^t
\]

with

\[
B = \begin{bmatrix}
b & b & b & b \\
0 & -bl & 0 & bl \\
-bl & 0 & bl & 0 \\
d & -d & d & -d
\end{bmatrix}
\]

The fault interpreted is a partial loss of effectiveness of one or two motors. The faults affect directly rotor speeds which results in a sudden decrease in the speed of the faulty motor(s), and the faulty control vector becomes

\[
u_f = BF \begin{bmatrix}
\Omega_1^2 \\
\Omega_2^2 \\
\Omega_3^2 \\
\Omega_4^2
\end{bmatrix}^t
\]

with \(F\) is a 4x4 diagonal matrix containing the fault magnitude which varies between 0 and 1 (1 : no fault, 0.5 : 50% fault, 0 : 100% fault).

### 3.2. Passive FTC Sliding Mode Controller Design

Although Sliding Mode Control is not straightforward, its main idea is intuitive and very simple. The first step in Sliding Mode design is to define a surface in the phase plane so that the controlled system exhibits a desired response when trapped on. The desired response is defined by the designer, and could be as simple as making the states error to decrease exponentially to zero. Whenever the state trajectory of the controlled system reaches the sliding surface, it slides on this surface until the states error becomes zero in finite time, thus ensuring the stability of the system. The controller part responsible for maintaining the system on the sliding surface (i.e. equivalent control) is designed to achieve \(\dot{s}\). The second step in Sliding Mode design is made to ensure that the states of the system are attracted to the sliding surface. A switching control law is designed using discontinuous functions so that the system states always navigate towards the sliding surface, and then are maintained on the surface thereafter by the equivalent controller. The SMC law has then two components, equivalent and discontinuous control laws

\[
u = u_{eq} + u_{dis}
\]
3.2.1. Regular SMC Design

In this part, a regular SMC controller is designed for the quadrotor. Let \( \mathbf{x}_d \) be the desired states of the quadrotor, and \( \mathbf{e} \) be the state error vector with \( e_i = i_d - i \) for \( i = z, \phi, \theta, \psi \). The sliding surface is chosen as \( s_i = \hat{e}_i + c_i e_i \), which means that when the system reaches the sliding surface \( (s_i = 0) \), the state error converges exponentially to zero \( (\dot{e}_i = -c_i e_i) \). \( c_i \) is the sliding surface slope which affects the conversion speed of the state error to zero and defined for the quadrotor as \( \mathbf{c} = [c_z \ c_\phi \ c_\theta \ c_\psi]^t \).

The equivalent controller responsible for maintaining the system on the sliding surface is chosen so that \( \dot{s}_i \) is zero when the system dynamics are on the sliding surface (i.e. \( s_i = 0 \) and \( \dot{e}_i = -c_i e_i \)). The dynamics of the sliding surfaces are \( \dot{s}_i = s_i (\dot{e}_i + c_i e_i) = i_d - i + c_i \dot{e}_i \). By setting \( \dot{s}_i = 0 \) and using the quadrotor dynamics (equations (1.3) to (1.6)), the equivalent control laws are found as

\[
\begin{align*}
\dot{z}_{eq} &= \frac{m}{\cos \theta \cos \phi} [c_z \dot{\hat{e}}_z + \ddot{\hat{z}}_d + g] \\
\dot{\phi}_{eq} &= I_x [c_\phi \dot{\hat{e}}_\phi + \ddot{\hat{\phi}}_d - \frac{I_y-I_z}{I_x} \ddot{\hat{\theta}}_d - \frac{I_y}{I_x} \dot{\hat{\phi}}_d] \\
\dot{\theta}_{eq} &= I_y [c_\theta \dot{\hat{e}}_\theta + \ddot{\hat{\theta}}_d - \frac{I_z-I_x}{I_y} \ddot{\hat{\psi}}_d + \frac{I_z}{I_y} \dot{\hat{\phi}}_d] \\
\dot{\psi}_{eq} &= I_z [c_\psi \dot{\hat{e}}_\psi + \ddot{\hat{\psi}}_d - \frac{I_x-I_y}{I_z} \dot{\hat{\phi}}_d]
\end{align*}
\]

(3.9)

Note that by assuming that the desired values are changing slowly, the acceleration of the desired states \( \ddot{i}_d \) (with \( i = z, \phi, \theta, \psi \)) is taken as zero.

With the equivalent controls above the states of the system remain on the sliding surface, and the closed-loop system behaves as a reduced-order system with motion independent of the control [70].

Let \( V_i = \frac{1}{2} s_i^2 \) be Lyapunov functions associated with the quadrotor. We intend to design a control law that makes the distance between the sliding surface and all the system trajectories decreases. In other words, the derivatives of the Lyapunov functions should be negative \( \dot{V}_i = s_i \dot{s}_i < -\eta_i |s_i| \). This is called the reachability condition, which means that the trajectory of the system states must always points towards the sliding surface. \( \eta_i \) is a small positive design scalar that ensures the satisfaction of the reachability condition despite the presence of some uncertainty and disturbance.

The discontinuous control is designed so that the reachability condition is fulfilled. In other words, the system states could start at any position in the plane, and it is the discontinuous part of the controller which is responsible to bring the states on the sliding surface. The discontinuous controller pulls the states upwards if they are located below the sliding manifold, and pushes them downwards if they are above. The signum function or the saturation function (to avoid discontinuities) can be used to form the controller. The discontinuous control is chosen as

\[
\dot{u}_{dis} = -k_i \text{sat}(s_i)
\]

(3.10)

where \( \mathbf{k} = [k_z \ k_\phi \ k_\theta \ k_\psi]^t \) is a positive gain affecting the conversion speed of
the discontinuous control, and that should be chosen greater than $\eta$ to ensure the reachability condition despite the presence of disturbance. The saturation function is defined as

$$\text{sat}(s) = \begin{cases} 
1, & \text{if } s > 1 \\
s, & \text{if } |s| \leq 1 \\
-1, & \text{if } s < -1
\end{cases}$$

(3.11)

The discontinuous control is responsible for driving the states towards the sliding surface under modeling uncertainty and disturbance, and so it determines the robustness of SMC system. Note that when the system states reach the sliding surface, the continuous control becomes more dominant than the discontinuous control (because $s = 0$) and the system is driven to the steady state [70].

Simulation tests have shown that with the proposed controller, $\theta$ and $\psi$ angles of the quadrotor exhibit a small remaining error when one or more motors are affected by faults (less than 5% with 55% fault). This remaining error rises only when actuator faults exist, and no error is seen when the quadrotor motors are working normally. To compensate for the remaining error, the SMC is augmented by an integrator to become

$$\begin{align*}
U_1 &= \frac{m}{\cos \theta \cos \phi} [c_\phi \dot{\psi} + \ddot{\phi} - k_\phi \text{sat}(s_\phi)] - k_\phi \text{sat}(s_\phi) + K_I \int e_\phi(\tau) d\tau \\
U_2 &= I_x [c_\phi \dot{\psi} + \ddot{\phi} - k_\phi \text{sat}(s_\phi)] - k_\phi \text{sat}(s_\phi) + K_I \int e_\phi(\tau) d\tau \\
U_3 &= I_y [c_\theta \dot{\theta} + \ddot{\phi} - k_\theta \text{sat}(s_\theta)] - k_\theta \text{sat}(s_\theta) + K_I \int e_\theta(\tau) d\tau \\
U_4 &= I_z [c_\psi \dot{\psi} + \ddot{\psi} - k_\psi \text{sat}(s_\psi)] - k_\psi \text{sat}(s_\psi) + K_I \int e_\psi(\tau) d\tau
\end{align*}$$

(3.12)

where $K_I = [K_{I_z} \ K_{I_\phi} \ K_{I_\theta} \ K_{I_\psi}]$ is the integrator gain vector.

### 3.2.2. Cascaded SMC Design

To further improve the quadrotor response, a Cascaded SMC controller is developed. In [14] and [15], cascaded SMC’s are developed using multi-layer interlaced sliding surfaces. Here, the controller has two SMC control loops: an outer loop controlling the altitude and attitude variables ($z$, $\phi$, $\theta$, and $\psi$), and an inner loop controlling the velocities. Cascaded SMC of the quadrotor altitude is shown in Figure 2.1.

![Cascaded SMC control of the height.](image)
The sliding surface of the inner loop controller is chosen as [9], [71]

\[ s_{i(inner)} = e_{iv} + c_{i(inner)} \int e_{iv}(\tau) d\tau \tag{3.13} \]

where \( e_{iv} \) is the velocity error vector or \( e_{iv} = \dot{e}_i = i_d - \dot{i} = u_i(outer) - \dot{i} \) (according to Figure 2.1), and \( c_{i(inner)} \) is a positive design vector. Following the same procedure of regular SMC design, the inner loop SMC controller is found as

\[
\begin{align*}
U_1 &= m \cos \theta \cos \phi \left[ c_z(e_{iv}) e_{iz} + \dot{z}_d + g \right] - \dot{k}_z(s_{i(inner)}),
U_2 &= I_x \left[ c_\phi(e_{iv}) e_{i\phi} + \dot{\phi}_d - \frac{I_y - I_z}{I_x} \dot{\psi} + \frac{L_{outar}}{I_x} \dot{\theta}\gamma \right] - \dot{k}_ \phi(s_{\phi(inner)}),
U_3 &= I_y \left[ c_\theta(e_{iv}) e_{i\theta} + \dot{\theta}_d - \frac{I_z - I_x}{I_y} \dot{\psi} + \frac{L_{outar}}{I_y} \dot{\phi}\gamma \right] - \dot{k}_ \theta(s_{\theta(inner)}),
U_4 &= I_z \left[ c_\psi(e_{iv}) e_{i\psi} + \dot{\psi}_d - \frac{I_x - I_y}{I_z} \dot{\phi} - \dot{k}_ \psi(s_{\psi(inner)}) \right]
\end{align*}
\tag{3.14} \]

Terms \( e_{iv} = u_i(outer) - \dot{i} \) (input of the inner SMC controller) and \( u_i(outer) \) (input of the inner loop) will be found by designing the Sliding Mode controller of the outer loop. The sliding surface of the outer loop controller is chosen as

\[ s_{i(outer)} = e_i + c_i(outer) \int e_i(\tau) d\tau \tag{3.15} \]

where \( e_i = i_d - \dot{i} \), and \( c_i(outer) \) is a positive vector. Following the previous SMC design procedure, the control input of the outer loop is found as

\[ u_i(outer) = \dot{x}_i - c_i(outer) e_i - \dot{k}_i(outer) sat(s_{i(outer)}) \tag{3.16} \]

Similar to regular SMC design, the acceleration of the state vector \( \ddot{i}_d \) is taken zero. The controller equations contain the velocity matrix \( \dot{i} \) which is difficult to calculate practically in presence of noise and disturbance. The reader is directed to [72], [73], and [74] for more information on the differentiation of noisy signals.

### 3.3. Active FTC Sliding Mode Controller Design

In Active FTC Sliding Mode Controller, the amount of faults affecting the quadrotor actuators is used to recover the loss in control effort through adaptive means. This requires the design of a Fault Detection and Identification (FDI) unit that is able to estimate online the fault type and severity. Extended Kalman Filter is chosen to be the core of the FDI unit. The Kalman Filter (KF), Known also as Linear Quadratic Estimation (LQE) algorithm, is a linear algorithm that estimates recursively the states of a linear system using noisy sensor information. The Extended Kalman Filter (EKF) is an improved KF that can be used with nonlinear systems. The EKF approximates the system nonlinearities by linearizing
at each iteration the nonlinear stochastic system equations about the current estimation.

Assuming that all faults and anomalies affecting the quadrotor UAV can be represented by a time varying additive function $F(x, u, t)$, the quadrotor dynamics becomes

$$
\begin{align*}
\dot{x} &= f(x) + g(x)u(t) + F(x, u, t) + \omega \\
y &= Cx + \vartheta
\end{align*}
$$

(3.17)

where $\omega_k$ is the state noise, and $\vartheta_k$ is the measurement noise. These noises are assumed to be gaussian, centered white noises with variance $Q_k$ and $R_k$ respectively.

Note that all faults are represented as an additive function and not by changes affecting the state-space equations of the system. The aerodynamic effect is taken into consideration while designing the Active FTC so that the FDI unit is not mislead in severe maneuvers and faulty cases.

### 3.3.1. State Estimation Using Extended Kalman Filter

Similar to any observer equation, the estimation of a Kalman Filter at an instant is equal to the system dynamic equations at that instant plus a corrective part. The corrective part is formed of a gain multiplying the output residual. Unlike observers that use offline calculated gains evaluated at each step, KF's calculate their gains at each step based on previous predictions and measurements. Kalman filters are recursive processes with two steps, where the algorithm uses prior knowledge of the states to predict the estimates, and then uses current measurement data to correct the estimates. KF repeats the prediction and correction steps online and recursively during the run to get closer estimation at each step [75].

By assuming that we have sampled observations of the system with discrete time interval, the discrete equivalent of the quadrotor becomes

$$
\hat{x}_{k+1|k} = F_k \hat{x}_k + G_k u_k + \omega_k
$$

(3.18)

where

$$
F_k = e^{A\Delta T}
$$

(3.19)

and

$$
G_k = \int_0^{\Delta T} e^{At} B \, dt
$$

(3.20)

with $\Delta T$ the sample time, matrix $A$ is the linearization of the model around the current estimation [76], and matrix $B$ is the linearization of the model around
the control vector

\[ A_{i,j} = \frac{\partial f(i)}{\partial x(j)} \big|_{(\tilde{x}_{k-1}, u_{k-1}, 0)} \]  \hspace{1cm} (3.21)

\[ B_{i,j} = \frac{\partial g(i)}{\partial u(j)} \big|_{(\tilde{x}_{k-1}, u_{k-1}, 0)} \]  \hspace{1cm} (3.22)

The algorithm starts by a prediction step where the state estimate \((\hat{x}_{k+1|k})\) and the \textit{a priori} covariance matrix \((P_{k+1|k})\) are assigned

\[ \hat{x}_{k+1|k} = F_k \hat{x}_k + G_k u_k \]  \hspace{1cm} (3.23)

\[ P_{k+1|k} = F_k P_k |_{k} F_k^t + Q_k \]  \hspace{1cm} (3.24)

The Kalman gain \((L_{k+1})\) is then calculated, and used along with the measured states \((y_{k+1})\) and the prediction residue \((\tilde{y}_{k+1|k})\) to correct the algorithm predictions; next, the \textit{a posteriori} covariance matrix \((P_{k+1|k+1})\) is computed

\[ L_{k+1} = P_{k+1|k} C^t (R_k + C P_{k+1|k} C^t)^{-1} \]  \hspace{1cm} (3.25)

\[ \tilde{y}_{k+1|k} = y_{k+1} - \hat{y}_{k+1|k} \]  \hspace{1cm} (3.26)

\[ \hat{x}_{k+1|k+1} = \hat{x}_{k+1|k} + L_{k+1} \tilde{y}_{k+1|k} \]  \hspace{1cm} (3.27)

\[ P_{k+1|k+1} = (I - L_{k+1} C) P_{k+1|k} (I - L_{k+1} C)^t + L_{k+1} R_k L_{k+1}^t \]  \hspace{1cm} (3.28)

Next, results of the correction step are used in the next prediction step to find updated estimation, which by its turn is used in the next correction step. Prediction and correction steps are repeated recursively during the process to get more precise estimation of the system states despite the use of noisy measurement data.
The A matrix of the quadrotor is

\[
A = \begin{bmatrix}
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & -k_{f_{tx}}/m & 0 & 0 & 0 & 0 & a_{10} & 0 & a_{11} & 0 & a_{12} & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & -k_{f_{ty}}/m & 0 & 0 & a_{13} & 0 & a_{14} & 0 & a_{15} & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & -k_{f_{tz}}/m & a_{16} & 0 & a_{17} & 0 & a_{18} & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}
\]  

(3.29)

with

\[
a_1 = -k_{f_{ox}}/I_x x_8 \\
a_2 = (I_y - I_z)/I_x x_{12} \\
a_3 = (I_y - I_z)/I_x x_{10} \\
a_4 = (I_z - I_x)/I_y x_{12} \\
a_5 = -k_{f_{oy}}/I_y x_{10} \\
a_6 = (I_z - I_x)/I_y x_{8} \\
a_7 = (I_x - I_y)/I_z x_{10} \\
a_8 = (I_x - I_y)/I_z x_{8}
\]  

(3.30)
\begin{align*}
  a_9 &= -\frac{k_{fax}}{I_z} x_{12} \\
  a_{10} &= -\cos x_7 \sin x_{11} - \sin x_7 \cos x_{11} \sin x_9 \frac{U_1}{m} \\
  a_{11} &= -\cos x_7 \cos x_9 \cos x_{11} \frac{U_1}{m} \\
  a_{12} &= -\left(\cos x_{11} \sin x_7 - \sin x_{11} \cos x_7 \sin x_9\right) \frac{U_1}{m} \\
  a_{13} &= \cos x_{11} \cos x_7 + \sin x_{11} \sin x_9 \sin x_7 \frac{U_1}{m} \\
  a_{14} &= -\cos x_7 \cos x_9 \sin x_{11} \frac{U_2}{m} \\
  a_{15} &= \left(\sin x_{11} \sin x_7 + \cos x_7 \sin x_9 \sin x_{11}\right) \frac{U_1}{m} \\
  a_{16} &= \sin x_7 \cos x_9 \frac{U_1}{m} \\
  a_{17} &= -\cos x_7 \sin x_9 \frac{U_1}{m} \\
  a_{18} &= 0
\end{align*}

(3.32)

\begin{align*}
  a_{13} &= \cos x_{11} \cos x_7 + \sin x_{11} \sin x_9 \sin x_7 \frac{U_1}{m} \\
  a_{14} &= -\cos x_7 \cos x_9 \sin x_{11} \frac{U_2}{m} \\
  a_{15} &= \left(\sin x_{11} \sin x_7 + \cos x_7 \sin x_9 \sin x_{11}\right) \frac{U_1}{m} \\
  a_{16} &= \sin x_7 \cos x_9 \frac{U_1}{m} \\
  a_{17} &= -\cos x_7 \sin x_9 \frac{U_1}{m} \\
  a_{18} &= 0
\end{align*}

(3.33)

### 3.3.2. Fault Estimation

Kalman Filters have been used widely in Fault Detection and Isolation for dynamic systems. KF was merged with intelligent methods such as Fuzzy logic and Neural Networks, where the KF residual moving average is used in fault detection [77]. Here, a hybrid neuro-fuzzy inference system insures fault classification. In [78], a powerful statistical technique that detects faults by detecting irregularities of the residuals generated by an Unscented Kalman Filter (UKF) is developed. Sensor faults of a UAV are isolated using two-stage Kalman filters (TSKF) in [79]. A linear adaptive TSKF algorithm is also developed to estimate actuator loss of control effectiveness and stuck faults. Authors in [80] estimate the health degradation of an aircraft gas turbine engine to detect its sensor faults. In [81], Augmented Kalman Filter is used to develop a fault detection and diagnosis technique. Loss of actuator effectiveness faults are modeled as random walk processes, and then treated as additional states which makes the fault estimation unbiased.

To detect, identify, and estimate the actuator faults of the quadrotor, the fault
effect on the control signals is evaluated. Controls generated by the SMC using the Kalman Filter outputs are filtered and searched for any instantaneous change. The filtered control signals contain a plenty of information about the fault location and magnitude. Experiments show that the controls have distinctive response to each actuator fault. A fault injected at motor 1, for example, produces a positive change in $U_4$ and a negative change in $U_3$. Fault detection and identification is based on the following IF/THEN functions

- IF $(\Delta U_3 < 0$ and $\Delta U_4 > 0)$ THEN M1 fault
- IF $(\Delta U_2 > 0$ and $\Delta U_4 < 0)$ THEN M2 fault
- IF $(\Delta U_3 > 0$ and $\Delta U_4 > 0)$ THEN M3 fault
- IF $(\Delta U_2 < 0$ and $\Delta U_4 < 0)$ THEN M4 fault
- ELSE No fault

where $\Delta U_i$ is the rate of change of control $i$ calculated at time $t$ as $\Delta U_i(t) = U_i(t) - U_i(t - \Delta T)$, and $\Delta T$ is the sample time.

The configuration of the Fault Detection and Isolation (FDI) unit is shown in Figure 2.2. The control signals are filtered with an 8th order Butterworth low-pass filter of cut-off frequency 2 rad/s to remove the high frequency noise before being used in the FDI. Next, a fault detection unit uses the filtered signals to detect the fault location. Finally, the fault location and the $U_4$ signal are fed into the fault estimation unit to estimate the fault magnitude using an array of lookup tables (table 2.1). These tables are tuned offline for each motor using the $U_4$ signal.

