

https://dspace.org/

Intelligent fault-tolerant control using integrated flight-propulsion strategy for medium-scale rotorcraft

Mpanza, Lindokuhle Justice

2024

https://hdl.handle.net/10539/37386

Downloaded from a DSpace repository. https://dspace.org/

Intelligent Fault-Tolerant Control using Integrated Flight-Propulsion Strategy for Medium-Scale Rotorcraft

Lindokuhle Justice Mpanza

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

Johannesburg, May 2023

Declaration

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

Signed this _____ day of _____ 20 ____

_____

Lindokuhle Justice Mpanza (0302966N)

Copyright © 2023

Abstract

This thesis presents the development of a medium-scale, single-rotor rotorcraft unmanned aerial vehicles (RUAVs) nonlinear dynamic model to be used for model-based design of fault-tolerant control. This model includes the swashplate actuator dynamics and the engine propulsion model. Most rotorcraft controller design treat these two system components as linear first-order systems. The inclusion of actuator dynamics allows us to simulate the actuator faults with increased fidelity, which enables the development of robust integrated flight-propulsion controllers (IFPCs) that are seamlessly transferable to the true platform. IFPC is proposed for the RUAV in order to handle the deliberate variations in rotor speed and for its exploitation as a redundant thrust control input in the case of a single swashplate actuator fault. The presented RUAV model is first controlled by using proportional-integral-derivative (PID) controllers in six degrees-of-freedom (DOF). To obtain the best controller gains, computational intelligent optimisation techniques are used, these are: (i) ant-based: ant colony and antlion optimisation algorithms; (ii) flight-based: cuckoo search and firefly optimisation algorithms; and these are compared to (iii) particle swarm optimisation and genetic algorithm. The performance of the PID controllers are used for benchmarking the rest of the control strategies investigated. For fault tolerance, we present the passive fault-tolerant control (PFTC) based on sliding mode controller (SMC). Conventional SMC and super-twisting SMC are compared in terms of robustness to handle loss-of-effectiveness (LOE) faults in one of the three swashplate actuators and the deviation of the rotor speed from nominal. These controllers are also optimised using the same computational intelligent algorithms as in the PID case. The SMC-based controllers proved robust to actuator LOE faults. However, they fail to recover the rotorcraft from total actuator failure. For this problem, an active fault-tolerant control (AFTC) scheme based on dynamic neural networks (DNN) is employed. RUAV system identification using DNN is combined with feedback linearisation (FBL). Multi-objective optimisation algorithms are used to find the controller and FBL gains. DNN is also used for fault detection and diagnosis (FDD) for the AFTC. This applied indirect DNNFBL control strategy proved to be more suitable for IFPC and was able to recover from severe actuator faults and rejected rotor speed variations. The effectiveness of the proposed control strategies is evaluated in hardware-in-the-loop simulations (HILS) using an experimental swashplate rig of three electromechanical actuators. The experimental results validated the developed and simulated control strategies.

Scope and Contribution

The contributions resulting from this research are as follows:

1. The design of a medium-scale RUAV model simulation that enables the demonstration of the application of variable rotor speed;

2. The results of the study of integrating engine control into the flight control for variable rotor speed as compared to a distributed control strategy;

3. A comprehensive study and the development of computational intelligence integrated flight-propulsion control. These are compared to traditional classical control methods. In addition, the investigation and comparison of evolutionary algorithms such as Genetic Algorithm, Particle Swarm Optimisation, Ant Colony Optimisation, AntLion Optimisation, Cuckoo Search and Firefly Algorithm for optimal tuning of control parameters are presented;

4. A study of robust PFTC based on SMC and AFTC based on DNN to recover from actuator faults and complete failure using control allocation of redundant degree of freedom offered by variable rotor speed. New knowledge contribution is in the RUAV system evaluated for its resilience to system parameter changes due to variable rotor speed and actuator LOE faults; and

5. The development of an HILS experimental rig for validation and rapid prototyping of the control strategies algorithms developed.

Published Work

From this research thesis the following papers were published:

Journal Papers

1. L. J. Mpanza and J.O. Pedro, " Optimised Tuning of a PID-Based Flight Controller for a Medium-Scale Rotorcraft," Journal of Algorithms, Multidisciplinary Digital Publishing Institute , 2021, Volume 14, Issue 6, pages 178-200.

2. L. J. Mpanza and J.O. Pedro, "Optimisation of Fault-Tolerant Super-Twisting Sliding Mode Controller for a Medium-Scale Rotorcraft UAV," International Journal of Robust and Nonlinear Control. (**Under review**).

3. L. J. Mpanza and J.O. Pedro, "Dynamic Neural Network-based Feedback Linearisation Controller of a Medium-Scale Rotorcraft UAV using Multi-objective Optimisation," International Journal of Systems and Control Engineering. (**Revised and resubmit (provisionally accepted subject to revision)**).

4. L. J. Mpanza and J.O. Pedro, "Comparing Passive and Active Fault-Tolerant Control of Swashplate Actuator Faults in a Medium-Scale Rotorcraft UAV with Experimental Validation," Control Engineering Practice: Engineering Applications of Artificial Intelligence. (**Revised and resubmitted**).

Conference Papers

1. L. J. Mpanza, J.O. Pedro and J. Roberts, "Control-Allocated Sliding Mode Control for a Single-Axis Tilting Quadrotor UAV," In Proceedings of the 2021 Control Conference Africa (CCA) Muldersdrift, South Africa, 6 - 8 Dec. 2021, pages 121-126.

2. L. J. Mpanza and J.O. Pedro, " Sliding Mode Control of an Electromechanical Actuator Swashplate Using Cuckoo Search Optimisation Algorithm," In Proceedings of the 2020 3rd International Conference on Control and Robots (ICCR), Tokyo, Japan, 26 - 28 Dec. 2020, pages 204-210.

3. L. J. Mpanza and J.O. Pedro, "Nature-Inspired Optimization Algorithms for Sliding Mode Control Parameters Tuning for Autonomous Quadrotor," In Proceedings of the 2019 IEEE Conference on Control Technology and Applications (CCTA), Hong Kong, China, 19 - 21 Aug. 2019, pages 1087-1092

4. L. J. Mpanza and J.O. Pedro, "Sliding mode control parameter tuning using ant colony optimization for a 2-DOF hydraulic servo system," In Proceedings of the 2016 IEEE International Conference on Automatic Control and Intelligent Systems (I2CACIS), Selangor, Malaysia, 22 - 23 Oct. 2016, pages 242-247

5. L. J. Mpanza and J.O. Pedro, "Sliding mode control of a 2-DOF hydraulic servo system," In Proceedings of the 2015 16th International Carpathian Control Conference (ICCC), Szilvasvarad, Hungary 27 - 30 May. 2015, pages 321-327

Acknowledgements

Firstly, I wish to express my gratitude to my supervisor Prof. Jimoh O. Pedro for his guidance, continuous support, advice and enthusiasm throughout the course of this work. I also wish to thank him for the tireless efforts invested into making this journey as smooth as only he can make it possible. His wise sayings and cryptic parables kept me going even through the hardest of times.

Thanks to Denel Aeronautics for sponsoring parts of this study, not only for financial support, but also affording me the time, facilities and equipment to carry out the research. The information and conclusions presented in this thesis are the sole responsibility of the author and do not reflect the opinions of the company. The support of management and my colleagues, especially in the Engineering and Continued Airworthiness, is hereby acknowledged.

Thanks also go to my mechatronics students who taught me more than I did them.

I would like to thank my family, on both sides, for all the support they have given me throughout the course of this study.

I express my deepest gratitude to my wife, Selokela, for all the love, encouragement and support throughout the course of this journey. High appreciation for being the pillar of strength I needed to undertake this work to completion.

Lindokuhle Justice Mpanza

Johannesburg, May 2023

To my late Parents and Parents-in-law.

Contents

Declaration **i**

Abstract **ii**

Acknowledgements **vi**

List of Figures **xii**

List of Tables **xx**

Nomenclature **xxiii**

1 Introduction **1**

1.1 Research Background 1

1.2 Research Motivation 3

1.3 Literature Review 5

1.4 Identified Gaps in the Existing Literature 19

1.5 Research Hypothesis/Research Question 20

1.6 Research Objectives 20

1.7 Research Methodology 21

1.8 Research Contributions 22

1.9 Thesis Outline 23

2 Mathematical Modelling of the RUAV and Actuator Dynamics 26

2.1 Research Introduction 26

2.2 Description of the Rotorcraft Platform 29

2.3 Modelling Assumptions 30

2.4 Coordinate Reference Frames 30

2.5 Rotorcraft Model 31

2.6 Rotorcraft Simulation Model 43

2.7 Actuator Dynamics Models 45

2.8 Rotorcraft Modelling Parameters 53

2.9 Flight Control Performance Specifications and Scope 54

2.10 Chapter Summary 55

3 Flight Control Design using PID Controller and Computational Intelligence Optimisation Techniques 56

3.1 Introduction 57

3.2 Rotorcraft Trim Analysis 58

3.3 Proportional-Integral-Derivative (PID) Controller 64

3.4 Introduction to Computational Intelligence Optimisation Techniques . . . 67

3.5 Evaluation and Validation of the Optimisation Techniques 84

3.6 Simulation Experiment Setup 88

3.7 Chapter Summary 104

4 Passive Fault-Tolerant Control of Rotorcraft using Sliding Mode Controller 105

4.1 Passive Fault-Tolerant Control and Fault Modelling 105

4.2 Modelling of System's Parameter Variations, Actuators and Faults 106

4.3 Problem Formulation 110

4.4 Sliding Mode Control (SMC) 114

4.5 Simulation Results and Discussion 123

4.6 Chapter Summary 141

5 Active Fault-Tolerant Control of Rotorcraft for Integrated Flight-Propulsion Control by Application of Dynamic Neural Network-based Feedback Linearisation 143

5.1 Active Fault-Tolerant Control (AFTC) 143

5.2 DNN-based Feedback Linearisation 147

5.3 Methodology 150

5.4 Controller Implementation 156

5.5 Simulation Results and Discussion 157

5.6 DNN, Fault Diagnosis and Isolation 166

5.7 Chapter Summary 178

6 Hardware-in-the-Loop Simulation 180

6.1 Introduction 180

6.2 Experiment Setup Description 181

6.3 Methodology 183

6.4 Sensors and Data Collection Limitations 187

6.5 Actuator Control Design 187

6.6 Comparing the Rotorcraft Response with Actuator Dynamics 191

6.7 Robust SMC Response to Real Actuators 197

6.8 Active DNN Response to Real Actuators 199

6.9 Comparing the Responses of the Proposed Controllers 202

6.10 HILS Fault Tolerance 202

6.11 Chapter Summary 206

7 Conclusion and Recommendations for Future Work **207**

7.1 Conclusions 207

7.2 Recommendations for Future Work 210

References **211**

List of Figures

1.1 Comparing a fixed-wing (a)(Denel-Dynamics, 2018) to a single rotor rotary-wing (b)(MilitaryFactory, 2008) UAVs. 2

1.2 Flight profile of a rotorcraft with a fixed rotor speed. 7

1.3 Flight profile of a rotorcraft at three different rotor speeds. 8

1.4 Classification of FTC strategies (Edwards et al., 2010). 10

1.5 Active fault-tolerant control philosophy. Adapted from (Edwards et al., 2010). 12

1.6 Controller reconfiguration by control allocation. 14

1.7 Structure of the thesis. 25

2.1 The rotorcraft as a combination of subsystems working together to accomplish the flight task. 34

2.2 The rotorcraft configuration diagram with axis labels (Mpanza and Pedro, 2021). 35

2.3 Simplified actuator model. 46

2.4 Top view of the actuator model with actuator A pointing directly forward. 47

2.5 Algorithmic swashplate mixing. 47

2.6 Swashplate geometry with A, B and C are as the positions of the actuators. 48

2.7 The standard components of an ACDC motor-based linear electromechanical actuator. 50

3.1 The rotorcraft position responses to selected hover trim condition. 61

3.2 The rotorcraft Euler angles' responses to selected hover trim condition. . 61

3.3 The rotorcraft position response to selected forward flight trim condition. 63

3.4 The rotorcraft Euler angles' responses to selected forward flight trim condition. 63

3.5 The control architecture of the rotorcraft using PID controller closed-loop system. 66

3.6 GA computation flowchart (Aly, 2011). 70

3.7 The Michalewicz function used for evaluating optimisation algorithms. . . 86

3.8 The convergence history of the fitness values of each of the optimisation algorithms over 100 iterations (generations). 89

3.9 The nonlinear rotorcraft position response for the tuned PID controllers. . 91

3.10 The nonlinear rotorcraft Euler angles' response for the tuned PID controllers. 91

3.11 The rotorcraft positions response for the forward flight at 1 m/s. 92

3.12 The rotorcraft velocity history for the forward flight at 1 m/s. 93

3.13 The convergence history of the fitness values of each of the four optimisation algorithms over 100 iterations. 94

3.14 The rotorcraft positions response for the forward flight at 10 m/s. 96

3.15 The rotorcraft velocity history for the forward flight at 10 m/s. 96

3.16 The rotorcraft position response when operating at 10% increase in rotor speed. 97

3.17 The rotorcraft control input when operating at 10% increase in rotor speed. 98

3.18 The rotorcraft position response for the robustness testing with 50%, 100% and 200% of the nominal mass. 99

3.19 The rotorcraft position response for the robustness testing with 50%, 100% and 200% of the nominal inertia tensor. 100

3.20 The rotorcraft position response to wind gust disturbance in hover. 101

3.21 The rotorcraft speed response to wind gust disturbance in hover. 102

3.22 The rotorcraft position response to wind gust disturbance at 10 m/s forward flight. 102

3.23 The rotorcraft position response when there is an actuator fault of 30% and 50% LOE. 103

3.24 The control inputs into the rotorcraft when there is an actuator fault of 30% and 50% LOE. 104

4.1 The system divided into states that are directly and indirectly influenced by the control input. 111

4.2 The control system architecture of the rotocraft SMC. 116

4.3 The scheme used to find the STSMC switching function. 120

4.4 SMC convergence history of the fitness value for ACO and PSO over 100 iterations. 124

4.5 STSMC convergence history of the fitness value for ACO and PSO over 100 iterations. 125

4.6 Comparing PSO and ACO SMC displacement tracking in hover. 126

4.7 Comparing PSO and ACO SMC Euler angles' tracking in hover. 127

4.8 Comparing PSO and ACO SMC control inputs for the rotorcraft in hover. 127

4.9 Comparing PSO and ACO STSMC displacement tracking in hover. 128

4.10 Comparing PSO and ACO STSMC Euler angles' tracking in hover. 128

4.11 Comparing PSO and ACO STSMC control inputs for the rotorcraft in hover. 129

4.12 Comparing PSO and ACO SMC displacement tracking for the rotorcraft in forward flight. 130

4.13 Comparing PSO and ACO SMC velocities for the rotorcraft in forward flight. 131

4.14 Comparing PSO and ACO SMC control inputs for the rotorcraft in forward flight. 131

4.15 Comparing PSO and ACO STSMC displacement tracking for the rotorcraft in forward flight. 132

4.16 Comparing PSO and ACO STSMC velocities for the rotorcraft in forward flight. 132

4.17 Comparing PSO and ACO STSMC control inputs for the rotorcraft in forward flight. 133

4.18 SMC phase-plane portraits of the hover (a) and forward flight (b). 133

4.19 STSMC phase-plane portraits of the hover (a) and forward flight (b). . . . 134

4.20 Comparing robustness to mass variations for position in hover. 135

4.21 Comparing robustness to mass variations control inputs in hover. 135

4.22 Comparing robustness to inertia tensor variations for position in hover. . 136

4.23 Comparing robustness to inertia tensor variations control inputs in hover. 136

4.24 Comparing robustness to gust disturbance position in hover. 137

4.25 Comparing robustness to gust disturbance control inputs in hover. 137

4.26 Comparing robustness to gust disturbance position in forward flight. . . . 138

4.27 Comparing robustness to gust disturbance control inputs in forward flight. 138

4.28 Comparing robustness of the RUAV position for different levels of loss-of-effectiveness. 139

4.29 Comparing robustness of the RUAV control inputs for different levels of loss-of-effectiveness. 140

5.1 Schematic of the DNN-based system identification process for the rotorcraft system. 151

5.2 Reference signal used to excite the closed-loop rotorcraft. 152

5.3 A sample of the system input signals applied to the rotorcraft for DNN system identification. 153

5.4 The DNN-based input-output feedback linearisation (DNNFBL) architecture with off-line SOO algorithm for DNN identification and MOO algorithm for PID gains tuning. 158

5.5 The system identification system fitness changes over the number of iterations. 159

5.6 DNN predictions vs validation data collected. 160

5.7 PID controller system Pareto front over the number of iterations. 161

5.8 Comparing MOALO and MOACO PID-DNNFBL displacement tracking. 161

5.9 Comparing ALO and ACO PID control inputs. 162

5.10 PID-DNNFBL system Pareto front over the number of iterations. 163

5.11 Comparing ALO and ACO PID-DNNFBL displacement tracking. 164

5.12 Comparing ALO and ACO PID-DNNFBL velocities. 164

5.13 Comparing ALO and ACO PID-DNNFBL Euler angles. 165

5.14 Comparing ALO and ACO PID-DNNFBL control inputs. 165

5.15 The DNN-based AFTC with the FDI and the control reconfiguration to use the engine control as a redundant control variable. 167

5.16 The determination of the number of states for the training of the DNN. . 169

5.17 The training and validation of the DNN for actuator identification. 169

5.18 Residuals generated for a 50% LOE on actuator A. 172

5.19 Rotorcraft position response under the DNNFBL controller for a 50% LOE on actuator A. 173

5.20 Rotorcraft control inputs under the DNNFBL controller for a 50% LOE on actuator A. 173

5.21 Residual generated for a 50% LOE on actuator A for the rotorcraft in hover.174

5.22 Rotorcraft position response under the DNNFBL controller for a 50% LOE on actuator A in forward flight. 175

5.23 Rotorcraft control inputs under the DNNFBL controller for a 50% LOE on actuator A in forward flight. 175

5.24 Residual generated for a 50% LOE on actuator A for the rotorcraft in steady forward flight. 176

5.25 Rotorcraft position response under the DNNFBL controller for a 50% LOE on actuator A. 176

5.26 Rotorcraft position response under the DNNFBL controller for a 50% LOE on actuator A and the presence of $5m/s$ gust wind. 177

5.27 Rotorcraft control inputs under the DNNFBL controller for a 50% LOE on actuator A and the presence of $5m/s$ gust wind. 177

5.28 Residual generated for a 50% LOE on actuator A for the rotorcraft in the presence of $5m/s$ gust wind. 178

6.1 The experimental setup. 182

6.2 Actuator internal. 183

6.3 Flowchart of the methodology followed in conducting the HILS experiment. 184

6.4 Actuator full-scale deflection test. 185

6.5 Actuator model validation with full-scale deflection data. 186

6.6 The digital twin of the hardware rig created in V-Realm® and integrated in Simulink®. 186

6.7 Fitness values of changes over time using CS-based PID for the swashplate actuators. 188

6.8 Fitness values changes over time using CS-based SMC for the swashplate actuators. 188

6.9 CS-based PID and SMC controlled swashplate tracking response for the desired reference signals. 189

6.10 CS-based SMC swashplate errors command for the desired reference signals. 190

6.11 Robustness testing with parameter variations. 190

6.12 Ideal compared to real actuator response. 191

6.13 The response of the real actuators when affecting flight dynamics. 192

6.14 Rotorcraft response with real actuators under PID control. 192

6.15 Rotorcraft position response with real actuator models under the retuned PID controller. 193

6.16 Rotorcraft position response with HILS actuators under the retuned PID controller. 194

6.17 Rotorcraft Euler angles response with HILS actuators under the retuned PID controller. 195

6.18 Rotorcraft control inputs with HILS actuators under the retuned PID controller. 195

6.19 Rotorcraft position response with HILS actuators under PID controller transitioning from hover to forward flight. 196

6.20 Rotorcraft position response with real actuator models under the STSMC. 197

6.21 Rotorcraft Euler angles response with real actuator models under the STSMC. 198

6.22 Rotorcraft control inputs with HILS actuators under the STSMC. 198

6.23 Rotorcraft position response with real actuator models under the DNNFBL. 199

6.24 Rotorcraft Euler angles response with real actuator models under the DNNFBL. 200

6.25 Rotorcraft control inputs with HILS actuators under the DNNFBL. . . . 200

6.26 Rotorcraft position response with real actuator models under the DNNFBL in forward flight. 201

6.27 Rotorcraft control inputs with HILS actuators under the DNNFBL in forward flight. 201

6.28 Residual and rotor speed with HILS actuators under the DNNFBL for a LOE of 50%. 203

6.29 Rotorcraft position response with real actuator models under the DNNFBL for a LOE of 50%. 203

6.30 Rotorcraft Euler angles response with HILS actuators under the DNNFBL for tracking with LOE of 50%. 204

6.31 Rotorcraft control inputs with HILS actuators under the DNNFBL for tracking with LOE of 50%. 204

6.32 Rotorcraft position response with real actuator models under the DNNFBL for tracking with LOE of 50%. 205

6.33 Rotorcraft control inputs with HILS actuators under the DNNFBL for tracking with LOE of 50%. 205

List of Tables

2.1 The system parameters of the complete EMA swashplate system. 52

2.2 The parameters used in the rotorcraft modelling. 53

3.1 Parameters for rotorcraft maintenance of hover trim condition. 60

3.2 Parameters for rotorcraft maintenance of forward flight of 10 m/s trim condition. 62

3.3 The setup parameters for the GA algorithm. 71

3.4 The setup parameters for the PSO algorithm. 74

3.5 The ACO tuning parameters and their effect on the algorithm. 76

3.6 The setup parameters for the continuous ACO algorithm. 76

3.7 The setup parameters for the ALO algorithm. 79

3.8 The setup parameters for the CS algorithm. 81

3.9 The setup parameters for the FA. 84

3.10 Optimisation benchmark functions. 85

3.11 The benchmark comparison of the six computational intelligence optimisation algorithms proposed for use in the tuning of controller gains. 87

3.12 The fitness, the running times, mean fitness and standard deviation of the PID controller gains optimisation process for hover condition. 89

3.13 The PID controllers gains for hover conditions found using the optimisation algorithms. 90

3.14 The fitness, the running times, mean fitness and standard deviation of the PID controller gains optimisation process for forward flight condition. . . 93

3.15 The PID controllers gains for forward flight condition found using the using optimisation. 94

4.1 Tuned SMC parameters using the proposed optimisation techniques. . . . 124

4.2 Tuned super-twisting SMC parameters using the proposed optimisation techniques. 125

4.3 Statistical test for comparing PSO and ACO used to tune the SMC. . . . 140

4.4 Statistical test for comparing PSO and ACO used to tune the STSMC. . 141

4.5 Statistical test computation for comparing conventional SMC and STSMC. 141

5.1 The setup parameters for the ACO algorithm in tuning DNN models. . . 159

5.2 The setup parameters for the ALO algorithm in tuning DNN models. . . 159

5.3 The PID controller gains found using multi-objective optimisation algorithms.160

5.4 The PID controller gains and the DNNFBL parameters found using multi-objective optimisation algorithms 162

5.5 The Pareto fronts for MOO-based PID controller gains and the FBL parameters found using optimisation. n-d means number of non-dominated solutions. 163

5.6 The use of residuals to identify faults in actuators a, b and c. μ_a, μ_b and μ_c are the fault thresholds. 170

5.7 The response determined in order to react to a fault in a single swashplate actuator. 171

6.1 The summary of the flight control methods compared in this chapter. . . 181

6.2 CS setup parameters for PID and SMC tuning for the swashplate actuators. 188

6.3 The new PID controller gains found after retuning using ACO algorithm. 193

6.4 Comparing the HIL simulation results for the three controllers. 202

Nomenclature

Acronyms

ACDC	:	Armature-Controlled Direct Current
ACO	:	Ant Colony Optimization
AFCS	:	Automatic Flight Control System
AFTC	:	Active Fault-Tolerant Control
AHRS	:	Altitude Heading Reference System
ALO	:	AntLion Optimisation
ANN	:	Artificial Neural Network
BET	:	Blade Element Theory
CG	:	Centre of Gravity
DC	:	Direct Current
DGPS	:	Differential Global Positioning System
DNN	:	Dynamic Neural Network
DOF	:	Degree-of-Freedom
FA	:	Firefly Algorithm
FBL	:	Feedback Linearisation
FDD	:	Fault Detection and Diagnosis
FDI	:	Fault Detection and Isolation
FLC	:	Fuzzy Logic Control
FTC	:	Fault-Tolerant Control
GA	:	Genetic Algorithm
GNC	:	Guidance, Navigation and Control
GPS	:	Global Positioning System
HILS	:	Hardware-in-the-Loop Simulation
HOSMC	:	High-Order Sliding Mode Control
IFPC	:	Integrated Flight-Propulsion Control
ISR	:	Intelligence, Surveillance and Reconnaissance
LMI	:	Linear Matrix Inequality
LOC	:	Loss-of-Control
LOE	:	Loss-of-Effectiveness
LPV	:	Linear Parameter Varying
LQG	:	Linear Quadratic Gaussian
LQR	:	Linear Quadratic Regulator
LTI	:	Linear Time Invariant
MIMO	:	Multiple-Input Multiple-Output

MLP	:	Multi-Layer Perceptron
MOO	:	Multi-Objective Optimisation
MPC	:	Model Predictive Control
MT	:	Momentum Theory
MTOM	:	Maximum Take-Off Mass
NN	:	Neural Network
PEM	:	Predicted Error Minimisation
PFTC	:	Passive Fault-Tolerant Control
PID	:	Proportional-Integral-Derivative
PSO	:	Particle Swarm Optimisation
RBF	:	Radial Basis Function
RMS	:	Root Mean Square
RTOS	:	Real-Time Operating System
RUAV	:	Rotorcraft Unmanned Aerial Vehicle
SDRE	:	State-Dependent Riccati Equation
SISO	:	Single-Input Single-Output
TPP	:	Tip-Path-Plane
SMC	:	Sliding Mode Control
STSMC	:	Super-Twisting Sliding Mode Control
UAS	:	Unmanned Aerial System
UAV	:	Unmanned Aerial Vehicle
VSC	:	Variable Structure Control
VTOL	:	Vertical Take-Off and Landing
VRS	:	Variable Rotor Speed

Variables

a_M, a_T	:	Rotor lift-curve slope
a_{lc}, b_{ls}	:	Longitudinal and lateral flapping angles
a, b, c	:	Swashplate actuator index position
A	:	Area of the rotor disc
b_M, b_T	:	Rotor number of blades
c_M, c_T	:	Rotor blade chord
C_D	:	Drag coefficient
C_{D_0}	:	Zero-lift drag coefficient
g	:	Acceleration due to gravity
k_t	:	The main to tail rotor reduction ratio
L, M, N	:	Aerodynamic moments about the x, y, z axes respectively
p, q, r	:	Rotorcraft roll, pitch and yaw rates respectively
R	:	Main rotor radius
$\mathbf{R}$	:	Direction cosine transformation matrix
S	:	Surface area
$\mathbf{T}$	:	Differential transformation matrix
u, v, w	:	Body-fixed axes velocities
v	:	Induced velocity
V	:	Freestream velocity
W	:	Rotorcraft weight

x, y, z	:	Body translation position in the inertial frame
X, Y, Z	:	Aerodynamic forces along the x, y, z axes respectively

Greek

α	:	Rotorcraft angle of attack
α_b	:	Angle of attack of the blades
α_f	:	Angle of attack of the fuselage
α_r	:	Angle of attack of the tip-path-plane
β	:	Rotorcraft sideslip angle
β_f	:	Flapping angle
θ, ϕ, ψ	:	Rotorcraft pitch, roll and yaw angles respectively
θ_b	:	Blade pitch angle
$\theta_0, \theta_{1c}, \theta_{1s}$	:	Collective, lateral and longitudinal cyclic pitch angles
λ	:	Inflow ratio
μ	:	Advance ratio, $V/\Omega R$
μ_i	:	The i-th residual threshold
ρ	:	Air density
σ	:	Blade solidity factor
σ	:	Sliding surface
Ψ	:	Azimuth angle of blade position
Ω	:	Blade rotation speed

Suffixes

b	:	of the blade
c, C	:	in climb
D	:	drag
e	:	of engine
h	:	hover value
H	:	horizontal stabiliser
i	:	induced
L	:	lift
lat	:	along the lateral axis
lon	:	along the longitudinal axis
M	:	of main rotor
max	:	maximum
o	:	constant value
P	:	power
Q	:	torque
t	:	blade tip
tw	:	blade twist
T	:	of tail rotor
T	:	thrust
V	:	vertical stabiliser
$wind$	:	component of wind velocity

Chapter 1

Introduction

This chapter presents the background and motivation for the research undertaken. A brief literature review is given to identify the knowledge gaps. The methodology followed for conducting the research is given and research contributions to the knowledge are enumerated. The chapter concludes by describing a roadmap for the remainder of the thesis.

1.1 Research Background

The past two decades have seen an unprecedented increase in application of aerial robots. This is largely due to advancements in technology such as: high processing speeds, electronic miniaturisation and lighter materials, that have led to affordable, simpler, smaller and more effective unmanned aerial vehicles (UAVs) (Green and Gómez, 2020). Consequently, due to their small size and the relatively low cost of development and operation, UAVs are increasingly becoming attractive for civilian applications, such as national parks surveillance, crime fighting, disaster observation, farming, parcel delivery, and filming.

A UAV is defined as a powered aircraft that is flown without an on-board pilot (Valavanis and Vachtsevanos, 2015). In essence, it is considered a flying robot. This robot can either operate autonomously or be a remotely piloted component of a larger integrated unmanned aerial system (UAS). The most popular UAVs in deployment are fixed-wing type aircraft (as shown in Figure 1.1 (a)). While fixed-wing aircraft are ideal for long range flights and large payloads; their disadvantage is that they require a runway or special launch and recovery equipments for take-off and landing. They also need to



(a) (b)

Figure 1.1: Comparing a fixed-wing (a)(Denel-Dynamics, 2018) to a single rotor rotary-wing (b)(MilitaryFactory, 2008) UAVs.

maintain high velocity to produce lift. This limits fixed-wing UAVs to payload drop-off and fly-over missions. There has been a disproportionately larger growth in interest towards rotorcraft unmanned aerial vehicles (RUAVs) in recent years than in fixed-wings. Unlike fixed-wing aircraft, RUAVs rely on the rotor speed and collective blade pitch to generate both lift and thrust. This means that rotorcraft have the capability of versatile flight profiles such as: vertical take-off and landing (VTOL) and the ability to hover over a target position. This is a unique advantage for rotorcraft as it eliminates the need for long runways and maintaining high airspeeds. Hence they can be put to service in diverse situations and also be able to manoeuvre complex environments. Making it a good candidate for intelligence, surveillance and reconnaissance (ISR) mission and parcel delivery. Despite this, fixed-wing aircraft are still more prevalent in operation than RUAVs. This is because of their simple structure, their ability to achieve higher cruise airspeeds, and have better endurance and long range. Regardless of these limitations, RUAVs have been proven successful in military and government applications, and are steadily gaining popularity in civilian applications. This was effectively demonstrated by NASA's Jet Propulsion Laboratory when they successfully flew and landed a RUAV on planet Mars (Saez et al., 2021).

Controlling rotorcraft is dynamically more complex than fixed-wing aircraft. As the RUAVs become more accessible, the need for developing new flight control strategies to promote high-performance, reliable and cost-effective systems for use in academic research, military and civilian applications increases. Since RUAVs are safety-critical systems that lack human judgement on-board, autonomous operation require that they must have enough situational intelligence to be able to interact with uncontrolled dynamic environ-

ments and have the self awareness necessary to be able to diagnose and recover from possible system faults (Böhm et al., 2021).

1.2 Research Motivation

Rotorcraft UAVs are inherently unstable and require continuous feedback control to keep them stable (Kim and Tilbury, 2004; He and Han, 2010). They are also nonlinear and dynamically-coupled with regard to aerodynamic forces and moments (Raptis and Valavanis, 2010). In addition, they are underactuated, that is, the number of control inputs is less than the number of degrees-of-freedom (DOFs). Also, the parameters of the rotorcraft cannot be exactly known to be captured in the design model (Raol et al., 2017). This leads to modelling uncertainties. As a result, the design and development of RUAV autonomous flight control system (AFCS) is quite challenging.

Secondly, the total loss of the aircraft system is undesirable due to the high cost of the equipment and the payload being transported. Conversely, the major motivation for using UAVs is for dull, dangerous and harsh environments and this puts UAVs in situations subject to high risk of faults, failures, and eventual damage and total loss of the system. This means that, to achieve a comparable control and stability levels as manually-piloted aircraft, extra effort is required to monitor the condition of the RUAV and take appropriate action when required to recover from adverse conditions (Böhm et al., 2021).

Additionally, the increased interest in the application of UAVs for civilian purposes means that they have to use the public airspace. This means that system failures are more likely to occur in areas where injury and death of civilians is possible (Austin, 2011). Previously, UAVs were only operated in controlled military environment with large margins of error, away from civilian, obstacles and other civilian aircraft (Drozeski et al., 2005). In civil airspace there is a high probability of endangering lives and infrastructure on the ground or mid-air collisions should there be a system failure. The concern is for human safety, airworthiness and operational control regulations (Eyerman, 2013). There is also more trust for piloted aircraft than autonomous ones even though pilot error has contributed substantially to aircraft crashes in recent years. Hence, there is need for robust, fault-tolerant control systems (FTCS) for UAVs in order to achieve the imposed airworthiness and reliability targets and earn the public trust.

The key motivation is to ensure the survivability and reliability of the system under unanticipated faults and failures. Emphasis in this research is directed to the development of control strategies that will maintain overall RUAV performance while being tolerant to faults: system damage and external environmental uncertainties (Ren et al., 2012). The increase in the complexity of RUAV systems leads to an increased probability of error development and propagation of this error to other parts of the system. Also, the uncertainties in the operating environment, and the wear-and-tear degradation of the aircraft system can cause adverse faults that can be hard to isolate and recover in a highly-coupled system. These objectives can be achieved by novel aircraft design and intelligent fault-tolerant control (IFTC) strategies, as proposed herein. A number of research efforts has been proposed to try and optimise the control performance of rotorcraft and improve their survivability (Hall, 2009; Guo, 2009; Ma et al., 2020; Yuke et al., 2022).

Full-scale rotorcraft have the benefit of a human pilot with natural situational awareness to avoid danger and account for certain faults and failures that could result in the loss-of-control (LOC) of the rotorcraft. Experienced pilots find it difficult to operate in adverse environmental conditions such as fog and brownouts. A brownout is a condition created when a rotorcraft is flying close to the ground resulting in the dust being blown around making it difficult to observe the surrounding area (Solem, 2011): which means, even with intelligent pilots onboard, full-scale rotorcraft still fail and survivability is not guaranteed, when inadequate information is available for decision making (Caldwell and Splitt, 2021).

In unmanned aircraft, the guidance, navigation and control (GNC) systems must ensure the stability, manoeuvrability, correct mission execution, and safe completion of tasks without a pilot (Ducard, 2009). However, this cannot be accomplished during an identified actuator failure. The accepted contingency in aviation, as in many safety-critical systems, is to implement hardware redundancy (Goupil and Marcos, 2011) i.e., use two or more actuators such that when one fails another one is available to take its place. Larger aircraft implement physical redundancy with large weight penalties.

Medium-scale rotorcraft are classed as tactical UAVs (Fahlstrom and Gleason, 2012). The motivation for using medium-scale instead of small/miniature-scale UAV as the literature trends suppose we should be targeting, is for long endurance, long distance and the ability to carry larger loads (Barnhart et al., 2021). For this research targeted platform, hardware redundancy is not an option and this generally applies most to medium-scale aircraft in

this category where the mass budget is very stringent. There are two issues that make medium-scale rotorcraft of particular interest when compared to full-scale or miniature aircraft:

1. they are not large enough for additional hardware actuator redundancy to be negligible; and
2. they are big enough to cause serious damage to people and property if not able to recover from actuator faults.

Therefore, it is the objective of this research to come up with an alternative fault tolerance methodology to maintain the same (or better) level of reliability and survivability of a medium-scale rotorcraft UAV as in the case of manned one. That is, in case of a fault or a failure, the system must be able to use some fault tolerance mechanism to either:

1. continue with the mission in a degraded mode;
2. abort the mission and safely return to base; or
3. land the aircraft as soon as possible.

A number of strategies based on classical linear control theories and nonlinear control are presented in the literature. However, they have proven to be limited when it comes to fault tolerance. The successful implementation of intelligent controllers in other fields (Pedro and Dahunsi, 2011; John and Pedro, 2013) shows promise in solving the complexity of the problem under discussion.

The advent of fly-by-wire and the application of software-only flight control means analytical redundancy is possible and can offer better response to failures. It is the aim of this research to demonstrate this assertion on a medium-scale rotorcraft design.

1.3 Literature Review

This section presents a review of the material that is deemed as foundational to the research presented in this thesis. Since the central point of this research is flight control system (FCS), we first review the work that has been done in terms of integrated flight propulsion control. The second part presents the most recent developments in linear and

nonlinear flight control methods that are found in literature. This section also delves into the application of FTC and concludes with a review of computational intelligence techniques that are applicable to control strategies.

1.3.1 Integrated Flight-Propulsion Control

Majority of rotorcraft are designed with a regulated or governed rotor speed, i.e., to operate at a constant setting where the rotation speed does not change much with respect to the reference (Ren et al., 2012; Iwata, 1996). By so doing, the flight controller and engine propulsion controller are decoupled and are often designed separately with a small degree of interaction. However, taking into account the highly nonlinear, coupled dynamics and time varying nature of rotorcraft systems with the drivetrain and engine system, the dynamic interaction of these subsystems cannot be ignored. Thus, in order to optimise the overall rotorcraft performance, the engine controller has to be integrated with the FCS. In the past two decades, there has been a revived interest into designing rotorcraft with rotor speeds that are variable (Guo, 2009; Amri et al., 2016).

Integrated flight-propulsion control (IFPC) is introduced to account for the dynamic coupling of the rotorcraft dynamics and engine systems. Iwata (1996) conducted an extensive study on integrated flight-propulsion control for variable rotor speed (VRS) application on a UH-60A Black Hawk. A feedback regulator for small motion disturbance rejection and a feed-forward trajectory generator for large aggressive manoeuvres were developed and tested in the wind tunnel. He concluded that control integration performed better than distributed controllers especially for large automated manoeuvres. The shortcoming of the method, as he concluded, was that the study was based on linear models and did not cover a wide flight envelope.

Drozeski (2005) first proposed the utilisation of variable rotor speed as a redundant control input for a rotorcraft. This work was conducted on the famous GTMax of Georgia Tech. Although the results were promising, most of the developments were based on linearised models. Guo (2009) conducted a thorough work investigating the optimal rotor speed and used the results with model-following and model-inversion integrated control with state-feedback. The attempt at continuous speed scheduling resulted in running too close to torque and control margins. In conclusion, a finite and discrete rotor speed scheduler was recommended.

Bowen-Davies and Yeo (2017) conducted a comprehensive analysis on the performance and loads of the UH-60A to determine the limits of the aircraft at high advance ratios. In this study, the aim was to avoid compressibility effects at high rotor tip speeds by either varying the rotor blade radius or by varying the rotor speed. The study showed that the range and airspeed of a rotorcraft can be increased by applying variable rotor speeds. However, the study did not show how the variation is achieved and the impact on the control and stability of the aircraft were not considered.

Chi et al. (2019) showed that during one engine inoperative flight condition, the reduction of engine speed reduces the power requirement during steady flight. Using a model based on the UH-60A, it was shown that the rotor speed can be reduced by as much as 25% without an adverse effect on aircraft stability, while the upper limit was set at +7%. Their results also show that reduced speed makes for better landing performance when compared to a fixed rotor-speed helicopter.

Figures 1.2 and 1.3 show the result of an experiment conducted to determine the effect of variable rotor speed for different performance indices in a rotorcraft and the power requirements. As depicted, the rotor speed can be changed to optimise the rotorcraft performance for a specific flight regime. For example: maximum endurance at lower loiter airspeeds will demand a lower rotor speed Ω_1, while maximum airspeed will be achieved at higher rotor speed Ω_3.

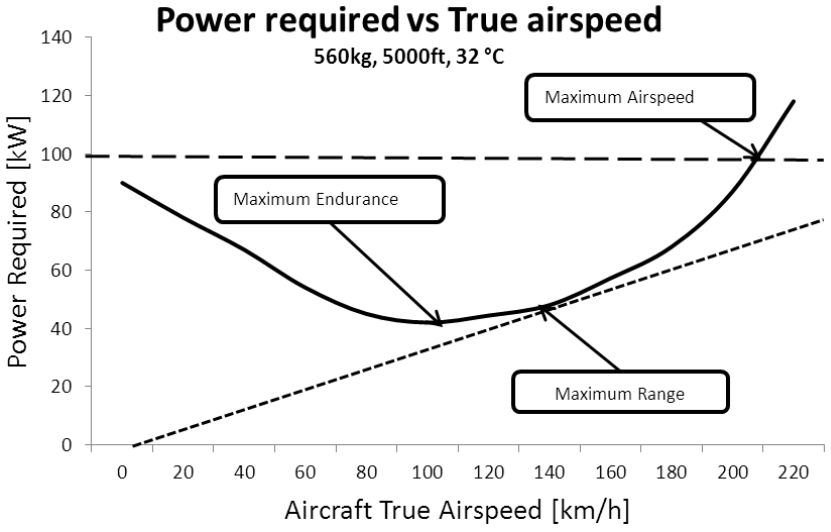


Figure 1.2: Flight profile of a rotorcraft with a fixed rotor speed.

It has been shown also that rotor speed can be used as a redundant flight control option.

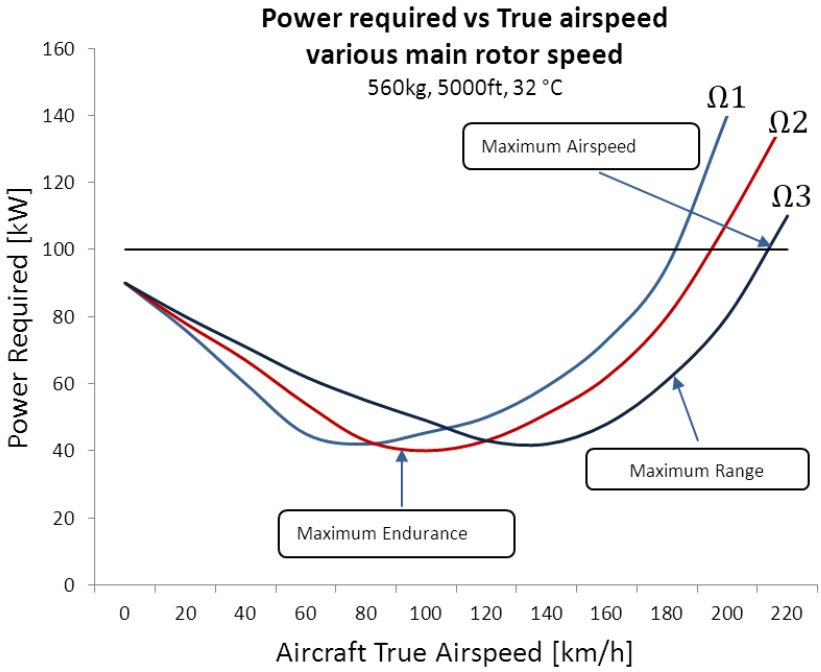


Figure 1.3: Flight profile of a rotorcraft at three different rotor speeds.

The unfortunate drawback of variable speed rotor is the introduction of unfavourable effects such as, rotor loads, vibration, and stability, and the limitations on the transmission system. However, given the advantages of using rotor speed variation, there is a need for application of control strategies that makes this possible.

1.3.2 Fault-Tolerant Control Strategies

Several control strategies have been proposed for rotorcraft FCS, these are: linear control such as PID control (Budiyono et al., 2009; Alvarenga et al., 2014), optimal control (Abdulhamitbilal and Jafarov, 2011; Yu, 2005), model predictive control (Kim et al., 2002), and nonlinear control such as: feedback linearisation also known as nonlinear dynamic inversion (Chen and Huang, 2015; Nemati et al., 2007), and backstepping (Zhou and Wen, 2008; Ahmed and Pota, 2009). These strategies are concerned with the development of closed-loop systems that perform according to design specifications. However, the assumption of these strategies is that everything will continue to function as designed and no faults occur, which is far from reality (Noura et al., 2009). No equipment is as good as it was when it was installed and given enough time, all equipments *will* fail (Williams et al., 1994). There is a high safety requirement that must be met by the rotorcraft control systems. These systems must demonstrate high integrity and reliability even in

the presence of faults.

The principle of redundancy is popular in aerospace engineering in dealing with faults. Physical redundancy is the duplication of sensors, actuators and control system logic that are used in critical systems (Enns and Si, 2003). This is done so that when a physical component fails, there is a duplicate item to take over its function. Physical redundancy is common practice in aircraft development and considerable success has been achieved. The problem with redundancy is the reduction in efficiency of the system by installing two or more components to achieve the same objective. As a result, the complexity of the system is increased, making it more susceptible to faults (Valavanis and Vachtsevanos, 2015). Moreover, the weight, size and power consumption, as well as the time for design, verification, and testing are increased and ultimately the cost of the aircraft is also increased (Venkataraman et al., 2019). When designing a hardware redundancy, there must be a trade-off between safety and efficiency. Analytical redundancy is an approach that uses multiple techniques on the system of interest to obtain its state information or to actuate it to a desired output after a loss of primary function. In this configuration there is no duplication but by using different combinations of sensors and actuators, a desired, albeit decreased, level of performance can be achieved even when one or more sensor or actuator pairs are malfunctioning.

The introduction of FTC enables the system of interest to have built-in tolerance to faults which allows the system continue to operate while avoiding complete system failure. In this case, a *fault* is defined as an unacceptable state deviation from normal behaviour that can lead to a failure. A *failure*, on the other hand, is the condition that disables a certain function of the system leading to the system being unusable. Faults can occur at the *actuators*, *sensors* or *system components* (Ducard, 2007, 2009). Regardless of the cause of the faults, their consequences can be catastrophic and require sophisticated control system designs to avoid, isolate and recover from them with limited human intervention (Zhang and Chamseddine, 2012). As shown in Figure 1.4, FTC systems are classified as passive and active.

A larger amount of FTC work has been dedicated to fixed-wing aircraft (Rudin et al., 2020; Zogopoulos-Papaliakos et al., 2021). It is only recently that the application to RUAVs has gained attention. This sudden interest is due to the fact that RUAVs are increasingly being introduced to operate in difficult and unpredictable environments where reliability

and safety are critical (Sridhar et al., 2022).

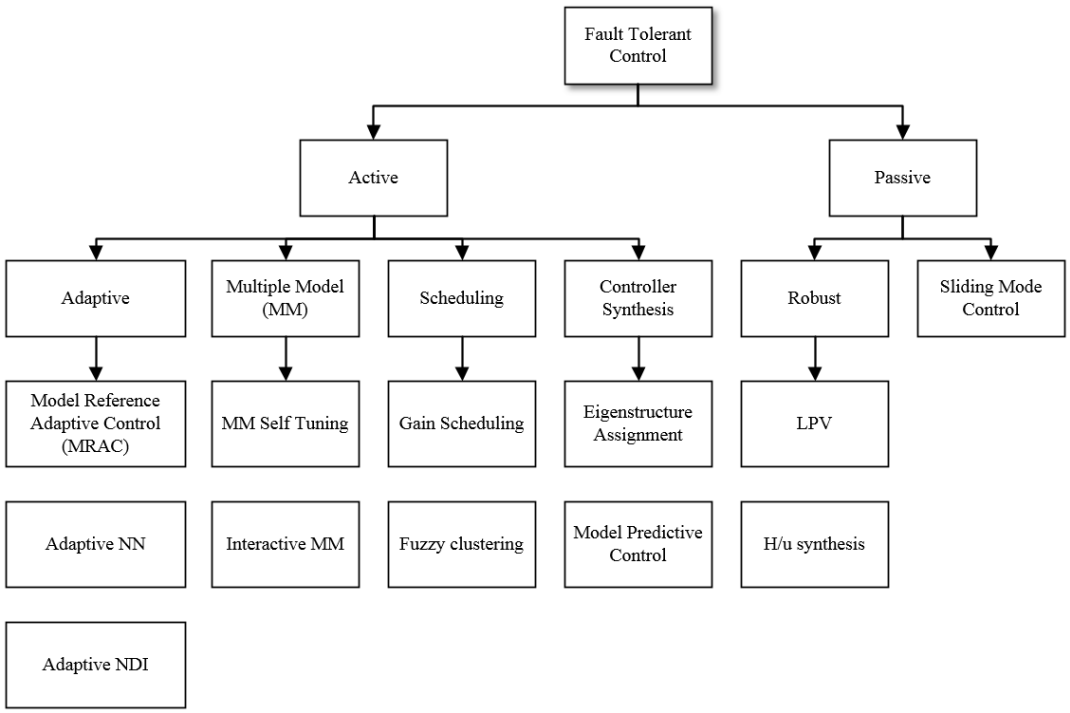


Figure 1.4: Classification of FTC strategies (Edwards et al., 2010).

1.3.2.1 Passive fault-tolerant control

With passive fault-tolerant control (PFTC), a single fixed-structure and fixed-parameters robust controller is developed to accommodate a wide range of specific faults (Zolghadri et al., 2014). Applying a single controller has the advantage of reduced complexity, predictability and ease of implementation since there is no online detection of faults resulting in a computational effective controller. However, this configuration cannot account for all fault modes. As a result, a total loss of the aircraft is possible should these unaccounted for fault modes occur. Also the increased robustness is often at the expense of decreased nominal performance (Edwards et al., 2010)

Xu (2011) demonstrated a nonlinear state-dependent Riccati equation (SDRE) which proved to stabilise a damaged Boeing 747. Shi et al. (2014) showed the application of $\mathcal{H}_\infty$ for PFTC on a windmill with pitch actuator gain and generator gain faults.

A Lyapunov-based strategy which provides robust, rigorous stability proof is the sliding mode control (SMC) approach (Edwards and Spurgeon, 1998). The theory of SMC involves driving the system towards a designed sliding manifold and then keeping it on this

manifold (Mpanza and Pedro, 2015). It is able to transform higher-order system models into first-order equivalents (Slotine and Li, 1991). This provides a method for dealing with unknown nonlinear dynamics. SMC demonstrates robustness to uncertainties, parameter variations and disturbances. Zhang and Chamseddine (2012) implemented a passive technique using SMC on a Qball-X4 quadrotor testbed. The results showed that SMC is robust to a partial damage to one of the propellers. The drawback of SMC is the chattering that arises in the on-off switching method that is used. Sridhar et al. (2022) presented the theoretical and experimental implementation of SMC on a tilt-rotor quadcopter. The study investigated the robustness of the controller in the presence of wind disturbances and one rotor actuator LOE fault. The authors further discussed the issues related to transitioning from simulation environment to experimental tests.

Halbe and Hajek (2020) proposed a SMC-based trajectory tracking controller which is robust to internal and external uncertainties. The controller was separated into an inner loop for stabilising attitude dynamics and outer loop for trajectory tracking. Using simulations, the SMC was shown to be accurate in tracking and insensitive to disturbances. Wang and Zhang (2018) proposed an adaptive SMC with control allocation to adjust the controller gains for a multirotor helicopter in an effort to eliminate tracking errors and to maintain closed-loop stability when subjected to actuator faults. It was shown, theoretically, that system stability is guaranteed. This controller was compared to the conventional SMC and linear quadratic regulator (LQR) and proved to be superior to both. Amini and Akbari (2017) employed an inner-loop SMC supported by a PID controller for the navigation outer-loop. This combination was shown to be robust to gust disturbances. Jiang et al. (2019) developed an integral SMC for trajectory tracking and disturbance reduction on a small-scale helicopter. Simulation results demonstrated the usefulness and robustness of the developed method. However, the developed controller was based on a linearised helicopter model.

Sira-Ramirez et al. (1994) first proposed a chatter-free application in miniature rotorcraft systems. By investigating the vertical lift of the rotorcraft, the authors were able to achieve lift to a reference point smoothly without chattering. Shakev and Topalov (2015) demonstrated the application of continuous SMC on a quadrotor. The application proved the suitability of this method in stabilising under-actuated systems. Simulation results showed that the proposed SMC achieved asymptotic stability. Integral SMC was used successfully in (Hamayun et al., 2013, 2016). Utkin et al. (2020) showed that high-order

sliding mode control (HOSMC) has better chattering suppression and as a result has better disturbance rejection when compared to conventional SMC.

1.3.2.2 Active fault-tolerant control

In active fault-tolerant control (AFTC), faults are detected, isolated and then the information about the fault is used to reconfigure the controller accordingly in order to make up for the system's defect in terms of magnitude and number of faults (Noura et al., 2009). Control reconfiguration can be achieved by selecting from pre-allocated controllers or by online redesigning the control law. Figure 1.5 shows the philosophy of AFTC. There are two parts of interest in AFTC, namely: fault diagnosis and isolation (FDI); and controller reconfiguration.

Fault Detection and Isolation (FDI): The early detection of a fault, isolating its location and classifying its nature successfully are first and most important step in AFTC design (Zolghadri et al., 2014). The major problem with FDI is that *false positives* (normal conditions detected as faulty) can result in degradation of aircraft FCS, even worse so, *false negatives* (faulty conditions detected as normal) can result in undetected faults which could lead to eventually the complete loss of the aircraft (Drozeski et al., 2005). Therefore, the FDI must be robust to parameter changes, noise and environmental disturbances. Hitherto, the failure of FTC design has been the assumption that the FDI is a perfect black-box system. One of the best methods for FDI is to use an observer (Nejati and Faraji, 2021).

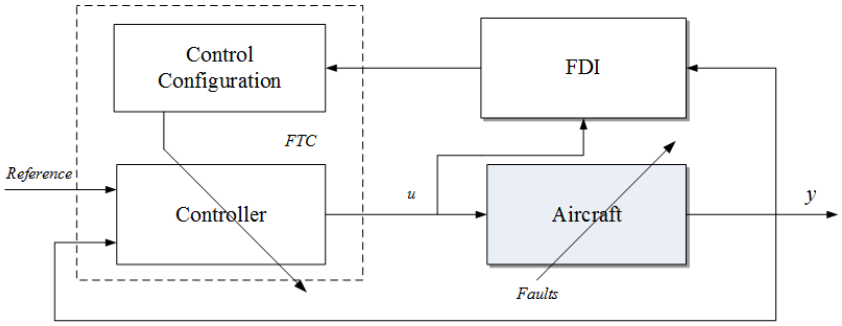


Figure 1.5: Active fault-tolerant control philosophy. Adapted from (Edwards et al., 2010).

Reconfigurable Control: After a fault has been successfully detected and isolated, a mechanism is required in order to achieve fault tolerance (Noura et al., 2009; Zolghadri

et al., 2014). Reconfigurable control is about using the FDI information to find a new controller in order to maintain control of the system. The system is reconfigured to reduce the deterioration and enables it to safely complete the task. This is achieved by applying system redundancy, either physical or analytical. The primary goal of reconfigurable control is to maintain system stability during an occurrence of a fault. Regaining rudimentary control of the rotorcraft is secondary and depending on the safety-criticality of the rotorcraft design, this is included in the cost of control system complexity (Richter, 2011). AFTC reconfiguration methods in Figure 1.4 are detailed in the next sub-sections.

Adaptive control methods have the ability to adapt the control law to uncertain conditions such as when there are changes in the plant, the external environment or a fault occurs (Åström and Wittenmark, 1989; Fabri and Kadirkamanathan, 2001; Kim et al., 2003). Model reference adaptive control consists of an adjustable controller and the mechanism to adjust this controller. *Indirect* adaptation uses system identification to estimate the model parameters of the plant and from this model, the controller is adjusted. In *direct* adaptation the attempt is to adapt the control parameters in such a way that the plant will track the reference model (Edwards et al., 2010).

Bechlioulis and Rovithakis (2010) presented a prescribed performance adaptive control methodology for application to multi-input multi-output (MIMO) affine-in-control systems. With this method, the authors showed that by the selection of a proper robust control Lyapunov function (RCLF), the tracking error can converge to an arbitrarily small value and at a specified rate. Drozeski (2005) developed a multi-tiered control architecture to allow RUAVs to complete their tasks despite system failures. This architecture enables the inclusion of *a priori* information and adaptive control to recover from faults and optimise performance. The added benefit of using FTC is that it also recovers well from gust and wake turbulence which are treated as external faults. The added advantage is that it also offers the trade-off between detailed model and control complexity.

Adaptive control techniques are well documented in the literature, as summarised in (Zhang and Jiang, 2008). They are mostly limited to single-input single-output (SISO) linear systems (Zhou and Wen, 2008) and theoretical in nature. This is not suitable for rotorcraft systems, because they are MIMO nonlinear and time-varying systems (Padfield, 2008). Dydek (2010) demonstrated, with simulation and flight-test, the adaptive control

for flight and guidance control of aircraft with time-delayed dynamics. This proved to attain the performance objective and stability.

Multiple model-based AFTC uses predetermined sets of models and pre-computed control laws. Each law is selected based on the identified model of a known fault mode (Ducard, 2007). In this method, the error between the output and the reference model informs the model selection in order to minimise the error cost function. This strategy is limited to restricted number of operating points or faults that are assumed to be known *a priori* (Murray-Smith and Johansen, 2020). Should the rotorcraft encounter an undetermined fault, the control of the system is lost due to the lack of a preprogrammed response to that fault.

Gain scheduling is used to select controller gain from a predetermined control gains based on the current system dynamics as informed by the FDI (Sadeghzadeh, 2015). This method is mostly used, not only for fault tolerance, but also for changes in operating environment.

Control allocation or control synthesis is a popular reconfiguration control technique for actuator faults. This method uses a combination of the remaining actuators to produce the desired actuation after a loss of one or more control actuators. This method produces a cycle-average of forces and moments by employing the remaining actuators (Valavanis and Vachtsevanos, 2015). Figure 1.6 shows the controller reconfiguration by control allocation FTC. In the figure, the *control block* can be either a linear controller i.e., PID or LQR (Khan et al., 2013) or nonlinear controller that takes the input from the FDI (Edwards et al., 2010). The following is a list of control allocation types used in literature.

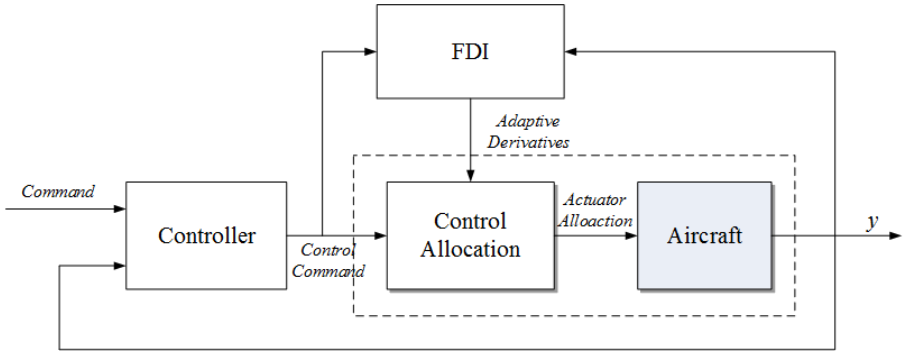


Figure 1.6: Controller reconfiguration by control allocation.

1. *Daisy Chaining:* This is a special type of control allocation. The control effort

is distributed to actuators by using priority analytical redundancy. The priority ensures that the control effort is distributed to the primary actuators which have the highest impact on the output first, then the effort is distributed to the secondary actuators once the primary actuators have been saturated (Alwi et al., 2011). This process is repeated until the desired control effort is achieved or all the actuators have been saturated.

2. *Pseudo Inverse:* The matrix factorisation computational expense of daisy chaining makes them less desirable in complex applications. Pseudo inverse uses a simple solution to the system. The pseudo inverse uses linear control allocation by optimising the generalised inverse with the following equation: $\mathbf{y}_d = \mathbf{Bu}$, therefore $\mathbf{u} = \mathbf{B}^{-1}\mathbf{y}_d$, where $\mathbf{u}$ is found by inverting the control effectiveness matrix (provided $\mathbf{B}$ is invertible). This approach is suitable for online reconfiguration (Ding, 2008).

3. *Eigenstructure Assignment:* This is a method of choosing the eigenvalues of the closed-loop system. The reconfigurable control FDI uses linear fault models to choose the possible faults and then send the input to a predefined eigenvalues (Edwards et al., 2010). The gain K is chosen to stabilise the closed-loop systems.

4. *Model Predictive Control (MPC):* This is a method that employs an iterative algorithm for fine-tuning control action. MPC has become popular due to its capability to handle system constraints. Since faults normally send the system to its limits, MPC is able to protect further deterioration of the system by changing/modifying the constraints. The drawback of MPC is the computational burden when applied to real-time control system and it is dependent on the availability of FDI information in order to optimise the tuning of constraints (Edwards et al., 2010).

1.3.3 Computational intelligence techniques in fault-tolerant control

1.3.3.1 Artificial neural network (ANN)

The increase in the application of ANNs is attributed to their capability to model any complex system's input-output relationships. The development of ANN is inspired by attempts to mimic the mechanism of a biological nervous systems (Lapedes and Farber, 1987; Hopfield, 1988; Kruse et al., 2013). They are universally distributed, adaptive and

generally nonlinear approximators without the need for *a priori* information. They are also capable of extracting useful system dynamics from the data.

The basic components of ANN are connected neurons that form an information processing network. This interconnection of neurons gives the ANN the learning ability, generalisation and parallelism (Bishop, 2003). The performance of an ANN application depends on the proper selection of the network architecture, i.e., the number of hidden nodes, activation functions, learning algorithm and the learning rate, which are used to fine-tune the ANN performance.

The two types of commonly used ANN are the multi-layer perceptron (MLP) and the radial basis function (RBF). By incorporating feedback, an ANN results in a dynamic space model. This is done by putting delayed input and output data at the input. This delay is used to modify the internal parameters of the ANN so that it adapts to the input-output relationships. Nonlinear AutoRegressive eXogenous (NARX) is a popular dynamic neural network (DNN) that can approximate any nonlinear system. NARX is adapted using a real-time recurrent learning algorithm. Plett (2003) used DNN for adaptive inverse control on four simple nonlinear SISO systems. This strategy proved to be effective against imposed disturbance.

ANN model identification is used a great deal in literature for the approximation of nonlinear plants. It does not require detailed knowledge of the plant only the input-output relationship is necessary. In addition, it requires a large amount of training data. System modelling using MLP has been applied in (Putro et al., 2009; Samal et al., 2008). Pedro and Kantue (2011) applied an RBF for online parameter estimation. Both supervised and unsupervised training have been reported (Fabri and Kadirkamanathan, 2001). Tijani et al. (2014) used a NARX model for nonlinear identification of a small unmanned helicopter. Since indirect adaptive control involves model adaptation using the difference between the error and the reference model, ANN-based identification methods offer an attractive prospect.

1.3.3.2 Fuzzy logic control

Fuzzy logic control (FLC) is arguably the most applied intelligent control scheme. First conceived by Zadeh (1965), it derives its power from the ability to reason with vague and imprecise knowledge. Instead of using crisp *"on" or "off"* logic, fuzzy logic uses vague

approximation to determine the required control action. For example, instead of using concrete boundaries between what is *hot* or *cold*, fuzzy logic employs a sliding degree of belonging to either class. It relates input space called the antecedent to an output space called the consequent using linguistic *if-else* rules and membership function for processing information.

A number of successful FLC applications for flight control systems has been documented in the literature. Khizer et al. (2014) developed an inner-loop and an outer-loop fuzzy logic controllers in a flight control system for tracking attitude and height of a small RUAV. Velagic and Osmic (2010) presented a detailed design and experimentation of a FLC applied to a 2DOF helicopter. In this work, the robustness of the FLC was proven and benchmarked against the PID. The application of adaptive FLC was also presented in (Yu, 2005). This adaptive FLC proved to be superior to the classical PID which requires re-tuning for different flight conditions. The challenge with FLC design is that the definition of rules requires detailed *a priori* knowledge of the system by an expert. However, when the expert information is available it can be readily incorporated into the controller.

1.3.3.3 Hybrid intelligent control

What makes ANNs less acceptable for safety critical systems is that they are a *"blackbox"*. The information is represented as numerical. On the other end, the shortcomings of FLC are that it requires *a priori* information from the system's experts. The combination of neural networks with fuzzy logic affords an opportunity to explore the advantages of both methods resulting in what is called Neuro-Fuzzy (NF) (Fabri and Kadirkamanathan, 2001).

Neuro-fuzzy refers to a hybridisation, which results from the synergistic combination of the learning ability and parallel processing of the ANN and the knowledge representation and human-like reasoning of the FLC. It is represented by the sum of weighted inputs with a logistic activation function mimicking an *if-else* decision point. This way it presents the transparency required in terms of membership-like logistic functions (Kolman and Margaliot, 2009).

Kurnaz et al. (2010) presented an autonomous flight controller for the fixed-wing UAV. The controller implemented an adaptive neuro-fuzzy inference system (ANFIS) to control

the altitude, heading and airspeed. A simulation was conducted on an Aerosim blockset and FlightGear and flight test was conducted on an Aerosonde UAV platform. This work demonstrated that ANFIS is able to accomplish fast responses while maintaining stability. In (Premkumar and Manijandan, 2014) an ANFIS was developed through simulations for controlling a brushless motor at various load and speeds. The results were benchmarked against the PI controller and fuzzy-tuned PID controller. The ANFIS outperformed its counterparts in terms of rise time, overshoot, settling time and steady-state error. It also demonstrated robustness to load variations. Shafiekhani et al. (2015) proposed an adaptive critic-based NF system. In this work, the adaptive NF system used back-propagation with dynamic programming for online learning to control the roll angle of an unmanned bicycle. Through simulation and experimentation they demonstrated the applicability of NF controller to an inherently unstable nonlinear system and got results that are better than the FL controller.

Since SMC demonstrates robustness to uncertainties, parameter variations and disturbances, it is not surprising that it is extensively applied in PFTC. The major drawback of SMC is that it suffers from chattering caused by the discontinuous control strategy used. A hybrid of sliding mode with other control strategies has been proposed in literature. Herein the focus is the SMC combination with NF to form a neuro-fuzzy-sliding mode controller (NFSMC) which has been proven to reduce chattering. Boubakir et al. (2009) applied the NFSMC to the coupled-tanks control problem. In this study, a standard SMC was compared to the NFSMC from tracking water levels. The NFSMC showed considerable improvement and reduced actuator action over the non-hybrid SMC. Javadi-Maghaddam and Bagheri (2010) presented an adaptive neuro-fuzzy sliding mode control (ANFS) against an adaptive neuro-fuzzy sliding mode control based on genetic algorithm (ANFSGA). The authors used the SMC for guiding the ensemble into stability (sliding surface) and used the ANFIS for generating the optimal control action. The genetic algorithm was used to get the best control action by finding the solution candidate that is closest to the sliding surface.

Backstepping has been applied successfully for nonlinear control and disturbance rejection. However, it is sensitive to parameter variations. Mohd Basri et al. (2015), developed a hybrid controller based on backstepping with an adaptive FL for regulation and trajectory control of a quadrotor with varying dynamics. The backstepping is used as a primary controller, while the adaptive fuzzy controller is used to compensate the system when

it varies from nominal. The hybrid was proved to be stable via the Lyapunov stability theorem. Similar control techniques have been reported in (Wang et al., 2017, 2015) with only simulation results presented.

1.3.3.4 Summary of computational intelligence techniques

The ideas of computational intelligence are to implicitly develop adaptive control techniques that are able to sustain stability and controllability under a variety of disturbances and uncertainties. In a way, they are able to tolerate system faults better than the linear and nonlinear controllers discussed earlier. There have been questions about the stability of nonlinear adaptive controllers. How to prove that the adaptation of parameters leads to convergence of parameters and that closed-loop stability is maintained is a topic of interest among researchers (Zhou and Wen, 2008; Liu, 2017).

1.4 Identified Gaps in the Existing Literature

The literature presents an insight into the state of the art of the control strategies currently employed on rotorcraft FTC. As such, the knowledge gaps identified in the literature are as follows:

1. There is a need for rotorcraft UAVs that have high payload capability, range and endurance, while operating at optimal power with less environmental impact. Variable rotor speed has been proposed as one of the promising approaches. However, there is no research focused on the control application for this strategy.