![Fault Detection and Isolation Unit](image.png)

Figure 3.2 – FDI unit configuration.

The table 2.1 is read as follows: If a fault in M1 is detected and $\Delta U_4 = 0.00076$, then the fault magnitude is 15%. These lookup tables are formed by injecting different fault magnitudes to each motor and measuring the magnitude of $\Delta U_4$ signal.

Simulation results show that the FDI unit is able to detect with acceptable precision multiple successive faults despite the presence of states and measurement
Table 3.1. – Look-Up Tables Used To Estimate Fault Magnitude

<table>
<thead>
<tr>
<th>M</th>
<th>15%</th>
<th>25%</th>
<th>35%</th>
<th>45%</th>
<th>55%</th>
</tr>
</thead>
<tbody>
<tr>
<td>M1</td>
<td>0.00076</td>
<td>0.00154</td>
<td>0.00271</td>
<td>0.00458</td>
<td>0.00783</td>
</tr>
<tr>
<td>M2</td>
<td>0.0007688</td>
<td>0.0015479</td>
<td>0.002727</td>
<td>0.0045854</td>
<td>0.0078424</td>
</tr>
<tr>
<td>M3</td>
<td>0.0007618</td>
<td>0.001544</td>
<td>0.0027157</td>
<td>0.0045854</td>
<td>0.0078424</td>
</tr>
<tr>
<td>M4</td>
<td>0.00076268</td>
<td>0.0015462</td>
<td>0.0024827</td>
<td>0.0045854</td>
<td>0.0078424</td>
</tr>
</tbody>
</table>

noise. It is important to state that when multiple successive faults occur, the fault estimation exhibits an error of 5%. This error has minimal effect on the system and the robustness of the SMC is able to tackle for this uncertainty.

3.3.3. Design of Adaptive SMC

After the fault is detected and estimated, it becomes possible to use this fault estimation to adapt the controller in order to minimize the fault effects. An active fault tolerant control SMC could be developed by updating the SMC designed previously with the fault magnitude. Let us denote by \( u_R(t) \) the control part that remains after a fault has occurred (used previously as \( u \) in (2.17)). The faults affect the speeds of the motors, and result in a sudden drop in the effectiveness of the motors. The remaining control after the fault occurrence is \( u_R = u - u_f \), with \( u \) is the adaptive controller needed for the safe control of the faulty quadrotor, and \( u_f \) is the control part lost when the fault occurs. Our aim is to find the control \( u \) and apply it to control the faulty system. The equivalent part of the remaining control \( u_{eq\ R} \) is equal to the regular SMC presented in equation (2.12). Note that the integrator gain in the Active FTC is taken as zero.

The discontinuous control of the regular SMC (equation (2.10)) is used to form the discontinuous part of the adaptive controller. The design of the adaptive controller is finalized by finding a parametric interpretation of \( u_f \) similar to the methods used in [8] and [34]. The relation between the quadrotor’s controls and motor speeds is expressed by equation (2.5). For faulty situation, the control-speed relation (equation (2.5)) is updated by the fault magnitude

\[
  u_R = B_R\Omega = [B - B_f]\Omega \tag{3.34}
\]

with \( \Omega = [\Omega^2_1\ \Omega^2_2\ \Omega^2_3\ \Omega^2_4]^t \), \( B_R \) is the remaining control effectiveness matrix, and \( B_f \) is the faulty control effectiveness matrix defined as

\[
  B_f = \begin{bmatrix}
  F_1b & F_2b & F_3b & F_4b \\
  0 & -F_2bl & 0 & F_4bl \\
  -F_1bl & 0 & F_3bl & 0 \\
  F_1d & -F_2d & F_3d & -F_4d 
  \end{bmatrix} \tag{3.35}
\]
Where the elements of diagonal matrix $F_{4 \times 4}$ are
\[
F_i = \begin{cases} 
0, & \text{if there is no fault} \\
\frac{E_i}{100}, & \text{if a fault of percentage } E_i \text{ exists}
\end{cases} \tag{3.36}
\]
with $E_i, i = 1, 2, 3, 4$ is the percentage of fault affecting motor $i$.

We can express $B_f$ in terms of $B$ as $B_f = BF$, and the faulty control becomes
\[
u_f = BF\Omega \tag{3.37}
\]

Control correction is made by adding the lost controls to the remaining controls $u = u_R + u_f$ or
\[
\begin{align*}
U_1 &= u_{Rz} + b(F_1\Omega^2_1 + F_2\Omega^2_2 + F_3\Omega^2_3 + F_4\Omega^2_4) \\
U_2 &= u_{R\phi} + bl(-F_1\Omega^2_1 + F_3\Omega^2_3) \\
U_3 &= u_{R\theta} + bl(-F_2\Omega^2_2 + F_4\Omega^2_4) \\
U_4 &= u_{R\psi} + d(F_1\Omega^2_1 - F_2\Omega^2_2 + F_3\Omega^2_3 - F_4\Omega^2_4)
\end{align*} \tag{3.38}
\]

$u$ here is the adaptive controller capable of driving the quadrotor under fault, and $u_R$ are the controls at equation 2.12 with no integral part. Note that $\Omega^2_i$ which is the squared speed of motor $i$ is found from the remaining controls using the following equations
\[
\begin{align*}
\Omega^2_1 &= \frac{1}{4b}u_{Rz} - \frac{1}{2bl}u_{R\theta} + \frac{1}{4d}u_{R\psi} \tag{3.39} \\
\Omega^2_2 &= \frac{1}{4b}u_{Rz} - \frac{1}{2bl}u_{R\phi} - \frac{1}{4d}u_{R\psi} \tag{3.40} \\
\Omega^2_3 &= \frac{1}{4b}u_{Rz} + \frac{1}{2bl}u_{R\theta} + \frac{1}{4d}u_{R\psi} \tag{3.41} \\
\Omega^2_4 &= \frac{1}{4b}u_{Rz} + \frac{1}{2bl}u_{R\phi} - \frac{1}{4d}u_{R\psi} \tag{3.42}
\end{align*}
\]

### 3.4. Tuning The Controllers

Sliding Mode Controllers designed in the previous sections require a good selection of the sliding surface slope vector $c$, and the discontinuous control convergence vector $k$. Good selection of these vectors ensures the reachability condition ($\dot{V} = ss < -\eta|s|$, with $\eta$ is positive real), and results with good robustness against disturbances and faults. On the other hand, poor selection of the gain matrices may result in unstable controllers. Classic SMC design methods use the reachability condition equations to find constraints on $c$ and $k$ vectors. Any values that respect these constraints ensure the stability of the controller,
but do not ensure the minimum conversion time. Moreover, classic SMC design methods require high mathematical skills and careful solution procedure.

In this section, Ecological Systems Algorithm (ESA) developed in the previous chapter will be used to tune the proposed passive controllers. The active SMC controller is a regular SMC updated with adaptive part, and uses the same regular SMC gains found using ESA. The algorithm imitates the motion of a zebra herd in the Savannah to suggest random values for $c$ and $k$ vectors. The suggested gains are then checked and weighted using a fitness function based on the output response of the quadrotor. Gain matrices are then updated using ecological rules and checked again. The last step is repeated iteratively until gains giving the highest fitness are found. Controller tuning process of the quadrotor is shown in Figure 2.3.

![Figure 3.3. – Tuning the controllers using ESA.](image)

### 3.4.1. Tuning Process

Gain vectors of the Regular SMC controller have four elements each. This means that we have eight gains to tune, one $c$ and one $k$ gain for each control variable $z$, $\phi$, $\theta$, and $\psi$. On the other hand, Cascaded SMC controller has a two-loop configuration, which means that the number of gain matrices for inner and outer loops is sixteen. Tuning process starts by assigning a search space containing zebra and lion herds for each control variable. A typical search space is shown in Figure 2.4 which represents the search space of the height controller with zebra and lion individuals spread across the space. Here, the $x$-coordinates of zebra individuals are candidate gain $c_z$, while the $y$-coordinates are candidate gain $k_z$. 

76
Gains of the control variables are used to control the quadrotor system with step function inputs. Position errors of the quadrotor response are calculated, and used to validate a Mean Squared Error based fitness function (equation (2.44)). The more this fitness function gives high value, the more \( c \) and \( k \) are efficient.

\[
Fitness_i = \frac{1}{\text{MSE}_i + 0.1} \tag{3.43}
\]

\[
\text{with } \text{MSE}_i = \frac{\sum_{t=0}^{\text{tf}} e_{it}^2}{\text{size}(E_i)} \tag{3.44}
\]

\( i \) is the control variable \((z, \phi, \theta, \text{ or } \psi)\), and \( e_{it} \) is the error of variable \( i \) at time \( t \).

Step functions used to validate the fitness function are a step function from 0 to 1 \( m \) at \( t = 0 \) for the height, and step functions from 0 to 0.3 \( \text{rad} \) at \( t = 0 \) for the three remaining variables \( \phi, \theta, \text{and } \psi \). Tuning of cascaded SMC is made in two steps, attitude variable gains are tuned first, and errors of only angle variables are used to construct the fitness function. Second, the best angle gains are used in the tuning process of the cascaded height controller. This method reduces the tuning time of the cascaded controller remarkably. Table 2.2 shows the best \( c \) and \( k \) vector element values for both regular and cascaded SMCs. As stated before, the Active FTC is a passive regular controller updated with an adaptive part, and it uses the same \( c \) and \( k \) gains of regular Passive FTC (PFTC) shown in Table 2.2.

Note that tuning the regular SMC is made without the integrator part of the controller. This is because the tuning procedure is made in fault free case, and the
remaining error (which necessitates the use of an integrator) shows effect only when a fault exists. The integrator gain vector $K_I$ is found by trial and error, and has all its elements equal to 2. The active SMC uses the same gain matrices of the regular SMC controller. Tuning the controllers is made offline, and in some rare cases the algorithm search individuals extinct and the search process failed to find optimal gains. ESA succeeded to find optimal gains of the controllers in 12 to 55 seconds using a 2.4GHz dual core computer with 3 GB RAM.

### 3.4.2. ESA Parameters

In ESA, search spaces are chosen to be bounded, and any step that violates the boundaries is negated. Parameters like the initial agent and predator population, their initial health values and step sizes, along with the minimum distance that a predator can detect its prey are all set before starting the search. When a zebra individual forages, its health increases by a factor of AugA (3 in regular SMC tuning, 25 in cascaded SMC tuning). Zebra's health decreases by DecA (0.98 in regular, 0.999 in cascaded) when it does not eat, and by a factor Dec (0.98 in regular, 0.99 in cascaded) when it suffers from injuries after being attacked by a lion. Similarly, the lion’s health increases by factor Aug (1.2 in regular and cascaded) if it forages, and decreases by DecP (0.8) when it does not manage to eat a prey. The maximum iteration life of an individual is 10 iterations, its initial health is 100, and its health threshold is 20. When the health of an individual falls below the health threshold value, it dies directly. A fitness value of EpsF (0.01) is the minimum accepted value to increase the health of a zebra individual, and a population of 4 zebras is the minimum accepted population for the algorithm to continue its execution. Table 3.2, Table 3.4, and Table 3.5 show respectively the parameters used in ESA to tune the regular and the cascaded SMCs of the quadrotor. The parameters of the algorithm are chosen carefully by trial and error. A very high AugA factor value for example causes slow elimination of the individuals found in bad regions, resulting in slow conversion. Contrarily, if AugA is low, all the agent individuals will die before the algorithm converges.

<table>
<thead>
<tr>
<th></th>
<th>$z$</th>
<th>$\phi$</th>
<th>$\theta$</th>
<th>$\psi$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$c$</td>
<td>16.56</td>
<td>6.84</td>
<td>8.14</td>
<td>4.37</td>
</tr>
<tr>
<td>$k$</td>
<td>155.5</td>
<td>0.916</td>
<td>7.86</td>
<td>2.68</td>
</tr>
<tr>
<td>$c_{(inner)}$</td>
<td>16</td>
<td>10</td>
<td>11</td>
<td>7</td>
</tr>
<tr>
<td>$k_{(inner)}$</td>
<td>15</td>
<td>13</td>
<td>13</td>
<td>10</td>
</tr>
<tr>
<td>$c_{(outer)}$</td>
<td>26</td>
<td>7</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>$k_{(outer)}$</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>1</td>
</tr>
</tbody>
</table>
### Table 3.3. – Variables Used in ESA to Tune Regular SMC Gains.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\phi$, $\theta$, $\psi$ intervals</td>
<td>$c_i \in [1\ 20]$, $k_i \in [30\ 200]$</td>
</tr>
<tr>
<td>$z$ intervals</td>
<td>$c_i \in [1\ 20]$, $k_i \in [30\ 200]$</td>
</tr>
<tr>
<td>Angles step sizes</td>
<td>$c_i : 0.5$, $k_i : 5$</td>
</tr>
<tr>
<td>$z$ step size</td>
<td>$c_i : 0.2$, $k_i : 0.2$</td>
</tr>
<tr>
<td>Prey population</td>
<td>80</td>
</tr>
<tr>
<td>Angles attack distance</td>
<td>0.05</td>
</tr>
<tr>
<td>Predator population</td>
<td>5</td>
</tr>
<tr>
<td>Initial health</td>
<td>20</td>
</tr>
</tbody>
</table>

### Table 3.4. – Variables Used in ESA to Tune Cascaded SMC Angles Gains.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prey population</td>
<td>20</td>
</tr>
<tr>
<td>Attack distance</td>
<td>0.3</td>
</tr>
<tr>
<td>Step size</td>
<td>0.2</td>
</tr>
<tr>
<td>Predator population</td>
<td>2</td>
</tr>
<tr>
<td>$\phi_{(inner)}$ intervals</td>
<td>$c_i \in [1\ 20]$, $k_i \in [1\ 20]$</td>
</tr>
<tr>
<td>$\phi_{(outer)}$ intervals</td>
<td>$c_i \in [1\ 10]$, $k_i \in [1\ 10]$</td>
</tr>
<tr>
<td>$\theta_{(inner)}$ intervals</td>
<td>$c_i \in [1\ 20]$, $k_i \in [1\ 20]$</td>
</tr>
<tr>
<td>$\theta_{(outer)}$ intervals</td>
<td>$c_i \in [1\ 5]$, $k_i \in [1\ 5]$</td>
</tr>
<tr>
<td>$\psi_{(inner)}$ intervals</td>
<td>$c_i \in [1\ 10]$, $k_i \in [1\ 15]$</td>
</tr>
<tr>
<td>$\psi_{(outer)}$ intervals</td>
<td>$c_i \in [1\ 5]$, $k_i \in [1\ 5]$</td>
</tr>
</tbody>
</table>

### Table 3.5. – Variables Used in ESA to Tune Cascaded SMC Height Gains.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prey population</td>
<td>100</td>
</tr>
<tr>
<td>Attack distance</td>
<td>0.1</td>
</tr>
<tr>
<td>Predator population</td>
<td>5</td>
</tr>
<tr>
<td>Step size</td>
<td>0.5</td>
</tr>
<tr>
<td>$z_{(inner)}$ intervals</td>
<td>$c_i \in [1\ 30]$, $k_i \in [1\ 70]$</td>
</tr>
<tr>
<td>$z_{(outer)}$ intervals</td>
<td>$c_i \in [1\ 30]$, $k_i \in [1\ 5]$</td>
</tr>
</tbody>
</table>

### 3.4.3. Tuning the Controllers Under Fault

ESA tuning of the regular and cascaded controllers is made using a fault-free quadrotor model. This tuning results in controllers that can handle a good amount of fault percentage as will be shown in the results section. A good improvement of the proposed controllers is to embed an amount of fault in the controller design by using a faulty model during the tuning process. Tuning of the regular controller is then repeated with 10%, 20%, 30%, and 40% of faults.
respectively injected to the first motor. The Cascaded SMC was re-tuned with a model having 30% fault in its first motor. The best \( c \) and \( k \) gains of the controllers tuned with 30% faulty model are shown in Table 2.6. ESA parameters used in the new tuning process are identical to those in the fault-free case (Tables 2.3, 2.4, and 2.5). Finally, the fault used during tuning process is a partial loss of speed of motor 1.

### 3.5. Stability Test Of The Proposed Controllers

In this section, the stability of the proposed controllers is checked. Scalar case stability is checked, which means that each state is checked individually.

Choose \( V_i = (1/2)s_i^2 \), with \( i = z, \phi, \theta, \psi \), as a positive Lyapunov function; to satisfy the sliding condition (i.e. reachability condition), \( k_i \) has to be chosen so that \( \dot{V}_i \) is negative. By solving \( \dot{V}_i \) for \( \phi \) angle, \( \dot{V}_\phi = s_\phi \dot{s}_\phi = s_\phi (\ddot{\phi} + c_\phi \dot{e}_\phi) = s_\phi (\ddot{\phi}_d - \ddot{\phi} + c_\phi \dot{e}_\phi) \) we end up with

\[
\dot{V}_\phi = s_\phi (-\frac{k_\phi \text{sign}(s_i)}{I_x}) = -\frac{k_\phi |s_\phi|}{I_x} \tag{3.45}
\]

With \( k_\phi \) taken positive, \( \dot{V}_\phi = -\frac{k_\phi |s_\phi|}{I_x} \) is negative. The same result is reached for \( z, \theta, \) and \( \psi \) variables. This means that the sliding condition is satisfied if all the elements of vector \( k \) are chosen positive.

Because the speeds of the real quadrotor motors might not be accessible during flight, the term \( \gamma \) (equation (2.1)) might not be performed. Consequently, this term cannot be used in the calculation of the control laws and may cause stability issues. As a result, \( \ddot{\tilde{\phi}} \), which is identical to \( \dot{\phi} \) with no \( \gamma \) multiplier term, replaces it in the controller equations. The derivative of the Lyapunov function becomes
\[ V_\phi = s_\phi (\ddot{\phi} - \dot{\phi} - \frac{k_\phi \text{sign}(s_\phi)}{I_x}) = \]
\[ s_\phi (\ddot{\phi} - \dot{\phi}) - \frac{s_\phi k_\phi \text{sign}(s_\phi)}{I_x} = \]
\[ s_\phi (\ddot{\phi} - \dot{\phi}) - \frac{k_\phi |s_\phi|}{I_x} \]

(3.46)

If we can find a positive function \( F(\phi) \) with \(|\ddot{\phi} - \ddot{\phi}\| \leq F(\phi)\), then we can ensure the stability of the quadrotor by choosing \( k_\phi \geq I_x(F(\phi) + \eta_\phi) \), which leads to
\[
s_\phi F(\phi) - \frac{k_\phi |s_\phi|}{I_x} \leq -\eta_\phi |s_\phi|.
\]
The function \( F(\phi) \) is found by subtracting \( \ddot{\phi} \) and \( \ddot{\phi} \), which means that all its elements are zero except \( F(\phi) = I_{\text{rotor}} \dot{\theta} \gamma \). Similarly, \( F(\theta) = -I_{\text{rotor}} \dot{\phi} \gamma \), and \( F(z) = F(\psi) = 0 \). The minimum value of \( k_\phi \) is found by replacing its formula \( (k_\phi \geq I_x(F(\phi) + \eta_\phi)) \) with the maximum value of \( F(\phi) \).

As defined previously, \( \gamma = \Omega_1 - \Omega_2 + \Omega_3 - \Omega_4 \) so its maximum value is \( \gamma_{\text{max}} = 2\Omega_{\text{max}} = 1046 \text{rad/sec} \). The minimum value of \( k_\phi \) is then
\[
k_\phi \geq \frac{I_{\text{rotor}} \gamma_{\text{max}} \dot{\theta}}{l} = 0.4532 \dot{\theta}.
\]

By assuming that any angle of the quadrotor will not change from its minimum value to its maximum value in half a second, the maximum angular velocity is \( \dot{\phi}_{\text{max}} = \dot{\theta}_{\text{max}} = \frac{0.62}{0.5} = 1.24 \text{rad/sec} \). \( k_\phi \) (similarly \( k_\theta \)) is then found as \( k_\phi \geq 0.562 \).

Because \( F(z) \) and \( F(\psi) \) are zero, it is sufficient to choose \( k_z \) and \( k_\psi \) positive to ensure the stability of the controller.

Stability test of the inner loop of the cascaded SMC results in similar constraints shown in the previous proposition. For the outer loop, stability test follows the same steps.

To summarize, the Regular Sliding Mode Controller (equation (2.12)) is stable if the \( k \) vector attitude elements are chosen according to the following conditions
\[
\begin{cases}
k_\phi \geq 0.562 \\
k_\theta \geq 0.562 \\
k_\psi > 0 \text{ and } k_z > 0
\end{cases}
\]

(3.47)

For the outer loop \( \phi \) controller, \( \dot{V}_\phi = s_\phi (\text{outer}) \ddot{s}_\phi (\text{outer}) = s_\phi (\text{outer}) (\dot{e}_\phi + c_\phi (\text{outer}) e_\phi) = s_\phi (\text{outer}) (\ddot{s}_\phi (\text{outer}) - \ddot{\phi} + c_\phi (\text{outer}) e_\phi) \). By replacing \( \ddot{\phi} \) with its equivalent and discontinuous
values (equation (2.16)), we end up with

\[ s_\phi (\text{outer}) \dot{s}_\phi (\text{outer}) = s_\phi (-k_\phi (\text{outer}) \text{sign}(s_\phi (\text{outer}))) = -k_\phi (\text{outer})|s_\phi (\text{outer})| \]  

(3.48)

which is negative if \( k_\phi (\text{outer}) \) is chosen positive. Similar results are found for \( z, \theta \), and \( \psi \) controllers.

To summarize, the Cascaded Sliding Mode Controller is stable if the gains of the outer discontinuous controller \( k_{\text{outer}} \) are positive.

Stability test presented in this section has shown that \( c \) and \( k \) gain vectors found using ESA (Table 2.2) ensure the stability of the regular and cascaded controllers. Moreover, \( k \) values not only ensure the stability of the controllers, but also are the optimal values within the proposed environment.

Same procedure presented above is used to study the stability of the controllers under fault. Note that the faulty control vector (equation (2.7)) should be used in the stability test. By expressing the fault effect directly on the controls, we set the remaining control as \( u_i = f_i u_i \) for \( i = 1, 2, 3, 4 \) (with \( f_i \) is the fault amount of motor \( i \)), the derivative of the Lyapunov function is

\[ \dot{V}_\phi = s_\phi (\ddot{\phi} - \dot{\phi} + c_\phi \dot{\phi}) \]

with

\[ \ddot{\phi} = \frac{I_y - I_z}{I_x} \dot{\psi} + \frac{I_{\text{rotor}}}{I_x} \dot{\gamma} - \frac{K_{fax}}{I_x} \dot{\phi}^2 + \frac{f_2 U_2}{I_x} \]  

(3.49)

By replacing \( U_2 \) with its equation (equation ??), and solving for \( k_\phi \) to ensure the reachability condition, the constraint on \( k_\phi \) which guarantees the stability under fault is found as

\[ k_\phi \leq \frac{-I_x (1 - f_2)(\ddot{\phi} - \frac{I_y - I_z}{I_x} \dot{\psi} + \frac{I_{\text{rotor}}}{I_x} \dot{\gamma} - \frac{K_{fax}}{I_x} \dot{\phi}^2 + c_\phi \dot{\phi}) - \eta_\phi}{f_2 |s_\phi|} \]  

(3.50)

The calculation of the equation above is detailed in Appendix B. Finding the constraint on \( k_\phi \) requires finding the maximum of \( \ddot{\phi} - \frac{I_y - I_z}{I_x} \dot{\psi} + \frac{I_{\text{rotor}}}{I_x} \dot{\gamma} - \frac{K_{fax}}{I_x} \dot{\phi}^2 + c_\phi \dot{\phi} \) and use it in equation (2.50). Seeking simplicity and space, the fault range that the controllers can afford is checked experimentally, and will be given in the results section.

Figures 2.5 and 2.6 show the phase portrait or the variable behavior of the fault-free quadrotor system controlled with both type passive SMCs. The green part of the portrait shows the first 50 points, and the blue sign is the final point in the portrait. As can be seen clearly from the plots, the phase portrait of all the variables (except for the height in regular SMC) converge to the origin. This is expected, since the use of the saturation function introduces a boundary layer,
which ensures that the system converges to the vicinity of the origin instead of to the origin itself. Figures 2.7 and 2.8 show the phase portrait of the faulty quadrotor system controlled with both type passive SMCs. With faults introduced, the system phase portrait converges to the neighborhood of the origin (0.08 is the greatest distance). This is still acceptable since faults are assumed as disturbance that introduce performance degradation to the system. The phase portrait shows that the system error is not increasing, not zero either but stays within a boundary. This means that the quadrotor is Lyapunov stable, which can be seen clearly in the results section.

![Phase portrait of Regular SMC controlled quadrotor with no faults.](image)

**Figure 3.5.** – Phase portrait of Regular SMC controlled quadrotor with no faults.

### 3.6. Simulations And Results

#### 3.6.1. Simulations

In this section, the performance of the designed controllers is checked in SIMULINK environment. The quadrotor is set to follow a 3-D path in the space, and faults are injected to motors at different instants. The first path is a closed
Figure 3.6. – Phase portrait of Cascaded SMC controlled quadrotor with no faults.

square of 10m long sides at 5m height, and the second path is a helix of equation

\[ x_d = 10\cos(0.1t) + 10 \]  
\[ y_d = 10\sin(0.1t) + 10 \]  
\[ z_d = 0.1t \]

(3.51) (3.52) (3.53)

Sliding Mode Controllers designed in this chapter are used to control the attitude and altitude variables \((z, \phi, \theta, \psi)\) of the quadrotor. Moreover, a simple PD controller is used for path tracking. The desired altitude and attitude values are found using equations (2.54) to (2.56) [82], where \(\ddot{x}, \ddot{y}, \text{ and } \ddot{z}\) are chosen according to the PD controller (2.57).

\[ \phi_d = \arctan\left(\frac{-\ddot{y}}{\sqrt{(\ddot{x})^2 + (\ddot{z} + g)^2}}\right) \]  
\[ \theta_d = \arctan\left(\frac{\ddot{x}}{\ddot{z} + g}\right) \]  
\[ \psi_d = 0 \]

(3.54) (3.55) (3.56)

The simple PD position controller used with Passive FTC controllers (regular
and cascaded) has the following equation

\[ \ddot{i} = k_{ip}(i_d - i) - k_{id}\dot{i} \quad (3.57) \]

where \( i \) is \( x, y, \) or \( z \), and \( k_{ip} = 2 \) and \( k_{id} = 3 \) are respectively the proportional and derivative gains found by trial and error. The integrator gain matrix of equation (2.12) is found by trial and error to have all its elements equal to two. It is important to emphasize that the height output of the PD controller \( \ddot{z} \) is used only to generate desired angles and not used to control the height of the quadrotor, which is done using the proposed SMC controllers. For the active FTC, the quadrotor sensors are assumed to be subjected to a band limited white noise, with noise standard deviation of \( \sqrt{\text{noise power}} / \sqrt{\text{sample time}} \). Position sensors accuracy is assumed to be 10 cm, with \( 10^{-6} \) noise power for \( x \) and \( y \) directions, and \( 10^{-7} \) for \( z \) position. For the attitudes, the sensor accuracy is 0.2° (0.0035 rad), with \( 10^{-9} \) noise power for both attitude values and their angular speeds. The variance matrices of the state (equation (2.24)) and measurement (equation (2.25)) noise are taken as \( Q = 10^{-3} \text{eye}(12) \) and \( R = \text{diag}(10^{-6}, 10^{-6}, 10^{-6}, 10^{-9}, 10^{-9}, 10^{-9}) \) respectively.
The $x$ and $y$ position controller used with the Active FTC is a lead compensator that have little effect on higher frequencies (i.e. $\omega > 30 \text{ rad}$). This is essential because noise signals have high frequencies, and they affect severely the control of the quadrotor if they are multiplied by the gains. The position controller is presented in equation (2.58), and its Bode diagram is shown in Figure 2.9.

$$C(p) = \frac{K_ds + K_p}{0.1s + 1}$$

(3.58)

The proportional and differential gains are chosen by trial and error to be $K_p = 2$ and $K_d = 6$ for both $x$ and $y$ variables.

### 3.6.2. Results

The faults used to test the robustness of the proposed fault tolerant controllers are partial loss of efficiency faults injected to one or two motors. The speed of a
motor drops when the load torque increases as result of rust or debris, or when
the voltage across that motor changes as the internal resistance of the motor
changes.

Simulation results show that Regular SMC controlled system is able to handle
a 40% fault in motor 2 at \( t = 20s \) along with a 55% fault in motor 1 occurring
at instant \( t = 40s \). The fault percentage rises to 57% and 56% respectively for
motor 1 and motor 2 when the controller is tuned with a faulty system. Results of
Regular SMC controlled quadrotor are illustrated in Figures 2.10 to 2.13. A small
deviation from the desired path can be seen in this figures (clear in the angles
response) at the instant when the faults occur. Figure 2.13 shows the motor
speeds and controls applied during flight for Regular SMC tuned without faults.
Motor speeds are realistic and stay within the physical limits of the quadrotor.
One can recognize the instantaneous change in motor speeds and controls as
the controller compensates for the sudden faults. Finally, it can be seen clearly
that fault compensation does not occur directly at the fault injection instance,
but after a short delay. This delay is the cause of the small deviation in the path,
remedied directly by the controller.

Cascaded SMC tuned without fault exhibits more fault tolerant capabilities
compared with similar Regular SMC. The quadrotor can handle up to 54% fault
in motor 2 at \( t = 20s \) along with a 55% fault in motor 1 occurring at instant
\( t = 40s \). Moreover, the quadrotor does not deviate from its desired path as can be
seen in Figures 2.17 to 2.20. The small and short deviation in the angles response
reveals the importance of Cascaded SMC, where faults are compensated in the
fast velocity inner loop, before they affect the quadrotor position. This is clear in

---

**Figure 3.9.** – Bode diagram of the x-y position controller.
the figure showing the motor speeds of the quadrotor (Figure 2.20). Changes in motor speeds happen at the time the faults are injected ($t = 20s$ and $t = 40s$), which means fast compensation of any fault.

With the Cascaded SMC tuned under fault, the quadrotor can handle up to 56% faults in both motor 2 and motor 1 occurring at $t = 20s$ and $t = 40s$ respectively. Figures 2.21 to 2.23 show that fault compensation is made quickly with no deviation in the path.

It is important to state that the location of fault during tuning process might affect the controllers robustness. Controllers can tolerate better the same fault that was injected during their tuning. In our experiment, controllers tuned under fault tolerate a good amount of motor 1 fault because they were tuned with this fault. Controllers response would be very different if the fault was in different motor. To solve this issue and improve the fault tolerance of the controllers, an active fault tolerant controller can be designed based on the passive FTC controllers proposed in this chapter. The controllers are tuned under different fault locations and a controllers bank containing all the controllers is formed. A Fault Detection and Identification (FDI) unit is used to detect any fault online. Whenever a fault is detected, the appropriate controller is chosen from the controllers bank based on the fault location and magnitude. The design of such active FTC is left for future work.

Concerning the Active Fault Tolerant SMC, results show better tracking and higher robustness compared to Passive FTC SMC. The active controller can handle
up to 57% loss of effectiveness of one motor. This is close to the Passive SMC results but with no remaining error found at the attitude response.

Figures 2.24 to 2.30 show the active FTCSMC simulation results (equation (2.38)). The fault for helical path is a loss of effectiveness of 55% in motor 1 injected at \( t = 40 \text{s} \). For the square path, a fault of 15% is injected to motor 2 at \( t = 40 \text{s} \), followed by a 30% fault injected at motor 3 at \( t = 50 \text{s} \).
Figure 3.13. – Controls and Rotor speeds of Regular SMC experiment (tuned without fault).

Figure 3.14. – Path of Regular SMC controlled Quadrotor tuned under fault (2 faults of 57% at 40s and 56% at 20s).
Figure 3.15. – Attitude tracking error of Regular SMC controlled Quadrotor tuned under fault (2 faults of 57% at 40s and 56% at 20s).

Figure 3.16. – Altitude and heading tracking error of Regular SMC controlled Quadrotor tuned under fault (2 faults of 57% at 40s and 56% at 20s).
Figure 3.17. – Path of Cascaded SMC controlled Quadrotor tuned under no fault (2 faults of 55% at 40s and 54% at 20s).

Figure 3.18. – Attitude tracking error of Cascaded SMC controlled Quadrotor tuned under no fault (2 faults of 55% at 40s and 54% at 20s).
Figure 3.19. – Altitude and heading tracking error of Cascaded SMC controlled Quadrotor tuned under no fault (2 faults of 55% at 40s and 54% at 20s).

Figure 3.20. – Controls and Rotor speeds of Cascaded SMC experiment (tuned without fault).
Figure 3.21. – Path of Quadrotor controlled with Cascaded SMC tuned under fault (2 faults of 56% at 40s and 56% at 20s).

Figure 3.22. – Attitude tracking error of Quadrotor controlled with Cascaded SMC tuned under fault (2 faults of 56% at 40s and 56% at 20s).
Figure 3.23. – Altitude and heading tracking error of Quadrotor controlled with Cascaded SMC tuned under fault (2 faults of 56% at 40s and 56% at 20s).

Figure 3.24. – Paths of the Active SMC controlled Quadrotor.
Figure 3.25. – Attitude response of the Active SMC controlled Quadrotor with 55% fault (helical path, 55% fault to M1 at t = 40s).

Figure 3.26. – Heading and altitude response of the Active SMC controlled Quadrotor with 55% fault (helical path, 55% fault to M1 at t = 40s).
Figure 3.27. – Controls and rotor speeds of the Active SMC controlled Quadrotor (helical path, 55% fault to M1 at t = 40s).
As a conclusion, passive FTC schemes are still reserving an important place in fault tolerant control. Active FTC outperforms its Passive counterpart when one motor fault is considered. The maximum tolerated amount of fault is 35% for Passive and 55% for Active FTC. By upgrading the Passive FTC with an integrator, its performance is improved to tolerate 55% faults. Tuning the passive controllers with faulty system pushes their robustness to the limits. The Passive FTC can now tolerate up to 57% fault in one motor.
Figure 3.30. — Controls and rotor speeds of the Active SMC controlled infected Quadrotor (square path, two faults at $t = 40s$, and $t = 50s$).
4. Emergency Controller for Quadrotor UAVs

In this chapter, an emergency controller based on the quadrotor to trirotor conversion maneuver is designed for quadrotors to tolerate the total loss of one rotor. The new emergency controller redistributes the quadrotor controls among healthy rotors to end up with a trirotor system. Quadrotors are underactuated UAV systems, they have only four actuators to achieve the control of their six variables $x$, $y$, along with $z$ coordinates, and $\phi$, $\theta$, and $\psi$ angles. A complete failure of only one rotor causes the total loss of control, and the crash of the UAV.

In Trirotors, three rotors are used to control the system as shown in Figure 3.1. Roll angle control is achieved using the difference between left and right rotor thrusts, while pitch angle is controlled using difference between front rotors (left and right rotors), and the rear rotor thrusts. To control the yaw angle, different techniques can be used. One technique is to tilt the rear rotor around the $x$-axis and generate a torque around $z$-axis [83]. Another technique is to generate torque around $z$-axis by placing two small motors with rotors on an axis parallel to the axis holding the left and right rotors [84]. Thrust difference between the new introduced motors generates the yaw moment. A third way to control the heading is to use tilting mechanisms for both rotor 1 and rotor 2 while fixing rotor 3 [85]. Without using any of the previous techniques, it is impossible to control the yaw angle, and the trirotor ends up spinning around its $z$-axis. Similar to quadrotor UAVs, the trirotor height is controlled by the total thrust of their three rotors. If one can develop a method to transform quadrotors into trirotors, a powerful Fault Tolerant Controller could be achieved. Whenever a rotor is affected by a fault, this rotor is stopped, the trirotor configuration is activated, and the quadrotor continues its mission as a trirotor UAV.

4.1. Trirotor Dynamics

Quadrotor and trirotor UAVs have similar dynamics, represented in equations (1.1) to (1.6) [84]. The main structural difference between quadrotor and trirotor UAVs is in the location of their Centers of Gravity (CoG). The CoG in quadro-
tors is located at the intersection point of its $x$ and $y$ axes. The CoG of trirotors is located at the rear of the $x$-$y$ axes intersection point. This allows trirotors to change their pitch angles by changing the thrust difference between front and rear motors. Moreover, the speed-control equations of quadrotor and trirotor UAVs are also different.

![Trirotor schematic](image)

(a) Trirotor with tilting mechanism for Yaw control.

(b) Trirotor with two opposite motors for Yaw control.

Figure 4.1. – Trirotor schematic.

The speed-control equations of a quadrotor UAV are

\[ U_1 = b(\Omega_1^2 + \Omega_2^2 + \Omega_3^2 + \Omega_4^2) \]  \hspace{1cm} (4.1)
\[ U_2 = bl(-\Omega_2^2 + \Omega_4^2) \]  \hspace{1cm} (4.2)
\[ U_3 = bl(-\Omega_1^2 + \Omega_3^2) \]  \hspace{1cm} (4.3)
\[ U_4 = d(\Omega_1^2 - \Omega_2^2 + \Omega_3^2 - \Omega_4^2) \]  \hspace{1cm} (4.4)

here $U_1$ is the thrust control, $U_2$ is the roll angle control, $U_3$ the pitch angle control, and $U_4$ is the control of the heading. $\Omega_i$ with $i = 1, 2, 3, 4$ is the speed
of motor $i$, $l$ is the distance between the CoG of the quadrotor and the center of each rotor, and $b$ and $d$ are respectively the thrust and drag factors of the quadrotor.

The rotor speed-control equations of a trirotor with tail tilting mechanism are [83]

\begin{align*}
U_1 &= b(\Omega_1^2 + \Omega_2^2) + b\Omega_3^2 \cos(\mu) \quad (4.5) \\
U_2 &= bl_1(\Omega_1^2 - \Omega_2^2) \quad (4.6) \\
U_3 &= -bl_2(\Omega_1^2 + \Omega_2^2) + bl_3\Omega_3^2 \cos(\mu) \quad (4.7) \\
U_4 &= dl_3(-\Omega_1^2 + \Omega_2^2) + l_3d\Omega_3^2 + bl_3\Omega_3^2 \sin(\mu) \quad (4.8)
\end{align*}

where $\mu$ is the angle of the tail rotor controlled with the tilting mechanism as shown in Figure 3.1.

4.2. Quad-Tri Conversion

Without using the yaw control techniques presented in the previous section, it is impossible for $\psi$ angle to follow a desired value. However, even without being able to control the yaw angle, it is still possible to control the remaining control variables of the trirotor. When the tail angle is kept zero ($\mu = 0$), the trirotor rotor speed-control relations are simplified by setting $\cos(\mu) = 1$ and $\sin(\mu) = 0$ in the equations (3.5) to (3.8).

Following the similarities between the two UAVs, a question rises here automatically: is it possible to turn off one of the quadrotor motors and control it as a trirotor? A small comparison between the two UAVs reveals the fact that without changing the CoG location of the quadrotor, it is impossible to control its pitch angle in trirotor configuration. If the CoG remains at the intersection point of the diagonals, $l_2$ in equation (3.7) is zero resulting in uncontrolled pitch as $U_3 = -bl_3\Omega_3^2$. To summarize, it is impossible to control the quadrotor as trirotor without shifting its Center of Gravity towards the rear rotor. This is applicable by increasing the weight of the tail rotor, and turning off the front rotor (Figure 3.2).

4.2.1. Different Trirotor Configurations

The quadrotor to trirotor conversion in Figure 3.2 is made by turning off the front rotor of the quadrotor, and shifting the CoG towards the rear rotor. In the new configuration, rotor speeds needed to perform the UAV controls change, and the control allocator responsible for generating the controls out of rotor speeds changes. With different rotors turned off, four different configurations are available to use the quadrotor as trirotor. Figure 3.3 shows the four different
configurations of UAV conversion which can be used only if the CoG is shifted towards the appropriate rear rotor and the right control allocator is chosen.