2. Rotorcraft are highly nonlinear, coupled systems and require nonlinear treatment to deal with model uncertainties, time-varying parameters and dynamic coupling. Application of variable rotor speed makes the dynamics even more complicated, as it requires the inclusion of high fidelity engine and actuator models.

3. To avoid the loss of the RUAV system during an event of a flight control actuator fault, FTC has been proposed with promising results. However, there is limited results showing the application of rotor speed variation for actuator redundancy.

4. Computational intelligence has been applied in industry. However, it is still treated with suspicion in the aviation industry where most applications are only for research

and hobbies. Investigating the combination of FTC with computational intelligence may prove worthwhile for the rotorcraft design. A comprehensive study to develop computational intelligence control strategies is required.

5. Any proposed control strategy must be developed in a way that is practically applicable. The challenges of introducing advanced fault-tolerant strategies have been the certification of the control systems in both the hardware and the software. To the author's knowledge, there is no research output on the topic of certification implications of intelligent FTC strategies applied to rotorcraft.

1.5 Research Hypothesis/Research Question

In response to the literature reviewed so far and the gaps that are identified, the proposed research aims to confirm the following hypothesis:

A rotorcraft UAV that employs a rotor speed variation technique can demonstrate flight control actuator fault tolerance by the application of an integrated flight-propulsion control using computational intelligence techniques.

1.6 Research Objectives

From the above motivation and the gaps identified, the objectives of this research project are as follows:

1. To investigate the possibility of improving rotorcraft performance using variable rotor speed and the control of the rotorcraft using integrated flight-propulsion strategy. This has a potential of also improving the fault tolerance of the rotorcraft system with an additional DOF in the swashplate that can be used as redundant control.

2. To develop a high-fidelity model for the medium-scale rotorcraft including engine and actuator dynamics with the aim of evaluating the ability of the variable rotor speed and integrated flight-propulsion combination in demonstrating resilience to actuator faults.

3. To investigate the application of computational intelligence in integrated flight-propulsion control. The aim is to quantify the performance improvements of com-

putational intelligence strategies over the limitations of conventional linear control strategies found in literature. The end goal being to propose the most suitable control strategy for a variable rotor speed aircraft.

4. To develop an intelligent fault-tolerant integrated flight-propulsion control strategy. In order to improve system reliability and avoid total loss, tolerance to main rotor actuator faults are investigated with the goal of demonstrating acceptable level of confidence for flights in civilian airspace.

5. The proposed flight control strategy must eventually fulfil its requirement by being implemented on a real system. Therefore, the last objective is to analyse the practicality i.e., development and certifiability of the proposed control methodology. To use hardware-in-the-loop simulation strategies interface with a real actuators to uncover practical issues that would, otherwise, not be visible in a simulation.

1.7 Research Methodology

This research project focuses on developing a nonlinear model of the rotorcraft. Actuator models are integrated for the simulation of the nonlinear dynamics. The resulting system model is used to develop an intelligent fault-tolerant control strategy. The SMC, a PFTC strategy, and DNN, an intelligent AFTC strategy are investigated. These methods are benchmarked against the popular PID controller. For determining and optimisation of control parameters, a number of optimisation techniques based on evolutionary algorithms such as ant colony optimisation, firefly, cuckoo search, ant lion and genetic algorithm are investigated.

The following listing summarises the approach that is taken in this research.

1. A nonlinear rotorcraft UAV model is developed. Emphasis is toward computational intelligence tools;

2. Use the best model to design control strategies using the control tools investigated in literature;

3. Investigate the application of computational intelligence and evolutionary optimisation algorithms for controller parameters tuning;

4. Apply the model to develop linear PID controller with the optimisation techniques;

5. Apply the model to develop passive fault-tolerant controller using sliding mode control technique;

6. Apply the model to develop active fault-tolerant controllers using different computational intelligence tools;

7. Critically evaluate the developed FTC controllers by comparing controllers from items (4), (5) and (6).

The proposed methods are developed and evaluated in numerical simulation environment. The laboratory and field evaluation of the methods being investigated is proposed as recommendation for future work.

1.8 Research Contributions

This research project presents the development of control strategies in order to tackle the issues of increasing rotorcraft performance and, most importantly, improve safety and reliability. This research project resulted in the following key contributions:

1. The design of a medium-scale RUAV model simulation that enables the demonstration of variable rotor speed;

2. The results of the study of integrating engine control into the flight control for variable rotor speed as compared to a distributed control strategy;

3. A comprehensive study and the development of computational intelligence integrated flight-propulsion control. These are compared to traditional classical control methods. In addition, the investigation and comparison of evolutionary algorithms such as particle swarm optimisation, genetic algorithm, continuous ant colony optimisation, antlion optimisation, cuckoo search and firefly algorithm for optimal tuning of control parameters are presented;

4. A study of robust PFTC based on SMC and AFTC strategies to recover from actuator faults and complete failure using control allocation of redundant degree of freedom offered by variable rotor speed. New knowledge contribution is in the RUAV system evaluated for its resilience to system parameter changes due to variable rotor speed and actuator loss-of-effectiveness faults; and

5. The development of an HILS experimental rig for validation and rapid prototyping of the control strategies algorithms developed. Certification considerations present a formal methodology for developing the evidence required for control software certification.

1.9 Thesis Outline

The purpose of this research project is to explore and evaluate the challenges in the RUAV applications and their reliance on classical control strategies as shown in the literature. This thesis presents the evidence to argue for the necessity of investigating the application of intelligent FTC solutions to the problem of variable rotor speed control.

Chapter 1 forms the core of this thesis by discussing motivation, the gaps in literature and setting objectives and defining the plan and the approach taken to fill the knowledge gaps.

Chapter 2 presents the details of the experimental platform, modelling and analysis of the nonlinear rotorcraft that is used to evaluate the control algorithms proposed in the rest of the thesis. It contains the rotorcraft equations of motion including rotor dynamics. The actuator dynamics are also presented in order to simulate faults and failures in the rotorcraft model.

Chapter 3 presents the rotorcraft trim analysis and the design of PID stabilising controllers for hover and forward flight. This is presented here as it is the benchmarking controller in aviation and it is used to mark the starting point to which advanced nonlinear controllers can be referred. The optimal PID controller parameters are found using computational intelligence optimisation techniques.

Chapter 4 presents the design, the analysis using Lyapunov method and implementation of the sliding mode controller. The controller design also includes the use of ant colony optimisation, particle swarm optimisation techniques for finding optimal controller parameters.

Chapter 5 presents the design and implementation of the DNN with feedback linearisation. The DNN is investigated for its applicability as an active fault-tolerant controller. The DNN is also used for FDI to enable AFTC. This chapter also intro-

duces the application of multi-objective optimisation to find the parameters of the DNNFBL controller.

Chapter 6 presents hardware-in-the-loop simulation of the controllers developed in the preceding chapters. In this chapter, the simulated actuators are replaced with experimental electromechanical actuators to validate the proposed control strategies and the computational intelligence optimisation algorithms applied in a real-life system.

Chapter 7 reflects on the research approach and the results presented in the thesis. In this chapter, the conclusions are drawn in terms of the ability for the intelligent control algorithms used for integrated-flight propulsion for fault tolerance and recommendations for future improvements of the proposed methods are enumerated.

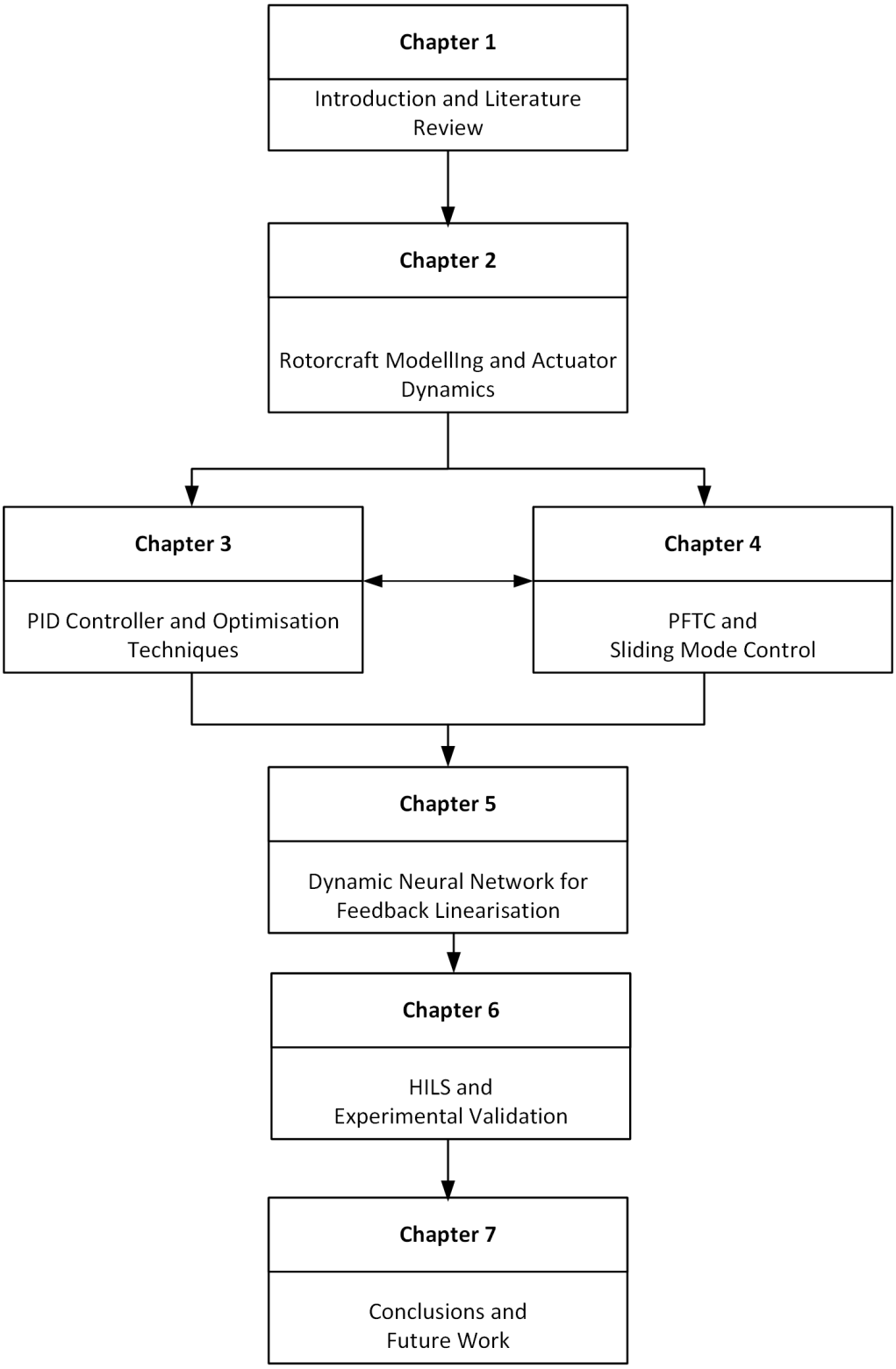


Figure 1.7: Structure of the thesis.

Chapter 2

Mathematical Modelling of the RUAV and Actuator Dynamics

This chapter describes the rotorcraft model used for the intelligent FTC investigation. It begins with the discussion of the coordinate reference frames used for the derivation of the rotorcraft's equations of motion. The rotorcraft dynamic model includes the main and tail rotor actuators and the engine dynamics. This is necessary when investigating integrated-flight propulsion control. The objective is to end up with an affine-in-control nonlinear model which is suitable for the flight control algorithms developed in subsequent chapters.

2.1 Research Introduction

To be able to design a flight control system, a mathematical model that reflects the flight dynamics of the rotorcraft of interest is required. The model of a helicopter is much more complex compared to that of a fixed-wing aircraft. The most common configuration for rotorcraft is a single main and tail rotors, which is the platform of choice of this research investigation. Other configurations such as compound rotor, tandem rotor and quadrotor are not discussed, even though they are becoming common and just as interesting (see Johnson (2013) for more details). Researchers in the academia and industry have recently shown interest in rotorcraft UAV research (Valavanis and Vachtsevanos, 2015).

Rotorcraft are inherently unstable nonlinear systems, with coupled and time-varying dynamics. They are a class of multi-input, multi-output (MIMO) systems that are underactuated, which means that they have more degrees of freedom than they have control

inputs. This makes the development of flight control systems challenging even under fault-free conditions. For such developments, the dynamics of the rotorcraft need to be well understood. Therefore, a high-fidelity model is essential in making sure that the FTC is designed to a model that is as close to the real rotorcraft representation as possible. At the same time, model-based development avoids design and development using the real rotorcraft system which may be prone to accidental damage to the aircraft.

A model of a system is an approximation of the true system and can never be perfect. The model must have high enough fidelity to give confidence that the flight control algorithms developed are representative of the control of the real system. The approach to model-based control development relies on the system model that is used to synthesise the control laws. Therefore, in this thesis we distinguish between the following modelling methodologies.

1. White box: analytical model developed from first principle;
2. Black box: computational model using system identification based on large sets of flight data;
3. Grey box: a combination of first principle and parameter identification from flight test data.

The analytical treatment of a rotorcraft model can be derived from Momentum Theory (MT) and Blade Element Theory (BET) (Prouty, 1984). The two seemingly different theories offer the same explanation from different perspectives. Analytical models have been covered extensively in literature over the past few decades (Johnson, 1980; Prouty, 1995; Bramwell et al., 2001; Padfield, 2008) and a more recent coverage is found in (Venkatesan, 2014; Johnson, 2015). Unfortunately, these models usually have high dimensions and order, making them difficult to use for FTC development.

Simplified models have also been presented in literature (Mettler et al., 1999; Shim et al., 2000; Gavrilets, 2003). These simplified models have received prominence during the advent of UAVs at the turn of the century. The simplification involves neglecting most of the transient behaviour of the rotor system, such as induced inflow and rotor aeroelastic modes. While these methods offer a simple model for controller development, they are only applicable in a smaller flight envelope where the selected assumptions hold. This limits their application for nonlinear FCS development.

Mettler et al. (2002) proposed a parametric helicopter model which combines analytical modelling with system identification. The system identification is conducted in the frequency domain. The parameters of the model are deduced from flight data, thus creating a linear model around the collected flight data points. This helicopter modelling philosophy continues to be popular among researchers to this day (Kim et al., 2002; Raptis and Valavanis, 2010; Ren et al., 2012; Alvarenga et al., 2014). The reason for this popularity is the availability of a large knowledge-base for linear control techniques, which makes the FCS design manageable.

The problem with the preceding approach is that the accuracy of the model is reduced due to the presence of the control feedback loop which is required for data collection from an inherently unstable system such as the rotorcraft (Gavrilets, 2003; Isermann and Münchhof, 2010). As a result, the rotorcraft model is limited to a small set of flight manoeuvres usually around hover and slow forward flight. In order to create a model that covers the whole (or larger) fight envelope, a full nonlinear model is required (Raptis and Valavanis, 2010). This however, requires high computational resources to run in real-time to facilitate FCS design (Pourrezaei Khaligh, 2014). However, with the development in computer technology, the models that were previously deemed too computationally expensive or intractable for FCS are now readily available and practical (Budiyono et al., 2010).

A number of aircraft modelling tools and software systems have been developed in recent years to assist with rapid prototyping, such as the commercial FlightLab (Zang et al., 2017), CIFER (Tischler and Remple, 2012), NASA's NDARC (Johnson, 2015) and Delft's SIMONA (Smaili et al., 2008; Pavel et al., 2016). These tools give the aerodynamicist an opportunity to investigate rotorcraft performance, optimise flight parameter and simulate different design scenario. The drawback of these tools is that they are proprietary and the source code is not made available to the user, thus exhibiting a black-box nature. This makes it difficult to understand what these tools are doing in the background. They also do not easily incorporate FCS development. For this reason, models based on the MATLAB® / Simulink® environment are popular in practice (Budiyono et al., 2010; Chachou et al., 2014; Munna et al., 2020).

The initial part of this thesis focuses attention at white-box modelling of the rotorcraft. As such, the developed model is based on first principle theory and dimensions from

the actual rotorcraft designs. Later the grey-box approach is used in applying advanced computational intelligence algorithm. The information from the experimental platform is incorporated into the analytical model as detailed later.

In the previous chapter we proposed the application of variable rotor speed for improving the rotorcraft performance and also make available this extra DOF to successfully control the aircraft in the event that one of the primary actuators experiences a fault. For such a fault-tolerant system to be possible, the details of the actuator and their failure mechanisms are required. Therefore, the model developed makes provision for inclusion of the actuator dynamics. The medium-scale rotorcraft under investigation is intended to be used with an electrical engine to provide power to the rotor system. To control the rotor speed successfully, a model of an engine is also required. Which means that despite the already complex rotorcraft dynamics, two more levels of complexities are added by incorporating the actuator dynamics and the motor drive system.

2.2 Description of the Rotorcraft Platform

This section describes a model of an experimental RUAV currently under development at Denel Aeronautics. The platform is designed to be 630 kg, with the ability to carry 80 kg payload. The power plant is based on a 90 kW electric motor with a 15 kWh Lithium Polymer battery pack, with an option for a 90 kW petrol internal combustion engine. It also includes a 5-blade fully-articulated rotor system. The main and the tail rotors employ symmetrical airfoils based on the NACA0021. The rotorcraft system is designed to be completely Fly-by-Wire (FBW) based on the success of UH-60M upgrade (Solem, 2011). The avionics is designed to support line-of-sight (LOS) communication with a ground station and equipped with a return to home base functionality should communication with ground station fail. The flight control algorithm is executed on a mission computer running a real-time operating system (RTOS). The FTC allows for automatic take-off, route steering, airspeed hold, height (altitude) hold, automatic hover and automatic landing.

The platform's operating environment is a dusty and sandy area, and possibly marine environment. Hence, this rotorcraft system must be tolerant to faults that these environments may pose. This research effort is directed at dealing with actuator faults. The detailed experimental setup to investigate FTC on the physical rotor flight control

actuators is presented in Chapter 6.

2.3 Modelling Assumptions

Since a model is an imperfect representation of a real existing agent, there is a need to draw some simplifying assumptions. This ensures that the model is simple enough to make the controller derivation process practical. At the same time, the model must not be too simple that the generality and accuracy of the platform are lost and the validity of conclusions drawn is affected. The derived model must still account for a broad flight envelope. As such, the assumptions imposed on the target rotorcraft model are as follows:

Assumption 1. *The rotorcraft is considered to be 6-DOF rigid, only the movement of the blades is considered; (Raptis and Valavanis, 2010);*

Assumption 2. *The body axis has longitudinal symmetry about the centre of gravity;*

Assumption 3. *The fuselage's centre of pressure coincides with the c.g.;*

Assumption 4. *There is uniform inflow across the rotor area, there is no reverse flow;*

Assumption 5. *The directional vector of the thrust developed is always perpendicular to the rotor disc tip-path-plane;*

Assumption 6. *The local properties of the air/flight medium do not vary significantly;*

Assumption 7. *The curvature and rotation of the Earth are locally insignificant.*

The first assumption is included in order for the Newton-Euler rigid body dynamics to apply in fixed mass and inertia tensor. Assumption number two is necessary to reduction in model complexity. However, due to the presence of the tail rotor assembly on the right of the vertical stabiliser, the rotorcraft is not entirely symmetrical about the x-axis but the difference is negligible. Assumption number four is valid since the designed maximum velocity of the rotorcraft is 180 km/h and the tip speed is $\Omega R = 177.8\ m/s$, this results in an advance ratio of $\mu < 0.3$, which small enough to ignore reverse flow according to Seddon and Newman (2002). The sixth and the last assumptions are sufficient as the rotorcraft is expected to operate at low altitude and over a short range close to the Earth surface.

2.4 Coordinate Reference Frames

The rotorcraft has many moving parts. Therefore, to describe the motions of these parts it is necessary to describe the coordinate reference frames in which these motions occur.

Four coordinate reference frames are defined for the work presented here:

1. The Earth-fixed reference frame F_E: this is used as an inertial reference frame in which Newton's laws are applicable and valid.
2. The Body-fixed reference frame F_B: this is the reference frame that coincides with the rotorcraft's centre of gravity (CG).
3. The Hub-fixed reference frame F_h: this is the reference whose origin coincides with the top centre of the rotor hub.
4. The tip-path reference frame F_t: this frame is in alignment with the tip-path-plane (TPP).

2.5 Rotorcraft Model

The rotorcraft is controlled by pitching of the main rotor blades. Controlling the vertical displacement (z) is achieved by varying the pitch on all blades equally i.e., collective pitch. Controlling the pitch (θ) the roll (ϕ) is by the cyclic varying in the respective direction of motion. Controlling the rotorcraft yaw (ψ) is by varying the pitch of the tail rotor blades collectively, and the difference in the counter-torque of main rotor. The rotorcraft has six DOF. However, it only has four inputs. As mentioned earlier, this type of system is known as under-actuated i.e., the number of system inputs is less than the number of outputs (i.e., number of the DOF).

The important parameter for rotorcraft flight dynamics are:

1. The disk area which is given by:
$$A = \pi R^2. \tag{2.1}$$
2. The disk solidity is given by:
$$\sigma = \frac{bc}{\pi R}. \tag{2.2}$$
3. The Lock number is given by:
$$\gamma = \frac{\rho a c R^4}{I_\beta}. \tag{2.3}$$

where R, b, c, a and I_β are the rotor radius, the number of rotor blades, equivalent chord, the blade lift-curve slope and the moment of inertia of the blade about the flapping hinge, respectively.

2.5.1 Position kinematics

The dynamics of the rotorcraft are derived using the Newton-Euler approach. two reference frames are required for navigation and control, i.e., the Earth-fixed reference frame $F_E = \{RO, x, y, z\}$ and the body-fixed reference frame $F_B = \{R_B O_B, x_B, y_B, z_B\}$, whose centre coincides with the CG of the rotorcraft. Therefore, the CG of the rotorcraft with reference to the F_E frame is given by $\boldsymbol{\xi} = [x\ y\ z]^T$. The rotational angles are $\boldsymbol{\eta} = [\phi\ \theta\ \psi]^T$ of the F_B frame with respect to the F_E frame which are for pitching $\theta \in (-\pi/2, \pi/2)$, rolling $\phi \in (-\pi/2, \pi/2)$ and yawing which is unrestricted. These are referred to as Euler angles. The translational and angular velocities in the F_B frame are given by $\mathbf{v}^B = [u\ v\ w]^T$ and $\boldsymbol{\omega}^B = [p\ q\ r]^T$ respectively.

Due to the two reference frames, there is a need to transform from F_B frame to the F_E frame and vice-versa. For this purpose we define, $\mathbf{R}$, the transformation matrix as a function of Euler angles given by:

$$\mathbf{R}(\boldsymbol{\eta}) = R_\psi(\psi) R_\theta(\theta) R_\phi(\phi), \tag{2.4}$$

$$\mathbf{R}(\boldsymbol{\eta}) = \begin{bmatrix} \cos\psi\cos\theta & \cos\psi\sin\theta\sin\phi - \sin\psi\cos\phi & \cos\psi\sin\theta\cos\phi + \sin\psi\sin\phi \\ \sin\psi\cos\theta & \sin\psi\sin\theta\sin\phi + \cos\psi\cos\phi & \sin\psi\sin\theta\cos\phi - \cos\psi\sin\phi \\ -\sin\theta & \cos\theta\sin\phi & \cos\theta\cos\phi \end{bmatrix}. \tag{2.5}$$

Therefore, the position and the velocity vectors in the F_B frame are translated to the F_E frame using the following:

$$\dot{\boldsymbol{\xi}} = \mathbf{R}\mathbf{v}^B. \tag{2.6}$$

2.5.2 Orientation kinematics

Raptis and Valavanis (2010) described the properties of a special orthogonal group matrix $SO3$. This group is used for relating orientation in the F_B frame translate to the F_E frame

using the differential transformation matrix, $\mathbf{T}$, which is defined as:

$$\mathbf{T} = \begin{bmatrix} 1 & \sin\phi\tan\theta & \cos\phi\tan\theta \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi/\cos\theta & \cos\phi/\cos\theta \end{bmatrix}. \tag{2.7}$$

Such that the body rates vector relates to the Euler angles as follows:

$$\dot{\boldsymbol{\eta}} = \mathbf{T}\boldsymbol{\omega}^B. \tag{2.8}$$

This transformation matrix, $\mathbf{T}$, has a singularity at $\pm 90°$. This is a disadvantage of using Euler angles for representing aircraft orientation. However, the manoeuvres under consideration in this thesis are not too aggressive for this to be considered as a problem. If the objective is to design an acrobatic RUAV with aggressive manoeuvres, the use of quaternions representation of angles is recommended (see Cai et al. (2011); Chen et al. (2021) for more).

2.5.3 Rotorcraft dynamics

The sum of forces and moments acting on the rotorcraft is expressed in the Newton-Euler rigid-body equations of motion as follows:

$$m\frac{d\mathbf{v}^B}{dt} + m(\boldsymbol{\omega}^B \times \mathbf{v}^B) = \mathbf{F}, \tag{2.9}$$

$$\mathbf{I}\frac{d\boldsymbol{\omega}^B}{dt} + (\boldsymbol{\omega}^B \times \mathbf{I}\boldsymbol{\omega}^B) = \mathbf{M}, \tag{2.10}$$

where m and $\mathbf{I}$ are the rotorcraft mass and the moment of inertia respectively. All external forces and moments acting on the rotorcraft with respect to the CG are combined into the triples $\mathbf{F} = [X\ Y\ Z]^T$ and $\mathbf{M} = [L\ M\ N]^T$ respectively. These forces and moments are contributions are derived from different rotorcraft components. The contributing system components are the main and tail rotors, fuselage, vertical and horizontal stabilisers respectively, and are depicted in Figure 2.1.

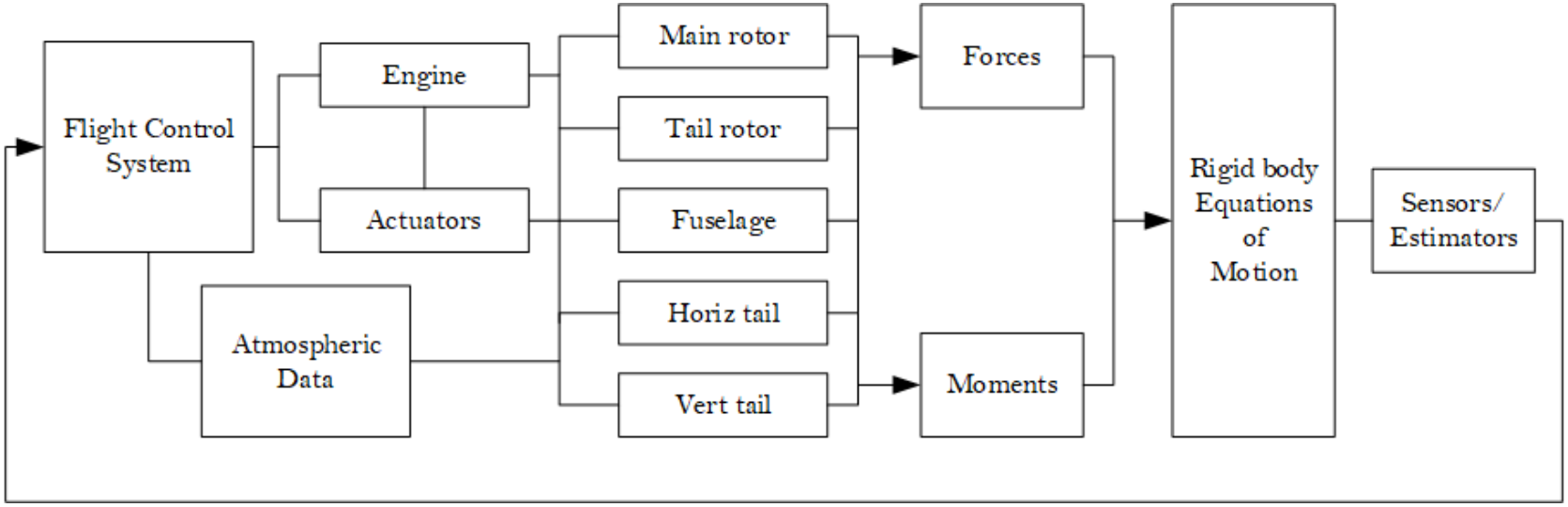


Figure 2.1: The rotorcraft as a combination of subsystems working together to accomplish the flight task.

The dynamics of the rotorcraft in 6-DOF result from the summation of all contributing forces and moments in the R_B frame and summarised as follows:

$$\begin{aligned} X &= X_M + X_F + X_g^B, \\ Y &= Y_M + Y_F + Y_T + Y_V + Y_g^B, \\ Z &= Z_M + Z_F + Z_H + Z_g^B, \\ L &= L_M + Y_M h_M + Y_T h_T + Y_V h_V, \\ M &= M_M - X_M h_M + M_T - Z_H l_H, \\ N &= N_M + Y_T l_T + Y_V l_V. \end{aligned} \tag{2.11}$$

The positions of the main rotor $\{0, 0, h_M\}$, tail rotor $\{x_T, y_T, h_T\}$, horizontal stabiliser $\{l_H,\ 0,\ h_H\}$ and vertical stabiliser $\{l_V,\ 0,\ h_V\}$ with reference to the CG. A simplified representation of the rotorcraft with subsystem locations is shown in Figure 2.2.

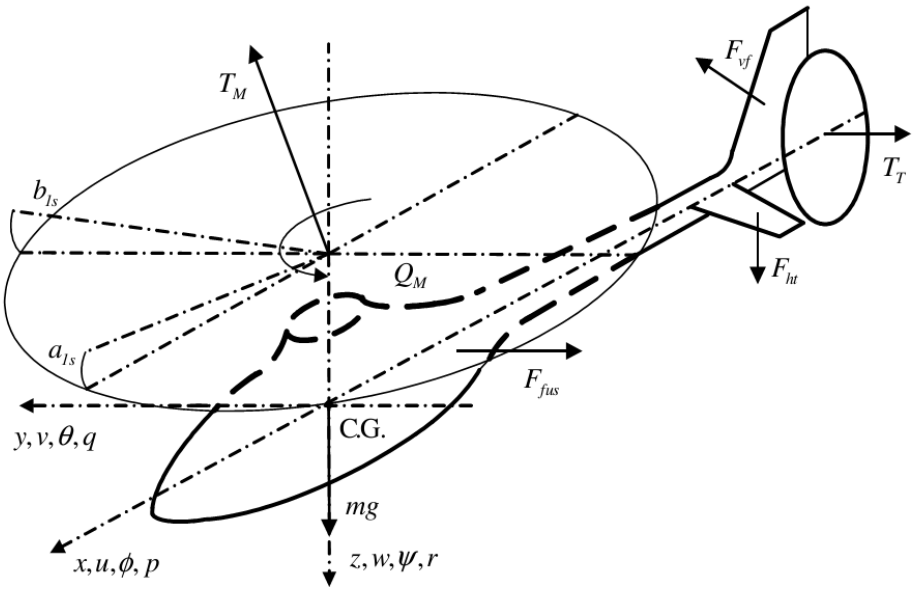


Figure 2.2: The rotorcraft configuration diagram with axis labels (Mpanza and Pedro, 2021).

The forces and moments are applied to the rotorcraft with respect to the $\{F_B\}$ frame. This is due to the fact that moments of inertia are time varying in the F_E frame and thus make analytical treatment a challenge (Raptis and Valavanis, 2010). Therefore, the force due to gravity, which pulls the rotorcraft downwards towards the Earth's centre is

transformed onto the F_B frame as follows:

$$\mathbf{F}_g = mg \begin{bmatrix} -\sin\theta \\ \sin\phi\cos\theta \\ \cos\phi\cos\theta \end{bmatrix}. \tag{2.12}$$

2.5.4 Main rotor

The helicopter convention outside the United States (most of the Western world) is that the rotor blades rotate clockwise when viewed from the top of the aircraft (Seddon and Newman, 2002). The main rotor produces lift to support the mass of the rotorcraft and the thrust to navigate by pitching the blades as they rotate about the rotor hub. While the tail rotor thrust is normally directed to the positive y-axis to oppose the counter rotation of the rotorcraft body. The rotor blades are free to move in three DOF. These degrees are *flapping*, *lead-lag* and *feathering*. There are several design configurations that facilitate the movement of blades as they rotate around the rotor hub. The three most known are *articulated*, *teetering* and *hingeless* rotors. This research focuses on the articulated rotor system that has a flapping and a lead-lag hinge for each blade. Controlling the blade's pitch angle is with respect to the feathering hinge via the pitch links attached to the blade horns. The effective control of the thrust, longitudinal and lateral forces, and moments is achieved through the blade's collective pitch angle, θ_0 and the cyclical variation pitch angles around the rotor azimuth Ψ. Conventionally, $\Psi = 0^o$ occurs when the blade points to the tail of the rotorcraft. The blade's pitch angle is periodic and the effective pitch angle can be represented by the first harmonics of the Fourier series which is a function of the azimuth angle Ψ (Layton, 1983).

$$\theta_M = \theta_0 + \theta_{lat}\cos(\Psi) + \theta_{lon}\sin(\Psi), \tag{2.13}$$

where θ_0, is the collective pitch angle responsible for controlling main rotor thrust. The lateral, θ_{lat}, and the longitudinal, θ_{lon}, cyclic pitch angles result in the orientation or tilting of the main rotor disc in the sideways and forward directions respectively.

The vertical movement and hovering are controlled by the main rotor thrust developed. The equation of thrust is derived from BET as follows:

$$dT_M = \rho\pi R_M^4 \Omega_M^2 C_T dr, \tag{2.14}$$

where dr is a small section of the blade. Integrating the thrust through the length of the blade gives:

$$T_M = \frac{\rho}{2}\Omega_M^2 a_M b c_M \int_0^R \left(\theta r^2 - \frac{v_{iM}(r)}{\Omega_M}\right) dr, \tag{2.15}$$

where a_M is the main rotor blade lift-curve slope, b is the number of blades, c_M is the main rotor blade chord length and v_{iM} is the main rotor induced velocity. For uniform flow across the rotor disc area, based on assumption 4, the main rotor produces thrust according to the following (Gavrilets, 2003):

$$T_M = C_T \rho A (\Omega_M R_M)^2 \tag{2.16}$$

$$C_{TM} = \frac{\sigma_M a_M}{4}\left((\mu_z - \lambda_{M0}) + \begin{bmatrix} \frac{2}{3} + \mu_x^2 + \mu_y^2 \\ -\mu_y \\ \mu_x \\ 0 \end{bmatrix}^T \boldsymbol{u}\right), \tag{2.17}$$

where $\boldsymbol{u} = [\delta_{col}\ \delta_{lon}\ \delta_{lat}\ \delta_{ped}]^T$ is the vector of rotorcraft inputs and λ_{M0} is the inflow ratio, i.e., $\lambda_{M0} = v_{iM}/(\Omega_M R_M)$. Therefore, the inflow is required for the calculation of the thrust. We define the hover induced velocity based on momentum theory as:

$$v_{iM} = \sqrt{\frac{T_M}{2\rho\pi R_M^2}}. \tag{2.18}$$

For general flight in any direction, the induced flow velocity is:

$$v_{iM} = \frac{T}{2\rho\pi R^2 V_\infty}, \tag{2.19}$$

where V_∞ is given by:

$$V_\infty = \sqrt{V_{hor}^2 + (V_{vet} + v_{iM})^2}, \tag{2.20}$$

where:

V_∞	:	is the velocity of the rotorcraft through air;
V_{hor}	:	is the component of rotor velocity parallel to the TPP;
V_{vet}	:	is the component of rotor velocity normal to the TPP.

The induced velocity v_{iM}, is normally solved through the use of iterative approach such as the Newton-Raphson method. However, for an affine-in-control model there is a need for the solution of thrust in closed form. Therefore, the simplified Glauert approximation function that has been shown to yield good induced velocity estimation is used in this model as follows (Johnson, 2013):

$$v_{iM} = \begin{cases} -\frac{V_a}{2} - \sqrt{\frac{V_a^2}{4} - 1} & V_a \leq -2 \\ 1 - \frac{V_a}{2} + \frac{25}{15}V_a^2 + \frac{7}{6}V_a^3 & -2 < V_a < 0 \\ -\frac{V_a}{2} + \sqrt{\frac{V_a^2}{4} + 1} & V_a \geq 0 \end{cases} \cdot \tag{2.21}$$

For a rotorcraft translating forward or sideways there is a planar horizontal force developed (H-force). This is a drag force resulting from asymmetry in the angle of attack along the azimuth. The rotor drag in the hub-plane is given by:

$$C_{xH} = \frac{H_{xM}}{\rho(\Omega_M R_M)^2 \pi R_M^2}, \tag{2.22}$$

$$C_{yH} = \frac{H_{yM}}{\rho(\Omega_M R_M)^2 \pi R_M^2}. \tag{2.23}$$

The rotor drag coefficient is found as follows:

$$C_{xH} = -\frac{\sigma C_{D_{M0}}}{4}\mu_x + \frac{\sigma_M a_M}{8}\left[(\mu_z - \lambda_{M0})(2\mu_x\delta_{col} + b_{1s})\right], \tag{2.24}$$

and

$$C_{yH} = -\frac{\sigma C_{D_{M0}}}{4}\mu_y + \frac{\sigma_M a_M}{8}\left[(\mu_z - \lambda_{M0})(2\mu_y\delta_{col} - a_{1c})\right]. \tag{2.25}$$

The main rotor a reaction torque that is applied to the body below. This torque is a result of the blade stiffness at the root (hub stiffness) and rotor anti-torque. The main rotor develops torque according to the following:

$$Q_M = C_{Q_M}\rho(\Omega_M R_M)^2 \pi R^3, \tag{2.26}$$

$$C_{Q_M} = \frac{C_{D_{M0}}\sigma_M}{8}\left(1 + \frac{7}{3}\mu^2\right) - \frac{\sigma_M a_M}{8}(\mu_z - \lambda_{M0})\left(2(\mu_z - \lambda_{M0}) + \begin{bmatrix} \frac{4}{3} \\ -\mu_y \\ \mu_x \\ 0 \end{bmatrix}^T \mathbf{u}\right), \tag{2.27}$$

where $C_{D_{M0}}$ is the zero-lift drag coefficient. The first term in Equation 2.27 represents the profile drag and the other represents the induced torque during thrust development.

The thrust is applicable in the vertical motion of the rotorcraft. To include longitudinal and lateral motion, the flapping dynamics are required as detailed below.

2.5.4.1 Flapping Motion

A model of the the flapping dynamics is used to calculate the developed forces and moments acting on the CG. The flapping angle is periodic in terms of the azimuth position of the blade Ψ, and therefore can be represented by a Fourier series:

$$\beta(\Psi) = \beta_0 + \sum_{k=1} a_{kc} \cos \Psi + \sum_{k=1} b_{ks} \sin \Psi. \tag{2.28}$$

Since flapping is periodic it can be approximated using a first-order harmonic, a truncated Fourier series (Layton, 1983).

$$\beta(\Psi) = \beta_0 + a_{1c} \cos \Psi + b_{1s} \sin \Psi. \tag{2.29}$$

This results in the coning of the TPP according to the value of β_0, and the tilting of the TPP in the direction of β_{1c} and β_{1s}, along the longitudinal and lateral axes, respectively. In literature a_{1c} and b_{1s} are replaced with a and b. Without loss of generality, the flapping angle dynamics can be approximated by two first-order differential equations as follows:

$$\tau_f \dot{a}_{lc} = -a_{lc} - \tau_f q + \left(\frac{\partial a_{lc}}{\partial \mu} \mu_x + \frac{\partial a_{lc}}{\partial \mu_z} \mu_z \right) + K_{lon} \delta_{lon}, \tag{2.30}$$

$$\tau_f \dot{b}_{ls} = -b_{ls} - \tau_f p + \left(\frac{\partial b_{ls}}{\partial \mu_v} \mu_y \right) + K_{lat} \delta_{lat}, \tag{2.31}$$

where K_{lon} and K_{lat} are the longitudinal and lateral control gains for cyclic pitch inputs δ_{lon} and δ_{lat} respectively. While τ_f is the main rotor flapping damping time constant given approximately by:

$$\tau_f = \frac{16}{\gamma \Omega_M} \approx 0.0257 \ s. \tag{2.32}$$

This means that the flapping angles settle faster than the revolution frequency of the rotor blades, which in this case is 59.27 rad/s. Considering Assumption 6 that the developed thrust is always perpendicular to the TPP, the following equations represent the longitudinal components of the thrust due to the flapping of the TPP in the respective directions:

$$\begin{aligned} X_M &= -T_M \sin a_{lc}, \\ Y_M &= T_M \sin b_{ls}, \\ Z_M &= -T_M \cos a_{lc} \cos b_{ls}. \end{aligned} \tag{2.33}$$

The tilting of the TPP also means that the main rotor torque is deviated from the z axis accordingly. Using the flapping angles, the components of moments due to the main rotor

become:

$$\begin{aligned} L_M &= \left(\frac{\partial L_M}{\partial b_{ls}}\right) b_{ls} - Q_M \sin a_{lc}, \\ M_M &= \left(\frac{\partial M_M}{\partial a_{lc}}\right) a_{lc} - Q_M \sin b_{ls}, \\ N_M &= -Q_M \cos a_{lc} \cos b_{ls}, \end{aligned} \tag{2.34}$$

where $\left(\frac{\partial L_M}{\partial b_{ls}}\right)$ and $\left(\frac{\partial M_M}{\partial a_{lc}}\right)$ are as a result of the out-of-plane resistance of the blades due to the attachment to the rotor hub. This is a result of a hinge-less rotor or stiff flapping hinges and are representative of a torsional spring constants. The rotor stiffness term is given by:

$$\left(\frac{\partial L_M}{\partial b_{ls}}\right) = \left(\frac{\partial M_M}{\partial a_{lc}}\right) = \frac{e}{4R} b m_b R(\Omega R)^2, \tag{2.35}$$

where e, b and m_b are the blade's hinge offset, number of blades, and moment of inertia respectively.

2.5.5 Tail rotor

The tail rotor is modelled to be proportional to the main rotor. The tail rotor does not have cyclic pitch, but other than that, the thrust is developed is similarly to the main rotor. Also, since the main and tail rotor are driven from the same power plant through a gear system, the tail rotor speed is simply given by $\Omega_T = k_T \Omega_M$, where k_T is the reduction gear ratio and Ω_M is the main rotor speed.

The tail rotor equation of thrust is given as follows:

$$dT_T = \rho \pi R_T^4 k_T^2 \Omega_M^2 C_T dr. \tag{2.36}$$

Integrating the thrust through the length of the tail rotor blade and the azimuth angle gives:

$$T_T = \frac{\rho}{4} k_T^2 \Omega^2 a_T b c_T \int_0^{R_T} \int_0^{2\pi} \left(\theta r^2 - \frac{v_{iT}(r)}{k_T \Omega_M}\right) d\Psi dr. \tag{2.37}$$

For uniform flow across the tail rotor disc area, this expression is equivalent to:

$$C_{T_T} = \frac{\sigma_T a_T}{4} \left(\delta_{ped} \left(\frac{2}{3} + \mu_x^2 + \mu_z^2\right)\right), \tag{2.38}$$

where σ_T, a_T and c_T are the tail rotor's disc solidity, the blade's lift-curve slope and chord length respectively.

The tail rotor only has a thrust component in the y-direction, i.e., $Y_T = -T_T$. The component of torque is about the longitudinal axis, $M_T = -Q_T$.

2.5.6 Fuselage

The fuselage experiences lift and drag. The rotor downwash pushes the fuselage downward during hover. In forward, level flight and lateral motion, the fuselage experiences parasite drag. The forces acting on the fuselage are as follows:

$$\begin{aligned} X_F &= \frac{1}{2}\rho S_x^F V_\infty (u - u_{wind}), \\ Y_F &= \frac{1}{2}\rho S_y^F V_\infty (v - v_{wind}), \\ Z_F &= \frac{1}{2}\rho S_z^F V_\infty (w - w_{wind} + v_{iM}), \end{aligned} \tag{2.39}$$

where S_i^F, is the characteristic areas of the fuselage as seen from the $i \in \{x, y, z\}$ axes. Flat plane area that is perpendicular to the axis of interest is used. The dynamic pressure of the fuselage is $V_\infty = \sqrt{(u - u_{wind})^2 + (v - v_{wind})^2 + (w - w_{wind} + v_{iM})^2}$. The main rotor induced velocity, v_{iM}, influences the dynamic pressure on the fuselage.

2.5.7 Horizontal and vertical tail stabilisers

The horizontal and the vertical tails produce aerodynamic lift and drag as the rotorcraft flies through air. The vertical tail aerodynamic forces and moments produced are as follows:

$$Y_V = S_V \left(C_{L\alpha}^V V_V + |v_V| \right) v_V, \tag{2.40}$$

$$L_V = -Y_V h_V, \tag{2.41}$$

$$N_V = Y_V l_V, \tag{2.42}$$

where S_V is the vertical tail area, $V_V = \sqrt{(u - u_{wind})^2 + (w - w_{wind} + l_V q - K_\lambda v_{iM})^2}$ is the axial velocity, $v_V = v - v_{wind} - l_V r - \epsilon^T v_{iT}$ is the velocity normal to the vertical tail, K_λ is the intensity of the main rotor wake experienced at the empennage and ϵ^T is the fractional area exposed to the tail rotor induced velocity.

The horizontal tail aerodynamic forces and moments produced are as follows:

$$Z_H = \frac{1}{2}\rho S_H \left(C_{L\alpha}^H |u - u_{wind}| + v_H \right) v_H, \tag{2.43}$$

$$M_H = -Z_H l_H, \tag{2.44}$$

where $v_H = w - w_{wind} + l_H q - K_\lambda v_{iM}$ is the velocity and S_H is the effective surface area of the horizontal stabiliser and l_H is the horizontal stabiliser longitudinal distance from the C.G.

2.5.8 Drive-train dynamics

The present rotorcraft system employs an electric engine. Electric cars such as the Tesla Model 3 have been successful in operating electric motor-based drive trains. They are built on the premise of having low impact on the environment compared to internal combustion engines. Other benefits for using electric motor include:

- Electric motors have high-energy efficiency and power density.
- Reduced vibrations.

The NASA's Green Flight Challenge (Administrator, 2013) has inspired the research and development of fuel-efficient flight. The focus is on fuel efficiency, however other benefits such as flight safety and reduced noise signatures have been witnessed.

For the power-plant, the engine is modelled as an armature-controlled direct current (ACDC) motor. The equations describing the dynamics of the ACDC motor are:

$$\begin{aligned} v_a &= R_a i_a + L_a \frac{di_a}{dt} + K_e \Omega_e, \\ Q_e &= K_Q i_a, \quad P_e = Q_e \Omega_e, \\ I_{eq} \dot{\Omega}_e &= Q_e - Q_M - k_e Q_T - Q_{loss}, \end{aligned} \tag{2.45}$$

where $v_a, i_a, R_a, L_a, K_e, K_Q, I_{eq}, Q_e, Q_M, Q_T, Q_{loss}$ and Ω_e, are the motor input voltage, armature current, resistance, inductance, emf proportionality constant, torque constant, the equivalent moment of inertia of the motor and the rotorcraft downstream load, the torque developed by the engine, the main rotor torque, the tail rotor torque and the torque due to gearbox and miscellaneous systems. These parameters are found through system identification of the motor itself. The equivalent moment of inertia as seen by the motor is found by lumping all the moments of inertia upstream. To the motor, variations in load torque is akin to disturbance on the speed which must be corrected promptly to avoid affecting rotorcraft dynamics.

The motor sizing, power electronics and the optimisation of the power requirements for the batteries and inverter are beyond the scope of this thesis. The assumption is that the selected powerplant has enough power to meet the nominal rotorcraft flight profiles and planned range.

To the motor, variations in load torque is akin to disturbance on the regulated speed

setting which must be corrected properly to avoid rotor speed dropping below the required value. To the rest of the flight controller, changes in engine is also treated as a disturbance that the controller must be robust against.

2.6 Rotorcraft Simulation Model

As discussed above, the CG coincides with the rotorcraft geometric centre and the origin of the F_B frame. For the purpose of flight control design, a mathematical model that is square and affine-in-control model is required (Saffarian and Fahimi, 2008). In this section the simulation model of the rotorcraft is presented. Recall from Equations 2.6, 2.8, 2.9 and 2.10, the rotorcraft equations of motion are combined and expressed as:

$$\begin{aligned} \dot{\mathbf{v}}^B &= \frac{1}{m}\mathbf{F} - \boldsymbol{\omega}^B \times \mathbf{v}^B, \\ I\dot{\boldsymbol{\omega}}^B &= -\boldsymbol{\omega}^B \times (I\boldsymbol{\omega}^B) + \mathbf{M}, \\ \dot{\boldsymbol{\xi}} &= \mathbf{R}\mathbf{v}^B, \\ \dot{\boldsymbol{\eta}} &= \mathbf{T}\boldsymbol{\omega}^B, \end{aligned} \tag{2.46}$$

where $\mathbf{F}$, $\mathbf{M}$, $\mathbf{R}$ and $\mathbf{T}$ are the external forces and moments acting on the CG of the rotorcraft, and the translational and rotational transformation matrices respectively.

2.6.1 Rotorcraft model with ideal actuators

The first rotorcraft model discussed is the one without actuator dynamics. The assumption here is that the actuator responds to input instantaneously and follows the command without error. This is a *gross* simplification, however it is used extensively in literature (Gavrilets, 2003). Later in the thesis, modelled actuators and real experimental actuators are included in this full rotorcraft model. As a result, this model has sixteen states: the twelve rigid-body states, the two flapping states and the two drive-train states.

The non-affine-in-control state-space model of the rotorcraft dynamics is given by:

$$\begin{aligned} \dot{\boldsymbol{x}} &= \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}), \\ \boldsymbol{y} &= \boldsymbol{h}(\boldsymbol{x}), \end{aligned} \tag{2.47}$$

where $\boldsymbol{x} \in \Re^n$ is the state vector, $\boldsymbol{u} \in \Re^m$ is the system input vector, and $\boldsymbol{y} \in \Re^p$ is the system output vector. The functions $\boldsymbol{f}(.)$ and $\boldsymbol{h}(.)$ are generally nonlinear. The state

vector is:

$$\begin{aligned}\boldsymbol{x} = &\ [x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9, x_{10}, x_{11}, x_{12}, x_{13}, x_{14}, x_{15}, x_{16}]^T \\ = &\ [v,\ u,\ w,\ p,\ q,\ r,\ x,\ y,\ z,\ \phi,\ \theta,\ \psi,\ a_{lc},\ b_{ls},\ i_a,\ \Omega_M]^T.\end{aligned} \tag{2.48}$$

The system input vector is $\boldsymbol{u} = [\delta_{col}\ \delta_{lat}\ \delta_{lon}\ \delta_{ped}\ \delta_\Omega]^T$. The fifth control input is included to aid fault-tolerant flight control development. This is used to control the speed of the engine and by extension the speed of the rotor.

The simplified system state-space representation is given as follows:

$$\begin{aligned}\dot{v} &= (vr - wq)/m + s_\theta g + (-T_M(\delta_{col}, \Omega_M)s_a + X)/m, \\ \dot{u} &= (wp - ur)/m + s_\phi c_\theta g + (T_M(\delta_{col}, \Omega_M)s_b + Y + T_{TR})/m, \\ \dot{w} &= (uq - vp)/m + c_\phi c_\theta g + (-T_M(\delta_{col}, \Omega_M)c_a c_b + Z)/m,\end{aligned} \tag{2.49}$$

$$\begin{aligned}\dot{p} &= \frac{I_{yy} - I_{zz}}{I_{xx}}qr - \frac{I_{xz}}{I_{xx}}(\dot{p} - pq) + \frac{L}{I_{xx}}, \\ \dot{q} &= \frac{I_{xx} - I_{zz}}{I_{yy}}pr + \frac{I_{xz}}{I_{yy}}(r^2 - p^2) + \frac{M}{I_{yy}}, \\ \dot{r} &= \frac{I_{xx} - I_{yy}}{I_{zz}}pq + \frac{I_{xz}}{I_{zz}}(\dot{p} - qr) + \frac{N}{I_{zz}},\end{aligned} \tag{2.50}$$

$$\begin{aligned}\dot{x}^I &= c_\theta c_\phi u + (s_\theta c_\phi c_\psi - c_\phi s_\phi)v + (s_\theta c_\phi c_\psi + s_\psi s_\phi)w, \\ \dot{y}^I &= c_\theta c_\psi u + (s_\theta s_\phi s_\psi + c_\theta c_\psi)v + (c_\theta s_\phi s_\psi - c_\psi s_\phi)w, \\ \dot{z}^I &= -s_\theta u + c_\theta s_\phi v + c_\phi c_\theta w,\end{aligned} \tag{2.51}$$

$$\begin{aligned}\dot{\phi} &= p + s_\phi t_\theta q + c_\phi t_\theta r, \\ \dot{\theta} &= c_\phi q - s_\phi r, \\ \dot{\psi} &= \frac{s_\phi}{c_\theta}q + \frac{c_\phi}{c_\theta}r,\end{aligned} \tag{2.52}$$

$$\tau_f \dot{a}_{lc} = -a_{lc} - \tau_f q + \left(\frac{\partial a_{lc}}{\partial \mu}\mu_x + \frac{\partial a_{lc}}{\partial \mu_z}\mu_z\right) + K_{lon}\delta_{lon}, \tag{2.53}$$

$$\begin{aligned}\tau_f \dot{b}_{ls} &= -b_{ls} - \tau_f p + \left(\frac{\partial b_{ls}}{\partial \mu_v}\mu_y\right) + K_{lat}\delta_{lat}, \\ \dot{i}_a &= (-R_a i_a - K_e \Omega_M)/L_a + \delta_\Omega / L_a,\end{aligned}$$

$$\dot{\Omega}_M = (K_Q i_a - Q_R - nQ_T - Q_{loss})/I_{eq}, \tag{2.54}$$

where $s_\theta = sin(\theta)$, $c_\theta = cos(\theta)$, $t_\theta = tan(\theta)$, $s_\phi = sin(\phi)$, $c_\phi = cos(\phi)$, $s_\psi = sin(\psi)$, and $c_\psi = cos(\psi)$. It is clear that this model is not affine-in-control. However, most of control theory presented in literature requires an affine-in-control model for model-based controller design. In Chapter 4, it is shown how this model can be transformed into an affine-in-control equivalent.

2.7 Actuator Dynamics Models

The modelling of actuators is essential in order to get the response of the rotorcraft model to be as close as possible to the real aircraft. The dynamics of actuators are usually ignored when developing FCS in literature (Halbe et al., 2021). Therefore, when the FCS is implemented on a real aircraft it does not work as expected. Since actuators form the basis of the FTC strategies studied here, they are given a detailed treatment. The first actuator considered is the electromechanical actuator which is used in the main rotor (three of them) and one for the tail rotor. Popular actuators in aviation are:

1. Hydraulic actuators;
2. Electrohydraulic actuators;
3. Electromechanical actuators.

Before each actuator type is discussed, it is necessary to understand the swashplate mechanism as this is how the actuators are arranged to translate control into the blades.

2.7.1 Detailed swashplate mechanism

The swashplate is a mechanism for converting linear travel of a set of actuators into height and inclination of the rotor hub. It is complex and nonlinear. Rotorcraft flight control is managed by varying the blades' pitch angles collectively and cyclically. This section discusses the kinematics of a three-actuator swashplate as depicted in Figure 2.3. In the figure, the swashplate supports a flat non-rotating surface mass (with equivalent mass of the hub and rotor head) on three actuators. The surface is able to move in three DOFs by individually controlling the vertical travel of each actuator, x_i, where $i \in \{a,\ b,\ c\}$ for front, left rear and right rear actuators respectively. The front actuator is at a distance l_f from the rotor shaft centre. The rear actuators are at a distance l_r behind the rotor shaft centre and are $l_{rr} = l_{rl}$ apart as shown on the vertical view shown in Figure 2.4.

The kinematics equations of the actuators and the swashplate are as follows:

$$z_a = z + l_f \sin\theta, \tag{2.55}$$

$$z_b = z + l_{rl} \sin\phi - l_r \sin\theta, \tag{2.56}$$

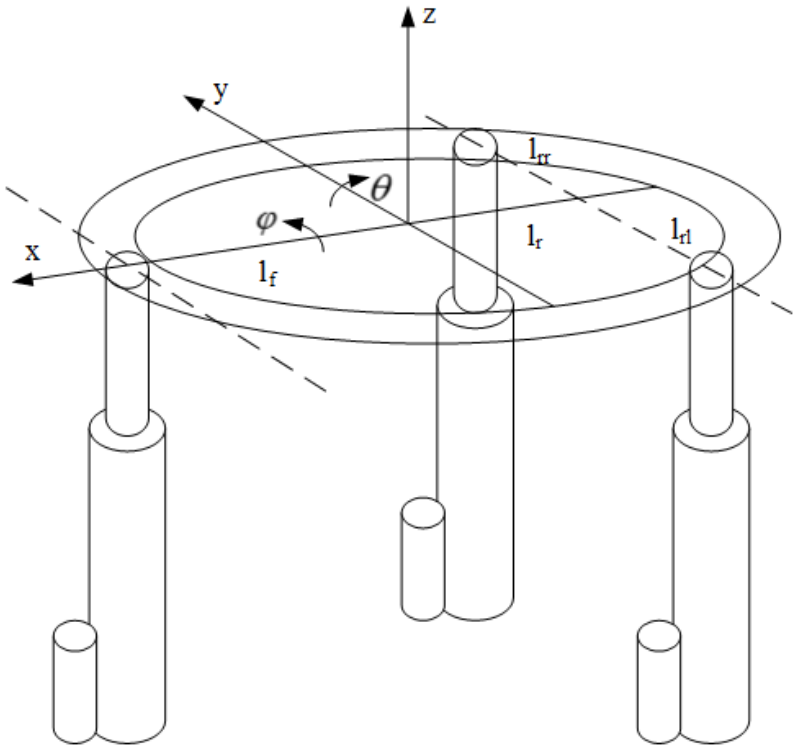


Figure 2.3: Simplified actuator model.

$$z_c = z - l_{rr} \sin\phi - l_r \sin\theta. \tag{2.57}$$

The dynamics for swashplate vertical translation are derived by Newton's 2^{nd} law as follows (Bilyaletdinova and Steblinkin, 2017):

$$M_s \ddot{z} = F_a + F_b + F_c - M_s g, \tag{2.58}$$

where M_s is the mass of the swashplate combined with the masses of the individual actuators. This is distinguished from m, which is the mass of the entire aircraft. The rotation dynamics for pitch and roll are given by:

$$I_x \ddot{\phi} = l_{rr} \left(F_b - F_c\right) \cos\phi, \tag{2.59}$$

$$I_y \ddot{\theta} = \left(l_f F_a - l_r F_b - l_r F_c\right) \cos\theta, \tag{2.60}$$

where $I_x = I_y$ are the moments of inertia of the swashplate disc upon the x and y axes, l_r , is the distance to the two rears actuator along the x-axis, l_f, is the distance from the centre to the front actuator, and $l_{rr} = l_{rl}$ are the distances to the right and the left of the swashplate centre line of the respective rear actuators.

In order to be consistent with established literature, the values $\{z, \phi, \theta\}$ can be replaced by $\{\sigma_0, \sigma_{lc}, \sigma_{1s}\}$. According to (Layton, 1983; Johnson, 2013), the tilting of the swashplate is

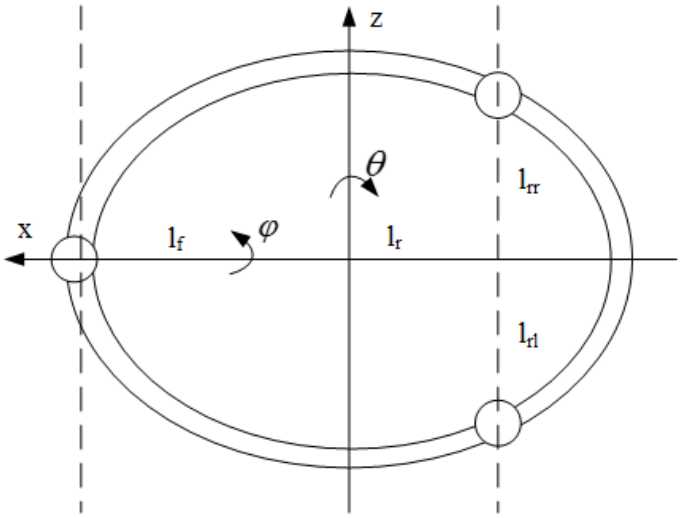


Figure 2.4: Top view of the actuator model with actuator A pointing directly forward.

out of phase with reference to the hub rotor plane by $\pi/2$. The swashplate with reference to the azimuth angle Ψ is given below as:

$$\sigma = \sigma_0 + \sigma_{lc} \sin \Psi + \sigma_{ls} \cos \Psi, \tag{2.61}$$

where σ_0, σ_{lc} and σ_{ls} are collective pitch, lateral and longitudinal cyclic pitch angles. This relates to the pitching angle in the following:

$$\begin{aligned} \sigma &= \sigma_0 + \sigma_{lc} \sin(\Psi + \pi/2) + \sigma_{ls} \cos(\Psi + \pi/2), \\ \sigma &= \sigma_0 + \sigma_{lc} \cos(\Psi) - \sigma_{ls} \sin(\Psi). \end{aligned} \tag{2.62}$$

In traditional swashplates the mixing is done mechanically via linkages. In modern fly-by-wire systems (Saffarian and Fahimi, 2008), the mixing is done by a mixing software algorithm that mimics the mechanical links and sends the electrical signal to the actuators as shown in Figure 2.5.

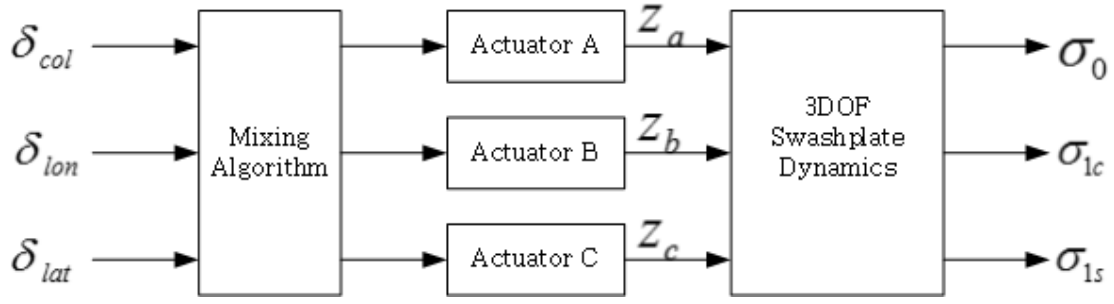


Figure 2.5: Algorithmic swashplate mixing.

The geometry to translate between the actuators, swashplate and the blades is shown in Figure 2.6 (Wang et al., 2020). The angle χ is used to off-centre the actuators so that actuator A is not aligned with the $x - axis$ of the rotorcraft. For this investigation the actuator are placed at 120^o from each other. This leads to the following swashplate equations:

$$\begin{aligned} z &= \frac{1}{3}\left(z_a + z_b + z_c\right), \\ \theta &= \frac{1}{3}\left(-z_a 2\cos(\chi) + z_b(\cos(\chi) + \sqrt{3}\sin(\chi)) + z_c(\cos(\chi) + \sqrt{3}\sin(\chi))\right), \\ \phi &= \frac{1}{3}\left(z_a 2\sin(\chi) - z_b(\sin(\chi) + \sqrt{3}\cos(\chi)) - z_c(\sin(\chi) - \sqrt{3}\cos(\chi))\right), \end{aligned} \tag{2.63}$$

where χ is a design attribute that can be chosen freely. For simplicity without loss of generality, χ may be chosen as 0^o, however, this restricts the motion of the swashplate should one actuator experience a fault or failure (Enns and Si, 2003).

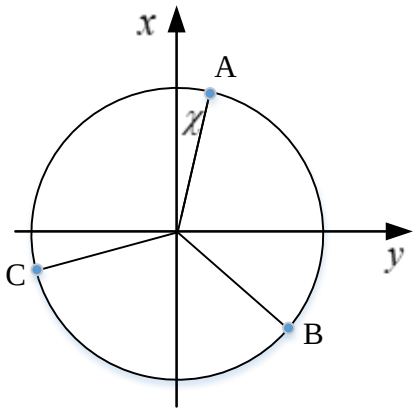


Figure 2.6: Swashplate geometry with A, B and C are as the positions of the actuators.

The desired swashplate translation (z) and orientation (ϕ and θ) are found from the desired actuator translations. In order to translate from the individual actuator displacements z_a, z_b and z_c, to the swashplate height and orientation, the following equations is used:

$$\begin{bmatrix} z \\ \theta \\ \phi \end{bmatrix} = \begin{bmatrix} \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ -\frac{2}{3}\cos(\chi) & \frac{1}{3}\cos(\chi) - \frac{\sqrt{3}}{3}\sin(\chi) & \frac{1}{3}\cos(\chi) + \frac{\sqrt{3}}{3}\sin(\chi) \\ \frac{2}{3}\sin(\chi) & -\frac{1}{3}\sin(\chi) - \frac{\sqrt{3}}{3}\cos(\chi) & -\frac{1}{3}\sin(\chi) + \frac{\sqrt{3}}{3}\cos(\chi) \end{bmatrix} \cdot \begin{bmatrix} z_a \\ z_b \\ z_c \end{bmatrix}. \tag{2.64}$$

The mixing algorithm employs the inverse of Equation 2.64 to find the desired actuator position to effect the desired swashplate height and orientation. However, the flight controller will only send the desired height to the mixing algorithm. These signals must be sent to the individual actuators as follows:

$$\begin{bmatrix} z_a \\ z_b \\ z_c \end{bmatrix} = \begin{bmatrix} 1 & -\cos(\chi) & \sin(\chi) \\ 1 & -\frac{\sqrt{3}}{2}\sin(\chi) + \frac{1}{2}\cos(\chi) & -\frac{\sqrt{3}}{2}\cos(\chi) - \frac{1}{2}\sin(\chi)) \\ 1 & \frac{\sqrt{3}}{2}\sin(\chi) + \frac{1}{2}\cos(\chi) & \frac{\sqrt{3}}{2}\cos(\chi) - \frac{1}{2}\sin(\chi) \end{bmatrix} \cdot \begin{bmatrix} z \\ \theta \\ \phi \end{bmatrix}. \tag{2.65}$$

When the flight controller detects that the desired height and orientation cannot be achieved, it is flagged as a fault in one of the actuators.

2.7.2 Electromechanical actuators-based rotorcraft

Electromechanical actuators (EMA) are used in the development of the medium-scale rotorcraft platform. The EMA have recently become popular in the aviation industry due to the fact that they are lightweight; they are fast and precise; and they do not require high maintenance when compared to hydraulic actuators.

An armature-controlled direct current (ACDC) motor electromechanical actuator is under consideration. This ACDC motor converts electrical energy into mechanical rotational motion which drives a nut-screw which in turn converts the rotation of the shaft into translational motion, as shown in Figure 2.7.

Both the electrical and the mechanical sides are approximated by linear differential equations. The electrical side of the ACDC motor system is given by this equation:

$$v_a(t) = L\frac{di(t)}{dt} + Ri(t) + e(t), \tag{2.66}$$

where $e(t)$ is the electromotive force (emf) due to the rotating shaft, which can be calculated as follows:

$$e(t) = K_e\dot{\theta}_e, \tag{2.67}$$

where K_e is the is the motor's emf constant. The torque produced by the armature current is found by:

$$T(t) = K_t i(t), \tag{2.68}$$

where K_t is the torque constant.

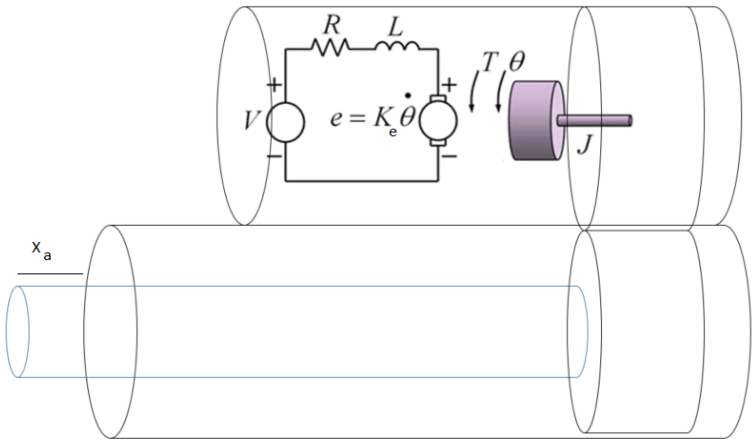


Figure 2.7: The standard components of an ACDC motor-based linear electromechanical actuator.