If Rotor 1 is turned off, the control allocator equations of the trirotor become
\[ U_1 = b(\Omega_4^2 + \Omega_2^2 + \Omega_3^2) \]  
\[ U_2 = bl_1(\Omega_4^2 - \Omega_2^2) \]  
\[ U_3 = -bl_2(\Omega_4^2 + \Omega_2^2) + bl_3\Omega_3^2 \]  
\[ U_4 = -dl_4(\Omega_4^2 + \Omega_2^2) + l_3d\Omega_3^2 \]  

(4.9) \[ U_1 = b(\Omega_1^2 + \Omega_3^2 + \Omega_4^2) \]  
\[ U_2 = bl_1(\Omega_1^2 - \Omega_3^2) \]  
\[ U_3 = -bl_2(\Omega_1^2 + \Omega_3^2) + bl_3\Omega_4^2 \]  
\[ U_4 = -dl_4(\Omega_1^2 + \Omega_3^2) - l_3d\Omega_4^2 \]  

(4.10) \[ U_1 = b(\Omega_2^2 + \Omega_4^2 + \Omega_1^2) \]  
\[ U_2 = bl_1(\Omega_2^2 - \Omega_4^2) \]  
\[ U_3 = -bl_2(\Omega_2^2 + \Omega_4^2) + bl_3\Omega_1^2 \]  
\[ U_4 = -dl_4(\Omega_2^2 + \Omega_4^2) + l_3d\Omega_1^2 \]  

(4.11) \[ U_1 = b(\Omega_3^2 + \Omega_1^2 + \Omega_2^2) \]  
\[ U_2 = bl_1(\Omega_3^2 - \Omega_1^2) \]  
\[ U_3 = -bl_2(\Omega_3^2 + \Omega_1^2) + bl_3\Omega_2^2 \]  
\[ U_4 = -dl_4(\Omega_3^2 + \Omega_1^2) - l_3d\Omega_2^2 \]  

(4.12)

On the other hand, if Rotor 2 is turned off the equations become

\[ U_1 = b(\Omega_4^2 + \Omega_2^2 + \Omega_3^2) \]  
\[ U_2 = bl_1(\Omega_4^2 - \Omega_2^2) \]  
\[ U_3 = -bl_2(\Omega_4^2 + \Omega_2^2) + bl_3\Omega_3^2 \]  
\[ U_4 = -dl_4(\Omega_4^2 + \Omega_2^2) + l_3d\Omega_3^2 \]  

(4.13) \[ U_1 = b(\Omega_1^2 + \Omega_3^2 + \Omega_4^2) \]  
\[ U_2 = bl_1(\Omega_1^2 - \Omega_3^2) \]  
\[ U_3 = -bl_2(\Omega_1^2 + \Omega_3^2) + bl_3\Omega_4^2 \]  
\[ U_4 = -dl_4(\Omega_1^2 + \Omega_3^2) - l_3d\Omega_4^2 \]  

(4.14) \[ U_1 = b(\Omega_2^2 + \Omega_4^2 + \Omega_1^2) \]  
\[ U_2 = bl_1(\Omega_2^2 - \Omega_4^2) \]  
\[ U_3 = -bl_2(\Omega_2^2 + \Omega_4^2) + bl_3\Omega_1^2 \]  
\[ U_4 = -dl_4(\Omega_2^2 + \Omega_4^2) + l_3d\Omega_1^2 \]  

(4.15) \[ U_1 = b(\Omega_3^2 + \Omega_1^2 + \Omega_2^2) \]  
\[ U_2 = bl_1(\Omega_3^2 - \Omega_1^2) \]  
\[ U_3 = -bl_2(\Omega_3^2 + \Omega_1^2) + bl_3\Omega_2^2 \]  
\[ U_4 = -dl_4(\Omega_3^2 + \Omega_1^2) - l_3d\Omega_2^2 \]  

(4.16)

If Rotor 3 is turned off, the control allocator equations are

\[ U_1 = b(\Omega_4^2 + \Omega_2^2 + \Omega_3^2) \]  
\[ U_2 = bl_1(\Omega_4^2 - \Omega_2^2) \]  
\[ U_3 = -bl_2(\Omega_4^2 + \Omega_2^2) + bl_3\Omega_3^2 \]  
\[ U_4 = -dl_4(\Omega_4^2 + \Omega_2^2) + l_3d\Omega_3^2 \]  

(4.17) \[ U_1 = b(\Omega_1^2 + \Omega_3^2 + \Omega_4^2) \]  
\[ U_2 = bl_1(\Omega_1^2 - \Omega_3^2) \]  
\[ U_3 = -bl_2(\Omega_1^2 + \Omega_3^2) + bl_3\Omega_4^2 \]  
\[ U_4 = -dl_4(\Omega_1^2 + \Omega_3^2) - l_3d\Omega_4^2 \]  

(4.18) \[ U_1 = b(\Omega_2^2 + \Omega_4^2 + \Omega_1^2) \]  
\[ U_2 = bl_1(\Omega_2^2 - \Omega_4^2) \]  
\[ U_3 = -bl_2(\Omega_2^2 + \Omega_4^2) + bl_3\Omega_1^2 \]  
\[ U_4 = -dl_4(\Omega_2^2 + \Omega_4^2) + l_3d\Omega_1^2 \]  

(4.19) \[ U_1 = b(\Omega_3^2 + \Omega_1^2 + \Omega_2^2) \]  
\[ U_2 = bl_1(\Omega_3^2 - \Omega_1^2) \]  
\[ U_3 = -bl_2(\Omega_3^2 + \Omega_1^2) + bl_3\Omega_2^2 \]  
\[ U_4 = -dl_4(\Omega_3^2 + \Omega_1^2) - l_3d\Omega_2^2 \]  

(4.20)

The control allocator equations of the trirotor when Rotor 4 is turned off are

\[ U_1 = b(\Omega_3^2 + \Omega_2^2 + \Omega_4^2) \]  
\[ U_2 = bl_1(\Omega_3^2 - \Omega_2^2) \]  
\[ U_3 = -bl_2(\Omega_3^2 + \Omega_2^2) + bl_3\Omega_4^2 \]  
\[ U_4 = -dl_4(\Omega_3^2 + \Omega_2^2) + l_3d\Omega_4^2 \]  

(4.21) \[ U_1 = b(\Omega_1^2 + \Omega_3^2 + \Omega_4^2) \]  
\[ U_2 = bl_1(\Omega_1^2 - \Omega_3^2) \]  
\[ U_3 = -bl_2(\Omega_1^2 + \Omega_3^2) + bl_3\Omega_4^2 \]  
\[ U_4 = -dl_4(\Omega_1^2 + \Omega_3^2) - l_3d\Omega_4^2 \]  

(4.22) \[ U_1 = b(\Omega_2^2 + \Omega_4^2 + \Omega_1^2) \]  
\[ U_2 = bl_1(\Omega_2^2 - \Omega_4^2) \]  
\[ U_3 = -bl_2(\Omega_2^2 + \Omega_4^2) + bl_3\Omega_1^2 \]  
\[ U_4 = -dl_4(\Omega_2^2 + \Omega_4^2) + l_3d\Omega_1^2 \]  

(4.23) \[ U_1 = b(\Omega_3^2 + \Omega_1^2 + \Omega_2^2) \]  
\[ U_2 = bl_1(\Omega_3^2 - \Omega_1^2) \]  
\[ U_3 = -bl_2(\Omega_3^2 + \Omega_1^2) + bl_3\Omega_2^2 \]  
\[ U_4 = -dl_4(\Omega_3^2 + \Omega_1^2) - l_3d\Omega_2^2 \]  

(4.24)

Note that when rotor \( i \) is turned off, it is enough to shift the CoG of the UAV a very short distance towards the opposite rotor.

### 4.2.2. Study of the Trirotor Static Equilibrium

Shifting the CoG can be done using pre-installed weights under each rotor, and releasing all the weights except the one opposite to the infected motor (using solenoid or motor mechanisms). Similar maneuver could be done by releasing the weight under only the infected motor/rotor. By studying the static equilibrium of the trirotor, it is possible to find some constraints on the masses that should be...
added to the quadrotor in order to realize the conversion effectively. Figure 3.4 shows the quadrotor converted to trirotor by stopping rotor 1 and releasing the mass underneath.

![Diagram of quadrotor and trirotor setup](image)

Figure 4.4. – Shifting the CoG of the quadrotor towards the tail rotor.

$m_m$ is the mass of one motor, $m_l$ is the mass added for each motor so that the total mass of the motor side is $m = m_m + m_l$. By assuming static equilibrium along $x - axis$, the sum of torques around the new CoG should be zero or

$$(mg - m_l g)(l_1 + l_2) = mgl_3$$

(4.25)

by using $m = m_m + m_l$, $l_1 = l_2 + l_3$, and $m_m = 57g$ and $l_1 = 24cm$ for the Pelican quadrotor, the relation between the released mass $m_l$ and the distance $l_2$ is found as

$$m_l = \frac{0.114l_2}{0.24 - l_2}$$

(4.26)

or

$$l_2 = \frac{0.24m_l}{0.114 + m_l}$$

(4.27)

The $m_l - l_2$ relation is shown in Figure 3.5. The figure shows clearly that to shift the CoG 5 cm backwards, a mass of about 30 g is needed to be connected under all the motors except the infected one.

The connection of extra weights under each motor could be used in an intelligent manner. It is possible for the quadrotors to carry extra batteries in their missions, and whenever quadrotor to trirotor conversion is needed these extra batteries are released. This will extend the mission life and help to increase the reliability of the UAVs.
4.2.3. Study of the Trirotor Dynamic Equilibrium

When the quadrotor flies near hovering condition, its roll and pitch angles are zero, and the gyroscopic coupling effects in the quadrotor/trirotor dynamics ($\dot{\theta}$, $\dot{\phi}$, $\dot{\psi}$, and $\dot{\theta}$) are small and can be neglected. Moreover, the total thrust of the UAV is equal to its weight. This simplifies the angular dynamics of the quadrotor to

\[
\ddot{\theta} = U_1 - M_{\text{trirotor}}g = 0 \quad (4.28) \\
\dot{\phi} = \frac{U_2}{I_x} = 0 \quad (4.29) \\
\ddot{\theta} = \frac{U_3}{I_y} = 0 \quad (4.30) \\
\ddot{\psi} = \frac{U_4}{I_z} \quad (4.31)
\]

by using the control/rotor speed equations of the trirotor (equations (3.9) to (3.12) for rotor 1 failure), it is possible to find constraints on $l_2$ based on the motor speed constraints. At hover,

\[
U_1 = b(\Omega_2^2 + \Omega_3^2 + \Omega_4^2) = M_{\text{trirotor}}g \quad (4.32) \\
U_2 = bl_1(\Omega_4^2 - \Omega_2^2) = 0 \quad (4.33) \\
U_3 = -b.l_2.(\Omega_4^2 + \Omega_2^2) + b.l_3.\Omega_3^2 = 0 \quad (4.34)
\]

Solving the above equations gives the following relation between $l_2$ and the
motor speeds

$$\Omega_2^2 = 3.7708 e^5 (0.24 - l_2)$$  \hspace{1cm} (4.35)$$

or

$$\Omega_3^2 = 7.54 e^5 l_2$$  \hspace{1cm} (4.36)$$

and

$$l_2 = 1 - 1.105 e^{-5} \Omega_2^2$$  \hspace{1cm} (4.37)$$

With $$\Omega_4 = \Omega_2$$. The relationships between the motor speeds and the length $$l_2$$ are shown in Figure 3.6. Figure shows that the more the new CoG is shifted away from the old CoG, the more rear rotor (rotor3 in case of rotor 1 failure) suffers to stabilize the pitch angle at the equilibrium point, but the less side rotors (rotor 2 and 4 in our case) need speed to stabilize the roll angle. Rotor speeds needed to stabilize the trirotor for a CoG shifting of 5 cm are shown on the figure.

![Graphs showing relationships between rotor speeds and $$l_2$$.](image)

(a) Side rotors  \hspace{1cm} (b) Rear rotor

**Figure 4.6.** – Relations between $$l_2$$ and rotor speeds.

### 4.2.4. Trirotor Moment of Inertia Calculation

With the CoG out of the center point, the moment of inertia of the trirotor becomes different than that of the quadrotor. The inertia matrix of the trirotor for an off-centered CoG is
\[
J = \begin{bmatrix}
J_x & -J_{xy} & -J_{xz} \\
-J_{xy} & J_y & -J_{yz} \\
-J_{xz} & -J_{yz} & J_z
\end{bmatrix}
\] (4.38)

where the inertia matrix coefficients are \([86]\)

\[
J_{x,y,z} = \sum_{i=1}^{4} I_{m_{ix,y,z}} + I_{c'_{x,y,z}} + I_{b'_{x,y,z}}
\] (4.39)

\[
J_{xy,xz,yz} = \sum_{i=1}^{4} I_{m_{ixy,xz,yz}} + I_{c'_{xy,xz,yz}} + I_{b'_{xy,xz,yz}}
\] (4.40)

\(I_{m_j}, I_{c_j}, \) and \(I_{b_j}\) with \(j = x, y, z\) are respectively the moment of inertia of the motors, the quadrotor cross, and the payload added to shift the CoG to its new location of coordinates \((x_{CoG}, y_{CoG}, z_{CoG})^T\). \(I_{c'_{j}}, \) and \(I_{b'_{j}}\) are the trirotor moment of inertia of the same elements. If \(\alpha, \beta, \) and \(\gamma\) are the CoG coordinates of the added payload, \(m_m\) is the weight of one motor, \(m_c\) the weight of the quadrotor cross, and \(m_p\) the weight of the payload, then the trirotor CoG coordinates can be found as

\[
(x_{CoG}, y_{CoG}, z_{CoG})^T = \left(\frac{m_p\alpha}{m}, \frac{m_p\beta}{m}, \frac{m_p\gamma}{m}\right)^T
\] (4.41)

The trirotor motor inertia, cross inertia, and payload inertia are shown in Appendix B.

### 4.3. Emergency Control of Quadrotor UAVs

Using the quadrotor improperly (as trirotor) results in high disturbance and aerodynamic changes that may affect severely the UAV. To overcome the new situation, Passive Fault Tolerant Sliding Mode Controller (PFTSMC) developed in previous chapters (equation (2.12)) is chosen to control the UAV because of its inherent robustness essential to overcome the newly emerged disturbance. The controller gains are shown in table 2.2, and were found previously using ESA. An FDI unit similar to the one designed for Active Fault Tolerant Control of quadrotor UAV is used here to detect actuator faults affecting the quadrotor rotors.

The emergency controller of the quadrotor is developed based on the quadrotor to trirotor conversion maneuver and Passive Fault Tolerant SMC. When the FDI unit detects a fault equal or greater than 40%, the infected motor is turned off, the relevant weight redistribution is applied, and the trirotor controller is activated. The application of conversion maneuver saves the UAV from crashing, and allows it to continue its path but while spinning around its z-axis. This is an undesired behavior but is acceptable to prevent the crash of the quadrotor un-
til it reaches the nearest landing zone. Without weight redistribution, the pitch angle becomes uncontrollable and the quadrotor will crash as soon as one of its motors is severely damaged. Weight redistribution is realized by attaching extra equal weights under each rotor. This does not affect the position of the CoG, and the sum of the new weights is added to the total weight of the quadrotor. When one of the rotors fails, the FDI unit detects the damage, changes the controller, and sends a command to a solenoid mechanism to throw the extra weight under the damaged rotor. This step shifts the center of gravity towards the opposite rotor, assumed now as the rear rotor.

4.4. Simulation and Results

The same lead compensator used with the Active FTC (equation (2.58)) is used with the emergency controller. This controller has little effect on high frequency noisy signals, thus decreasing their effect to the minimum. The proportional and differential gains are respectively $K_p = 2$ and $K_d = 6$ for both $x$ and $y$ variables. The quadrotor sensors are assumed to be subjected to the same band limited white noise presented in chapter 2.

The emergency controller is used in Simulink environment to control the quadrotor performing a helical path in the space. Note that the same noise standard deviation used in previous chapter is applied in this chapter. Tests with different rotor failures show that the UAV conversion was successful in maintaining the path despite the stop of one of the rotors. Figures 3.7 to 3.10 show the quadrotor response after the failure of rotor 2 at $t = 50s$.

![3D path](image1)

![2D path](image2)

Figure 4.7. — Quadrotor path (Failure of the second rotor).
Figure 4.8. – Attitude of the quadrotor (Failure of the second rotor).

Figure 4.9. – Heading and height of the quadrotor (Failure of the second rotor).
4.5. Experimental Work: Emergency Controller Application on AR Drone 2 Platform

AR Drone 2 is a quadrotor UAV manufactured by Parrot company as a toy for augmented reality games. The quadrotor is controlled over Wi-Fi using a smartphone or similar devices, and has a range of 50m depending on the signal strength. With the indoor protective hull (Figure 3.11), AR Drone is of 517 × 517 mm² dimension and 420g weight, (451 × 451 mm² and 380g with the external hull), which makes it a member in the Micro Aerial Vehicles (MAV) family. Although the quadrotor is meant to be a toy, its low cost, open source Application Program Interface (API), and redundant sensors make AR Drone 2 an ideal platform for robotic and control research [87].

4.5.1. Technical Specifications of AR Drone 2

The AR Drone 2 has four brushless DC motors, each controlled with an 8 MIPS AVR microcontroller, and that can achieve a speed of 28500 rpm with a power of 14.5W. These motors allow AR Drone 2 to fly with a cruising speed up to 10 m/s, and are equipped with emergency stop controlled by software. The quadrotor is equipped with a 3 cell high grade Lithium-Polymer battery with 1000 mAh capacity at 11.1 volts, which gives enough energy for 13 minutes of flight. The battery has a Protection Circuit Module (PCM) to protect it against over charge, over discharge, and short circuits.
AR Drone 2 is equipped with a redundant number of sensors. To measure roll, pitch, and yaw angles, a 3-axis gyroscope with rotational speed of 2000 deg/s is used. To provide more accurate measurements, gyroscope results are also combined with the results of a 3-axis magnetometer with precision up to 6 degrees. The acceleration of the quadrotor in x, y, and z directions is measured using a 3-axis accelerometer with precision up to ±0.05 g. The height of the quadrotor is measured using two types of sensors. When the AR Drone 2 flies in low altitude (less than 6 m height), an ultrasound sensor is used to provide the controller with height information. For higher altitudes, a pressure sensor with a precision up to 10 Pa is used to predict the height of the quadrotor. The internal software of the AR Drone 2 uses the 480 × 360 pixels / 45° × 45° field of view 60 fps camera mounted at the bottom to enhance the drone’s stability by estimating the movement in x and y directions. The quadrotor is also equipped with a 1280 × 720 pixels / wide angle lens / 30 fps color image HD frontal camera for surrounding inspection.

To deal with the extensive number of sensor data and complex control requirements, AR Drone 2 is equipped with a 1 GHz, 32 bit ARM Cortex A8 processor with 1 GByte of DDR2 RAM running at 200 MHz. The Operation System is a GNU Linux with kernel version 2.6.32 and a versatile command line program BusyBox. Finally, video processing is achieved with the TM320DMC64x DSP unit running at 800 MHz.

To control the AR Drone 2 and develop the emergency controller, the AR Drone Simulink Development-Kit V1.1 available at MATLAB/File Exchange board is used. The kit consists of simulink blocks for real-time and simulation of the Parrot AR Drone 2 quadrotor. The real-time blocks send commands and read the
states of the quadrotor online. The kit was developed by David Escobar Sanabria and Pieter J. Mosterman in the context of the 2013 MathWorks summer research internship project [89].

4.5.2. Dynamics of AR Drone 2

Unlike Asctec Pelican quadrotor used earlier in this thesis, AR Drone 2 quadrotor has a cross flight configuration (Figure 3.12). This means that each control variable of the quadrotor is controlled with at least two pairs of rotors instead of only two rotors. The roll angle of the AR Drone 2 (rotation around $x$-axis, and thus its motion along $y$-axis) is controlled using the thrust difference between rotor1/rotor4 set of rotors, and rotor2/rotor3 set of rotors (left and right sets of rotors). Similarly, rotation around $y$-axis (pitch angle, motion along $x$-axis) is ensured by applying a thrust difference between front and rear set of rotors (rotor1/rotor2, and rotor3/rotor4).

![Figure 4.12: AR Drone 2 body frame.](image)

Dynamics of AR Drone 2 is identical to all quadrotor dynamics, and is shown in equations (1.1) through (1.6). Note that the $z$-axis equation is multiplied by -1 because the axis is pointing downwards in AR Drone 2. Table 3.1 shows the variables of AR Drone 2 used in system modeling and controller development [90]. Note that the rotor speed imbalance disturbance effect (term multiplier of $\gamma$) is not taken into consideration in this part.