Referring to the mechanical side, the ACDC motor generates shaft rotation motion. However, only actuator shaft translational movement is of interest for controlling of the swashplate. Since the motor is rotating at thousands of radians per second, a reduction gear to reduce the motor speed at the output shaft is implemented as follows:

$$\theta_n = \theta_e \frac{R_1}{R_2} = \theta_e N_r, \tag{2.69}$$

where θ_e is the motor angular displacement and θ_n is the angular displacement at the nut-screw mechanism. The ratio of the motor to shaft gears R_1/R_2 is written as N_r. The nut-screw converts motor shaft angular displacement into actuator shaft translational movement as follows:

$$z_i = \theta_{ni} \frac{pitch}{rev} = \theta_{ni} K_r, \tag{2.70}$$

where z_i is the actuator translational travel, $i = \{a, b, c\}$ the subscript for the three individual actuators, and K_r is the pitch nut-screw (in mm/radians).

The equivalent mass due to motor's moment of inertia and the actuator rotating parts when translated to the translational side become $m_e = J(N_r K_r)^2$. This equivalent mass, plus the mass of the shafts and the swashplate disc are included as part of the translational side of the equation. The motor force developed and transmitted to the shaft is $F_i = T_i \frac{N_r}{R_s}$, where R_s is the actuator shaft's radius.

2.7.2.1 Electromechanical actuator dynamics

The three DOFs of the swashplate are controlled via the switching algorithm that sets the desired position of each EMA. Then the controller determines the input voltage required to achieved the desired actuator position. Then the input voltages, $\boldsymbol{u} = \{u_a, u_b, u_c\}$, are sent to the three EMAs resulting in the control of the vertical displacement, and roll and pitch of the swashplate.

The state-space model of the EMA swashplate system is given in the affine-in-control form as follows:

$$\begin{aligned} \dot{\boldsymbol{x}} &= \boldsymbol{f}(\boldsymbol{x}) + \boldsymbol{g}(\boldsymbol{x})\boldsymbol{u}, \\ \boldsymbol{y} &= \boldsymbol{h}(\boldsymbol{x}). \end{aligned} \tag{2.71}$$

The state vector is:

$$\begin{aligned} \boldsymbol{x} = &\ [x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9]^T, \\ = &\ [z, \theta, \phi, \dot{z}, \dot{\theta}, \dot{\phi}, i_a, i_b, i_c]^T. \end{aligned} \tag{2.72}$$

The system input vector is:

$$\boldsymbol{u} = [u_a, u_b, u_c]^T = [v_a, v_b, v_c]^T. \tag{2.73}$$

The system matrix is:

$$\boldsymbol{f}(\mathbf{x}) = [f_1, f_2, f_3, f_4, f_5, f_6, f_7, f_8, f_9]^T, \tag{2.74}$$

$$f_1 = x_4,\ f_2 = x_5,\ f_3 = x_6, \tag{2.75}$$

$$f_4 = \frac{K_t N_r}{M.R_s}(x_7 + x_8 + x_9) - Mg, \tag{2.76}$$

$$f_5 = \frac{1}{I_{yy}}\frac{K_t N_r}{R_s}(l_f x_7 - l_r(x_8 + x_9))\cos x_2, \tag{2.77}$$

$$f_6 = \frac{1}{I_{xx}}\frac{K_t N_r l_{rr}}{R_s}(x_8 - x_9)\cos x_3, \tag{2.78}$$

$$f_7 = -\frac{R}{L}x_7 - \frac{K_e N_r}{L K_r}(x_4 - l_f \cos x_2 x_5), \tag{2.79}$$

$$f_8 = -\frac{R}{L}x_8 - \frac{K_e N_r}{L K_r}(x_4 + l_r x_5 \cos x_2 - l_{rr} x_6 \cos x_3), \tag{2.80}$$

$$f_9 = -\frac{R}{L}x_9 - \frac{K_e N_r}{L K_r}(x_4 + l_r x_5 \cos x_2 + l_{rr} x_6 \cos x_3). \tag{2.81}$$

The control input matrix is:

$$g(x) = \begin{bmatrix} 0 & \dots & 1/L & 0 & 0 \\ 0 & \dots & 0 & 1/L & 0 \\ 0 & \dots & 0 & 0 & 1/L \end{bmatrix}^T, \tag{2.82}$$

and the output is:

$$\boldsymbol{h}(\boldsymbol{x}) = [x_1, x_2, x_3]^T. \tag{2.83}$$

The dynamics of the EMA swashplate are modelled for simulation in MATLAB®/Simulink® environment. Table 2.1 lists values of the EMA swashplate parameters used in this thesis.

Table 2.1: The system parameters of the complete EMA swashplate system.

Parameters	Value
Swashpate Mass, M_s	20 kg
Swashplate Radius, l_f	100×10^{-3} m
Moment of inertia, I_x, I_y	30.1×10^{-3} kg/m^2
Gear Reduction, N_r	1 : 20
Mass Moment of inertia of EMA, J_m	3.0×10^{-4} kg/m^2
Motor torque constant, K_t	0.04 Nm/A
Motor emf constant, K_e	0.04 Vs/rad
Motor Inductance, L_e	0.05 H
Motor Resistance, R_e	0.61 Ω
Nut-screw Radius, R_r	10×10^{-3} m
Nut-screw Pitch, K_r	3×10^{-3} m/rev

2.8 Rotorcraft Modelling Parameters

The rotorcraft subsystem discuss up this point are collected into a single model for simulation in MATLAB®/Simulink®. Table 2.2 gives values of the rotorcraft systems' parameters used in this investigation.

Table 2.2: The parameters used in the rotorcraft modelling.

Parameters	Description	Value
Main rotor		
R_M	Main rotor radius	3 m
c_M	Main rotor blade cord length	0.231 m
b	Number of rotor blades	5
I_b	Moment of inertia of the rotor blade	5 kgm^2
$a_M(C_{L_\alpha})$	Main rotor lift curve-slope	7.06 $/rad$
θ_{tw}	Blades twist	-0.21 rad
N	Number of blades	5
Ω	Main rotor angular speed	59.29 rad
h_M	Main hub position above CG	0.928 m
Tail rotor		
R_T	Tail rotor radius	0.587 m
c_T	Tail rotor blade cord	0.077 m
$a_T(C_{L_{\alpha T}})$	Tail rotor lift-curve slope	5.6 $/rad$
Ω_T	Tail rotor angular speed	345.54 rad
l_T	Tail rotor hub's x distance behind the CG	3.628 m
h_T	Tail height above the CG	0.7 m
Fuselage		
m	Rotorcraft mass	630 kg
I_{xx}	Moment of inertia about x	116 kgm^2
I_{yy}	Moment of inertia about y	654 kgm^2
I_{zz}	Moment of inertia about z	624 kgm^2
I_{xz}	Product of inertia in xz	8.13 kgm^2
h_H	Tail horizontal stabiliser from CG	0.7 m
h_V	Tail vertical stabiliser from CG	3.628 m

2.9 Flight Control Performance Specifications and Scope

The FCS developed must complement the operational concept of the RUAV. The FCS accepts references and steering commands from an external system i.e., human ground station pilot or an autonomous high-mode trajectory generation system on board the rotorcraft. This research is focused on autonomous flight control, however, trajectory generation is outside the scope of this work.

Other supporting systems that are required for the successful execution of the FCS but are outside the scope of this research are: air data system (ADS), the attitude and heading reference system (AHRS), radio altimeter (Radalt), and global position system (GPS) units. These systems are required to supply the information used to form a state feedback control system. For the rest of this study, these data sources are assumed present and correct. Faults in the sensors and the associated data loss correction are also outside the scope of this research project.

An additional fifth control input is used in case of failure in one of the other traditional four control inputs. These control inputs are used together in an integrated flight-propulsion control strategy. The actuators are:

1. Three main rotor control inputs that are mixed via the swashplate to control the collective pitch, longitudinal and lateral cyclic pitch;
2. One tail rotor actuator to control the tail rotor collective pitch; and
3. One engine actuator to control the motor engine rotation speed.

The intelligent FTC strategy is tested based on the following flight regimes:

- Hover out of ground effect;
- Cruise at two determined speeds;
- Trajectory following a preplanned flight path.

The investigation is limited to hover and slow forward flight with the speed of up to 23 m/s. As a result, the lateral acceleration is only based on the rated of change of lateral velocity and not the turn rate. The ADS-33 distinguished between predicted

handling qualities and assigned handling qualities. The former is a quantitative measure of the rotorcraft's dynamic response, while the latter is a qualitative feel by the pilot. Since this study is restricted to RUAV only the predicted HQs are analysed.

Also, the rotorcraft is tested when faults are present in the main rotor actuators. The failure of the tail rotor actuator, even though it is more severe compared to main rotor actuator failure, it is not covered in this investigation. Integrated flight-propulsion assists in redundancy when one of the three actuators fail.

Validation is used to determine whether the derived model simulates the real rotorcraft behaviour. Fidelity is judged by comparing real experimental data to the model's simulated response when similar inputs are applied (Padfield, 2008).

Two types of fidelity are discussed: functional and physical.

1. Functional fidelity: is the model compliant to design requirements such as predicting flying qualities?
2. Physical fidelity: does the model capture the underlying physics correctly?

In the next chapter, the rotorcraft trim analysis is presented. This includes the stabilisation of the system using PID controllers. Computational intelligence optimisation techniques are used to tune the PID controllers. The inclusion of a stabilisation controller is to enable the study of intelligent FTC strategies that follow, which are the main subject of this research project.

2.10 Chapter Summary

This chapter presented the mathematical model of the rotorcraft UAV used for FTC system development. The dynamics and kinematics of the swashplate are also presented for electromechanical actuators. A model of the electrical motor is also presented. From the fully developed model, it is interesting to note that the rotor flapping angles settling time is inversely proportional to the rotor speed. This increases coupling if the rotor speed is to be used as a control input. In addition, the chapter presented the model in a non-affine-in-control format and this representation is not suitable for FTC development to follow. The performance specifications to be met by the flight control strategy to be developed and the research scope were outlined.

Chapter 3

Flight Control Design using PID Controller and Computational Intelligence Optimisation Techniques

This chapter presents the design and application of the PID-based flight controller for the rotorcraft system model described in the previous chapter. Since the presented rotorcraft system is MIMO and non-affine-in-control, the selection and tuning of PID controller gains is not a trivial task. For this reason, we employ the use of computational intelligence optimisation techniques to obtain the gains that are close to optimal as possible for this FCS The computational intelligence optimisation algorithms are presented in two groups: ant-based and flight-based algorithms. These algorithms are also compared to the genetic algorithm and the particle swarm optimisation in order to benchmark their performance. The results show that the PID controller is able to control the rotorcraft effectively around selected trim points. This was demonstrated in numerical simulations for regulation hover, steady forward flight, tracking a small amplitude circle and rejection of moderate wind gust disturbances of up to $5\mathrm{m}/s$*. Also the optimised gains proved to perform better than the manually-selected ones. An actuator failure in the swashplate was demonstrated leading up to fault tolerance.*

3.1 Introduction

In this chapter, we start to tackle the rotorcraft problem by providing a flight control solution. As a benchmarking chapter, the methods presented here should be at the same level as those found in contemporary literature and forms the baseline to which the upcoming chapters' advanced control strategies are compared.

Firstly, let us review the conclusions from the previous chapter: that rotorcraft are difficult to control owing to the following:

1. Highly nonlinear dynamic systems;
2. Strong degree of inter-axis coupling; and
3. Underactuated making them highly unstable.

A PID controller for a rotorcraft is developed in this chapter. Even though a PID controller does not handle nonlinearities well (unless gain scheduling is used), it has several advantages for a good departure point for comparing the advanced controllers that are discussed in the sequel. In addition, since a rotorcraft is inherently unstable, a rudimentary stable controller is required in order to safely conduct experiment on the platform without loss of control, such as data collection for system identification. The proposed flight controller is modelled for the following scenarios:

- To stabilise the rotorcraft about a selected trim point. This is a *regulation problem.*
- To track the a desired flight plan. This is a *tracking problem.*
- To recover from imposed disturbances on the attitude. This is a *disturbance rejection problem.*
- To evaluate the system's resilience to parameter changes and the injection of an actuator fault. This is a *fault tolerance problem.*

The PID controller designs are based on the rotorcraft model developed in Chapter 2. The adopted model presented in Equation 2.46 is repeated here for quick reference and completeness. This system representation is expressed in concise form of the rotorcraft

dynamics as given by the following:

$$\dot{\mathbf{v}}^B = \frac{1}{m}\mathbf{F} - \boldsymbol{\omega}^B \times \mathbf{v}^B,$$
$$I\dot{\boldsymbol{\omega}^B} = -\boldsymbol{\omega}^B \times (I\boldsymbol{\omega}^B) + \mathbf{M},$$
$$\dot{\boldsymbol{\xi}} = \mathbf{R}\mathbf{v}^B,$$
$$\dot{\boldsymbol{\eta}} = \mathbf{T}\boldsymbol{\omega}^B.$$

The PID controller is developed for the rotorcraft at two trim points: hover and 10 m/s forward speed. Trim analysis of these points is treated next.

3.2 Rotorcraft Trim Analysis

In order to conduct experiments on a rotorcraft, it is necessary to perform trim analysis which helps in defining the operating condition where the aircraft is deemed stable for a given set of inputs. This is the point around which the PID controllers are designed. Trim is concerned with making sure that the rotorcraft is able to maintain some equilibrium state given a precisely chosen set of inputs in a given operational environment (Padfield, 2008).

What gives rotorcraft a supreme advantage over other types of aircraft is its ability for vertical take-off and landing, and hovering. Hovering is one of the most challenging of the rotorcraft manoeuvres. In hover, the flight controller's objective is to position the aircraft in a specific position x, y and z above the ground. It requires precise selection of forces and torques inputs to maintain a steady equilibrium. Selecting these inputs is an optimisation problem, hence optimisation techniques are used to find the candidate solution for inputs. The trim states are found by solving for the control input vector $\mathbf{u}$ in the rotorcraft dynamic equation that result in zero state rate; which means finding the zeros of the following equation:

$$\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}) = \mathbf{0}. \tag{3.1}$$

In an optimisation problem formulation Equation 3.1 can be expressed as:

$$\boldsymbol{f} : \boldsymbol{u} \rightarrow \mathbf{0}. \tag{3.2}$$

This is a strict condition for trim known as uniformly asymptotic stable. However in real rotorcraft the sufficient condition for equilibrium is uniform ultimately bounded stability

in a small region around round trim defined by $\epsilon > 0$, so that the following condition is true:

$$\|\boldsymbol{x}_0\| < \delta(\boldsymbol{\epsilon}) \Rightarrow \|\boldsymbol{x}(t)\| < \boldsymbol{\epsilon} \text{ , for all } t \geq t_0. \tag{3.3}$$

This means that if the rotorcraft's initial state, $\boldsymbol{x}_0$, is in a small region $\delta(\boldsymbol{\epsilon})$ at time $t = t_0$, any time after that it is found to be in the region $\boldsymbol{\epsilon}$, that is, it is bounded.

A number of trim analysis tools have been proposed. Dai et al. (2008) used a hybrid GA to find the trim state of a helicopter. This method is hybrid because it combined quasi-Newton with GA. The hybrid proved to be superior to both individual algorithms in terms of convergence time and minimising the objective function. Wang et al. (2019) used a similar hybrid GA to trim a helicopter in a coordinated turn. Gümüşboğa and İftar (2019) showed that PSO can trim an F-16 model successfully for level flight and coordinated turn. Demirel (2019) investigated the application of Nelder-Mead (NM) simplex to find the trim solution of the GENHEL-based UH-60 helicopter. This method was chosen simply because it is less computationally expensive when compared to advanced methods such as GA. The MATLAB *fminsearch* algorithm is based on this method and is generally used for trim (Lagarias et al., 1998).

3.2.1 Rotorcraft in hover

The analysis presented in this thesis begins with the trimming of the rotorcraft model in hover, 10 m above the ground. The position above ground was chosen in order to avoid ground-effects that take place when the rotorcraft is less than the diameter of the main rotor above the ground. The second step is to trim at steady low speed forward flight of 10 m/s. In hover trim analysis, the condition is characterised by zero translation velocity, $\mathbf{v}^I = [0\ 0\ 0]^T$, and zero angular rates, $\boldsymbol{\omega}^B = [0\ 0\ 0]^T$. A set of inputs in main rotor collective and cyclic and tail rotor collective setting is required to achieve this equilibrium state. This is achieved by iterative means. The algorithm for hover trim analysis is given as follows.

Algorithm 1: Rotorcraft hover trim analysis.

Initialisation

while *Trim condition not met* **do**

1. Choose the collective setting for the thrust to support the rotorcraft weight.
2. Choose tail rotor thrust to counteracts the generated main rotor toque .
3. Choose lateral cyclic to tilt the main rotor to counteract the tail rotor side force.
4. Choose longitudinal cyclic to tilt the main rotor backwards to counteract the tail rotor upthrust due to fuselage side tilt.

end

Save File ← trim data

The steady-state values for rotorcraft attainment and maintenance of hover trim condition are shown in Table 3.1.

Table 3.1: Parameters for rotorcraft maintenance of hover trim condition.

Parameters	Value	Parameters	Value
T_M	$= 5423.2\ N$	C_{TM}	$= 0.0083$
λ_{M0}	$= 0.0501$	v_{iM}	$= 8.905\ m/s$
Q_M	$= 1252.62\ N.m$	C_{QM}	$= 0.00038$
δ_{0M}	$= 0.109\ rad$	ϕ	$= 0.062$
T_T	$= 379.61\ N$	C_{Tt}	$= 0.011$
λ_{T0}	$= 0.0446$	v_{iT}	$= 11.405\ m/s$
Q_T	$= 11.59\ N.m$	C_{QT}	$= 0.00035$
δ_{0T}	$= 0.115\ rad$		
a_{1c}	$= 0.0019\ rad$	b_{1s}	$= 0.0001\ rad$

Figures 3.1 and 3.2 show the displacement and Euler angles responses of the rotorcraft to the calculated hover trim values. The rotorcraft is in the vicinity of the desired state with the oscillation in the order of $10^{-4}\ m$. Which is insignificant when compared to the size of the rotorcraft and the resolution of the distance sensor used (DGPS resolution is about 200 mm). In Figure 3.2 the Euler angles are also shown to be relatively low. The exception is made for the roll angle ϕ, which is required to produce a force to balance the tail rotor force. The inflow ratio, $\lambda_{M0} = 0.0501$, is also consistent with the result predicted by momentum theory to be between 0.05 and 0.07 (Johnson, 2013).

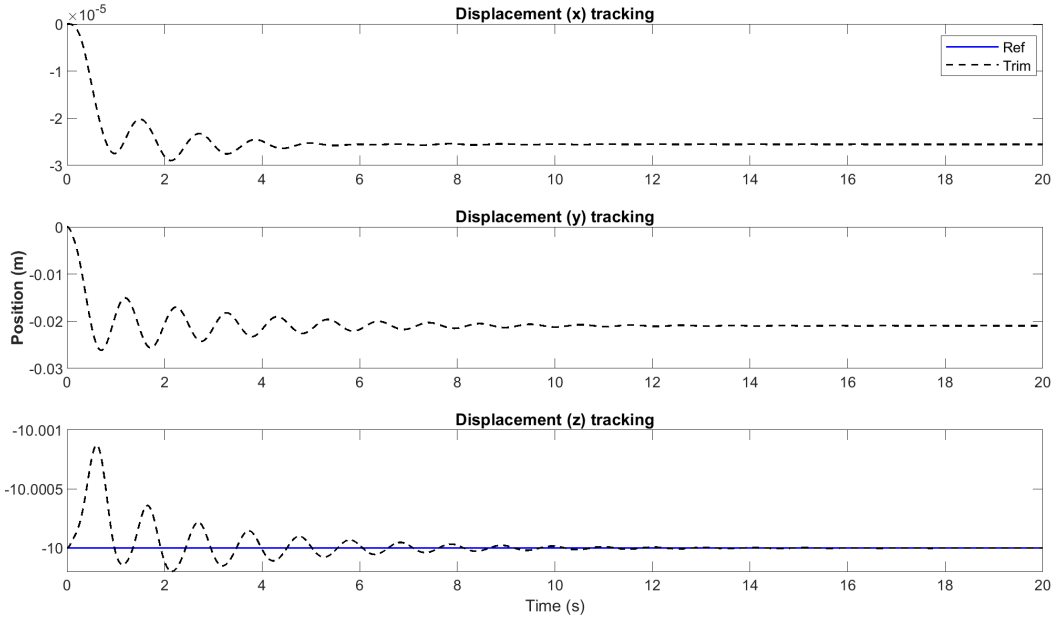


Figure 3.1: The rotorcraft position responses to selected hover trim condition.

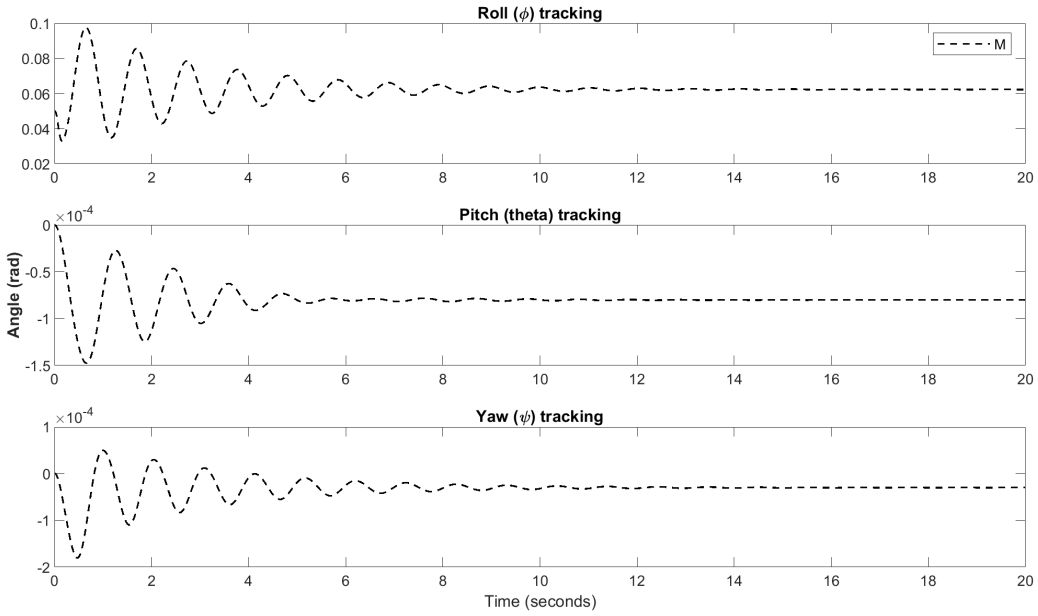


Figure 3.2: The rotorcraft Euler angles' responses to selected hover trim condition.

3.2.2 Rotorcraft in forward flight

A new flight trim condition was developed for the rotorcraft. This happens at low steady forward speed of 10 m/s. The condition is specified by 10 m/s longitudinal translation velocity, zero translation velocities elsewhere, $\mathbf{v}^I = [10\ 0\ 0]^T$, and zero angular rates, $\boldsymbol{\omega}^B = [0\ 0\ 0]^T$. The algorithm for forward trim analysis is given as follows:

Algorithm 2: Rotorcraft 10 m/s forward flight trim analysis.

Initialisation

while *Trim condition not met* **do**

1. Choose longitudinal cyclic to produce the pitch angle required to create the force that will balance 10 m/s drag;
2. Choose the collective setting for the thrust to support the rotorcraft weight and the force required in 1 above.
3. Choose the tail collective setting for the tail rotor thrust required to counteract the main rotor toque produced.
4. Choose lateral cyclic to tilt the main rotor to counteract the tail rotor side force.
5. Choose longitudinal cyclic to tilt the main rotor backwards to counteract the tail rotor upthrust due to fuselage side tilt.

end

Save File ← trim data

The steady-state values for rotorcraft attainment and maintenance of forward flight trim condition are shown in Table 3.2.

Table 3.2: Parameters for rotorcraft maintenance of forward flight of 10 m/s trim condition.

Parameters	Value	Parameters	Value
T_M	$= 5394.2\ N$	C_{TM}	$= 0.0049$
λ_0	$= 0.0484$	v_{iM}	$= 8.603\ m/s$
Q_M	$= 1223.7\ N.m$	C_{QM}	$= 0.00037$
δ_{0M}	$= 0.107\ rad$	ϕ	$= 0.060$
T_T	$= 306.24\ N$	C_{TT}	$= 0.0056$
λ_{0T}	$= 0.0446$	v_{iT}	$= 11.115\ m/s$
Q_T	$= 11.59\ N.m$	C_{QT}	$= 0.00035$
δ_{0T}	$= 0.2269\ rad$		
a_{1c}	$= 0.0016\ rad$	b_{1s}	$= 0.0001\ rad$

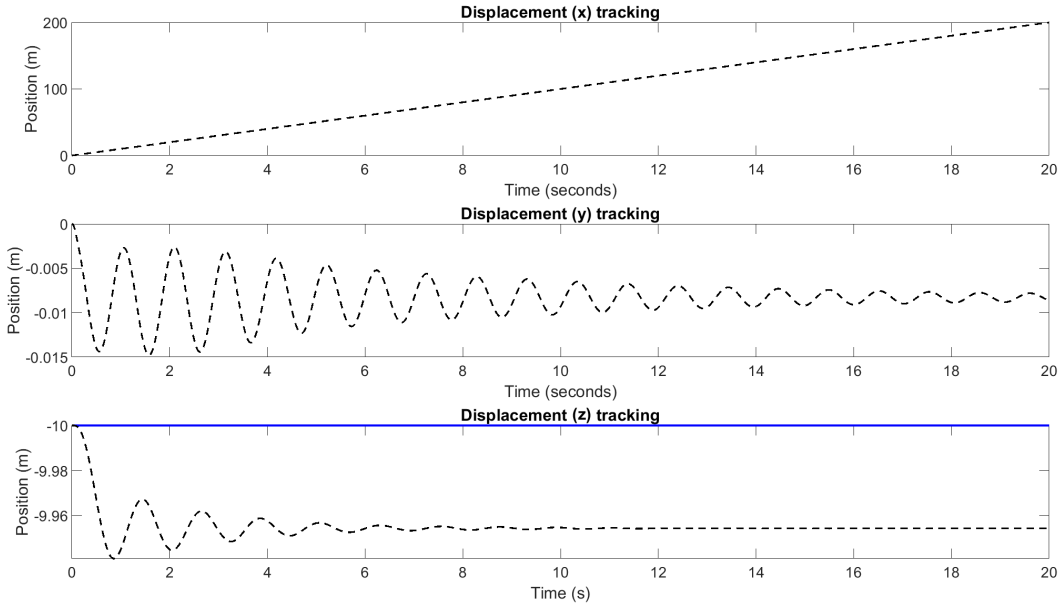


Figure 3.3: The rotorcraft position response to selected forward flight trim condition.

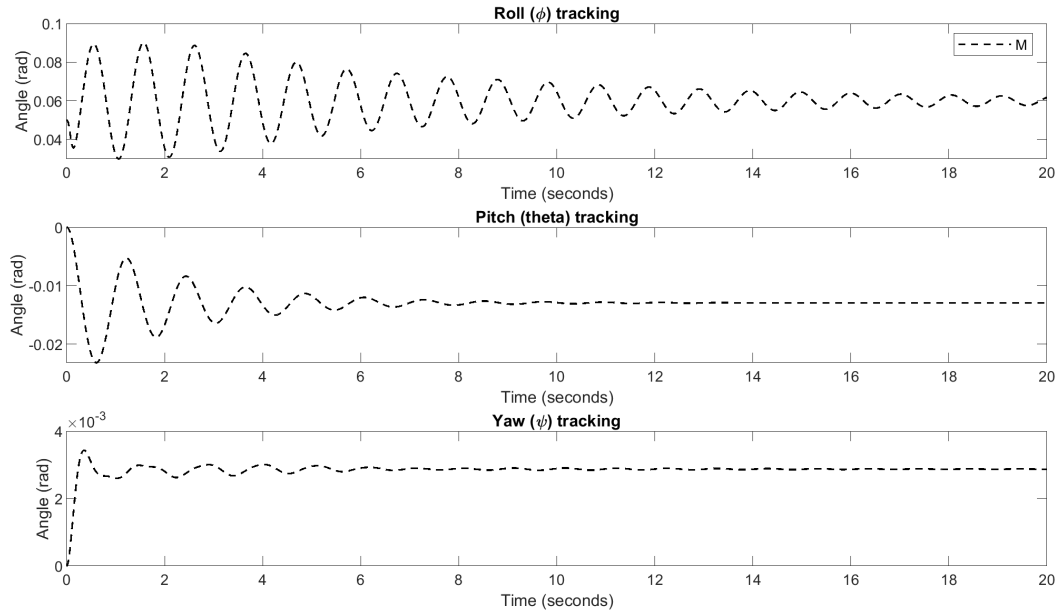


Figure 3.4: The rotorcraft Euler angles' responses to selected forward flight trim condition.

Figures 3.3 and 3.4 illustrate the response of the rotorcraft in forward flight with the trim values in Table 3.2. The rotorcraft is able to hold a stable forward flight as the x value continue to increase. The y channel has more oscillation than was observed in hover. There is also an error of 60 mm in the $z-$direction. Since the vertical tail contributes to anti-torque in forward flight, the tail rotor thrust has reduced and so has the roll angle.

3.3 Proportional-Integral-Derivative (PID) Controller

To ensure a stable response for the rotorcraft at the selected trim conditions and an ability to reject disturbances such as parameter variations and wind gust, an active closed-loop controllers are used. In this chapter, the controller of choice is PID. The PID controller has been in use for close to a 100 years. It is claimed that 90% of all automation and control systems are based on PID controllers (O'dwyer, 2009). It is easy to apply and tune PID controller even in the absence of a system model. The PID controller is made up of sum of three terms: the proportional term, the integral term and the derivative term. The simplified representation of a PID controller is derived in the time-domain as follows (Pedro et al., 2016):

$$u_i = K_p e_i(t) + K_i \int e_i(t) d\tau + K_d \frac{d}{dt} e_i(t), \tag{3.4}$$

$$e_i = y_{di} - y_i, \tag{3.5}$$

where u_i, e_i, K_p, K_i and K_d are the control variable, the error signal that has to be driven to zero, the proportional, the integral and the derivative gains respectively. The subscript i indicates that there could be more than one PID controller in the system of interest. As it is, in fact, six PID controllers in the case for rotorcraft with six-DOF. Despite the long standing history and popularity, (Van Overschee and De Moor, 2000) reported that 80% of PID controllers currently in operation are badly tuned. As discussed in the previous section, the model of the RUAV is too complex for an analytical solution. When combined with PID controller, it is solved by means of numerical methods in the MATLAB® / Simulink® environment. Because there is no analytical solution for this model, to get the correct PID controller gains, the popular method of Ziegler-Nichols tuning is used.

1. Adjust the proportional parameter to influence the rise time;
2. Adjust the derivative gain to reduce the overshoot; and
3. Then adjust the integral gain to reduce the steady-state error.

PID has been applied extensively to the control of rotorcraft (Mpanza and Pedro, 2021). Recently, there has been an increase towards hybrids of controllers to take advantage of their positives and eliminate the negatives. Sanchez et al. (2007) combined the PID

controller with a FLC for stabilising the inner loop and tracking in the outer loop. The authors showed controller effectiveness in simulations. Godbolt et al. (2013) implemented a model-based PID controller in an experiment of a helicopter UAV. This work experimentally validated the applicability of the two-loops PID controllers for a RUAV.

The developed PID controller for the rotorcraft is analysed. Since there are four control inputs and six DOFs, the pitch and roll dynamics are coupled to the x and y translation dynamics. This allows us to develop PID controllers in the inner and outer loops for faster and slower dynamics respectively. Then the controller tuning process starts in the inner loop to guarantee attitude stability then followed by the outer loop for x and y tracking accuracy.

The position of the rotorcraft with reference to F_E is controlled by the outer loop PID controller. The transformation matrix, $\mathbf{R}$, converts the position reference signals, x_d, y_d and z_d, to control loops in F_B. Due to coupling, the output of the outer loop controllers output becomes the desired values of roll and pitch angles of the rotorcraft that are required to produce the desired positions as shown in Figure 3.5 (Sanchez et al., 2007).

3.3.1 Controller objectives and evaluation criteria

The control problems discussed in this chapter are for regulation, tracking and disturbance rejection. For the PID controller design, we present two control loops: one for tracking the position of the rotorcraft and others for disturbance rejection of the rotorcraft attitude.

The controller design must meet the following performance specifications to be deemed satisfactory (Gavrilets, 2003):

1. The controller must exhibit general stability;
2. The steady-state error must be less than 5%;
3. The percentage overshoot must be less than 5%;
4. It must track the following signal $z_d = 2.5\sin(2\pi ft)$;
5. The handling qualities of the helicopter must be within the limits recommended in (ADS-33E, 2000), i.e., Level 1.

The architecture of the system used for PID controller tuning is shown in Figure 3.5.

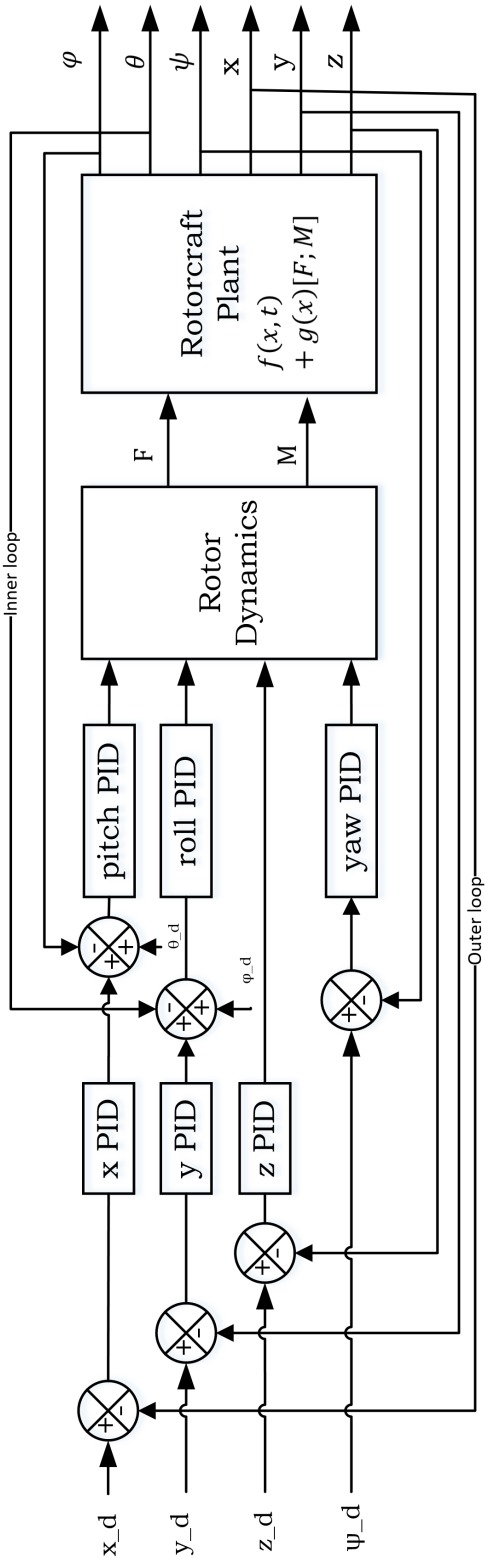


Figure 3.5: The control architecture of the rotorcraft using PID controller closed-loop system.

In order to develop a set of properly tuned PID controllers the following objective function is used:

$$J = \frac{1}{T}\int_0^T \Bigg[\left(\frac{x_d - x}{x_{max}}\right)^2 + \left(\frac{y_d - y}{y_{max}}\right)^2 + \left(\frac{z_d - z}{z_{max}}\right)^2 + \left(\frac{\theta_d - \theta}{\theta_{max}}\right)^2 + \left(\frac{\phi_d - \phi}{\phi_{max}}\right)^2 + \left(\frac{\psi_d - \psi}{\psi_{max}}\right)^2 + \left(\frac{\delta_{col}}{\delta_{col_{max}}}\right)^2 + \left(\frac{\delta_{lon}}{\delta_{lon_{max}}}\right)^2 + \left(\frac{\delta_{lat}}{\delta_{lat_{max}}}\right)^2 + \left(\frac{\delta_{ped}}{\delta_{ped_{max}}}\right)^2\Bigg] dt, \tag{3.6}$$

where x_d, y_d and z_d are the desired positions of the rotorcraft with respect to the Earth-fixed reference frame; θ_d, ϕ_d and ψ_d are the desired Euler angles of the rotorcraft; and δ_{col}, δ_{lon}, δ_{lat} and δ_{ped} are the collective, longitudinal cyclic, lateral cyclic and tail rotor collective inputs respectively. And $\delta_{col_{max}}$, $\delta_{lon_{max}}$, $\delta_{lat_{max}}$ and $\delta_{ped_{max}}$ are the maximum permissible actuator deflections. This is an exampler single-objective optimisation problem whose aim is to reduce the tracking error and while minimising the actuator effort applied.

The PID controller gains tuning procedure presented in this section has significant coverage in literature. As seen in Figure 3.5, the number of parameters is greatly increased and the coupling of the variables make the tuning process complex. The contribution contained in this chapter is based on the expansion of this procedure by including and comparing computational intelligence algorithms for tuning controller gains/parameters. These techniques are presented in the next section.

3.4 Introduction to Computational Intelligence Optimisation Techniques

Computational intelligence techniques are proposed for the optimal selection of controller parameters. Although this chapter is concerned with PID controllers, the optimisation techniques discussed here are also applicable to the nonlinear control strategies to be discussed in subsequent chapters. The optimisation techniques of interest here include: genetic algorithm (GA), particle swarm optimisation (PSO), ant colony optimisation (ACO), antlion optimisation (ALO), firefly algorithm (FA) and cuckoo search (CS) algorithm. The merits of these methods have been discussed in literature; however they have never been compared for controller gains tuning, especially in relation to rotorcraft control. This is

the focus of this thesis.

Optimisation is the act of choosing the best possible results within a given set of constraints. In order to understand how an optimisation algorithm operates it is crucial to understand the concept of an optimisation problem first. An optimisation problem satisfies Equation 3.7 (Yao et al., 2003; Socha and Dorigo, 2008).

$$P = (S, f)\,, \tag{3.7}$$

where $S \in \Re^n$ is the problem search space which is a set of all possible solutions and f is an objective function expressed as:

$$f : S \to \Re. \tag{3.8}$$

The function returns a real value that is used to measure the cost value of an individual solution s. The aim of the optimisation process is to discover $s \in S$, that results in the lowest cost value possible for the problem, i.e., finding $\forall s \in S : f(s^*) \leq f(s)$ in a reasonable amount of time.

A number of objective functions have been proposed in literature for optimally tuning controllers, such as the following (Solihin et al., 2011):

1. The Integral of Squared Error (ISE)

$$J_1 = \int_0^T e^2(t)dt. \tag{3.9}$$

2. The Integral of Absolute Error (IAE)

$$J_2 = \int_0^T |e(t)|\, dt. \tag{3.10}$$

3. The Integral of Time Multiply Squared Error (ITSE)

$$J_3 = \int_0^T te^2(t)dt. \tag{3.11}$$

4. The Integral of Time Multiply Absolute Error (IMAE)

$$J_4 = \int_0^T t\,|e(t)|\, dt. \tag{3.12}$$

There are three types of optimisation techniques found in the literature. They are defined as follows (Mpanza, 2012; Platt et al., 2018):

Definition 3.1. *Analytical Search* is a mathematically-based search algorithm. Because it is based in calculus, it is guaranteed to find a solution but this solution can be a local optimal solution.

Definition 3.2. *Blind search* is an exhaustive search approach that searches for every possible solution. This method is guaranteed to find a global optimal solution but it is computationally expensive and for large search spaces it is intractable.

Definition 3.3. *Heuristic search* is a guided search mechanism that is not guaranteed to find the optimal solution. This method, however, does find a satisfactory solution in a reasonable amount of time.

While the blind and analytical search techniques are attractive for simple optimisation problems, due to their exactness in optimal solutions, they fall short in large combinatorial optimisation problems. Heuristic search algorithms are able to approximate the solutions of hard optimisation problems(Mpanza, 2012), such as in controller gains optimisation problems and system identification. These problems do not have any minima or maxima analytical solutions. However, a near optimal solution for these problems are found and used successfully in practice (Okwu and Tartibu, 2020). Therefore, this thesis investigates the application of heuristic search algorithms to the problem of tuning controller gains. A subset belonging to the class of computational intelligence is of special interest in this thesis and the description of each algorithm is given in the sequel.

3.4.1 Genetic algorithm (GA)

GA is a heuristic population-based optimisation strategy that was developed to imitate the evolution of a population in a natural environment by natural selection. This is based on Darwin's theory "survival of the fittest." GA has been successfully used in complex and high-dimensional optimisation problems with solutions found in a relatively short time. The setup of GA is that the candidate solutions called *chromosomes*. The solution evolves over a number of prescribed generation. At each generation, new offspring chromosomes are generated through the reproduction of two parent chromosomes. The new chromosomes can also undergo mutation at a prescribed probability. Once all the chromosomes have been processed the process is repeated for a prescribed number of generations. This GA optimisation process is summarised in the Figure 3.6.

The process followed by the GA is: (i) creating a random initial population solution of *chromosomes*, (ii) each solution in chromosome is evaluated for fitness, (iii) the *fittest*

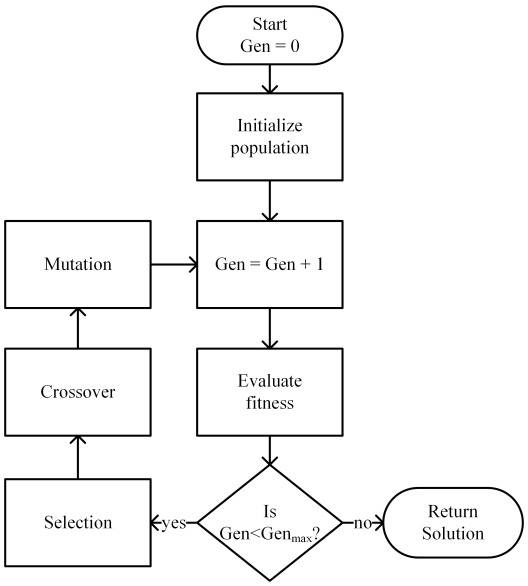


Figure 3.6: GA computation flowchart (Aly, 2011).

members of the population are allowed to reproduce, by *crossover*, for the next generation, (iv) the new chromosomes in the new population are subject to *mutation*. The process is repeated until a stopping condition is reached. This condition may be either that the solution was found or the number of specified generations have been reached. The latter case meaning a satisfactory solution was not reached.

The pseudocode for the GA optimisation processes is shown in Algorithm 3 (Bindu and Namboothiripad, 2012):

The best GA parameters for the present problem were found by numerical experiments and are listed in Table 3.3.

Unlike many optimisation procedure, GA has a high probability for converging to a global minimum solution. In (Phillips et al., 1996), the GA was used to discover rules for a fuzzy-based controller of a UH-1 helicopter with a need for pilot knowledge. Alkamachi and Erçelebi (2017) used GA to tune PID controllers for a high-speed racing quadrotor helicopter. This controller combination was able to reject disturbance of up to 30% weight change and injected Gaussian noise into the feedback loop. In (Mpanza and Pedro, 2019) a GA was used for SMC gain tuning.

As a tool for adaptive control, GA is used to find the fittest model representing the current system states and operating conditions. It is also useful in system identification where it is able to choose the best fitting parameters from a number of generations (Fabri and

Algorithm 3: GA optimisation.

1. Initialise new population with N chromosomes;
2. Initialise $iGen = 0$;

while $(iGen < maximum\ generation)||(stopping\ criteria)$ **do**

1. Update the number of generation $iGen = iGen + 1$.
   for *each chromosome* $i = 1, ..., N$ **do**
   // Evaluate the fitness of the chromosome;
   end
   // Generating off-springs;
2. *Selection*: Create the elite by selecting the chromosomes with high fitness value to participate in the reproduction.
3. *Crossover*: Random Roulette is used for pairing parent chromosomes and these are used to reproduce new chromosomes by crossover with a probability P_c.
4. *Mutation*: mutation is applied to a percentage, P_m, of the offspring to simulate adaptation.

end

Save File ← best solution

Table 3.3: The setup parameters for the GA algorithm.

Parameters	**Value**
Population size, N	500
Crossover point, P_c	0.8
Mutation probability, P_m	0.05
Maximum generation, Gen_{max}	100

Kadirkamanathan, 2001). However, the drawbacks of GA are the computational intensity and the indeterminate speed at which the solution is arrived at. This makes it a bottleneck in on-board real-time computation where resources are limited.

3.4.2 Particle swarm optimisation (PSO)

PSO is a multi-agent meta-heuristic optimisation algorithm that imitates the social behaviour of a group of animals acting as a single system. This behaviour is found in animals such as in flock of birds, school of fish, swarm of insects, etc. These animals are referred to as particles in a swarm. PSO is a random search that possesses the following characteristics:

- A single collection of individuals that is capable of accomplishing a difficult task;
- This task would normally be impossible for the individual to perform in the environment;
- The problem is distributed to the population making it easy to manage.

The idea of PSO was first proposed in 1995 by Eberhart and Kennedy and it has been very popular among heuristic evolutionary algorithms(Eberhart and Kennedy, 1995). There are similarities between PSO and GA in that they are both based on a population evolving the solution over a number of generations/iterations. However, while GA has evolution properties such as crossover and mutation, PSO does not.

Instead, the PSO algorithm uses a population of particles that are swarming through the search space with given individual velocities. Since each PSO particle at any iteration has velocity and position, the best solution is found by tracking the particle with the best solution cost to a specific position. The velocity of each particle is calculated by the following equation:

$$v_i(k+1) = mv_i(k) + c_1.r_1(pbest_i(k) - p_i(k)) + c_2.r_2.(gbest(k) - p_i(k)), \tag{3.13}$$

where v_i and p_i are the velocity and current position of the i-th particle respectively. *pbest* is the i-th particle's best solution so far, and *gbest* is the best solution in the global set of all the particles. The parameter m is the velocity gain used for controlling exploration versus exploitation trade-off. The parameters r_1 and r_2 are random numbers in a range

$[0, 1]$ sampled from a normal distribution. While the parameters c_1 and c_2 are positive acceleration constants (Engelbrecht, 2007).

The velocity of the particle allows it to move in the direction towards *gbest*. The next position of each particle is found by integrating velocity according to the following equation:

$$p_i(k+1) = p_i(k) + v_i(k+1) \tag{3.14}$$

In Equation 3.14, the particles are only updated to $p_i(k+1)$ if the fitness of the current position $f(p_i(k)) < f(p_i(k+1))$. The iteration number is incremented once all particles in the swarm have updated their positions (Engelbrecht, 2007).

Algorithm 4: Particle swarm optimisation.

1. Create and initialise an ns-dimensional swarm;
2. Initialise $k = 0$

while $(k < maxIteration)||(stopping\ criteria)$ **do**
  // Update iterations $k = k + 1$;
  for *each particle* $i = 1, ..., N$ **do**
    // Evaluate the fitness of the particle;
    if $f(p_i) < f(pbest_i)$ **then**
      // update the i-th particle's best position
      $pbest_i = p_i$
    end
    if $f(pbest_i) < f(gbest_i)$ **then**
      // update the global best position
      $gbest = pbest_i$;
    end
  end
  for *each particle* $i = 1, ..., ns$ **do**
    The velocity v_i is updated using Equation (3.13);
    The position p_i is updated using Equation (3.14);
  end
end
Save File ← best solution

In Algorithm 4, N is the number of particles, p_i and $pbest_i$ are the solutions of each particle and its best position so far, *gbest* is the global solution. The particle moves to its best possible position $pbest_i$ and then shares that information with the other particles Equation (3.13). After positions of every particle have been evaluated, the best overall position is designated as the global best position, *gbest*. This global best position is returned as the solution after reaching the stopping condition.

The parameters that are used in the setup of the PSO algorithm are shown in Table 3.4.

Table 3.4: The setup parameters for the PSO algorithm.

Parameters	**Value**
Number of particles, N	250
Crossover factor, F	0.5
Crossover probability, C_R	0.5
Particle inertia, m	0.9
Acceleration factors, c_1,c_2	$1, 2$
Maximum iteration, k	100

PSO has been popular in controller tuning given the number of findings reported in literature. Solihin et al. (2011) presented the result of PID controller tuning for the DC motor. The results were compared with Zeigler-Nichols tuning. Pedro et al. (2018) compared a passive vehicle suspension system with an PSO optimised active system and showed that the inclusion of PSO improved the vehicle handling and ride comfort. Moreover, the PSO was employed for tuning using different objective functions (Equations 3.9 3.10 3.11 and 3.12). Fan and Jen (2019) compared the traditional PSO with a newly developed enhanced partial search (EPS), i.e., a PSO with co-swarms that are able to share information between particles.

3.4.3 Ant colony optimisation (ACO)

ACO is also a population-based meta-heuristic optimisation method falling into the swarm intelligence category. Swarm intelligence aims to exploit the emergent collective intelligence of group of simple agents (Engelbrecht, 2007). It was first proposed by Dorigo (1992), after studying the natural behaviour of ants as they move in the environment searching for food.

The behaviour of the ACO algorithm is determined by selecting the number of concurrent ants, m. The i-th ant, $i = 1, \ldots, m$, represents a current solution of the optimisation problem being optimised. Given the current position and the pheromone trails of all the possible transition states, the ant selects, j, the new state move using a pheromone probability distribution function. The new ant solutions are evaluated and the pheromone trails are updated proportionally to the quality of the solutions. Thereafter, the process is repeated for each ant unit the maximum iteration number or stopping condition is reached.

The process outline above works for discrete combinatorial optimisation problems. However, the present problem of controller parameter tuning is optimisation over a continuous space. Socha and Dorigo (2008) developed $ACO_{\Re}$ which is ACO for $\Re$, a continuous domain. By comparison: the new $ACO_{\Re}$ has a continuous search space is $v_i \in D_i \subseteq \Re$, while ACO has a discrete countable search space $v_i \in D_i = \{v_i^1, ..., v_i^{|D_i|}\}$, where $|D_i|$ is the size of the search space. Also, unlike ACO, $ACO_{\Re}$ employs a probability density function (PDF) to find the next feasible solution, such that $P(s) \geq 0 \ \forall s$ and $\int_{-\infty}^{\infty} P(s) = 1$. For our purpose, a Gaussian PDF was selected for its ease of sampling.

Since each controller has more that one gain, a multi-variable optimisation problem is setup using a kernel, $G^i(s)$, that maps a sum of weighted Gaussian functions as follows:

$$G_i(s) = \sum_{l=1}^{k} \omega_l \frac{1}{\sigma_l^i \sqrt{2\pi}} e^{-\frac{(s-\mu_l^i)^2}{2(\sigma_l^i)^2}}, \tag{3.15}$$

where $i = 1, ..., n$ is variable index in a $n-$dimension search problem, μ_l^i and σ_l^i are mean and its standard deviation of the $l-th$ solution for each variable respectively. An archive table, T_{ij}, is used to store the current available solutions, μ and σ. The table is indexed by $l = 1, ..., k$, where k is the archive table size - an algorithm tuning parameter representing the size of available solutions at a given iteration. The first item in the table $s^l = \mu_l^1, ...\mu_l^n$. represents the best solution at a given iteration.

The $ACO_{\Re}$ pseudocode is shown in Algorithm 5.

The tuning parameters, q and ζ are for controlling exploration versus exploitation and pheromone evaporation rate respectively. The effect of these parameters on the algorithm performance are summarised in Table 3.5 (Socha and Dorigo, 2008).

The best ACO parameters for the present problem were found by numerical experimentation and are listed in Table 3.6.

There is not a lot of ACO application in literature for control gain tuning, let alone in aircraft flight control. Priyambodo et al. (2015) applied ACO to the tuning of PID gain for stabilising a quadrotor. The study investigated all four objective functions (Equations 3.9 3.10 3.11 and 3.12). Through numerical simulation, it was observed that ACO corrected the problems found in Zeigler-Nichols tuning. Boubertakh (2017) used ACO to tune a Fuzzy-PID controller for a 2-DOF helicopter supported on a gimble for azimuth and elevation tracking of a quadrotor. Using the ISE objective function, this method was shown to be superior to manual tuning in numerical simulations.

Algorithm 5: Continuous ant colony optimisation.

- Create and randomly initialise m ant ;
- Initialise an archive table of size k;
- Initialise $l = 0$;

while $(l < \textit{maximum iteration})||(\textit{stop criteria})$ **do**
// Update iterations $l = l + 1$;
for *each ant* $i = 1, ..., m$ **do**
Create new solution drawn from Equation (3.15)
Evaluate the solution's fitness $f(s_l)$;
end

1. Sort the solutions starting with the best fitness value $f(s_l)$;
2. Calculate probability weight ω_l for each solution using the following:

$$\omega_l = \frac{1}{qk\sqrt{2\pi}} e^{-\frac{(l-1)^2}{2q^2k^2}}.$$

3. Find σ_l^i for each candidate solution using the equation:

$$\sigma_l^i = \zeta \sum_{j=1}^{k} \frac{\left| s_j^i - s_l^i \right|}{k-1}.$$

4. Update the archive table T with μ and σ, of the best solutions up to k.

end
Save File ← best solution

Table 3.5: The ACO tuning parameters and their effect on the algorithm.

	High Value	Low Value
q	exploitation	exploration
ζ	fast forgetting, slow convergence	slow forgetting, fast convergence

Table 3.6: The setup parameters for the continuous ACO algorithm.

Parameters	**Value**
Number of ants, m	20
Number of archives, k	30
Forgetting constant, ζ	0.8
Pheromone constant, q	0.05
Maximum iteration, l	100

3.4.4 Antlion optimisation (ALO)

Similar to ACO, ALO is a meta-heuristic algorithm. It works to mimic the behavioural interaction, between antlions (predator) and ants (prey) in traps. It was first proposed by Mirjalili (2015). The following steps are involved in implementing the ALO: the ants move around randomly (seemingly random; as it is known that ants intelligently share environmental information with other ants using pheromones), the antlion hunt them by constructing pit traps; these traps capture ants, catching of prey, and the reconstruction of traps.

The motivation for the recent increase in using ALO is that it is gradient-free and provides high levels of exploration and exploitation. Exploration is by ants and antlion random walks. Exploitation is provided by reduction in the antlion boundary. The ALO algorithm is defined as a triple-tuple function $AOL(A_1, A_2, A_3)$ that estimates the best solution for the optimisation problems.

$$\mathbf{\Phi} \rightarrow_{A_1} \{G_{ant},\ G_{oa},\ G_{antlions},\ G_{oal}\}, \tag{3.16}$$

$$\{G_{ant},\ G_{antlion}\} \rightarrow_{A_2} \{G_{ant},\ G_{antlions}\},$$

$$\{G_{ant},\ G_{antlion}\} \rightarrow_{A_2} \{true,\ false\},$$

where G_{ant} is the ants' position matrix, $G_{antlion}$ is the antlions' gain matrix, G_{oa} contains the ants' fitness values and G_{oal} contains the antlions fitness values.

The ants and antlions are initialised randomly by applying a Roulette wheel to Equation 3.16. The ALO algorithm implements random walks as follows:

$$X^n(t) = [0, cumlsum(2l(t_1) - 1), \ldots, cumlsum(2l(t_n) - 1),] \tag{3.17}$$

where n is the maximum number of iterations for the algorithm run, $cumlsum(\cdot)$ is the cumulative sum and t is the instance of the random walk.

$$l(t) = \begin{cases} 1 & rand() > 0.5 \\ 0 & rand() \leq 0.5 \end{cases},$$

where $rand()$ generates a random number between $[0, 1]$ with a Gaussian distribution. The random walk are restricted to the boundary of the solution space Ω by normalisation as follows:

$$X_i^t = \frac{(X_i^t - a_i)(d_i - c_i^t)}{(d_i^t - a_i)} + c_i, \tag{3.18}$$

where a_i is the minimum random walk of the i-th ant, d_i is the maximum random walk of the i-th ant, c_i^t is the minimum of the i-th ant at the $t-th$ iteration and d_i^t is the maximum of the i-th ant at the t-th iteration (Mirjalili, 2015; Leke and Marwala, 2019).

Randomly walking ants fall into traps. The antlion traps ants which are modelled by the following:

$$c_i^t = Antlion_j^t + c^t, \tag{3.19}$$

$$d_i^t = Antlion_j^t + d^t, \tag{3.20}$$

where c^t is the lower bound of all the ants at the t-th iteration. d^t is the upper bound of all the ants at the t-th iteration and $Antlion_j^t$ is the location of the j-th antlion at the t-th iteration. These traps are proportional to the antlions' fitness values. Modelling how an ant trapped slides down the slope, the following applies:

$$c^t = \frac{c^t}{Z}, \tag{3.21}$$

$$d^t = \frac{d^t}{Z}, \tag{3.22}$$

where Z is a ratio specified as an algorithm parameter. $Z = 10^w \frac{t}{T}$ and w is chosen to adjust the trade-off between exploration and exploitation.

ALO also applies elitism by choosing the fittest antlion at iteration t. This allows the algorithm to maintain the best solution. Therefore, the position of every ant's random walk around a Roulette selected antlion and the elite antlion as follows (Mirjalili, 2015):

$$Ant_i^t = \frac{R_A^t + R_E^t}{2}, \tag{3.23}$$

where R_A^t is the random walk around a Roulette selected antlion and R_E^t is a random walk around the elite antlion.

The ALO pseudocode is shown in Algorithm 6,

In the setup of the ALO, the parameters that are used are shown in Table 3.7 (Leke and Marwala, 2019).

ALO literature is starting to gain popularity when it comes to controller optimisation. In (Al-Jodah et al., 2020), an ALO algorithm wa used for tuning a SMC for a 3-DOF micro-positioning mechanisms for trajectory tracking tasks in $XY\Theta$. Compared to manual tuning, ALO showed better tracking performance and this was shown in simulations and

Algorithm 6: AntLion Optimisation.

begin
  Create and randomly initialise n ants and antlions;
  Evaluate the fitness of ants and antlions;
  Apply elitism to the best antlions.
  while $(t < maxIteration)||(stopping\ criteria)$ **do**
    for *each ant* $i = 1, ..., n$ **do**
      Select an antlion using Roulette wheel;
      Update c and d using Equations 3.21 and 3.23;
      Create ants' random walks Equation 3.17 ;
      Normalise random walks using Equation 3.18;
      Update the position of the ants using Equation 3.23.
    end
    Evaluate ants' fitness;
    Replace antlion position with a fitter ant;
    Apply elitism to the best antlions.
  end
  Save File ← best solution: the elite antlion
end

Table 3.7: The setup parameters for the ALO algorithm.

Parameters	**Value**
Number of ant agents, n	20
Number of ants, k	50
Maximum iteration, t	100

experimental results. Oliveira et al. (2020) implemented ALO and compared it to another contemporary algorithm called the gray wolf optimiser (GWO) for tuning of SMC gains of a quadrotor helicopter. The two algorithms outperformed the PSO based on ISE objective functions for precise tracking.

3.4.5 Cuckoo search (CS) optimisation algorithm

CS is another meta-heuristic algorithm which derives its inspiration from the parasitic breeding behaviour of cuckoo birds. These birds are opportunistic in that they try to maximise the chance of survival of their off-springs without participating in incubating the eggs or feeding the hatchlings. Yang and Deb (2009) developed the cuckoo search optimisation algorithm.

A cuckoo bird lays its own eggs and conceals them among the eggs of other unsuspecting birds. This is done in the hope that the nest's owner will not notice the foreign eggs and will treat them as its own. Thereby relieving the cuckoo of its responsibilities. Occasionally, the cuckoo goes far as to cast out the other bird's eggs in order to increase its chicks survival rate. The cuckoo chick that hatches first also maximises its own chance of survival by disposing of the eggs in the nest. The chick that successfully survives is considered the best solution. The eggs laid by the cuckoo can be discovered by the host bird and discarded with a probability P_a. Yang and Deb (2009) notice that this aggressive breeding behaviour of cuckoo can be used as a search optimisation algorithm.

A number of nest k exists. There are fewer nest than there are birds, n. The access number of birds $k - n$ are cuckoo birds. The algorithm employs random walks for cuckoo to find suitable nests. The new solution for each cuckoo is found as follows Yang and Deb (2009):

$$x_i^{t+1} = x_i^t + L\acute{e}vy(\lambda), \tag{3.24}$$

where x_i is the position of the cuckoo, $L\acute{e}vy(\lambda)$ flight generates stochastic random walks from a $L\acute{e}vy = t^{-\lambda}$ distribution with the conditions $1 < \lambda < 3$. The nest has a high fitness value if the cuckoo egg deposited is better than the eggs that are already in the nest and the new offending eggs are not discovered.

The whole process summarising the cuckoo search algorithm is shown in Algorithm 7 (Yang and Deb, 2009).

Algorithm 7: Cuckoo search optimisation.

begin
  Initialise the population n and host nests $x_i : (i = 1, 2, 3...n)$
  Set the maximum search generations G
  while $(i < maxIteration)||(stop\ criteria)$ **do**
    1. Select a i-th cuckoo and generate a new solution by Levy flights
    2. Evaluated the solution's fitness F_i
    3. Randomly chose a nest among the n nests
    if $(F_i < F_j)$ **then**
      replace j-th solution with the i-th solution
    end
    Drop the solutions with the lowest fitness values.
  end
  Save File ← best solution
end

In the setup of the CS, the parameters that are used are shown in Table 3.8.

Table 3.8: The setup parameters for the CS algorithm.

Parameters	**Value**
Number of nests, m	20
Number of birds, n	30
Discovery rate, P_a	0.25
Maximum number of iterations, i	100

Jin et al. (2015) employed CS for PID tuning and compared the results to the classical Zeigler-Nichols tuning method. Three versions of CS are presented: traditional, improved and novel improved CS, all of which performed better than Ziegler Nichols method. In (El Gmili et al., 2019), the CS optimisation algorithm was compared to PSO for quadrotor PID controller tuning. The comparison here also applied the following objective functions (Equations 3.9 3.10 3.11 and 3.12). Even though the CS did not outperform the PSO, the hybrid PSO-CS outperform both individual algorithms.

3.4.6 Firefly algorithm (FA)

FA is one of the meta-heuristic algorithms that are inspired by naturally occurring phenomena. It is an example of swarm intelligence optimisation technique. In particular, the

algorithm models the flashing pattern and resulting behaviour of fireflies. The primary function of the firefly flashing pattern is to attract mating partners as well as to attract prey.

The algorithm was proposed by Yang and He (2013). The algorithm postulates that the intensity of bioluminescent light produced by a firefly can be a signal of fitness of the light source to other fireflies. That is, in a fixed population of fireflies, the brightest one signifies the best solution for the problem and the rest of them should move towards it for mating and resulting in progenies with better solutions. The rhythmic flashing and the rate of flash signals the fitness of the individual and therefore attracts appropriate mates. The FA is based on the following assumptions (Yang, 2013):

- Fireflies are unisex, i.e., all fireflies are attracted to all other fireflies regardless of their sex;
- The brighter the firefly the higher attractiveness. The brightness decreases with the increases in distance. Less bright firefly improve their apparent brightness by moving closer to brighter one;
- The fireflies are allowed to move randomly if there is a tie of brightness; and
- The firefly's brightness at a distance is a function of the environment.

To develop the optimisation problem, the brightness, I_i, of firefly i as seen by firefly j separated by a distance r_{ij} follows the square of distance law as given by:

$$I(r) = \frac{I_i}{r_{ij}^2}. \tag{3.25}$$

The brightness or attractiveness is associated with the objective function of the problem to be optimised. The attractiveness function is given by:

$$\beta(r) = \beta_0 e^{-\gamma r^2}, \tag{3.26}$$

where β_0 is the firefly brightness as seen at $r_{ij} = 0$ and γ is the coefficient of light absorption by the medium. The distance, r, between two fireflies x_i and x_j, is Cartesian distance $r_{ij} = \|x_i - x_j\|$ (Leke, 2017).

The movement of a firefly from its initial position, x_i, is either at random or towards the brightest firefly as determined by the following equation:

$$x_i(t+1) = x_i(t) + \beta_0 e^{-\gamma r^2}(x_i - x_j) + \alpha\epsilon_i, \tag{3.27}$$

where subscript i and j are the current and the brightest fireflies respectively. $\beta_0 e^{-\gamma r^2}(x_i - x_j)$ controls the movement of the firefly towards the brightest one. This means that the closer fireflies are attracted at a higher rate than farther ones. This portion of the movement represents the exploitation of the current best solution. The coefficient γ is used to control the exploitation rate of the algorithm. The higher the value of γ the less is the exploitation and encourages more exploration of the search space. The last term, $\alpha\epsilon_i$, is a randomisation parameter that encourages the exploration of the search space. The magnitude of exploration is governed by α, the exploration coefficient and ϵ_i is a new random solution drawn from a Gaussian distribution.

The algorithm is iterated according to the following pseudocode (Yang, 2013):

Algorithm 8: Firefly Algorithm.

```
begin
    Create and randomly initialise n fireflies x_i;
    Evaluate the attractiveness of each firefly by f(x_i);
    Apply elitism to the best fireflies.
    while (l < MaxIteration)||(stopping criteria) do
        for each firefly i : n do
            for each firefly j : i = 1, ..., n do
                // Light intensity of i is determined by f(x_i);
                if (I_j > I_i) then
                    The move by i towards j is found using Equation 3.27;
                end
                Evaluate solution and update flies light intensities;
            end
        end
        Rank fireflies according to the fittest found
    end
    Save File ← best solution
    Postprocessing and visualisation
end
```

This process is repeated for all fireflies in the populations. The best individual is returned as the solution to the optimisation problem. In our case the best individual contains the optimal gains for the controller gains.

The parameters used in this investigation are shown in Table 3.9.

Table 3.9: The setup parameters for the FA.

Parameters	**Value**
Number in the population, N	50
Randomisation factor, α	0.5
Light diffusion scale, γ	0.5
Maximum iteration, i	100

Bendjeghaba et al. (2013) compared FA to Ziegler-Nichols method for PID tuning of three different plants including one with time delay. A compound objective function composed of the ITAE, rise time, settling time, overshoot and steady-state error was used for comparison. Overall, the FA performed better even though in some cases the steady-state error did not converge to zero. Sababha et al. (2019) demonstrated the tuning ability of FA for a PID controller on a MIMO LTI system. The autotuning was applied in the MATLAB environment and simulation showed that FA performed within specifications. Literature at the time of writing this thesis did not have any implementation of FA to the control of rotorcraft.

3.5 Evaluation and Validation of the Optimisation Techniques

The proposed computational intelligence optimisation algorithms are evaluated in benchmarking tests before proceeding to the act of solving our engineering problem. The steps followed for the evaluation are indicated by Ratner (2011) as follows:

1. *Defining the problem:* This first step identifies the dependent variables which we aim to optimise.

2. *Determining technique:* This step presents the preferred method of optimisation for the selected type of problem.

3. *Use of competing techniques:* At this step, the preferred method is compared to

alternatives to increase the odds of finding the most suitable method.

4. *Rough comparison of efficacy:* A matrix is used to compare the chosen methods. At this step, some methods may be dropped and there may be a search for more alternatives.

5. *Comparing in terms of precise criterion:* At this step, the methods are compared based on the criteria that were identified to optimise the dependent variables identified in Step 1.

6. *Finding the best solution:* It must be noted at this stage that the best solution for a given problem cannot be assumed to be a suitable solution for any other problem.

There are a large number of optimisation algorithms proposed in literature. There are even more evaluation problems to validate the algorithms being proposed. In this section, a number of functions are used to validate the optimisation algorithms that were discussed in this section. The functions used are shown in Table 3.10.

Table 3.10: Optimisation benchmark functions.