Because of its cross flight configuration, controls of AR Drone 2 quadrotor are expressed in terms of rotor speeds according to the following set of equations.
<table>
<thead>
<tr>
<th>$I_x$</th>
<th>$0.002237568, N.m.s^2$</th>
<th>$b$</th>
<th>$8.048e^{-6}, N.s^2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$I_y$</td>
<td>$0.002985236, N.m.s^2$</td>
<td>$l$</td>
<td>$0.1785, m$</td>
</tr>
<tr>
<td>$I_z$</td>
<td>$0.00480374, N.m.s^2$</td>
<td>$d$</td>
<td>$2.423e^{-7}, m$</td>
</tr>
<tr>
<td>$m$</td>
<td>$0.429, Kg$</td>
<td>$g$</td>
<td>$9.81, m.s^{-2}$</td>
</tr>
</tbody>
</table>

The inverse relations between rotor speeds and controls are

$$U_1 = b(\Omega_1^2 + \Omega_2^2 + \Omega_3^2 + \Omega_4^2)$$

$$U_2 = bl(\Omega_1^2 - \Omega_2^2 - \Omega_3^2 + \Omega_4^2)$$

$$U_3 = bl(\Omega_1^2 + \Omega_2^2 - \Omega_3^2 - \Omega_4^2)$$

$$U_4 = d(-\Omega_1^2 + \Omega_2^2 - \Omega_3^2 + \Omega_4^2)$$

$$\Omega_1^2 = \frac{U_1}{4b} + \frac{U_2}{4bl} + \frac{U_3}{4bl} - \frac{U_4}{4d}$$

$$\Omega_2^2 = \frac{U_1}{4b} - \frac{U_2}{4bl} + \frac{U_3}{4bl} + \frac{U_4}{4d}$$

$$\Omega_3^2 = \frac{U_1}{4b} - \frac{U_2}{4bl} - \frac{U_3}{4bl} - \frac{U_4}{4d}$$

$$\Omega_4^2 = \frac{U_1}{4b} + \frac{U_2}{4bl} - \frac{U_3}{4bl} + \frac{U_4}{4d}$$

### 4.5.3. Using AR Drone 2 in Plus Configuration

In order to ensure a safe quadroto to trirotor conversion of the AR Drone 2, the quadrotor should be used first in plus configuration instead of cross configuration. This is essential in order to realize a safe and quick quad-to-tri-rotor conversion, and to provide the emerging trirotor with correct position information. Because the UAV uses the camera at its bottom to read its $x$ and $y$ positions, and because the trirotor configuration has always its $x-\text{axis}$ along the rear rotor direction, position information of the quadrotor will be $-45^\circ$ off the trirotor position. To simplify the quadrotor to trirotor conversion maneuver, the AR Drone 2 is first transformed to fly in the plus configuration by rotating its body $45^\circ$ around its $z-\text{axis}$. By applying this rotation, $x$ and $y$ positions returned by the position sensor should be rotated $45^\circ$ around $z-\text{axis}$ in order to get the real position of the quadrotor in plus configuration.

The cross-to-plus configuration conversion requires no change in the quadrotor controller, a simple reallocation of the controls among rotors is enough to
apply the conversion. The correction in $x$ position (pitch angle) will not be performed using all the rotors anymore, but using only two rotors. The cross-to-plus configuration conversion is accomplished through control redistribution (reallocation) in two steps: first, the controls found by the PID controller are used to find the plus configuration rotor speeds using the set of equations (3.54). Second, these rotor speeds are used to generate the cross configuration control laws using the cross configuration rotor speeds/controls relations (equations 3.42). It is important to put in mind that the AR Drone 2 platform accepts only cross configuration control laws, and any moves should be "translated" to this configuration. Figure 3.13 illustrates the cross-to-plus rotor1 configuration conversion, where rotor1 becomes the front rotor.

With the upper conversion applied (rotor1 becomes front rotor), the $x$ and $y$ positions provided by the position sensor should be rotated $45^\circ$ around $z-axis$. If rotor2 is used as front rotor, the position data should be rotated $-45^\circ$, $-135^\circ$ for rotor3 as front rotor, and $135^\circ$ for rotor4 to be front rotor. With each configuration (plus-rotor1, plus-rotor2, etc.) the equations responsible for finding the plus-configuration rotor speeds out of the control laws change accordingly.

The rotor speeds/controls relations for the plus configuration (with rotor1 in the front) are

$$U_{1+} = b(\Omega_{1+}^2 + \Omega_{2+}^2 + \Omega_{3+}^2 + \Omega_{4+}^2)$$

$$U_{2+} = bl(\Omega_{2+}^2 - \Omega_{4+}^2)$$

$$U_{3+} = bl(\Omega_{1+}^2 - \Omega_{3+}^2)$$

$$U_{4+} = d(-\Omega_{1+}^2 + \Omega_{2+}^2 - \Omega_{3+}^2 + \Omega_{4+}^2)$$

The inverse relations between rotor speeds and controls for the plus configuration (with rotor1 in the front) are
The new $x - y$ position of the quadrotor in plus configuration (with rotor1 in the front) will be calculated from the cross configuration position, found using bottom camera, by applying $45^\circ$ rotation transformation or

\[
x_+ = x \times \cos(\pi/4) - y \times \sin(\pi/4)
\]
\[
y_+ = x \times \sin(\pi/4) + y \times \cos(\pi/4)
\]

The roll and pitch angles of the plus configuration will be found by applying the same transformation to the roll and pitch angles of the cross configuration

\[
\phi_+ = \phi \times \cos(\pi/4) - \theta \times \sin(\pi/4)
\]
\[
\theta_+ = \phi \times \sin(\pi/4) + \theta \times \cos(\pi/4)
\]

Note that these values are found for the plus configuration where rotor1 is made the front rotor. This conversion requires rotation transformation around $z - axis$ for $45^\circ$ (or $\pi/4$). For different configurations, the same transformation is applied but with different angle values.

**4.5.4. Emergency Control of AR Drone 2 Quadrotor**

Seeking time and space, only the emergency controller when rotor1 is stopped will be shown below. Emergency controllers for other configurations can be used following similar steps. When rotor1 is damaged and need to be stopped, a weight redistribution maneuver should be applied to shift the CoG towards the rear rotor (rotor3 in this configuration). To ensure the shifting of the CoG towards rotor3, a small plastic weight is connected to the quadrotor frame close to rotor3 as shows Figure 3.14. This weight is connected before starting the experiment, and the PID controller of the healthy quadrotor is seen to accommodate for the model uncertainty emerged after the connection of this weight. Upon the connection of the weight, the CoG is assumed to be shifted $5cm$ towards rotor3 ($l_2 = 5cm$). No precise measurement of the weight and the new CoG position is
applied, and any error between real and estimated values is assumed as uncertainty that will examine the performance of the new controller.

The controls-rotor speeds relations of the trirotor in configuration 1 are

\[
U_{1T_{ri}} = b(\Omega_2^2 + \Omega_3^2 + \Omega_4^2) \\
U_{2T_{ri}} = bl_1(\Omega_2^2 - \Omega_4^2) \\
U_{3T_{ri}} = bl_1(\Omega_2^2 + \Omega_4^2) - bl_3\Omega_3^2 \\
U_{4T_{ri}} = d(\Omega_2^2 - \Omega_3^2 + \Omega_4^2)
\]  

The inverse relations can easily be found as

\[
\Omega_{1T_{ri}}^2 = 0 \\
\Omega_{2T_{ri}}^2 = \frac{U_1}{2b} - \frac{U_2}{2bl_1} - \frac{l_1 - U_3}{2b(l_1 + l_3)} \\
\Omega_{3T_{ri}}^2 = \frac{l_1}{2b(l_1 + l_3)} - \frac{U_3}{2b(l_1 + l_3)} \\
\Omega_{4T_{ri}}^2 = \frac{U_1}{2b} + \frac{U_2}{2bl_1} - \frac{l_1 - U_3}{2b(l_1 + l_3)}
\]

The experiments are conducted at the engineering lab of the Lebanese Inter-
national University (LIU), Tripoli campus. To provide the position sensor of the quadrotor with heterogeneous view and allow it to estimate the precise position, prototype boards of old students electronic projects are used as landmarks. Figure 3.15 shows the experimental set-up at the Lebanese International University, Tripoli campus.

![Figure 3.15](image)

Figure 3.15. – Experimental set-up at the Lebanese International University, Tripoli campus.

The emergency controller uses the same PID controller for both the quadrotor and the trirotor configurations. For the quadrotor configuration, the PID controls are fed directly to the drone. For the trirotor configuration, the PID controls are redistributed among the remaining healthy rotors. The quadrotor to trirotor conversion maneuver should be activated using a Fault Detection and Identification (FDI) unit that estimates the fault affecting the rotors. Whenever a rotor fault is seen to exceed an acceptable threshold this rotor is turned off, and the relevant emergency controller is activated. Note that the FDI unit is not interpreted in our study and left to future work. However, to be realistic an FDI delay is added to the control loop in our experiments. The delay between the fault occurrence and the emergency controller activation is taken as 0.5s.

The PID controller used to control the attitude angles and the height of both the trirotor and quadrotor systems has the following transfer function

\[
\frac{U(S)}{E(S)} = \frac{k_ds^2 + k_ps + k_i}{s}
\]  

(4.70)

Where \( k_d \), \( k_p \), and \( k_i \) are respectively the derivative, the proportional, and the integral gains of the controller. The gains of the PID Controllers are chosen by trial and error. The height and the yaw variable controllers have equal gains with \( k_p = 1, k_d = k_i = 0 \). The PID controllers of the roll and pitch variables have equal gains of \( k_p = 1.3, k_d = 0.15 \), and \( k_i = 1 \). Our experiments show that the choice
of the gains is not conservative, and the gains used for quadrotor control are suitable for trirotor control which adds simplicity to the design.

To control the position of the quadrotor/trirotor along $x$ and $y$ axes a simple PD controller similar to the one used in previous chapters is used. The proportional and derivative gains of the $x$ and $y$ controller are chosen by trial and error as $k_{ip} = 2$ and $k_{id} = 3$. Note that the quadrotor controller has a cascaded double loop architecture. The inner loop in this cascaded controller controls roll and pitch angles, while the outer loop provides the inner controller with the desired roll and pitch angle values, and controls the $x$ and $y$ positions of the system (Figure 3.16). The cascaded controller has the same structure of the controller suggested by [21], where a stability test of the outer loop path tracking controller of the system is also provided.

![Diagram](Image)

**Figure 4.16.** The quadrotor/trirotor control scheme.

Note that the controllability analysis of the trirotor with and without shifting of the CoG should be discussed in details. Using the quadrotor as a trirotor is not regular and subjects the vehicle to harsh flying conditions which renders it unable to hover [91]. This means that a classical linear analysis of the controllability of the trirotor system is not realistic since the system cannot be maintained at the equilibrium point. To check the controllability of the quadrotor used as a trirotor, novel methods should be used. Algebraic controllability analysis presented in [92], Available Control Authority Index (ACAI) method developed in [93], and Attainable Control Sets (ACS) introduced in [94] are possible candidate methods to check the controllability of the new system. The ACS method assesses the performance degradation and the static controllability of a multirotor UAV with arbitrarily distributed rotors subjected to actuator faults. This method suggests to assess the controllability of the faulty system without any yaw control in case a fault configuration renders the complete system uncontrollable.

It is important to emphasize that AR Drone 2 doesn’t allow to stop a rotor completely without changing the API. When a speed command of 0 rpm is sent to one of its motors, AR Drone 2 sets its speed to the minimum possible rotation speed which is 10350 rpm [95], [90]. This means that the emergency controller will not be controlling a real trirotor, and new uncertainties that push the controller to its limits emerge. The PID Controller of the UAV has gains of $K_{zp} = K_{z\theta} = 1$, $K_{zd} = K_{zi} = K_{z\theta_d} = K_{z\theta_i} = 0$, $K_{z\psi} = K_{z\theta} = 1.3$, $K_{z\phi} = K_{z\phi_d} = 0.15$, and $K_{z\psi} = K_{z\psi_d} = 1$.

Both simulation and experimental work using the AR Drone Simulink Development-
Kit V1.1 give good results. Figures 3.18 to 3.23 show the simulation results of the emergency controller controlled quadrotor following respectively a circular and a square path. The results of fault free, faulty case without emergency controller, and faulty case with emergency controller are all shown in the figures. The critical fault is injected at $t = 20\, s$ in the first experiment, and at $t = 17\, s$ in the second experiment. One can see clearly the performance degradation of the AR Drone 2 which results in its height oscillation and yaw angle loss. Despite the effect of the severe fault, the quadrotor controlled with the emergency controller is able to continue following its path as shown in the figures. A video recording of the experiments has been posted online [96].

![Diagram](image.png)

Figure 4.17. - Quadrotor to trirotor control conversion.

![Graphs](image2.png)

Figure 4.18. - Path of the quadrotor model without emergency controller with and without fault (following a circular path).
Path of the quadrotor model without emergency controller with and without fault (following a square path).

Path of the faulty quadrotor model controlled with emergency controller (following a circular path).
Figure 4.21. – Response of the faulty quadrotor model controlled with emergency controller (following a circular path).

Figure 4.22. – Path of the faulty quadrotor model controlled with emergency controller (following a square path).
The response of the fault free real AR Drone 2 controlled with the quadrotor controller is seen in figures 3.24 and 3.25. Figures 3.26 and 3.27 show the effects of critical faults on the AR Drone, controlled with nominal controller, performing a circular and a square path. The effect of the severe fault on the UAV is seen to vary from an experiment to an other. In some experiments, virtual stopping of one of the motors (its speed is decreased to its minimal possible value) causes the UAV to lose height and direction. This results in a sudden fall of the UAV with arbitrary $x$ and $y$ directions. In other experiments, the UAV controller increases the speeds of rotors, trying to correct angle responses, which results in a sudden increase in the UAV height. In both situations, the UAV loses control and crashes.

Figures 3.28, 3.29, 3.30, and 3.31 show the response of the AR Drone controlled with the emergency controller. Note that in these experiments, no FDI unit is used. The delay an FDI unit adds to the fault tolerant system (delay between fault occurrence and fault detection/emergency controller activation) is presented in our code by a 0.5s delay between the instant of the fault occurrence and the emergency controller activation. Experimental results show that despite the emergency controller activation delay, the uncertainties emerged from the imprecise CoG shifting, and the inability to stop the infected rotor, the simple PID controller is still able to control the quadrotor as a trirotor and ensures the completion of its path.

To examine the robustness of the emergency controller without weight redistribution maneuver, several experiments were conducted without the application of this maneuver. Results show that despite the pitch angle becomes less controlled, it is still possible for the emergency controller to prevent the UAV crash. In some experiments, the drop in pitch control didn’t affect the path se-
verely, and the UAV was able to complete its path with some errors. In other experiments, the UAV looses its path, but it remains under partial control. As a conclusion, if the weight re-distribution maneuver is impossible to apply, it is possible to use the emergency controller as an emergency landing controller. Whenever a motor is affected with a severe fault or whenever a rotor stops, the emergency landing controller is activated to prevent crashing and land the UAV safely. Figures 3.32 and 3.33 show the response of the AR Drone following a circular path, with the emergency controller activated without applying the weight re-distribution maneuver.

Figure 4.24. – Response of the real fault-free quadrotor experiment (following a circular path).
Figure 4.25. – Response of the real fault-free quadrotor experiment (following a square path).

Figure 4.26. – Response of the real faulty quadrotor without emergency controller (following a circular path).
Figure 4.27. – Response of the real faulty quadrotor without emergency controller (following a square path).

Figure 4.28. – Path of the real faulty quadrotor controlled with emergency controller (following a circular path).
Figure 4.29. – Response of the real faulty quadrotor controlled with emergency controller (following a circular path).

Figure 4.30. – Path of the real faulty quadrotor controlled with emergency controller (following a square path).
Figure 4.31. – Response of the real faulty quadrotor controlled with emergency controller (following a square path).

Figure 4.32. – Path of the real faulty quadrotor with emergency controller applied without weight redistribution (following a circular path).
The emergency and nominal PID Controllers’ gains are chosen by trial and error as $K_{zp} = K_{z\psi} = 1$, $K_{zd} = K_{zi} = K_{z\psi d} = K_{z\psi i} = 0$, $K_{\phi p} = K_{\theta p} = 1.3$, $K_{\phi d} = K_{\theta d} = 0.15$, and $K_{\phi i} = K_{\theta i} = 1$. The choice of gains is not restrictive, and the same gains (i.e. the same controllers) are used for both quadrotor and trirotor control.
5. Integrated FTC of Quadrotor UAVs in Noisy Environments Based on Fault Severity

Active Fault Tolerant Controllers (AFTC) are more reliable than Passive Fault Tolerant Controllers (PFTC), and can handle more severe faults. On the other hand, they suffer from complexity, high computational demand, and the extensive use of actuator resources which restricts their use in real systems. In this chapter, an Integrated Fault Tolerant Controller (IFTC) for quadrotor UAVs based on Sliding Mode Passive and Active FTCs, and trirotor conversion maneuver is proposed. Extended Kalman Filter (EKF) is used to develop an FDI unit that provides the new controller with the fault magnitude needed for fault tolerance. EKF based fault estimator is able to detect and estimate actuator faults in noisy and uncertain environments. A decision station uses the fault information to activate the suitable controller based on fault severity. The new controller ensures fault tolerance while saving actuator resources and processor computational effort, despite the use of noisy measurement and the presence of model uncertainties.

5.1. Integrated FTC Controller design

The heart of the proposed controller is the decision station which uses fault information provided by the FDI to decide which controller is optimal to deal with the fault. The first controller is the the Passive FTC (PFTC) Sliding Mode Controller which is used for fault free and small fault situations. The PFTC assumes all faults to be matched disturbances that can be compensated with its own robustness. The second controller is the Active FTC (AFTC) Sliding Mode Controller which uses fault magnitudes to compensate for their effects. The last controller is based on the emergency controller and quadrotor to trirotor conversion maneuver, and is used when one of the rotors is critically damaged or totally failed. The structure of the Integrated Fault Tolerant Controller (IFTC) is presented in Figure 4.1. All faults and anomalies that affect the quadrotor are represented by a time varying additive function $F(x, u, t)$, and the process and measurement noises are represented by a centered, white, Gaussian noise.
5.1.1. Passive FTC Sliding Mode Controller

The Passive FTC Sliding Mode Controller is

\[
\begin{align*}
U_1 &= \frac{m}{\cos\theta\cos\phi}[c_x \dot{e}_x + \ddot{z} + g] - k_z \text{sat}(s_z) + K_{iz} \int e_z(\tau) d\tau \\
U_2 &= I_x [c_\phi \dot{e}_\phi + \dot{\phi}_d - \frac{L_{\phi} - L_{\phi}}{I_x} \dot{\theta}_y - \frac{L_{\phi}}{I_x} \dot{\gamma}] - k_\phi \text{sat}(s_\phi) + K_{i\phi} \int e_\phi(\tau) d\tau \\
U_3 &= I_y [c_\theta \dot{e}_\theta + \dot{\theta}_d - \frac{L_{\theta} - L_{\theta}}{I_y} \dot{\phi}_y + \frac{L_{\theta}}{I_y} \dot{\gamma}] - k_\theta \text{sat}(s_\theta) + K_{i\theta} \int e_\theta(\tau) d\tau \\
U_4 &= I_z [c_\psi \dot{e}_\psi + \dot{\psi}_d - \frac{L_{\psi} - L_{\psi}}{I_z} \dot{\phi}_z] - k_\psi \text{sat}(s_\psi) + K_{i\psi} \int e_\psi(\tau) d\tau
\end{align*}
\]

The positive gain diagonal matrices \(c\) and \(k\) are found using Ecological Systems Algorithm (ESA) and are shown in table 2.2.

5.1.2. Active FTC Sliding Mode Controller

The dynamics of a quadrotor UAV affected with loss of effectiveness fault is

\[
\dot{x} = f(x) + g(x)[u(t) - u_f]
\]

with \(u_f\) is the control loss. The Active FTC updates the equation of the passive controller using the fault magnitudes

\[
\begin{align*}
U_{AFTC \ 1} &= u_R \dot{z} + b(F_1 \Omega_2^2 + F_2 \Omega_2^2 + F_3 \Omega_3^2 + F_4 \Omega_4^2) \\
U_{AFTC \ 2} &= u_R \dot{\phi} + b(-F_2 \Omega_2^2 + F_4 \Omega_4^2) \\
U_{AFTC \ 3} &= u_R \dot{\theta} + b(-F_1 \Omega_1^2 + F_3 \Omega_3^2) \\
U_{AFTC \ 4} &= u_R \dot{\psi} + d(F_1 \Omega_1^2 - F_2 \Omega_2^2 + F_3 \Omega_3^2 - F_4 \Omega_4^2)
\end{align*}
\]
Where $u_{Rj}$ with $j = z, \phi, \theta, \psi$ are the controls presented in equation 4.1 with the integral gains are taken equal to zero. Define $F_{i}^{1 \times 4}$ as the faults vector with $F_i = \frac{E_i}{100}$, with $E_i$ here is the percentage of error in motor $i$.

### 5.1.3. Emergency Controller of Quadrotor UAVs

Because quadrotor and trirotor UAVs have identical dynamics, it is possible to turn off one of the quadrotor’s motors and use it as a trirotor. Note that two differences exist between the two UAV configurations: the position of the center of Gravity (CoG), and controls/speeds relations.

The quadrotor is used as trirotor through four different configurations, and these four configurations can be used only if the CoG is shifted towards the rear rotor by means of weight distribution. The four different trirotor configurations require four control/rotor speed relations, and different control allocation is used with each configuration.

If Rotor 1 is turned off, the control effectiveness matrix $B$ should be used as follows

$$B = \begin{bmatrix}
0 & b & b & b \\
0 & -bl_1 & 0 & bl_1 \\
0 & -bl_2 & bl_3 & -bl_2 \\
0 & -dl_4 & dl_3 & -dl_4
\end{bmatrix}$$

(5.4)

If Rotor 2 is turned off, the control effectiveness matrix becomes

$$B = \begin{bmatrix}
b & 0 & b & b \\
bl_1 & 0 & -bl_1 & 0 \\
-bl_2 & 0 & -bl_2 & bl_3 \\
dl_4 & 0 & dl_3 & -dl_4
\end{bmatrix}$$

(5.5)

The control effectiveness matrix used when Rotor 3 is turned off is

$$B = \begin{bmatrix}
b & b & 0 & b \\
0 & bl_1 & 0 & -bl_1 \\
bl_3 & -bl_2 & bl_3 & -bl_2 \\
dl_3 & -dl_4 & 0 & -dl_4
\end{bmatrix}$$

(5.6)

Finally, if Rotor 4 is turned off, the control effectiveness matrix becomes

$$B = \begin{bmatrix}
b & b & b & 0 \\
-bl_1 & 0 & bl_1 & 0 \\
-bl_2 & bl_3 & -bl_2 & 0 \\
dl_4 & -dl_3 & dl_4 & 0
\end{bmatrix}$$

(5.7)

The weight redistribution is performed by installing small extra weights under each rotor, and releasing all the weights except the one under the desired rear rotors.
5.2. Simulation and Results

In this section, the response of the IFTC controller versus various type of faults will be examined. Moreover, the Pelican quadrotor is assumed to be controlled in presence of measurement noise and actuator faults. With the integrated controller, the passive SMC controller is responsible for the quadrotor control in fault-free and low fault situations. When a fault of magnitude greater than 15% affects one of the quadrotor rotors, the active SMC is activated for the UAV control. A wide range of fault types and magnitudes can be handled using this strategy. When a motor fault exceeds 35% the emergency controller is activated, the rotor is stopped, trirotor equations are introduced, and the UAV continues its path as a trirotor. The decision unit is a set of \textit{IF} \textit{ELSE} functions that examine the fault information and outputs the relevant decision based on fault location and severity. Note that the same lead compensator with little influence from higher frequencies used in the previous chapters is used in this chapter as well.

The decision unit contains the set of the following functions:

\begin{align*}
\text{IF } & E_1 \text{ and } E_2 \text{ and } E_3 \text{ and } E_4 < 15\%, \text{ THEN OUT} = 0 \\
\text{IF } & 15\% \leq (E_1 \text{ or } E_2 \text{ or } E_3 \text{ or } E_4) \leq 35\% , \text{ THEN OUT} = 5 \\
\text{IF } & E_1 > 35\% , \text{ THEN OUT} = 1 \\
\text{IF } & E_2 > 35\% , \text{ THEN OUT} = 2 \\
\text{IF } & E_3 > 35\% , \text{ THEN OUT} = 3 \\
\text{IF } & E_4 > 35\% , \text{ THEN OUT} = 4
\end{align*}

where $E_i$ is the estimated error percentage in motor $i$, and OUT is the output of the decision unit. Note that when OUT = 0, the quadrotor uses the Passive FTC. When OUT = 5, Active FTC is activated and the quadrotor controller is reconfigured according to the estimated fault magnitudes. When OUT = 1, 2, 3, or 4, the emergency controller regarding rotor 1, 2, 3, or 4 respectively is activated.

Figure 4.2 shows the path of the IntFTC controlled quadrotor infected by multiple successive partial loss in effectiveness faults in all its rotors. The first fault is a 20% fault in motor 3 at $t = 15s$, followed by a 10% fault in motor 4 at $t = 20s$. A third fault of 30% affects motor 1 at $t = 30s$, is followed by a 70% critical fault that affects motor 2 at $t = 45s$. The position, altitude, and attitude responses of the infected quadrotor are shown in Figure 4.4. Results show that the quadrotor needs some time to recover from the critical fault and catch the desired path. Despite the deviation, the controller exhibits good robustness against motor 3 failure.
Figures 4.2 to 4.4 show the quadrotor response for the first fault scenario. The decision unit output is shown in Fig. 4.2 (b). When the fault magnitude is less than 15% or when no fault exists, decision unit outputs zero, activating the PFTC. When the FDI unit detects a fault between 15% and 35%, the decision unit outputs 5, activating the AFTC. Severe faults and failures make the decision unit output 1, 2, 3, or 4 activating the relevant quadrotor to trirotor configuration and emergency controller.

Figures 4.5 to 4.7 show the results of a different fault scenario. In this scenario, a small fault of 13% affects rotor 3 at $t = 15$. This fault increases after 25s to reach 25%. As a result, the AFTCSMC is activated directly and a small deviation can be seen in the quadrotor path. After 15s the same rotor fails completely, the IFTC turns rotor 3 off and the quadrotor continues its path as a trirotor. The path followed by the quadrotor in the second fault scenario is the helical path presented in the previous chapters.

Finally, figures 4.8 to 4.10 show the response of the quadrotor following a square path in the space, with 20% fault injected to motor 2 at $t = 20s$, followed by a critical fault of 35% injected to motor 1 at $t = 50s$.

![Helical Path and Decision Unit Output](image)

Figure 5.2 - Helical path and decision unit output - first fault scenario.
(a) Controls

(b) Rotor speeds

Figure 5.3. -- Controls and rotor speeds - first fault scenario.
Figure 5.4. – Response of the quadrotor with the IntFTC - first fault scenario.
Figure 5.5. – Helical path and decision unit output - second fault scenario.

Figure 5.6. – Controls and rotor speeds - second fault scenario.
Figure 5.7. – Response of the quadrotor with the IntFTC - second fault scenario.
Figure 5.8. – Square path and decision unit output.

Figure 5.9. – Controls and rotor speeds - Square path.
Figure 5.10. – Response of the quadrotor with the IntFTC - Square path.
6. Fault Tolerant Control of Octorotor UAVs

Overactuated -or redundant- systems have more actuators than controlled variables, i.e. they have more actuators than strictly needed in order to perform the control objectives. These systems are widely used in applications where safety and reliability are main issues such as aerospace and maritime applications. Redundancy in systems allows to distribute the control effort among multiple existing actuators according to a given method or strategy. The distribution strategy, called also control allocation, allows the controller to achieve secondary goals other than realizing the control such as avoiding actuators saturation, tolerating actuator faults and failures, as well as reducing the cost, power consumption, wear, and tear of actuators, or any other important task. With Control allocation, the design of overactuated control systems is simplified to the design of a controller that generates one single virtual command for each control objective, and the design of a control allocator that assigns the optimal actuator for each command.

Octorotor UAVs have eight rotors to achieve the control of their $x$, $y$, $z$ coordinates, along with their roll, pitch, and yaw angles. Octorotors are overactuated UAVs with more rotors than their controlled variables. This means more payload capacity and higher fault tolerant capability compared to quadrotors.

6.1. Octorotor Model

In this chapter, the star-shaped Octorotor configuration developed by V. Adir [97] is used as test platform to develop new Fault Tolerant Controllers. The star-shaped octorotor (Figure 5.1) has the following model
\[ \ddot{x} = \frac{U_1}{m}(\sin \psi \sin \phi + \cos \psi \sin \theta \cos \phi) \quad (6.1) \]
\[ \ddot{y} = \frac{U_1}{m}(-\cos \psi \sin \phi + \sin \psi \sin \theta \cos \phi) \quad (6.2) \]
\[ \ddot{z} = \frac{U_1}{m} \cos \theta \cos \phi - g \quad (6.3) \]
\[ \ddot{\phi} = \frac{I_y - I_z}{I_x} \dot{\theta} \psi - \frac{I_{\text{rotor}}}{I_x} \dot{\theta} \gamma + \frac{l U_2}{I_x} \quad (6.4) \]
\[ \ddot{\theta} = \frac{I_z - I_x}{I_y} \dot{\phi} \psi + \frac{I_{\text{rotor}}}{I_y} \dot{\phi} \gamma + \frac{l U_3}{I_y} \quad (6.5) \]
\[ \ddot{\psi} = \frac{I_x - I_y}{I_z} \dot{\theta} \phi + \frac{U_4}{I_z} \quad (6.6) \]

Figure 6.1. - Octorotor schematic [97].

Octorotor dynamic equations resemble to the dynamic equations of a quadrotor; and the parameters in the equations above are defined in the first chapter following equation (1.6). Note that the aerodynamic and the drag forces are ignored in the octorotor equations. The parameters of the star-shaped octorotor according to [97] are shown in table 5.1.

The relation between the controls of the octorotor and the speeds of its motors is given by the following equation

\[ [U_1 \ U_2 \ U_3 \ U_4]^T = B \Omega \quad (6.7) \]

With \( \Omega = [\Omega_1^2 \ \Omega_2^2 \ \Omega_3^2 \ \Omega_4^2 \ \Omega_5^2 \ \Omega_6^2 \ \Omega_7^2 \ \Omega_8^2]^T \), \( \Omega_i \ (i = 1, \ldots, 8) \) is the speed of motor \( i \), and \( U_1, U_2, U_3, \) and \( U_4 \) are respectively the thrust, and the roll, pitch
and yaw torques. Matrix $B$ is the control effectiveness matrix defined as

$$B = \begin{bmatrix}
    b & b & b & b & b & b & b & b & b & b & b & b \\
    -bP & -bP & -bp & bp & bP & bP & bp & bp & bp & bp & bp & bp \\
    -d & -d & d & d & -d & -d & d & d & d & d & d & d
\end{bmatrix}$$

(6.8)

with $P = l \cos(22°30')$, and $p = l \sin(22°30')$.

$$\gamma = \Omega_3^2 + \Omega_4^2 + \Omega_5^2 + \Omega_7^2 - \Omega_1^2 - \Omega_2^2 - \Omega_5^2 - \Omega_6^2$$

(6.9)

Because the octorotor has no sensors measuring its motor speeds, the value of $\gamma$ is unavailable for the controller. Thus, the term $I_i \dot{j}\gamma$ with $i = x, y$, and $j = \phi, \theta$ is assumed as disturbance.

Following the simplification of motor speed - controls relations suggested in [97], each two motors of the octorotor are paired. Each pair of motors are assumed to rotate at the same direction and with the same speed. Motor 1 and motor 2 have speed of $\Omega_A$, $\Omega_B$ is the speed of motor 3 and motor 4, motors 5 and 6 rotate with speed of $\Omega_C$, and motors 7 and 8 with speed of $\Omega_D$. This assumption transforms motor speed - controls relations into a simple matrix form(equations (5.10) to (5.13)), which makes it easy to find the motor speeds out of the controls by means of matrix inversion process. It is important to note that the previous assumption holds only in fault free mode where the octorotor behaves like a quadrotor. However, when subjected to actuator fault coupling appears and the assumption becomes impractical.

$$U_1 = 2b(\Omega_A^2 + \Omega_B^2 + \Omega_C^2 + \Omega_D^2)$$

(6.10)

$$U_2 = 2bP(\Omega_D^2 - \Omega_B^2)$$

(6.11)

$$U_3 = 2bP(\Omega_C^2 - \Omega_A^2)$$

(6.12)

$$U_4 = 2d(\Omega_B^2 + \Omega_D^2 - \Omega_A^2 - \Omega_C^2)$$

(6.13)

By taking $x = [x_1 \ x_2 \ x_3 \ x_4 \ x_5 \ x_6 \ x_7 \ x_8 \ x_9 \ x_{10} \ x_{11} \ x_{12}]^T$ as the state vector, $y = [x \ y \ z \ \phi \ \theta \ \psi]^T$ as the output vector, and $u(t) = [U_1 \ U_2 \ U_3 \ U_4]^T$ as the control input vector, the octorotor dynamics can be expressed in the state

| $I_x$   | $44e^{-3}Kgm^2$ | $b$ | $10e^{-6}Ns^2$ |
| $I_y$   | $44e^{-3}Kgm^2$ | $l$ | $0.4m$         |
| $I_z$   | $88e^{-3}Kgm^2$ | $d$ | $0.3e^{-6}ms^2$ |
| $I_{rotor}$ | $90e^{-6}Kgm^2$ | $g$ | $9.8ms^{-2}$  |
| $m$     | $1.64Kg$        |     |                 |

Table 6.1 - Octorotor Variables
where $f(x)$ is a $12 \times 1$ column vector, $g(x)$ is $12 \times 4$ matrix, the control vector is $4 \times 1$, and $C$ is a $6 \times 12$ gain diagonal matrix with

$$
f(x) = \begin{bmatrix}
x_2 \\
0 \\
x_4 \\
0 \\
x_6 \\
-g \\
x_8 \\
\frac{I_x - I_z}{I_x} x_{10} x_{12} - \frac{I_{\text{rotor}}}{I_x} x_{10} \gamma \\
x_{10} \\
\frac{I_x - I_z}{I_y} x_8 x_{12} + \frac{I_{\text{rotor}}}{I_y} x_8 \gamma \\
x_{12} \\
\frac{I_x - I_y}{I_x} x_8 x_{10}
\end{bmatrix}$$

$$
g(x) = \begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\frac{\sin x_{11} \sin x_{1} + \cos x_{11} \sin x_{9} \cos x_{1}}{m} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\frac{- \cos x_{11} \sin x_{1} + \sin x_{11} \sin x_{9} \cos x_{1}}{m} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & \frac{\cos x_8 \cos x_{12}}{m} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & \frac{1}{I_y} & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{I_x} & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & \frac{1}{I_x} & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{I_y} & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{I_z}
\end{bmatrix}
$$

$$
C = \begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{bmatrix}
$$

In this chapter, an Extended Kalman Filter similar to the one used in chapter 2.
is used to estimate the octorotor states. The EKF is presented in equations (2.23) to (2.28). Here also, the same noise amplitude and standard deviation used in previous chapters is applied to the octorotor.

6.2. Control Allocation Problem

Control Allocation is an optimization problem used with redundant systems to distribute the control demand among available actuators. The desired control effort in overactuated systems can be achieved using different possible actuator configurations, so the optimization problem is solved to find controls that fulfill the control objectives and achieve secondary goals. Secondary goals range from respecting the actuator position and speed boundaries to minimizing the disturbance and actuator usage. Control allocation can be used as fault tolerant control method that minimizes fault effects by using only healthy actuators to realize the control effort. Note that for UAV’s, using control allocation methods is only possible if the thrust produced by the remaining healthy actuators is sufficient to compensate for the weight of the system.

The Control Allocation problem will be explained for linear systems [98], [36]. The linear system

\[
\begin{align*}
\dot{x} &= Ax + B_u u(t) \\
y &= Cx
\end{align*}
\]

(6.19) (6.20)

can be transformed to a new linear system by introducing a virtual control vector \( v \)

\[
\begin{align*}
\dot{x} &= Ax + B_v v \\
v &= B_u u(t) \\
y &= Cx
\end{align*}
\]

(6.21) (6.22) (6.23)

Where \( x \in \mathbb{R}^n \) is the state vector of the system, \( u \in \mathbb{R}^m \) is the control input, \( y \in \mathbb{R}^p \) is the system output, \( (A, B_v) \) is stabilizable, \( B_v \in \mathbb{R}^{n \times k} \), and \( B \in \mathbb{R}^{k \times m} \). \( v \in \mathbb{R}^k \), \( B \) and \( B_u \) have null space of dimension \( m - k \), and \( \text{rank}(B_u) = k < m \) so that \( B_u = B_v B \) can be set.

For the octorotor system, the control allocation problem is summarized as [37]: given a virtual control vector \( v \) generated by the controller, find the set of propeller speeds \( \Omega \) such that \( u = B \Omega \) with

\[
\begin{align*}
\Omega &= [\Omega_1^2 \Omega_2^2 \ldots \Omega_8^2]^t \\
u &= [U_1 \ U_2 \ U_3 \ U_4]^t \\
u &= B \Omega
\end{align*}
\]

(6.24) (6.25) (6.26)
Control allocation problem of the octorotor is illustrated in (Figure 5.2). The control allocation problem is solved by optimizing vector $B$.

6.2.1. Pseudo-Inverse Control Allocation

The simplest method to solve the control allocation problem is to transform it into the following optimization problem:

$$\min \| W_u u \| \quad \text{subject to } Bu = v \quad (6.27)$$

$W_u$ is a diagonal matrix weighting each actuator. This problem suggests to find the minimum control input while ensuring the relation between the virtual and the real controls. Moore-Penrose pseudo inverse method can be used to solve the control allocation problem as

$$u = W_u^{-1} \left( BW_u^{-1} \right)^{+} v \quad (6.28)$$

Where the pseudo-inverse of a matrix $M$ is $M^+ = M^T \left( MM^T \right)^{-1}$ [99].

6.2.2. Dynamic Control Allocation

Pseudo-Inverse is a simple and straightforward control allocation method, but it doesn’t take into account the position and speed limits of the actuators. This is critical in real applications where the required control effort might not be achievable with saturated actuators. This control allocation method is also static, and the output depends only on the current control demands, and in two consecutive moments, control distribution of the system could vary dramatically. If it is possible to relate the output of the control allocator to its previous value, actuator bandwidth could be taken into account, and the behavior of control distributor could be changed between transient and steady-state intervals [54]. Dynamic Control Allocation method finds the best control configuration that keeps the control input close to its desired steady state value while minimizing the diffe-
rence between the current and previous control inputs

\[ u(t) = \arg\min_{u(t) \in \Gamma} (\|W_1(u(t) - u_s(t))\|^2 + \|W_2(u(t) - u(t - T))\|^2) \] (6.29)

\[ \Gamma = \arg\min_{u_{\min} \leq u \leq u_{\max}} (\|W_v(Bu(t) - v(t))\|) \] (6.30)

The optimization problem suggested above states that given the set of controls that minimize the virtual control error \( Bu(t) - v(t) \) (weighted by \( W_v \)), choose the control input that minimizes both the difference between the actual and the steady state control input \((u(t) \text{ and } u_s(t) \text{ respectively})\), and the difference between the actual control and its previous value \((u(t) \text{ and } u(t - T) \text{ respectively})\).

\( W_1, W_2, \text{and } W_v \) are weighting matrices, \( B \) is the control effectiveness matrix, \( v \) is the virtual control, \( \|u\| = \sqrt{u^T u} \) is the Euclidean 2-norm, and the control constraints are \( u_{\min} \) and \( u_{\max} \).

For the octorotor, the Dynamic Control Allocation (DCA) becomes: given \( \Omega \) the set of rotor speeds that minimize the virtual control error \( B\Omega(t) - u(t) \) (weighted by \( W_u \)), choose the suitable control that minimizes both the difference between the actual and the steady state rotor speeds \((\Omega(t) \text{ and } \Omega_s(t) \text{ respectively})\), and the difference between the actual rotor speeds and their previous values \((\Omega(t) \text{ and } \Omega(t - T) \text{ respectively})\), or

\[ \Omega(t) = \arg\min_{\Omega(t) \in \Gamma} (\|W_1(\Omega(t) - \Omega_s(t))\|^2 + \|W_2(\Omega(t) - \Omega(t - T))\|^2) \] (6.31)

\[ \Gamma = \arg\min_{\Omega_{\min} \leq \Omega \leq \Omega_{\max}} (\|W_u(B\Omega(t) - u(t))\|) \] (6.32)

with \( \|\Omega\| = \sqrt{\Omega^T \Omega} \) is the Euclidean 2-norm, and the control constraints are \( \Omega_{\min} \) and \( \Omega_{\max} \).

With no actuator saturation, the optimal problem can be simplified into

\[
\min_{u(t)} (\|W_1(u(t) - u_s(t))\|^2 + \|W_2(u(t) - u(t - T))\|^2) \\
Subject \ to \ Bu(t) = v(t)
\] (6.33)

and for the octorotor,

\[
\min_{\Omega(t)} (\|W_1(\Omega(t) - \Omega_s(t))\|^2 + \|W_2(\Omega(t) - \Omega(t - T))\|^2) \\
Subject \ to \ B\Omega(t) = u(t)
\] (6.34)
The solution of the above control allocation problem is given as [55]

\[ u(t) = Fu(t - T) + Gv(t) \]  \hspace{1cm} (6.35)

or

\[ \Omega(t) = F\Omega(t - T) + Gu(t) \]  \hspace{1cm} (6.36)

where

\[ F = W^{-1} \left( I - B^T H^{-1} B W^{-1} \right) W_2 \]  \hspace{1cm} (6.37)

\[ W = W_1 + W_2 \]  \hspace{1cm} (6.38)

\[ H = BW^{-1} B^T \]  \hspace{1cm} (6.39)

\[ G = B^T H^{-1} \]  \hspace{1cm} (6.40)

\( W \) is nonsingular and \( W_1 \) and \( W_2 \) are symmetric matrices. A trade-off is made while choosing the weighting matrices: high \( W_1 \) value results in high convergence speed of the control to its desired value, while high \( W_2 \) value results in limited rate of change of actuator values.