Parameters	**Range**	**Value**
Michalewicz	$\boldsymbol{x} \in [0, \pi]^n$	$f(\boldsymbol{x}) = -\sum_{i=1}^{n} \sin(x_i) \sin^{2n}(\frac{ix_i^2}{\pi})$
Griewangk	$\boldsymbol{x} \in [-600, 600]^n$	$f(\boldsymbol{x}) = \frac{1}{4000} \sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos(\frac{x_i}{\sqrt{i}}) + 1$
Rosenbrock	$\boldsymbol{x} \in [-5, 10]^n$	$f(\boldsymbol{x}) = \sum_{i=1}^{n-1} 100(x_i^2 - x_{i+1})^2 + (x_1 - 1)^2$
Schwefel	$\boldsymbol{x} \in [-500, 500]^n$	$f(\boldsymbol{x}) = 418.9829n - \sum_{i=1}^{n} x_i \sin(\sqrt{\lvert x_i \rvert})$
Ackley	$\boldsymbol{x} \in [-32.768, 32.768]^n$	$f(\boldsymbol{x}) = -20e^{(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2})} - e^{(\frac{1}{n}\sum_{i=1}^{n} cos(2\pi x_i))} + 20 + e$
De Jong	$\boldsymbol{x} \in [-5.12, 5.12]^n$	$f(\boldsymbol{x}) = x_1^2 + x_2^2 + x_3^2$

An example of a Michalewicz function used for optimisation benchmarking is shown in Figure 3.7.

For each optimisation algorithms, the benchmarking function tests were run 100 times and the statistical variables were recorded together with the average running time. The statistical variables collected are: maximum, minimum, mean and standard deviation. The results of this computational experimentation are shown in Table 3.11.

It was observed that the PSO outperformed all the other algorithms in all but the De Jong function. It also has the fastest convergence time in three out of the six test functions. The ACO algorithm performed the second best and was able to find the global minimum in the

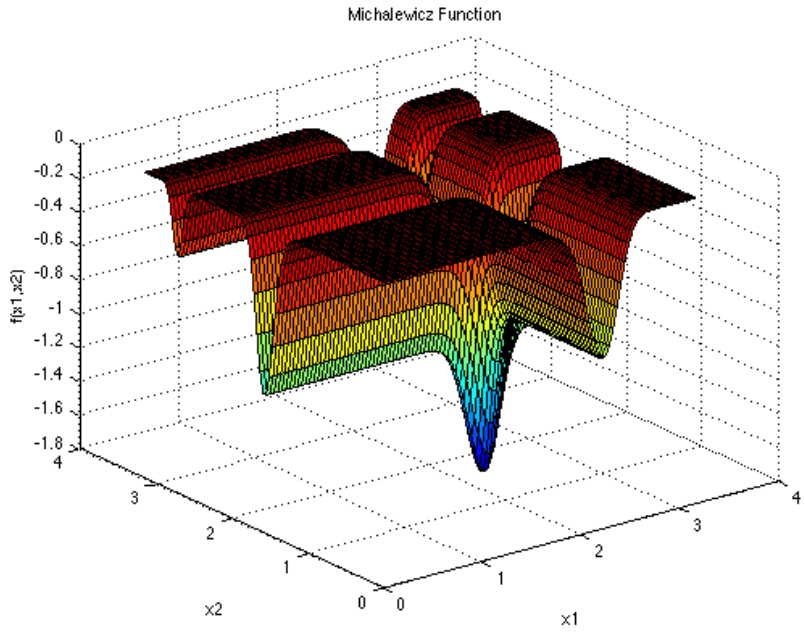


Figure 3.7: The Michalewicz function used for evaluating optimisation algorithms.

six test functions. For the Ackley function, it was the fastest and found the best solution. GA came in third and converged the fastest time in De Jong function. The performance of ALO and CS are comparable. However, the FA was the worst performing algorithm. It was able to find near optimal results in Michalewicz, but failed to converge to a solution for Schwefel and De Jong. It also had the slowest convergence of all the algorithms tested. PSO has the smallest standard deviation meaning its results are repeatable. It was closely followed by ACO.

The motivation for using computational intelligence techniques for the tuning of controller gains is that, in most controller designs the developed PID controllers are based on linearised models around a trim point. While these methods work, they do neglect/eliminate important rotorcraft information during linearisation resulting in less than optimal outcomes. This necessitates fine-tuning of the PID gains by the pilot during flight testing.

The optimisation methods presented herein tune the controller gains on the nonlinear model. Even though they are around a selected trim point, they incorporate the full nonlinear knowledge of the rotorcraft and search for the near optimal gains just as an experienced pilot would.

Table 3.11: The benchmark comparison of the six computational intelligence optimisation algorithms proposed for use in the tuning of controller gains.

Parameter		**GA**	**PSO**	**ACO**	**ALO**	**CS**	**FA**
Michalewicz	Max	-1.801	-0.801	-1.000	-1.000	-1.801	-1.782
	Min	-1.801	-1.801	-1.801	-1.801	-1.801	-1.801
	Average	**-1.801**	-1.771	-1.770	-1.785	**-1.801**	-1.799
	SD	**1.28e-8**	0.17	0.138	0.121	1.16e-6	2.22e-3
	Time	0.062	**0.048**	0.834	0.131	0.085	0.325
Griewangk	Max	0.205	**4.06e-8**	1.45e-1	4.68e-2	1.77e-2	15.5
	Min	2.29e-12	**0.00**	**0.00**	6.91e-13	5.30e-5	9.33e-2
	Average	1.58e-2	**5.04e-10**	1.65e-2	8.48e-3	5.61e-3	2.741
	SD	3.44e-2	**4.12e-9**	2.02e-2	9.22e-3	3.34e-3	1.99
	Time	0.069	**0.047**	0.049	0.137	0.100	0.103
Rosenbrock	Max	9.67	**6.08e-8**	2.00e-5	3.50	6.61e-2	1.26
	Min	4.66e-7	**6.08e-14**	2.42-13	8.74e-14	3.24e-2	3.34e-5
	Average	4.95e-1	**9.49e-10**	2.54e-7	8.98e-2	5.53e-3	1.90e-1
	SD	1.34	**6.28e-9**	2.00e-6	4.77e-1	8.90e-3	2.56e-1
	Time	0.155	0.053	**0.039**	0.128	0.056	0.331
Schwefel	Max	236.9	415.0	**118.4**	236.9	439.1	310.0
	Min	**2.55e-5**	**2.55e-5**	**2.55e-5**	**2.55e-5**	1.924e-4	5.33
	Average	23.49	138.3	45.01	43.82	**0.126**	120.6
	SD	52.2	168	57.49	64.12	**0.457**	75.37
	Time	0.083	**0.043**	0.046	0.122	0.083	0.331
Ackley	Max	2.58	2.58	**4.44e-15**	2.41e-5	5.49e-2	11.5
	Min	2.35e-6	4.45e-9	**8.88e-16**	4.15e-7	5.38e-4	2.08e-2
	Average	5.16e-2	2.58e-2	**1.03e-15**	6.41-6	1.03e-2	5.623
	SD	3.61e-1	2.57e-1	**6.96e-16**	3.78e-6	8.74e-3	2.667
	Time	0.08	0.06	**0.048**	0.125	0.339	0.081
De Jong	Max	17.37	10.76	23.81	20.15	**1.048**	52.73
	Min	**9.98e-1**	10.76	**9.98e-1**	**9.98e-1**	**9.98e-1**	1.094
	Average	5.294	10.76	6.66	5.31	**9.99e-1**	9.55
	SD	4.360	**7.97e-11**	3.78e-6	8.74e-3	5.791e-3	7.184
	Time	**0.078**	0.083	0.132	0.166	0.261	0.359

3.6 Simulation Experiment Setup

The numerical simulations of the rotorcraft system and the PID controllers for trajectory tracking of x, y and z and regulation of ϕ, θ, ψ were implemented in MATLAB®/ Simulink®. A Bogacki-Shampine solver was used with the sampling time of 1 kHz.

3.6.1 PID parameter optimisation

In the previous section, a number of optimisation techniques were introduced, and it was earlier mentioned that most PID controllers in industrial operations are under-performing due to lack of optimal tuning, despite the simple and intuitive structure of the controller. This is not evidence of the lack of know-how, but a prevailing problem that is inherent in systems that have large number of variables to tune.

From this experience and literature evidence, instinctively, we note that tuning an inherently unstable system such as a rotorcraft is a difficult and tedious task. The most common methodology for designing PID controller is by linearising the model of the system of interest and then using the available wealth of analytical linear control tools to find the gains that meet the performance specifications (Ogata, 2010). Other methods are empirical in nature and rely on the ability to excite the system and measure the output. An example from these is the Ziegler-Nichols method for finding the PID controller gains. The following, however, presents the results of PID controller gains tuned using optimisation techniques based on computational intelligence algorithms presented in the previous section. The following PID controller gains optimisation techniques are employed.

1. Manual tuning (MT) method;
2. Genetic Algorithm;
3. Particle Swarm Optimisation;
4. Continuous Ant Colony Optimisation;
5. Cuckoo Search Optimisation Algorithm.

The results are presented for the rotorcraft in hover and in 10 m/s forward flight.

3.6.1.1 PID control for a rotorcraft in hover

The algorithms presented in the previous section are used to tune PID controllers to minimise the objective function shown in Equation 3.6. Due to the stochastic nature of these algorithms, ten (10) trials were executed for each and the best results were recorded. Figure 3.8 shows the fitness value convergence history graphs for each of the optimisation algorithms tested.

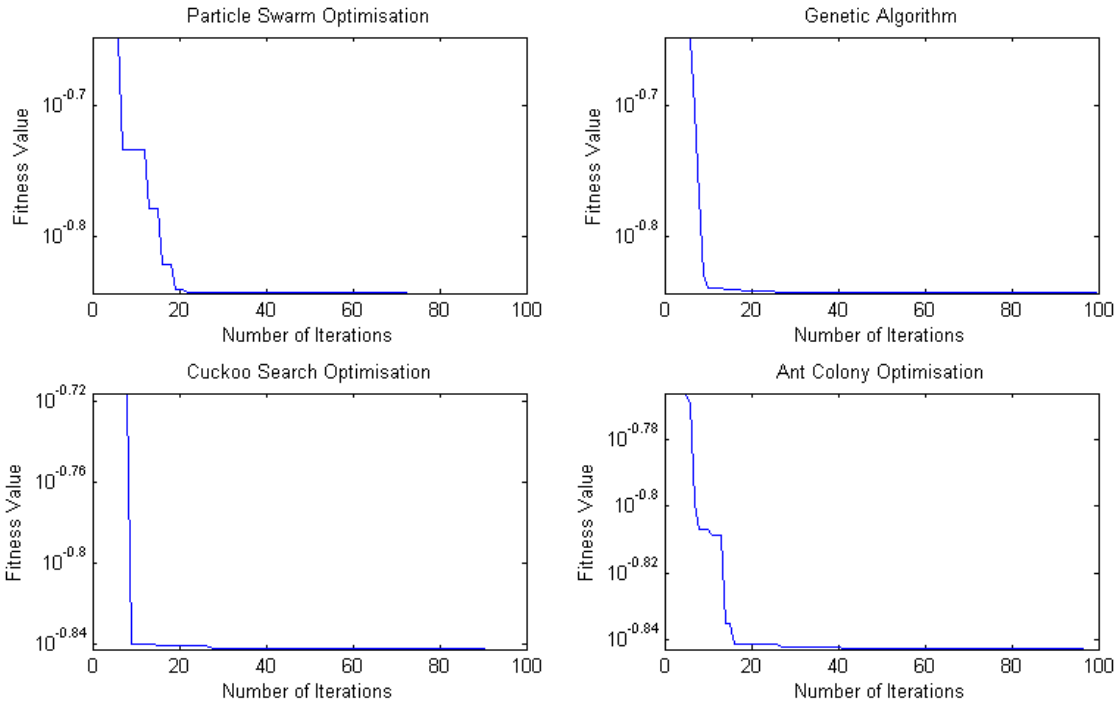


Figure 3.8: The convergence history of the fitness values of each of the optimisation algorithms over 100 iterations (generations).

Table 3.12: The fitness, the running times, mean fitness and standard deviation of the PID controller gains optimisation process for hover condition.

PID gains	**PSO**	**GA**	**ACO**	**CS**
Best fitness	1.499e-01	2.563e-01	1.340e-01	1.566e-01
Running time (min)	94.70	91.21	77.59	119.68
Mean	1.521e-01	2.608e-01	1.434e-01	1.763e-01
Standard deviation	0.020	0.065	0.119	0.240

Table 3.12 shows the fitness values for each algorithm and the respective running times. The ACO-based PID optimisation was completed in the fastest time and achieved the lowest average ISE score. The GA-tuned PID controller performed the worst, while the CS

had the longest running time. The gains found for the four PID controllers optimisation methods are shown in Table 3.13.

Table 3.13: The PID controllers gains for hover conditions found using the optimisation algorithms.

PID gains	Roll	Pitch	Yaw	Altitude	Lon	Lat
PSO K_p	-12.564	36.446	-20.986	-37.172	10.998	49.867
K_i	0	0	36.985	2.460	-45.444	19.936
K_d	-12.546	16.296	48.725	7.108	41.892	-47.418
GA K_p	21.000	-10.000	-14.000	-17.750	19.755	36.000
K_i	0	0	18.000	-10.000	11.000	33.250
K_d	0	0	31.000	0.931	-19.000	-8.000
ACO K_p	20.000	-9.757	-19.999	-19.997	0.975	19.999
K_i	0	0	8.669	-1.751	10.913	19.997
K_d	0	0	19.986	0.801	-17.124	-6.533
CS K_p	20.000	-8.862	-19.623	-20.000	17.650	20.000
K_i	0	0	20.000	-14.914	19.590	20.000
K_d	-4.370	-11.987	20.000	1.063	-18.6291	-8.603

The performance of the optimised PID controllers are evaluated for regulation of the elevation at hover time, i.e., the desired height above ground of, $z_d = -10 \pm 1$, and while withstanding a unit steps wind gust for y at 7 seconds. Figures 3.9 and 3.10 show the comparison of the tuned controllers. The ACO-based PID controller shows superior performance to others even though the others are also within performance specifications. The PSO-based PID controller is different from the controllers found by the other three algorithms. This is due to the stochastic nature of meta-heuristic algorithms, which means that even with similar initial conditions, the outcome can be vastly different, especially in cases where there are multiple minima.

On the graph it can also be seen at 5 seconds and 10 seconds that the height was disturbed. This droop was caused by the sudden change of the thrust vector components into the horizontal plane. Additionally, in Figure 3.10, the slowly growing pitch angle shows the unstable nature of the rotorcraft equilibrium. However, this is still below 0.001 rad after 20 seconds.

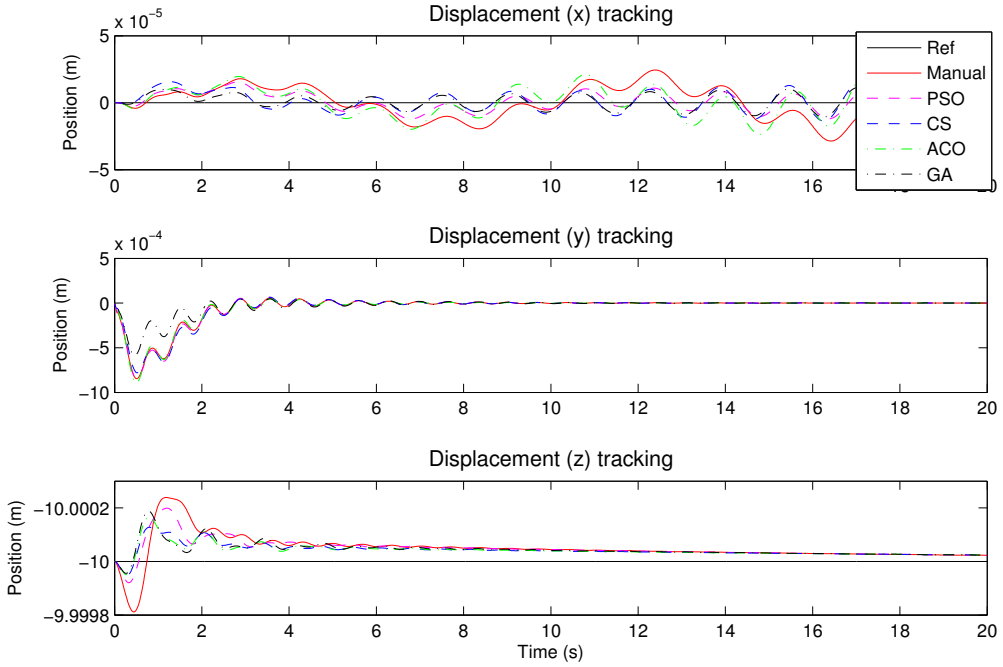


Figure 3.9: The nonlinear rotorcraft position response for the tuned PID controllers.

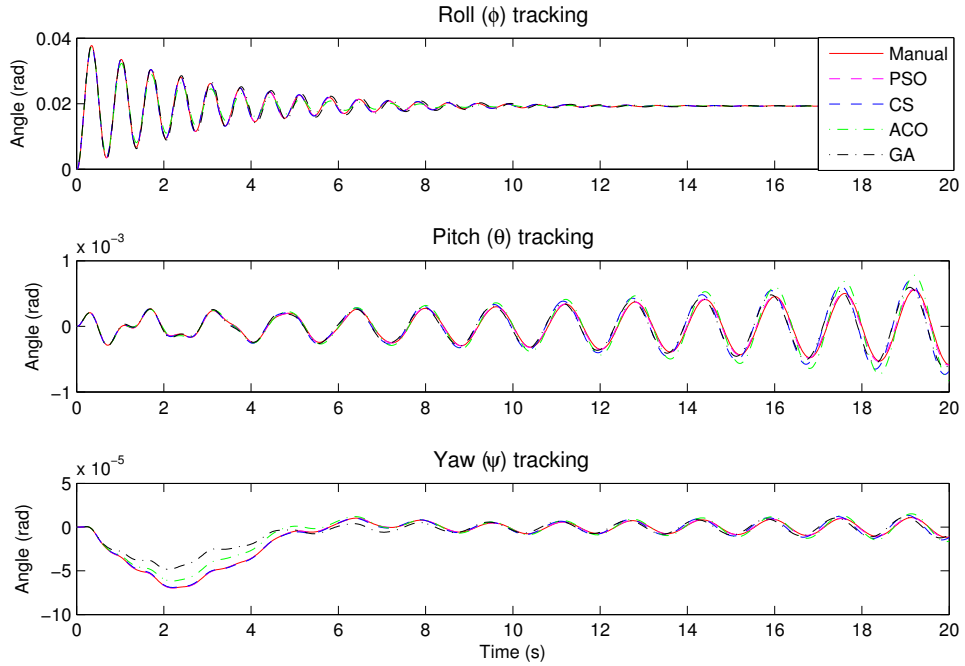


Figure 3.10: The nonlinear rotorcraft Euler angles' response for the tuned PID controllers.

PID controllers tend to have better regulation performance than tracking a desired signal due to the limited region of effectiveness around an equilibrium point. The PID controller

are evaluated for the tracking of sudden change in longitudinal displacement. In this case the controllers are able to track a small increase in forward speed of up to 2 m/s. Figures 3.11 and 3.12 shows the displacement and velocity responses of the rotorcraft. However, the ACO-based PID seems to be performing best. Also, it has less overshoot which is very desirable if the rotorcraft is performing in confined spaces. The PSO-based PID is the worst performing with velocity overshoot of 68%. With the controller gains an attempt to move the rotorcraft from trim with a velocity higher than 2 m/s result in control instability.

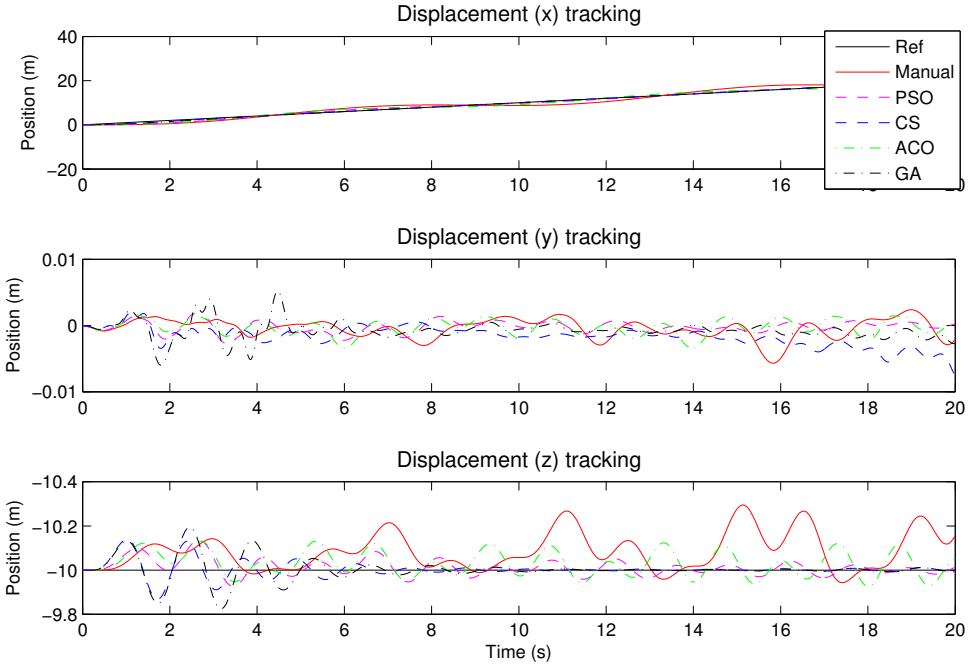


Figure 3.11: The rotorcraft positions response for the forward flight at 1 m/s.

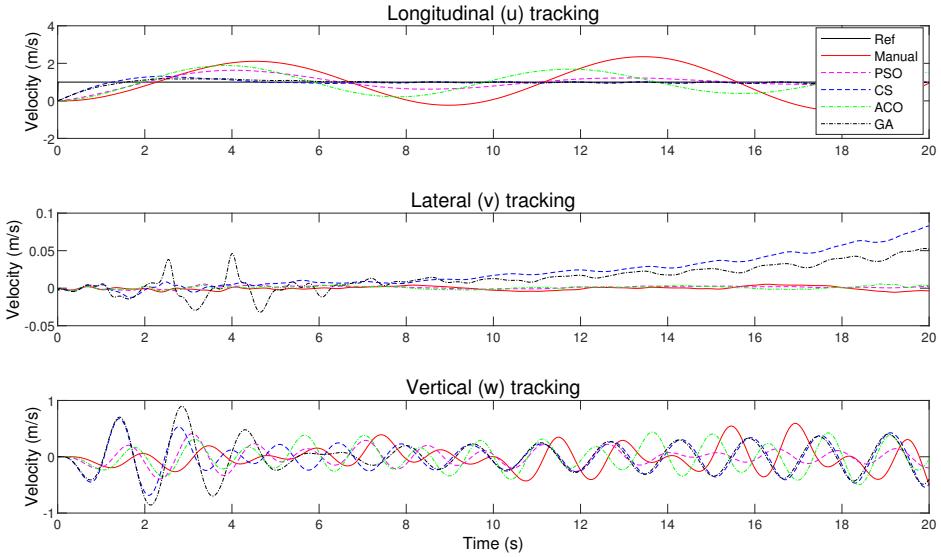


Figure 3.12: The rotorcraft velocity history for the forward flight at 1 m/s.

3.6.1.2 PID control for a rotorcraft in forward speed

The optimally-tuned PID controllers can give performance of up to 2 m/s, after which, the rotorcraft becomes unstable. Substantial move away from the selected conditions of the PID controller whose gains have been optimised results in the loss of performance. This is a well understood shortcoming of the PID controllers. The tuned system was tested on how well it withstands the translation from hover to 10 m/s forward flight, and found that a new set of controller gains is required for the new flight condition. The PID controllers were retuned using the four optimisation techniques for the 10 m/s forward flight, $\mathbf{v}^I = [10\ 0\ 0]^T$. Fitness value convergence history graphs for each of the optimisation algorithms tested are shown in Figure 3.13.

Table 3.14: The fitness, the running times, mean fitness and standard deviation of the PID controller gains optimisation process for forward flight condition.

PID gains	**PSO**	**GA**	**ACO**	**CS**
Best fitness	1.334e2	0.972e2	0.021e2	0.723e2
Running time (min)	91.11	91.21	77.59	130.02
Mean	1.176e2	1.076e2	0.034e2	0.812e2
Standard deviation	30.8	0.33	1.71	8.62

The training process for forward flight took about the same time as in hover. The ACO algorithm converged the fastest and also found the best gains. The velocity is regulated

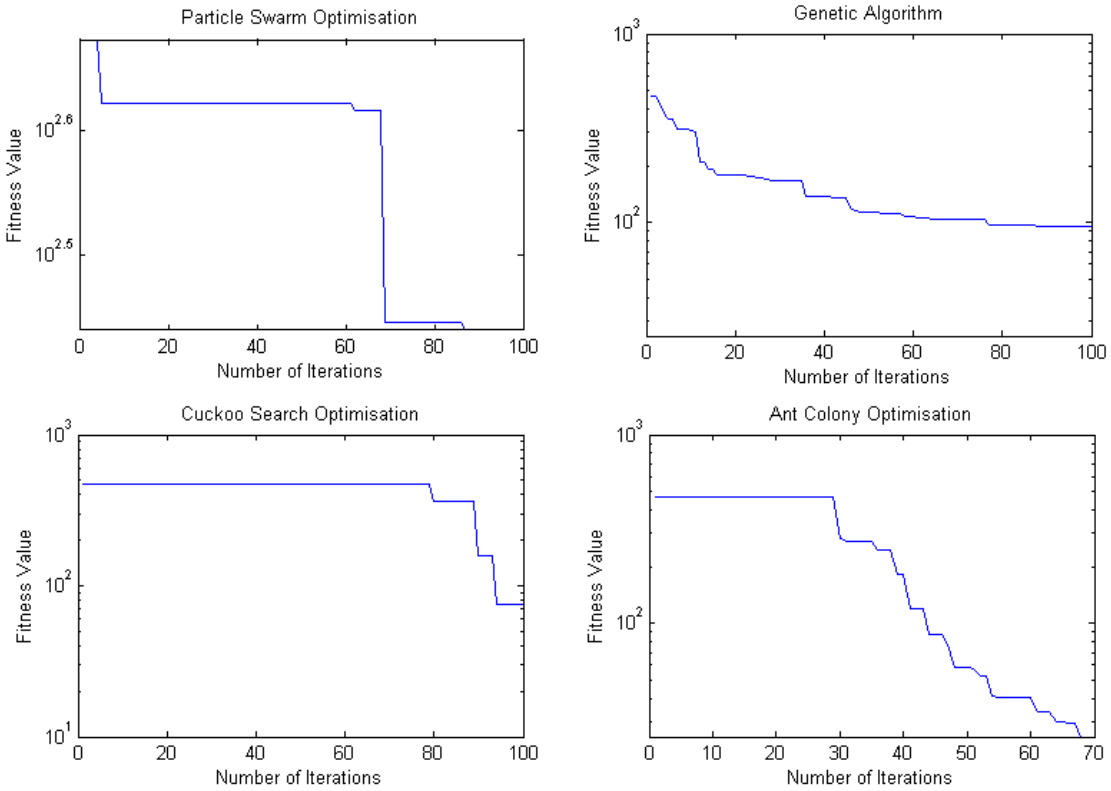


Figure 3.13: The convergence history of the fitness values of each of the four optimisation algorithms over 100 iterations.

at $10m/s$ and the distance is continually increasing. The fitness values were found to be larger than those found in hover. However, the performances of the algorithms are closely comparable. Table 3.15 shows the gains returned from the optimisation process.

Table 3.15: The PID controllers gains for forward flight condition found using the using optimisation.

PID gains	**Roll**	**Pitch**	**Yaw**	**Altitude**	**Lat**	**Lon**
PSO K_p	37.4465	-11.6261	4.4341	-36.7346	8.4551	11.6230
K_i	0	0	38.2470	5.9603	0.6901	1.7972
K_d	17.2960	-2.6046	2.4902	7.1013	0.1360	-7.9357
GA K_p	37.4465	-11.6261	4.4341	-36.7346	8.4551	11.6230
K_i	0	0	38.2470	5.9603	0.6901	1.7972
K_d	17.2960	-2.6046	2.4902	7.1013	0.1360	-7.9357
ACO K_p	37.3029	-11.6348	4.3475	-36.7444	8.4505	11.6536
K_i	0	0	38.2267	5.9482	0.6955	1.8379
K_d	17.4892	-2.5949	2.4355	7.1258	0.1196	-7.9047
CS K_p	37.4465	-11.6261	4.4341	-36.7346	8.4551	11.6230
K_i	0	0	38.2470	5.9603	0.6901	1.7972
K_d	17.2960	-2.6046	2.4902	7.1013	0.1360	-7.9357

A forward flight path the reference signal was used to test the optimised PID control. The controllers were analysed by comparison the position and the velocity responses of the rotorcraft in this forward flight are shown in Figures 3.14 and 3.15. The controllers are able to keep a constant velocity while steadily increasing the x-position as it is expected at constant velocity. Simulations show that the GA-based PID controller outperformed the other tuned controllers with regards to regulating the required velocity. However, the velocity percentage overshoot and settling time were 11% and $4s$ respectively. The rest of the controllers are also follow the desired forward displacement and velocity, but this was at the expense of loss of the vertical position of the rotorcraft. It is worth noting that the GA-based PID controller resulted in the worst performance when it come to regulating the y-axis. The underactuated and coupled nature of this problem becomes apparent in that optimising each rotorcraft DOF output channel comes at the expense of other output channels.

In Table 3.15, three of the control gains set seem to be similar within the quoted rounding error. However, in an optimisation problem, especially with a large search space or complex problem, a slight difference in the algorithm can result in significantly different results. This is because these optimisation algorithms involve iterative processes where small changes to the algorithm can compound over time and lead to divergent solutions. Hence, there will be significant differences in the performance of the corresponding control systems as shown in Figures 3.14 and 3.15.

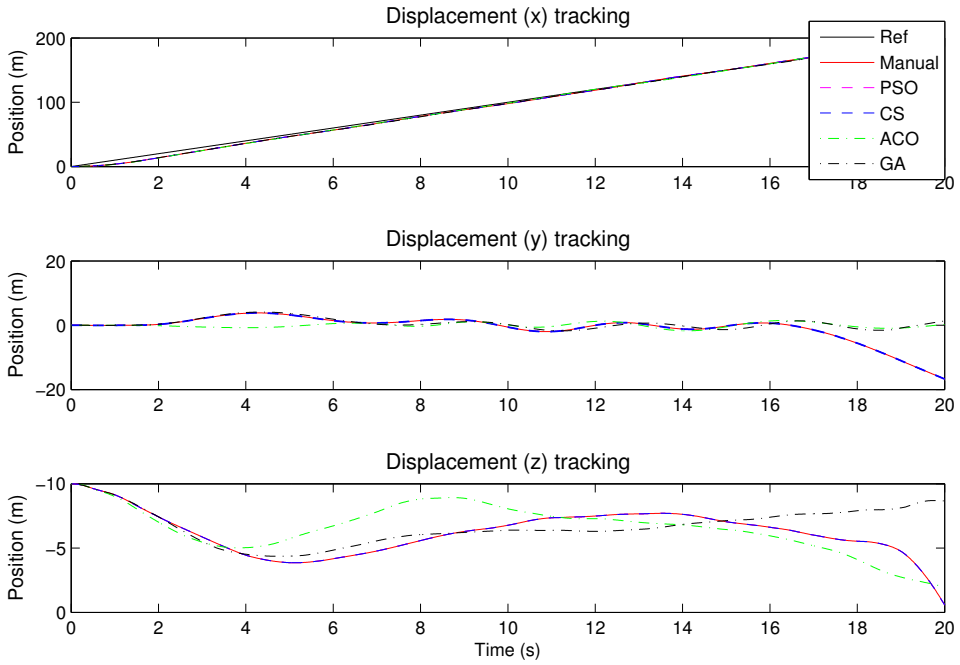


Figure 3.14: The rotorcraft positions response for the forward flight at 10 m/s.

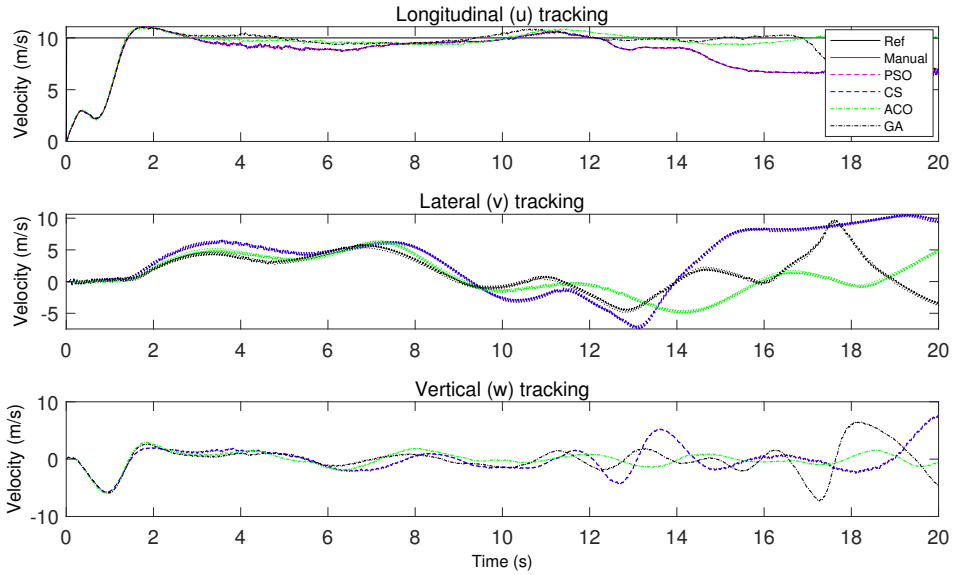


Figure 3.15: The rotorcraft velocity history for the forward flight at 10 m/s.

3.6.1.3 Effect of variations in rotor speed

One of the main objectives of this study is to integrate the flight controller with the propulsion controller. In this section of the thesis, the result of how the designed PID controller

handles the changes in rotor speed from nominal are presented. The engine speed and rotor speed are used interchangeably since the difference is in the gear transmission ratio.

The engine speed was changed from nominal to nominal+10% after 10 seconds of steady flight. Figures 3.16 and 3.17 show the time history of the position of the aircraft and how it responds to 10% increase in rotor speed, and the inputs that are required to keep the rotorcraft in steady flight.