### 6.3. First and Second Order Sliding Mode FTC

#### 6.3.1. First Order Sliding Mode Controller Design

\( s_i = e_i + c_i \int e_i(\tau)d\tau \) is chosen as sliding surface of the octorotor UAV, with \( e_i = i_d - \dot{i} \) for \( i = \phi, \theta, \psi \), and \( z \). \( V_i = \frac{1}{2}s_i^2 \) is chosen as the Lyapunov function, if \( \dot{V}_i \) is negative then the system trajectory is ensured to stay on the sliding surface. \( \dot{V}_i = s_i \dot{s}_i = s_i(\dot{e}_i + c_i e_i) = s_i(i_d - \dot{i} + c_i e_i) \). For \( \dot{V}_i \) to be negative \( i_d - \dot{i} + c_i e_i \) should be less than zero. Using the dynamics of the octorotor, the equivalent control is found as

\[ u_{ieq} = \frac{\dot{i}_d + c_i e_i - f_i(x)}{g_i(x)} \]  \hspace{1cm} (6.41)

where \( c = [c_z, c_{\phi}, c_{\theta}, c_{\psi}] \) is a positive gain vector. The discontinuous control is designed using the saturation function

\[ u_{idis} = -k_i sat(s_i) \]  \hspace{1cm} (6.42)

where \( k = [k_z, k_{\phi}, k_{\theta}, k_{\psi}] \) is a positive gain vector affecting the conversion speed of the discontinuous control. Finally, the Sliding Mode Controller of the octorotor is

\[ u_i = \frac{\dot{i}_d + c_i e_i - f_i(x)}{g_i(x)} - k_i sat(s_i) \]  \hspace{1cm} (6.43)

with \( u_z = U_1, u_{\phi} = U_2, u_{\theta} = U_3, u_{\psi} = U_4 \).
Variables Used In ESA To Tune the Octorotor SMC.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Search Space</th>
<th>Steps</th>
<th>Distance</th>
</tr>
</thead>
<tbody>
<tr>
<td>φ intervals</td>
<td>x : [0.2 10], y : [0.2 2]</td>
<td>x : 0.2, y : 0.1</td>
<td>0.08</td>
</tr>
<tr>
<td>θ intervals</td>
<td>x : [0.2 10], y : [0.2 10]</td>
<td>x : 0.2, y : 0.2</td>
<td>0.08</td>
</tr>
<tr>
<td>ψ intervals</td>
<td>x : [0.2 10], y : [0.2 5]</td>
<td>x : 0.2, y : 0.2</td>
<td>0.08</td>
</tr>
<tr>
<td>z intervals</td>
<td>x : [1 20], y : [30 200]</td>
<td>x : 0.5, y : 5</td>
<td>0.3</td>
</tr>
</tbody>
</table>

Best FOSMC Gain Values Found Using ESA

<table>
<thead>
<tr>
<th>z</th>
<th>φ</th>
<th>θ</th>
<th>ψ</th>
</tr>
</thead>
<tbody>
<tr>
<td>8.5387</td>
<td>4.6484</td>
<td>5.8688</td>
<td>4.9498</td>
</tr>
<tr>
<td>129.3544</td>
<td>1.3067</td>
<td>5.8096</td>
<td>2.5246</td>
</tr>
</tbody>
</table>

6.3.1.1. FOSMC Tuning Using Ecological Systems Algorithm

Designing the Sliding Mode Controller requires the choice of the gain of sliding surface slope vector $c$, and the gain of discontinuous control vector $k$. The choice of these gain matrices should be made carefully to ensure the sliding mode condition and to realize the best performance. The matrices contain elements for each control variable, which means that the number of variables to be optimized here is eight. Ecological Systems Algorithm (ESA) will be used to optimize $c$ and $k$ matrices.

The maximum age of each individual is set to 10, and its initial health is set to 100. 80 zebra and 5 lion individuals are generated at the beginning of the search for each control variable. If a zebra individual manages to move to a better location in a given iteration, its health increases by a factor of 3. On the other hand, its health decreases by a factor of 0.98 if the new location is worse. If a zebra survives a lion attack, its health decreases by a factor of 0.8. If a lion eats a zebra, its health increases by a factor of 1.2; and it decreases by 0.8 if it doesn’t eat in an iteration. If the health of any individual drops below 20, this individual dies. The stop criteria is when the value of the fitness function exceeds 7. Table 6.2 shows the search space boundaries of each control variables, and the step sizes of each zebra individual in $x$ and $y$ axes coordinates, as well as the maximum distances that allow lions to detect and attack their preys. The optimal gains for the octorotor SMC controller found using ESA are shown in Table 6.3.

6.3.1.2. FOSMC Stability Test

The stability of the octorotor SMC controller is tested using Lyapunov stability by choosing $V_i = \frac{1}{2}s_i^2$ as Lyapunov function. The derivative of the Lyapunov function is $\dot{V}_i = s_i(\dot{i} - \dot{i} + c_i e_i) = s_i(\dot{i} - f_i(x) - g_i(x)u_i + c_i e_i)$. By inserting the SMC law designed previously, the Lyapunov function becomes $\dot{V}_i = s_i(f_i(x) -$
Here, $\tilde{f}_i(x)$ term is identical to $f_i(x)$ but with no $\gamma$. Let $F_i(x) = f_i(x) - \tilde{f}_i(x)$, so that $\dot{V}_i = s_i F_i(x) - k_i g_i(x)|s_i|$ is negative if $k$ is chosen as $k_i \geq \frac{F_i(x) + \eta_i g_i(x)}{g_i(x)}$, where $\eta$ is a vector with small positive elements.

The value of $F(x)$ is found for each variable using the octorotor parameters, its constraints, and the maximum possible $\gamma$ value which is $\gamma_{max} = 3600$ (equation (5.9)). $I_i$ with $i = x, y, z$ are positive numbers, and because the octorotor angles are assumed to be within the $[-0.31, 0.31]$ rad interval, the function $g(x)$ is greater than zero. As a result, in order for the SMC controller to be Lyapunov stable, the gains $k$ should be $k_z \geq 0$, $k_\phi \geq 0.81\phi$, $k_\theta \geq 0.81\theta$, and $k_\psi \geq 0$. To ensure safe navigation of the octorotor a new constraint on the angular speed is added $\dot{i} \leq 1.24 rad/sec$, with $i = \phi, \theta$. With the introduction of the new constraint, the SMC is stable if $k_\phi$ and $k_\theta$ are greater than 1.0044. SMC controller gains found using ESA (table 5.4) respect the constraints found in this paragraph, this means that the SMC controller is Lyapunov stable.

### 6.3.2. Second Order Sliding Mode Controller

#### 6.3.2.1. Second Order Sliding Mode Controller Design

In Higher Order Sliding Mode Control (HOSMC) not only the sliding surface is set to zero, but also its high order derivatives. HOSMC gives a final control law with an integrated discontinuous function term which helps in chattering attenuation.

The sliding surface is chosen as $s_i = \dot{e}_i + c_i e_i$, where $e_i = i_d - i$ is the tracking error for $i = \phi, \theta, \psi$, and $z$. The first derivative of the sliding surface is $\dot{s}_i = \ddot{e}_i + c_i \dot{e}_i$, and by using equation (5.14) the equivalent control is found as

$$ u_{i\, eq} = \frac{\dot{i}_d + c_i \dot{e}_i - f_i(x)}{g_i(x)} \quad (6.44) $$

Where $c = [c_z c_\phi c_\theta c_\psi]$ is a positive gain vector.

The discontinuous part of the control is chosen according to the Super-Twisting Algorithm as [100], [101]

$$ u_{i\, dis} = -\lambda_i |s_i|^{0.5} \text{sign}(s_i) - \int \alpha_i \text{sign}(s_i) dt \quad (6.45) $$

where $\lambda = [\lambda_z \lambda_\phi \lambda_\theta \lambda_\psi]$ and $\alpha = [\alpha_z \alpha_\phi \alpha_\theta \alpha_\psi]$ are positive gain vectors affecting the conversion speed of the discontinuous control.

The octorotor second order SMC control is $u = u_{eq} + u_{dis}$.  

150
6.3.2.2. SOSMC Tuning

In order for the algorithm to converge in finite time, \( \lambda \) and \( \alpha \) vectors are chosen according to the following conditions \([102]\)

\[
\alpha_i > \frac{\Phi_i}{G_{i2}} \quad (6.46)
\]

\[
\lambda_i \geq \sqrt{\frac{4\Phi_i G_{i2}(\alpha_i + \Phi_i)}{G_{i1}^2 G_{i1}(\alpha_i - \Phi_i)}} \quad (6.47)
\]

where \( G_{i1}, G_{i2}, \) and \( \Phi_i \) are constraints on the second order derivative of the sliding surface

\[
\ddot{s}_i = \varphi_i(t, x, u) + G_i(t, x, u)\dot{u}_i \quad (6.48)
\]

\[
|\varphi_i(t, x, u)| \leq \Phi_i \quad (6.49)
\]

\[
0 \leq G_{i1} \leq G_i(t, x, u) \leq G_{i2} \quad (6.50)
\]

Choosing \( \alpha_i \) and \( \lambda_i \) is a hard task where a trade-off is made between dynamic performance of the system and chattering attenuation. A good choice of the controller gain vectors ensures the fast conversion of the system with reduced chattering. A bad choice of the gains within the suggested constraints will ensure the fast conversion of the system but with non attenuated chattering, or the slow conversion of the system with attenuated chattering. Ecological Systems Algorithm (ESA) is used to find the SOSMC controller gains.

To ensure that the SOSMC gains found using ESA are within the accepted range, the search space boundaries are chosen according to equations \((5.46)\) and \((5.47)\). The second order derivative of the sliding surface is

\[
\ddot{s}_z = \dddot{z}_d - \dddot{z} + c_z(\dddot{z}_d - \dddot{z}) \quad (6.51)
\]

\[
\ddot{s}_\phi = \dddot{\phi}_d - \dddot{\phi} + c_\phi(\dddot{\phi}_d - \dddot{\phi}) \quad (6.52)
\]

\[
\ddot{s}_\theta = \dddot{\theta}_d - \dddot{\theta} + c_\theta(\dddot{\theta}_d - \dddot{\theta}) \quad (6.53)
\]

\[
\ddot{s}_\psi = \dddot{\psi}_d - \dddot{\psi} + c_\psi(\dddot{\psi}_d - \dddot{\psi}) \quad (6.54)
\]
The third order derivatives of the control variables are

\begin{align}
\ddot{\varphi} &= \frac{U_1}{m}(-\sin \theta \cos \phi - \cos \theta \sin \phi) + \frac{\dot{U}_1}{m} \cos \theta \cos \phi \\
\ddot{\phi} &= \frac{I_y - I_z}{I_x} (\dot{\theta} \ddot{\psi} + \dot{\theta} \dot{\psi}) - \frac{I_{\text{rotor}}}{I_x} \ddot{\theta} + \frac{\dot{I}_2}{I_x} \\
\ddot{\theta} &= \frac{I_z - I_x}{I_y} (\dot{\phi} \ddot{\psi} + \dot{\phi} \dot{\psi}) + \frac{I_{\text{rotor}}}{I_y} \ddot{\phi} + \frac{\dot{I}_3}{I_y} \\
\ddot{\psi} &= \frac{I_x - I_y}{I_z} (\ddot{\phi} \dot{\theta} + \dot{\phi} \ddot{\theta}) + \frac{\dot{I}_4}{I_z}
\end{align}

with \( \ddot{i}_d \) accepted as zero for \( i = z, \phi, \theta, \) and \( \psi, \) the second order derivative of the sliding surface is transformed to the form \( \ddot{s} = \varphi(t, x, u) + G(t, x, u) \ddot{u} \) by setting

\[
\varphi = \begin{bmatrix}
\frac{U_1}{m} (\sin \theta \cos \phi + \cos \theta \sin \phi) + c_z (\ddot{z}_d - \ddot{z}) \\
-\frac{I_{\text{rotor}}}{I_x} \ddot{\theta} + \frac{I_{\text{rotor}}}{I_y} \ddot{\phi} + c_\phi (\ddot{\phi}_d - \ddot{\phi}) \\
-\frac{I_x - I_y}{I_z} (\ddot{\phi} \dot{\theta} + \dot{\phi} \ddot{\theta}) + c_\psi (\ddot{\psi}_d - \ddot{\psi})
\end{bmatrix}
\]

and

\[
G = \begin{bmatrix}
\frac{\dot{I}_1}{m} \\
\frac{\dot{I}_2}{I_x} \\
\frac{\dot{I}_3}{I_y} \\
\frac{\dot{I}_4}{I_z}
\end{bmatrix}
\]

First, we are going to find the accepted search boundaries for the controller gains. By assuming that the octorotor weight along with its maximum load is 3 Kg (\( U_1 \leq 30 N \)), the roll, pitch, and yaw angle interval is \([-50° 50°]\) \([-0.8727 0.8727] \text{rad}\), the angles are not allowed to change from minimum to maximum value in less than 0.5sec \((\text{angle}_{\text{max}} = 3.5 \text{rad/sec})\), and the angular velocities are not allowed to change from zero to their maximum values in less than one second \((\text{angle}_{\text{max}} = 3.5 \text{rad^2/sec})\), \( c_z \) is not more than 10 and \( c_{\text{angles}} \) are not more than 5, and the fact that \(-1 \leq \cos(.) \leq 1\), along with the octorotor constants yield \( G_1 = [0.1376 \ 9.09 \ -9.09 \ 11.36]^T \) and \( G_2 = [0.61 \ 9.09 \ -9.09 \ 11.36]^T \) and \( \Phi = [11.1 \ 2 \ -1.998 \ 0]^T \). By using these values, the search boundaries for \( \lambda \) and \( \alpha \) are found as \( \alpha > [7.9489 \ 1.3208 \ -2.7 \ -1.1]^T \) and \( \lambda > [6.9674 \ 0 \ 1.22 \ 0.7]^T \). By using these constraints as boundaries for the search space, any gains found using ESA are ensured to be valid and optimal.

The optimal gains for the SOSMC controller found with ESA while respecting the above constraints are shown in table 5.4.
Table 6.4. – Best SOSMC Gain Values Found Using ESA

<table>
<thead>
<tr>
<th></th>
<th>(z)</th>
<th>(\phi)</th>
<th>(\theta)</th>
<th>(\psi)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>13.3171</td>
<td>3.8910</td>
<td>4.7694</td>
<td>3.2163</td>
</tr>
<tr>
<td>(k)</td>
<td>2.1440</td>
<td>6.1732</td>
<td>4.3244</td>
<td>4.2674</td>
</tr>
<tr>
<td>(\alpha)</td>
<td>13.2635</td>
<td>1.6484</td>
<td>3.3275</td>
<td>4.4601</td>
</tr>
<tr>
<td>(\lambda)</td>
<td>8.1768</td>
<td>9.2096</td>
<td>4.5294</td>
<td>4.0075</td>
</tr>
</tbody>
</table>

Table 6.5. – Fault Effects On Motor Speeds

<table>
<thead>
<tr>
<th>Motor Fault</th>
<th>Change in (\omega_1^2)</th>
<th>Change in (\omega_2^2)</th>
<th>Change in (\omega_3^2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>M1 Fault</td>
<td>Change in (\omega_1^2)</td>
<td>Change in (\omega_2^2)</td>
<td>Change in (\omega_3^2)</td>
</tr>
<tr>
<td>M2 Fault</td>
<td>Change in (\omega_2^2)</td>
<td>Change in (\omega_3^2)</td>
<td>Change in (\omega_4^2)</td>
</tr>
<tr>
<td>M3 Fault</td>
<td>Change in (\omega_3^2)</td>
<td>Change in (\omega_4^2)</td>
<td>Change in (\omega_5^2)</td>
</tr>
<tr>
<td>M4 Fault</td>
<td>Change in (\omega_4^2)</td>
<td>Change in (\omega_5^2)</td>
<td>Change in (\omega_6^2)</td>
</tr>
<tr>
<td>M5 Fault</td>
<td>Change in (\omega_5^2)</td>
<td>Change in (\omega_6^2)</td>
<td>Change in (\omega_7^2)</td>
</tr>
<tr>
<td>M6 Fault</td>
<td>Change in (\omega_6^2)</td>
<td>Change in (\omega_7^2)</td>
<td>Change in (\omega_8^2)</td>
</tr>
<tr>
<td>M7 Fault</td>
<td>Change in (\omega_7^2)</td>
<td>Change in (\omega_8^2)</td>
<td>Change in (\omega_9^2)</td>
</tr>
<tr>
<td>M8 Fault</td>
<td>Change in (\omega_8^2)</td>
<td>Change in (\omega_9^2)</td>
<td>Change in (\omega_1^2)</td>
</tr>
</tbody>
</table>

6.4. Design of the FDI Unit

In this chapter, an FDI unit based on the solution of the control allocation optimal problem stated in equations (5.31) and (5.32) is developed. Note that because we are using paired motor speed configuration of the octorotor, using the control signals for fault detection gives poor results. Control signals were unable to give sufficient information to detect the precise fault location, and it is impossible to differ between paired motor faults. To detect the fault location, the FDI unit is developed based on the solution of the optimal control allocation problem given in equation (5.36) which gives the squared rotor speeds of the octorotor. Squared rotor speeds calculated using equation (5.36) are filtered with an 8th order Butterworth filter with cut off frequency equal to 2 rad/sec. The changes in the filtered signals are calculated each and used to detect the fault location based on table 6.5. This table shows the effect of different faults on the filtered squared rotor speed signals. To estimate the fault magnitude affecting a motor, the FDI unit first uses the changes in motor speeds and table 5.5 to detect the fault location. Once the faulty motor is found, the change in its speed is fed to table 6.6 to estimate the fault magnitude.

Tests emphasize the accuracy of the FDI unit when less than four severe faults

Table 6.6. – Look-Up Table Used To Estimate Fault Magnitude

<table>
<thead>
<tr>
<th>Motor i fault</th>
<th>15%</th>
<th>25%</th>
<th>35%</th>
<th>45%</th>
<th>55%</th>
<th>65%</th>
<th>75%</th>
<th>85%</th>
<th>95%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Motor i speed change</td>
<td>21</td>
<td>36.88</td>
<td>53.8</td>
<td>71.34</td>
<td>88.8</td>
<td>105.34</td>
<td>119.53</td>
<td>130</td>
<td>135.8</td>
</tr>
</tbody>
</table>
or four motor failures occur successively. When four or more severe faults occur, the accuracy decreases slightly without hitting the 10% error value. It is important to state that any disturbance on the octorotor affects the signals resulting in an inaccurate fault estimation. However, this undesirable effect adds robustness to the system since high disturbance that cannot be tolerated by the inherent robustness of the SMC is read as fault and is tolerated by the fault tolerant controller.

6.5. Simulations and Results

6.5.1. Fault Tolerant Control of Octorotor UAVs

The First Order and Second Order Sliding Mode Controller can be used as passive and active fault tolerant controllers. Small fault magnitudes are assumed as disturbance and are compensated using the inherent robustness of the Sliding Mode Controller used as passive fault tolerant controller. On the other hand, high faults cannot be ignored and it is essential to change the controller internal configuration or to redistribute controls among healthy rotors in order for the system to accommodate for faults and keep exhibiting the desired behavior.

Because the octorotor is a redundant system, it is possible to develop an active fault tolerant controller by redistributing the control effort among healthy rotors and minimize the effect of faulty rotors. This is possible by minimizing the gain of the damaged propeller in the gain matrices $W_\omega$ and $W_1$ respectively for Pseudo-Inverse control allocation (equation (5.28)), and Dynamic control allocation (equation (5.38)). In this case, the solution of the control allocation problems (equations (5.28) and (5.36)) will be used as the octorotor control input. We can use the fault information provided by the FDI unit to update the gain matrix so faults will have less effect on the system. The gain matrices in equation (5.34) will be $W_2 = diag([1; 1; 1; 1; 1; 1; 1; 1; 1])$, and $W_1 = diag([E_{r1}; E_{r2}; E_{r3}; E_{r4}; E_{r5}; E_{r6}; E_{r7}; E_{r8}])$, with $E_{ri}$ is the fault magnitude of rotor $i$ estimated by the FDI unit, and $E_{ri} = \frac{100 - \text{Fault percentage of rotor } i}{100}$.

For the Pseudo-Inverse control allocation method, the gain matrix will be $W_\omega = diag([E_{r1}; E_{r2}; E_{r3}; E_{r4}; E_{r5}; E_{r6}; E_{r7}; E_{r8}])$.

The fault tolerant controllers presented in the previous sections are compared based on their performances in controlling the star-shaped octorotor in MATLAB/SIMULINK environment. The octorotor is made to follow the same continuous helical path in the space while faults are injected in different instances. The same PD position controller used with the quadrotor (equation (2.57)) is used here with $k_{ip} = 2$ and $k_{id} = 6$ for $i$ is $x$, $y$, and $z$. 

154
6.5.2. Results

The FOSMC and SOSMC are used in passive and active configurations to control the octorotor subjected to different multiple faults. Faults are partial/total loss of effectiveness of motors where a part of the motor power (or rotor speed) is lost. Note that all tests were conducted on the octorotor with no payload available.

Passive FTC First Order Sliding Mode Controller controlled octorotor with no control allocation can handle the total loss of only two of its motors along with 50% and 60% loss in effectiveness of another two motors. This is a good result but less than expected: the UAV should have more reliability because it is double redundant. The use of Pseudo-Inverse method for fault tolerant control allocation gives very poor results. The UAV could handle only 50% loss of effectiveness of three motors, and 30% loss of a fourth motor. This was expected since the control allocator does not respect the motor constraints and many motors may suffer saturation. This means that the control effort part given to each motor could be greater than its capability, and the expected response could not be achieved. On the other hand, the octorotor Controlled with Dynamic Control Allocation Active FTC can afford the total loss of four of its motors along with 20% loss of effectiveness of two other motors. Note that any two of the four rotors totally failed should not be of paired motors. Figures 5.3 to 5.6 show the path, height, and attitude response of the First Order SMC DCA based Active FTC controlled octorotor, along with its position response, the controls applied, and the rotor speeds when rotors 1, 4, 5, and 7 receive a 100% fault respectively at t = 15, 45, 85, and 60 sec, along with 20% faults in motors 2 and 3 at t = 25, and t = 55 sec respectively.

Simulations show also that with multiple faults occurred, the FDI unit readings start to have a small error. This error is generated because the look-up tables used in the FDI unit to estimate the faults do not change with additional faults. Despite that, the FDI unit is still able to estimate the faults with sufficient accuracy.

The Passive and Pseudo-Inverse control allocation based Second Order SMC have shown better response compared with the First Order SMC results. Passive SOSMC was successful in tolerating the total loss of four motors. Pseudo-Inverse Active SOSMC has shown poor fault tolerant capabilities, and was able to handle 70% loss in three motors, along with 65% loss of effectiveness in a fourth motor. The Active Second Order SMC based on Dynamic Control Allocation took the fault tolerance of the system further by handling the failure of four motors, along with 20% loss of effectiveness in four other motors. Figures 5.7 to 5.10 show the 3D path followed, along with the height and attitude angles response, the position, the controls applied, and the rotor speeds of the Active SOSMC controlled octorotor. The faults injected in this experiment are total loss of rotors 1 (t = 20 sec), 3 (t = 40 sec), 5 (t = 60 sec), and 7 (t = 80 sec), and 20% loss in motors 2 (t = 80 sec), motor 4 (t = 50 sec), motor 6 (t = 70 sec), motor 8 (t = 90 sec).
It is important to state that using the discontinuous control with signum functions as in equation (5.45) produces a smooth response for the height and attitude angles. However, the rotor speeds exhibit some oscillations. When the signum function is replaced by a saturation function, the attitude and height response suffer from small oscillations, but the rotor speeds are smooth. Results shown here are generated using equation (5.45) with the signum replaced by saturation function.

Figure 6.3. – 2-D and 3-D Path of the octorotor controlled with FO DCA FTC (total loss of 4 motors).

Figure 6.4. – X and Y response of the octorotor controlled with FO DCA FTC (total loss of 4 motors).
Figure 6.5. – Height and attitude response of the octotor controlled with FO DCA FTC (total loss of 4 motors).
Figure 6.6. – Controls and Rotor speeds of the octorotor controlled with FO DCA FTC (total loss of 4 motors).
Figure 6.7. – 2-D and 3-D path of the faulty octorotor UAV controlled with Second Order SMC (4 total failures and 1 partial loss of effectiveness faults).

Figure 6.8. – X and y response of the faulty octorotor UAV controlled with Second Order SMC (4 total failures and 1 partial loss of effectiveness faults).
Figure 6.9. – Altitude and attitude response of the faulty octorotor UAV controlled with Second Order SMC (4 total failures and 1 partial loss of effectiveness faults).
Figure 6.10. – Controls and Rotor speeds of the faulty octorotor UAV controlled with Second Order SMC (4 total failures and 1 partial loss of effectiveness faults).
7. Conclusion

7.1. Thesis Outcome

In this thesis, a fault severity based Integrated Fault Tolerant Controller (IFTC) was developed for quadrotor UAVs. The novel controller reserves the quadrotor computational resources by activating the minimum required controller according to the fault situation. A robust controller based on Sliding Mode theory is enough to control the quadrotor in no fault situations. The same controller is used when the quadrotor is subjected to small fault magnitudes, since robust controllers can be used as Passive Fault Tolerant Controllers (PFTC). When the faults affecting the quadrotor reach the capability boundaries of the PFTC, an Active Fault Tolerant Controller (AFTC) is activated. AFTCs use the fault magnitudes estimated by a Fault Detection and Identification (FDI) unit to reconfigure the control laws accordingly. This type of controllers requires more computational effort than PFTCs since fault magnitudes are estimated online, and control laws are adapted on the fly (in real time). The main idea of the IFTC is that it is wise to sacrifice some of the quadrotor reliability (as results of delays resulting from excessive memory use) to prevent the total loss of the UAV. This idea is taken even further when a severe fault or a total failure of one motor/rotor is occurred. The total failure of a rotor is detected by the FDI, and the IFTC activates the emergency controller of the quadrotor. Once activated, the emergency controller stops the infected rotor, applies some weight re-distribution maneuver, and flies the quadrotor as a trirotor. This results in the total loss of heading control, a hard drawback that is only accepted because the emergency controller saves the quadrotor from a certain crash.

Sliding Mode based Passive Fault Tolerant Controllers for quadrotor UAVs were studied in details in this thesis. The fault tolerant capabilities of the regular Sliding Mode Controllers are investigated with the use of optimal search algorithms to tune the controller gains. Using an optimal search algorithm for controller tuning gives optimal gains that ensure the minimum reaching time and the minimum chattering. It was shown that when tuning process uses infected model of the quadrotor, the fault tolerance of SMC based FTC is improved. To further improve the capability of SMC FTC, a Cascaded Sliding Mode FTC is designed and shown to be more effective. The cascaded FTC compensates for the actuator faults in the fast velocity inner loop, preventing them from affecting the position
variables of the quadrotor.

Active Fault Tolerant Controllers (AFTC) based on Sliding Mode Control are also interpreted. The design of the FDI unit responsible for the fault magnitude estimation is realized using Extended Kalman Filters. The FDI unit is shown to give accurate fault estimation despite the occurrence of multiple faults that affect quadrotor rotors and the presence of measurement noise.

The thesis studied the possibility of using quadrotor UAVs as trirotors. This method will take Fault Tolerant Control of quadrotor UAVs to new aspects by gaining them the ability to handle critical faults and total failures of one rotor. Quadrotor to trirotor conversion maneuver was studied in details, and tests were conducted in simulations as well as experimentally. Experiments have shown that this conversion is possible even without the application of the weight redistribution maneuver that shifts the Center Of Gravity (COG) of the UAV to the rear. Applying the quadrotor conversion allows the drone to survive one actuator failure and keep following the desired path but with degraded performance that appears as oscillations. This performance degradation is acceptable, but restricts the use of the quadrotor to trirotor conversion to emergency cases only.

The thesis studied finally the capacity of FTCs applied to over-actuated UAV systems, octorotors in specific. Fault Tolerant Controllers suggested are based on First Order and Second Order Sliding Mode Control (FOSMC-SOSMC). The controllers are made Active by using control allocation methods to distribute the control effort among available actuators based on the fault amplitude affecting each rotor. Pseudo-Inverse Allocation / Dynamic Control Allocation methods, and SOSMC / FOSMC are compared based on simulation results.

The novel stochastic search algorithm based on natural ecological equilibrium developed by the author is also presented in this thesis. The new algorithm, called Ecological Systems Algorithm (ESA), uses primitive decision making individuals to develop an intelligent tool that is able to solve real engineering problems. The algorithm, shown to be effective in static and dynamic search environments, is compared with GA and BFA search algorithms, and was used to tune all the Sliding Mode Controllers used in this thesis. The algorithm not only finds gains that ensure the controller stability, but also finds optimal gains that give high Fault Tolerant capability.

### 7.2. Future Work

Ecological Systems Algorithm can be improved by the introduction of potential functions that establish inter-agent forces between the search individuals [103]. This helps in aggregating the search individuals in flocks, which is believed to fasten the search process. Moreover, and to avoid extinction of the search individuals, elitism concept could be used. Using elitism, the "best" search individuals in an iteration are always kept alive to the next iteration regardless of their health.
life values. This concept, taken from Genetic Algorithms, helps in conserving the best individuals through the iteration history. This is believed to give more precise and faster results.

An Active FTC could be developed based on the Passive FTCs presented in chapter 2. Because the PFTCs tuned with infected motors exhibit more fault tolerance capabilities than regularly tuned controllers, a bank of passive controllers tuned with different faults can be used with an FDI unit that activates the relevant best controller. This type of FTC is active, but it does not require the online computation of reconfigured controls. This results in an Active FTC that has a performance close to the AFTC presented in chapter 2, but which uses less computational effort.

The emergency controller presented in chapter three is a powerful FTC, but it still needs some improvement in order to be applied on real quadrotors. Instead of using multiple weights connected to each motor, one weight could be used connected at the COG of the UAV, and that is able to slide towards all the rotors. This means that more payload is reserved for the UAV applications. The performance of the Integrated FTC should be tested on real quadrotors in the presence of multiple successive faults. The experimental testing of the IntFTC provides valuable information on the performance of the decision unit, the best fault magnitudes to switch between controllers, and the effect of controller switching on the quadrotor stability.

The FDI unit uses look-up tables for each actuator to estimate its fault. This requires long and tricky design and tuning processes for each look-up table. A better approach would be the design of a filter that is able to find the fault estimation directly out of the observed states [104].

The existence of redundant actuators in over-actuated UAV systems provides more FTC capabilities. Active and Passive FTCs based on First and Second Order SMC presented in chapter 5 should be tested on real octorotor systems.

Model uncertainties and disturbances should be considered analytically in future work. This helps in increasing the robustness of controllers, and makes it possible to design robust FDI units. Using a robust FDI is essential to separate the real faults affecting actuators from disturbances and model uncertainties that stimulate false fault alarms.
A. Bibliography
Bibliography


[52] M. Saied, H. Shraim, C. Francis, et al. « Actuator Fault Diagnosis in an Octorotor UAV Using Sliding Modes Technique : Theory and Experimen-

[53] M. Saied. Fault-Tolerant Control of an Octorotor Unmanned Aerial Vehicle under actuators failures, PhD thesis. Department of Informatics Engineer-
ing, University of Technology of Compiègne. University of Technology of Compiègne, Compiègne - France., July 2016 (cit. on p. 29).

[54] O. Harkegard. « Dynamic Control Allocation Using Constrained Quadra-
tic Programming. » In: AIAA Guidance, Navigation, and Control Confe-

[55] Y. Choi and H. Bang. « Dynamic Control Allocation for Shaping Space-


[58] S. Bouabdallah. Design and control of quadrotors with application to au-


[61] M. Seyedkazemi. « Designing Optimal PID controller with Genetic Algo-


172


Appendix
A. Publications associated with this thesis

**Book Chapters**

**Journal papers**

**Conference papers**
B. Detailed calculations

SMC gain which guarantees the stability of PFTC under fault

The dynamics of angle $\phi$ is

$$\ddot{\phi} = \frac{I_y - I_z}{I_x} \dot{\psi} + \frac{I_{rotor}}{I_x} \dot{\gamma} - \frac{K_{f ax}}{I_x} \dot{\phi}^2 + \frac{U_2}{I_x}$$  \tag{1.1}

Assume that $f_\phi = \frac{I_y - I_z}{I_x} \dot{\psi} + \frac{I_{rotor}}{I_x} \dot{\gamma} - \frac{K_{f ax}}{I_x} \dot{\phi}^2$ and $g_\phi = \frac{1}{I_x}$. This assumption allows us to express the dynamics of angle $\phi$ as $\ddot{\phi} = f_\phi + g_\phi U_2$.

The control under fault is $U_2 f = F_2 U_2$, and the nominal passive SMC is

$$U_2 = I_x [c_\phi \dot{\phi} + \ddot{\phi} - \frac{I_y - I_z}{I_x} \dot{\psi}] - k_\phi \text{sat}(s_\phi)$$

The derivative of the Lyapunov function is

$$\dot{V}_\phi = s_\phi (\ddot{\phi} - \ddot{\phi} + c_\phi \dot{\phi})$$

by replacing $\ddot{\phi}$ and $U_2$ in the equation above with their values in the derivative of the Lyapunov function we get

$$\dot{V} = \ddot{\phi} - f_\phi + c_\phi \dot{\phi} - F_2 (\ddot{\phi} - f_\phi + c_\phi \dot{\phi}) + g_\phi F_2 k_\phi \text{sat}(s_\phi)$$

This gives

$$\dot{V} = (1 - F_2)(\ddot{\phi} - f_\phi + c_\phi \dot{\phi}) + g_\phi F_2 k_\phi \text{sat}(s_\phi) \leq \eta_\phi$$

Finally, the gain $k_\phi$ that ensures the stability of the controller is found to be

$$k_\phi \leq \frac{-I_x (1 - F_2)(\ddot{\phi} - \frac{I_y - I_z}{I_x} \dot{\psi} + \frac{I_{rotor}}{I_x} \dot{\gamma} - \frac{K_{f ax}}{I_x} \dot{\phi}^2 + c_\phi \dot{\phi}) - \eta_\phi}{F_2 |s_\phi|}$$
Trirotor Moment of Inertia

\[ I_{m1x} = I_{mz} + m_m(y_{cg}^2 + z_{cg}^2) \]
\[ I_{m1y} = I_{my} + m_m((l - x_{cg})^2 + z_{cg}^2) \]
\[ I_{m1z} = I_{mz} + m_m((l - x_{cg})^2 + y_{cg}^2) \]
\[ I_{m1xy} = m_m(l - x_{cg})y_{cg} \]
\[ I_{m1xz} = m_m(l - x_{cg})z_{cg} \]
\[ I_{m1yz} = m_my_{cg}z_{cg} \]

(2)

\[ I_{m2x} = I_{mx} + m_m((l - y_{cg})^2 + z_{cg}^2) \]
\[ I_{m2y} = I_{my} + m_m(x_{cg}^2 + z_{cg}^2) \]
\[ I_{m2z} = I_{mz} + m_m(x_{cg}^2 + (l - y_{cg})^2) \]
\[ I_{m2xy} = m_mx_{cg}(l - y_{cg}) \]
\[ I_{m2xz} = m_mx_{cg}z_{cg} \]
\[ I_{m2yz} = m_m(l - y_{cg})z_{cg} \]

(3)

\[ I_{m3x} = I_{mx} + m_m(y_{cg}^2 + z_{cg}^2) \]
\[ I_{m3y} = I_{my} + m_m((l + x_{cg})^2 + z_{cg}^2) \]
\[ I_{m3z} = I_{mz} + m_m((l + x_{cg})^2 + y_{cg}^2) \]
\[ I_{m3xy} = m_m(l + x_{cg})y_{cg} \]
\[ I_{m3xz} = m_m(l + x_{cg})z_{cg} \]
\[ I_{m3yz} = m_my_{cg}z_{cg} \]

(4)
\[ I_{m4x} = I_{m2} + m_m((l + y_{cg})^2 + z_{cg}^2) \]
\[ I_{m4y} = I_{m3} + m_m(x_{cg}^2 + z_{cg}^2) \]
\[ I_{m4z} = I_{m2} + m_m(x_{cg}^2 + (l + y_{cg})^2) \]
\[ I_{m4xy} = m_m x_{cg}(l + y_{cg}) \]
\[ I_{m4xz} = m_m x_{cg} z_{cg} \]
\[ I_{m4yz} = m_m(l + y_{cg}) z_{cg} \]  

\[ (5) \]

\[ I_{c'x} = I_{cx} + m_c(y_{cg}^2 + z_{cg}^2) \]
\[ I_{c'y} = I_{cy} + m_c(x_{cg}^2 + z_{cg}^2) \]
\[ I_{c'z} = I_{cz} + m_c(x_{cg}^2 + y_{cg}^2) \]
\[ I_{c'xy} = m_c x_{cg} y_{cg} \]
\[ I_{c'xz} = m_c x_{cg} z_{cg} \]
\[ I_{c'yz} = m_c y_{cg} z_{cg} \]  

\[ (6) \]

\[ I_{b'x} = I_{b2} + m_b((y_{cg} - \beta)^2 + (z_{cg} - \gamma)^2) \]
\[ I_{b'y} = I_{b3} + m_b((x_{cg} - \alpha)^2 + (z_{cg} - \gamma)^2) \]
\[ I_{b'z} = I_{b2} + m_b((x_{cg} - \alpha)^2 + (y_{cg} - \beta)^2) \]
\[ I_{b'xy} = m_b(x_{cg} - \alpha)(y_{cg} - \beta) \]
\[ I_{b'xz} = m_b(x_{cg} - \alpha)(z_{cg} - \gamma) \]
\[ I_{b'yz} = m_b(y_{cg} - \beta)(z_{cg} - \gamma) \]  

\[ (7) \]
## C. Quadrotor Data

### Pelican Quadrotor Data

<table>
<thead>
<tr>
<th>Component</th>
<th>Quantity</th>
<th>Mass (g)</th>
<th>Volume (mm²)</th>
<th>Density</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>motor</td>
<td>4</td>
<td>57</td>
<td>14568</td>
<td>0.003913</td>
<td>1</td>
</tr>
<tr>
<td>battery</td>
<td>1</td>
<td>372</td>
<td>173166</td>
<td>0.002148</td>
<td>2</td>
</tr>
<tr>
<td>propeller</td>
<td>4</td>
<td>6</td>
<td>2421.68</td>
<td>0.002478</td>
<td>3</td>
</tr>
<tr>
<td>GPS</td>
<td>1</td>
<td>18</td>
<td>24462.5</td>
<td>0.000736</td>
<td>4</td>
</tr>
<tr>
<td>antenna</td>
<td>1</td>
<td>12</td>
<td>4935.66</td>
<td>0.002431</td>
<td>5</td>
</tr>
<tr>
<td>antenna box</td>
<td>1</td>
<td>8</td>
<td>5871.6</td>
<td>0.001362</td>
<td>6</td>
</tr>
<tr>
<td>top box</td>
<td>1</td>
<td>13</td>
<td>9000</td>
<td>0.001444</td>
<td>7</td>
</tr>
<tr>
<td>sensor</td>
<td>1</td>
<td>19</td>
<td>6691</td>
<td>0.002840</td>
<td>8</td>
</tr>
<tr>
<td>integrated circuit 1</td>
<td>1</td>
<td>100</td>
<td>108000</td>
<td>0.000926</td>
<td>9</td>
</tr>
<tr>
<td>integrated circuit 2</td>
<td>1</td>
<td>30</td>
<td>21780</td>
<td>0.001377</td>
<td>10</td>
</tr>
<tr>
<td>battery support</td>
<td>1</td>
<td>20</td>
<td>59027</td>
<td>0.000339</td>
<td>11</td>
</tr>
<tr>
<td>base frame</td>
<td>1</td>
<td>17.41</td>
<td>52480</td>
<td>0.000332</td>
<td>12</td>
</tr>
<tr>
<td>integrated circuit 3</td>
<td>1</td>
<td>20</td>
<td>7056</td>
<td>0.002834</td>
<td>13</td>
</tr>
<tr>
<td>frame</td>
<td>1</td>
<td>612.6</td>
<td>245250.51</td>
<td>0.002498</td>
<td>14</td>
</tr>
</tbody>
</table>

| **Total weight**     | **1305.01** |         |              |          |           |

Figure 1. Pelican quadrotor data.
Figure 2. - Pelican quadrotor model using Solidworks.