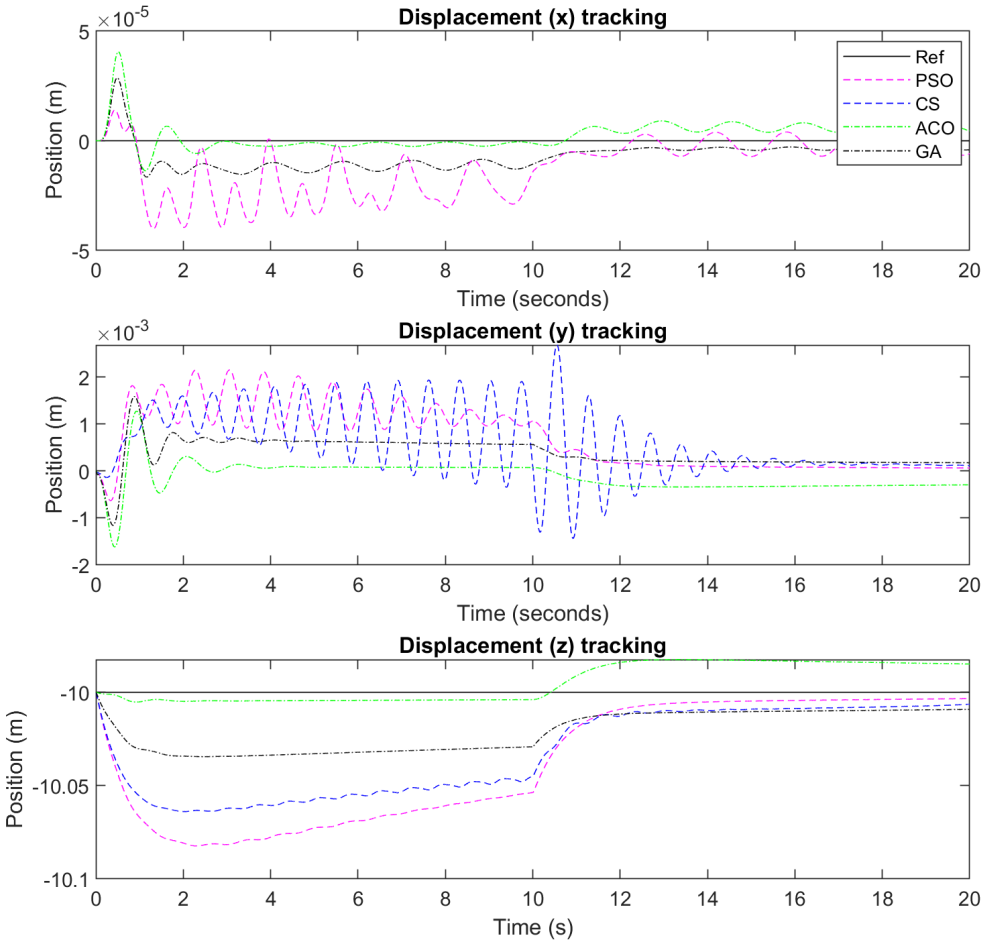


Figure 3.16: The rotorcraft position response when operating at 10% increase in rotor speed.

The engine speed was then changed to nominal - 10% after 10 seconds of steady flight. Figures 3.16 and 3.17 show the time history of the position of the rotorcraft and how it responds to 10% decrease in rotor speed from nominal, and the inputs that are required to keep the rotorcraft in steady flight.

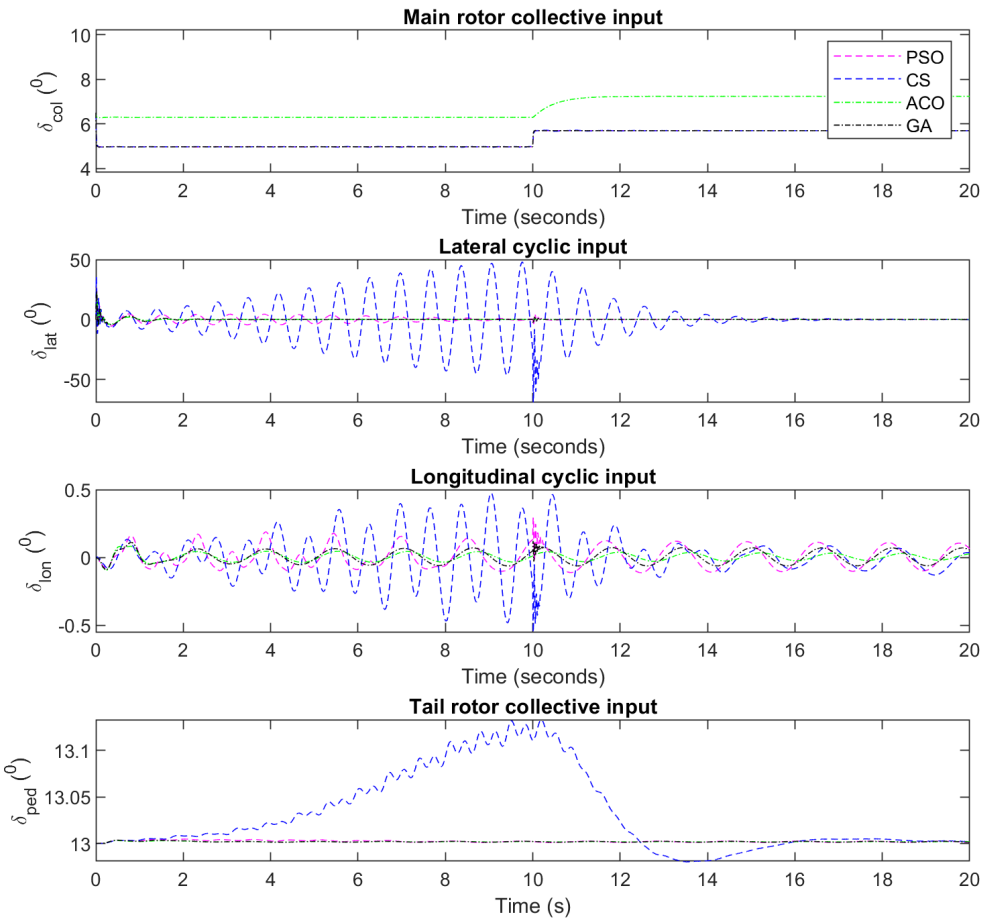


Figure 3.17: The rotorcraft control input when operating at 10% increase in rotor speed.

3.6.1.4 Robustness to parameter variations and external disturbances

No component is as good as it was when new. As a result, during its operation the parameters that define our rotorcraft will change over time: due to wear and tear, fault and damage. In this section, the rotorcraft is subjected to change in its parameters and the performance of the designed controllers is analysed. The following parameter variations and disturbances are imposed on the rotorcraft and to see how the respective controllers react:

- The mass of the rotorcraft will be varied from 50, 100 and 200%;
- The moments of inertia will be varied from 50, 100 and 200%;
- The rejection of wind gust of up to 5 m/s;

- The LOE of one of the three actuators by 10 and 30% effectiveness.

The results of mass experiment are shown in the Figure 3.18. As can be seen that the controllers are able to handle minor variations, but it is unable to control the rotorcraft if the variations are too large.

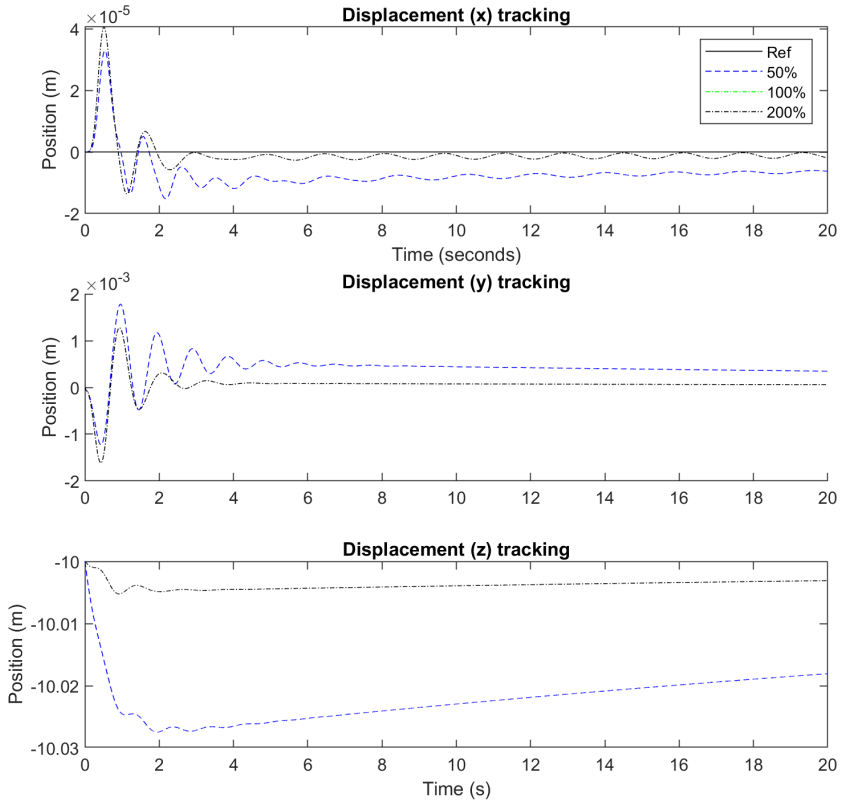


Figure 3.18: The rotorcraft position response for the robustness testing with 50%, 100% and 200% of the nominal mass.

The results of the moments of inertia variations show a similar pattern (Figures 3.19) in that for minor variations the control is robust enough to control the rotorcraft. However, this robustness is limited as it fails to control the system at larger changes to parameters.

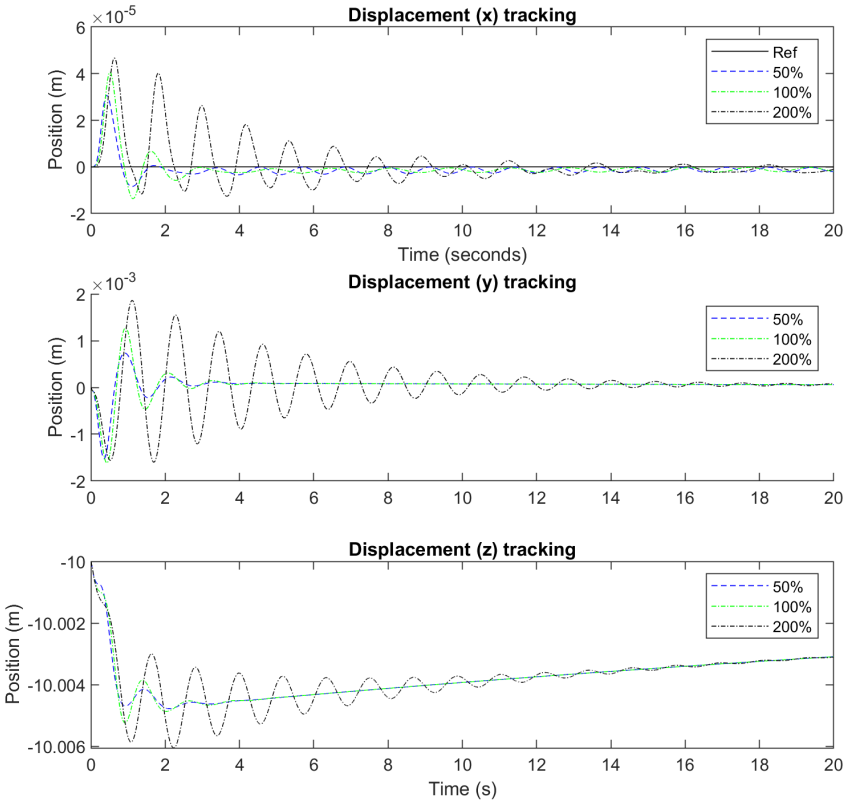


Figure 3.19: The rotorcraft position response for the robustness testing with 50%, 100% and 200% of the nominal inertia tensor.

Thirdly, to verify the robustness of the tuned controllers to external disturbance, the rotorcraft was subjected to a wind gust perturbation . After introducing the disturbance at time $t = 7$ seconds, the rotorcraft was observed to see if it will be able to return and maintain its equilibrium. The position and orientation of the rotorcraft when it is subjected to a 5 m/s wind gust from the starboard side are depicted in Figures 3.20 and 3.21. All the PID controllers tuned for hover are capable of recovering from the disturbance within 5 seconds as shown in Figure 3.20. Eventhough the rotorcraft is able to retrun to a zero y position, the vertical position, z, is lightly increased. Figure 3.21, shows velocity response of the rotorcraft. There is obeservable change at 7 seconds for all three axes. The velocities oscillate afterwards but eventually settles asymptotically. The longitudinal velocity, u, takes longer than others to settle. The GA-based controller rotorcraft in unable to recover at all.

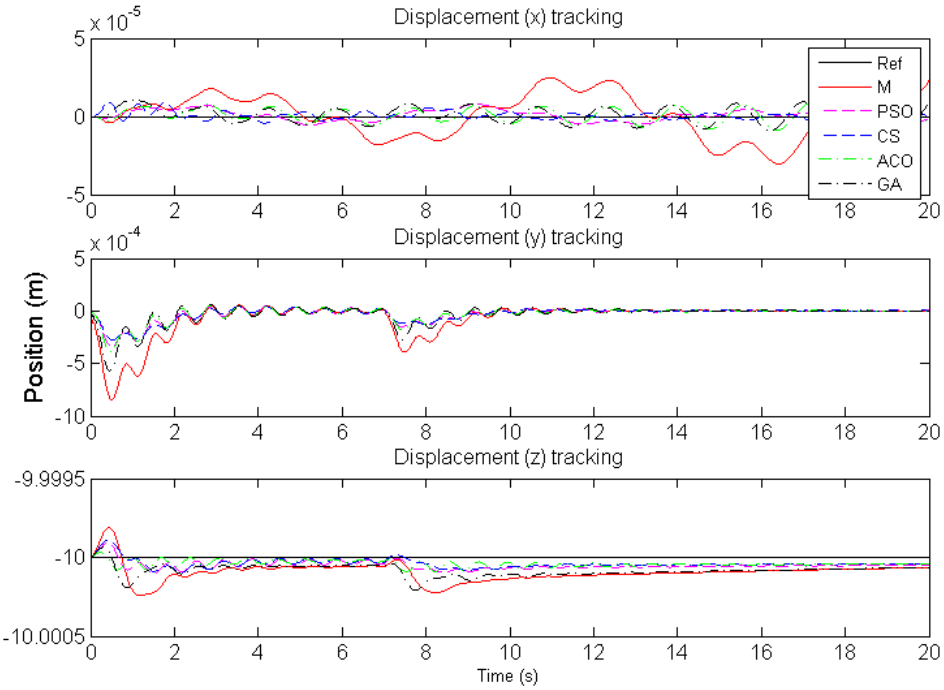


Figure 3.20: The rotorcraft position response to wind gust disturbance in hover.

During the rotorcraft wing gust perturbation in forward speed, it is still able maintain the correct forward speed, eventually it drifts and does not recover the desired position track. The ACO-based PID controller was the more effective in its disturbance rejection response while the GA-based PID controller responded the worst to wind gust in this flight regime. We suppose that GA's effectiveness is diminished outside the nominal trim conditions is that due to its strict adherence to the objective function during tuning, making it less robust.

The fourth experiment is to control the rotorcraft with a faulty actuator. Referring to Figure 2.3 of the previous Chapter, the LOE is injected in actuator A. The response of the rotorcraft is shown in Figure 3.23, the control inputs are shown in Figure 3.24.

The PID controllers are shown to maintain the stability of the system when there is an actuator fault of 30% and 50% LOE. However, during the faults, it is seen that the actuators are pushed to saturation. There is significant droop on the aircraft. The aircraft losses close to half a metre in height. If the rotorcraft is operating in a confined space, this could result in a crash and loss of the rotorcraft.

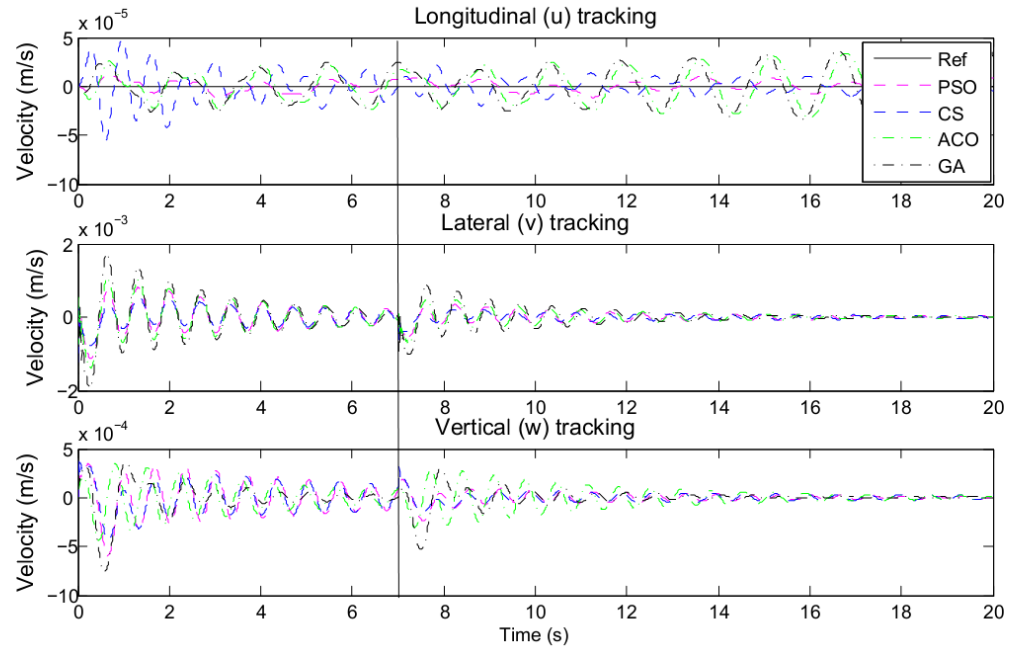


Figure 3.21: The rotorcraft speed response to wind gust disturbance in hover.

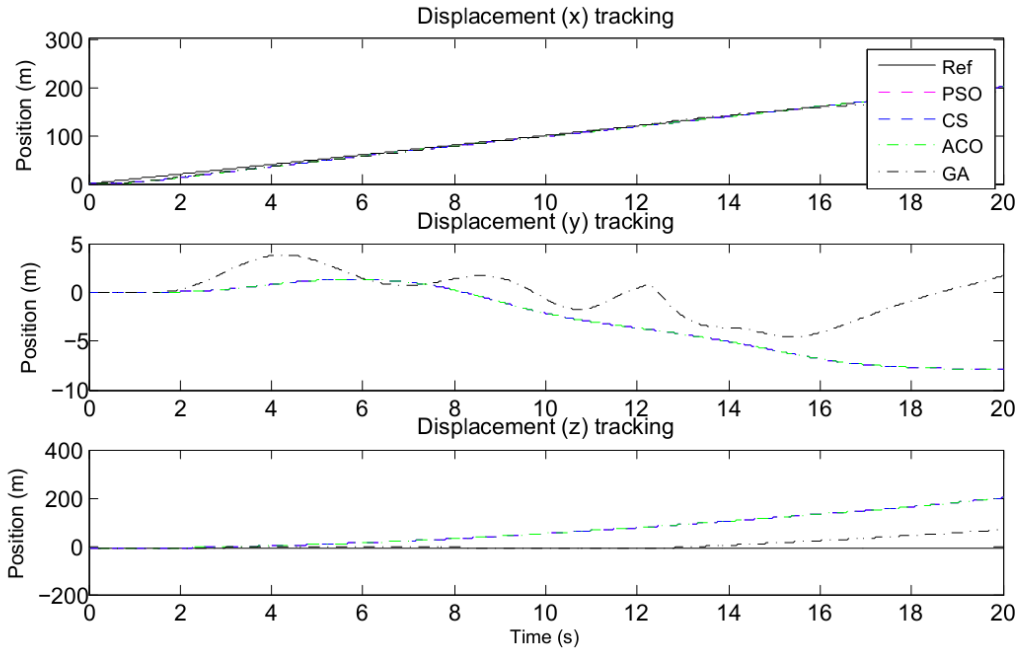


Figure 3.22: The rotorcraft position response to wind gust disturbance at 10 m/s forward flight.

The shortcomings of adaptive and intelligent control when applied to aircraft FCS and their lack of rigorous proofs leading to the large number of documented experimental aircraft incidents (Nguyen, 2018). However, in this thesis the PID controller gains presented are derived off-line. The result of the computational intelligence optimisation algorithms

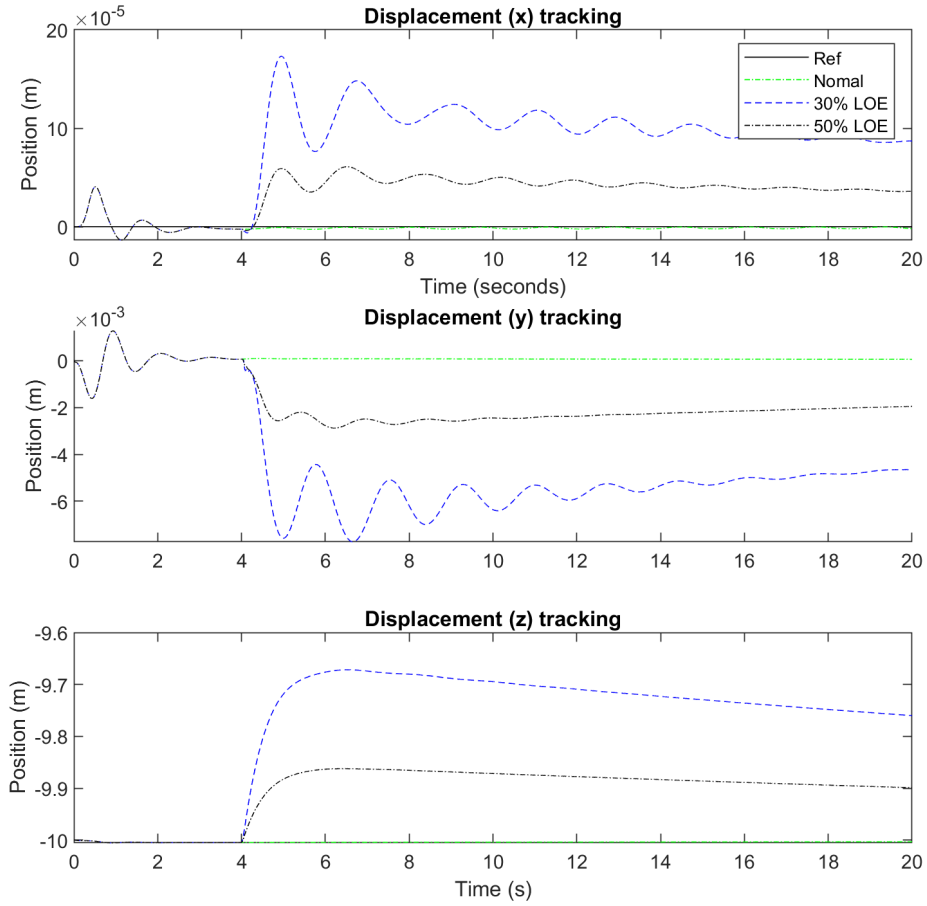


Figure 3.23: The rotorcraft position response when there is an actuator fault of 30% and 50% LOE.

are the controller gains. These gains become part of the FCS while the intelligent algorithms themselves do not end up in the aircraft FCS.

We showed earlier in numerical simulations that ACO-based PID controller performs better for hover while the GA-tuned PID controller performs better for forward flight. This might come across as a contradiction. However, this does not mean that the rotorcraft FCS employs the GA algorithm and later the ACO algorithm. In reality, the results of the optimisation process are incorporated into the FCS, while the optimising algorithm is not installed in the FCS. These results are subjected to the same scrutiny and rigour as the optimised PID controller gains from manual or other empirical means that are available to the designer, albeit not optimal.

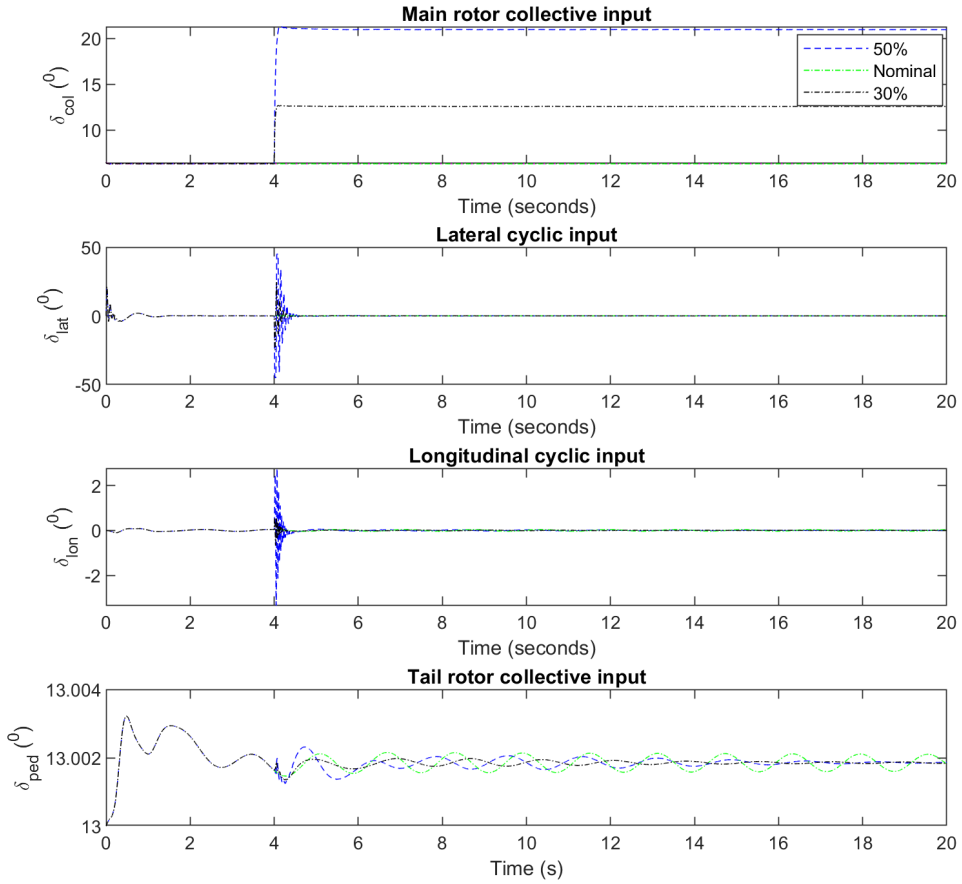


Figure 3.24: The control inputs into the rotorcraft when there is an actuator fault of 30% and 50% LOE.

3.7 Chapter Summary

This chapter presented the rotorcraft trim analysis for hover and forward flight. The position, attitude, altitude and heading control design using the PID controller was also presented. The PID controller gains were initially tuned manually. In order to get optimal gains computational intelligence optimisation algorithms were employed. The results presented in this chapter have been published in (Mpanza and Pedro, 2021). The contribution to knowledge is the application of continuous ant colony and cuckoo search for finding gains in a multi-input multi-output rotorcraft system. It was evident that the optimised controllers are able to maintain rotorcraft stability even after the system mass and moment have been halved or doubled. The rotorcraft was able to reject a wind gust of up to 5 m/s in the lateral axis. However, the controllers (both optimised and not) failed to meet the system's performance specifications and actuator fault with 50% LOE.

Chapter 4

Passive Fault-Tolerant Control of Rotorcraft using Sliding Mode Controller

This chapter presents the passive fault-tolerant control based on the sliding mode control on the rotorcraft model developed earlier. It is concerned with the development of the basics of sliding mode control and presents different variants which are applied to the rotorcraft system. The model-based controller design methodology is used. The application of sliding mode in numerical experimentation and simulation is detailed. In order to find optimal parameters for the system, the computational intelligence optimisation algorithms that were developed in previous chapter are applied. The fault tolerance is evaluated on actuator loss-of-effectiveness of up to 50%.

4.1 Passive Fault-Tolerant Control and Fault Modelling

The preceding chapter presented the rotorcraft control based on PID controller. This controller proved effective in controlling the rotorcraft in terms of regulating trim conditions for both hover and forward flights, and demonstrated disturbance rejection around a specified trim. However, the PID controller was ineffective in dealing with large deviations from trim, parameter variations, such as: changes in rotorcraft mass and inertia tensor, and control under actuator faults. Improvements were noticed when the PID controller is optimally tuned using computational intelligence algorithms. This chapter extends on the preceding effort by proposing passive fault-tolerant control (PFTC) strategies for dealing

with modelling uncertainties, disturbances in the environment and parameter variations of the rotorcraft in operation.

The two major aims of this Chapter are: to develop control strategies for the rotorcraft to recover from actuator faults; and be able to handle variations in rotor speed as required for variable rotor speed rotorcraft. PFTC is achieved when a single-fixed parameter or fixed-structure controller has the ability to tolerate faults without the need for fault information. This is also called robust or reliable controller. This type of control strategy guarantees nominal performance while providing tolerance to changes in the nominal system characteristics to a certain extent. PFTC is designed off-line and when in operation, it is robust to a limited number and types of faults. Trying to increase coverage of faults is at the expense of nominal performance.

In this thesis, we present the application of sliding mode control (SMC) for robust tolerance to actuator faults (Yu, 2016). Literature on robust controller and FTC refers to three issues that must be dealt with. However, there is no consensus on the proper definition of these items. Given herein is a brief definition of the SMC FTC formulation as used in this thesis in alignment with majority of the literature covering similar subject (Noura et al., 2009; Xu, 2011; Halbe and Hajek, 2020).

Definition 4.1. External disturbance involves the change of the system due to unintended external excitations such as gust winds, temperature, density, and altitude to name a few; These are a function of operation environment and can come and go.

Definition 4.2. Model uncertainties are due to the flaws in the system identification process. That a controller is designed based on the model which is not a true representation of the system to be controlled.

Definition 4.3. Parameter variation is due to the fact that the system itself changes over time due to aging or faults occurring internally. This differs from disturbances in that once the parameter changes, it does not return to a nominal value until external interventions are applied.

4.2 Modelling of System's Parameter Variations, Actuators and Faults

In a system that is in continuous operation, no component is as good as it was when new in terms of functionality. This is due to the fact that during operation, the parameters that define the rotorcraft system will change over time: due to wear and tear, faults,

damages and improper installation (Zhang et al., 2021). Therefore a controller must be able to handle minor deviations from the model that was used to derive the control laws. In this chapter we investigate the reaction of the rotorcraft when it is subjected to changes in its parameters and the performance of the proposed controllers is analysed based on their robustness.

Consider a linear time invariant (LTI) system representation for the FTC system with parameter variation as shown in Equation 4.1:

$$\dot{x} = (A + \Gamma A)x + Bu, \tag{4.1}$$

where $0.5 < \Gamma < 1$ is the parameter variation factor. In this case, the system parameter is allowed to vary by a factor of two. In this study, an assumption is made to ignore system component failure due to ΓA. This ΓA factor is treated as a system uncertainty. The developed controller is expected to deal with these uncertainties. The rest of this thesis will focus on modelling actuator faults.

Actuator faults

The representation of faults is important in gaining insight into the failure recovery mechanism. The rotorcraft has four actuators. Three actuators are for controlling the main rotor swashplate and one actuator for controlling the tail rotor thrust. The actuators are either electrohydraulic or electromechanical as described in Chapter 2. If we consider these actuator dynamics as "fast", the actuator positions response to inputs can be estimated as $x_i = Ku_i$: which simplifies the model-based design of the controllers. Since the tail rotor's main purpose is to counter the torque resulting from the rotation of the main rotor, its failure is unrecoverable by any means (Bramwell et al., 2001). As a result only the main rotor actuators are considered for fault tolerance in this thesis.

Actuator and sensor faults can be considered in two categories: additive and multiplicative. The ultimate goal of the FTC is to compensate for faults regardless of thier nature or origin. The fault illustration is simplest when considering a MIMO linear model. A system with different types of faults and can be represented as follows:

$$\dot{\boldsymbol{x}}_f = \boldsymbol{A}_f\boldsymbol{x}(t) + \boldsymbol{B}_f\boldsymbol{u}_f(t), \tag{4.2}$$

$$y_f = \boldsymbol{C}_f\boldsymbol{x}(t), \tag{4.3}$$

where $\boldsymbol{A}_f = \boldsymbol{A} + \delta\boldsymbol{A}$, $\boldsymbol{B}_f = \boldsymbol{B} + \delta\boldsymbol{B}$ and $\boldsymbol{C}_f = \boldsymbol{C} + \delta\boldsymbol{C}$ representing the change in system components, actuators and sensors, respectively. More importantly, the actuator fault (which is the subject of this thesis) is linked to the variation of $\boldsymbol{u}$, the system input vector (Alwi et al., 2011; Hamayun et al., 2016).

$$\boldsymbol{u}_f = \boldsymbol{u}_n + (\boldsymbol{I} - \boldsymbol{\Gamma})(\boldsymbol{u}_{fo} - \boldsymbol{u}_n),$$

where $\boldsymbol{u}_n$, $\boldsymbol{u}_f$, $\boldsymbol{u}_{fo}$ and $\boldsymbol{\Gamma}$ are the nominal input, the global fault input, the additive actuator fault (such as drift) and the multiplicative fault (change in effectiveness) respectively. The constant is given as: $\mathbf{0} < \boldsymbol{\Gamma} < \boldsymbol{I}$, $\boldsymbol{\Gamma} \in \Re^{m\times m}$ is an unknown diagonal positive definite matrix representing the actuator loss-of-effectiveness (LOE) constants of m-sized diagonal matrix given by:

$$\boldsymbol{\Gamma} = \begin{bmatrix} \alpha_1 & 0 & 0 & 0 \\ 0 & \alpha_2 & 0 & 0 \\ 0 & 0 & \alpha_3 & 0 \\ 0 & 0 & 0 & \alpha_4 \end{bmatrix}. \tag{4.4}$$

This is a weighting matrix used by the system to inject the faults and failures into the rotorcraft actuator dynamics. This matrix is used later when the effectiveness of the designed controller is being tested for robustness to changes and tolerance to actuator loss-of-effectiveness. The list below gives the fault that are possible for the given actuator model:

1. when $\alpha_i = 1$, the system is healthy;
2. when $0 < \alpha_i < 1$, the system has "actuator loss-of-effectiveness" fault;
3. when $\alpha_i = 0$, the actuator at i has completely failed.

The system representation for a biased FTC system is as shown in Equation 4.5

$$\dot{x} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{\Gamma}\boldsymbol{u}_n + \boldsymbol{B}(\boldsymbol{I} - \boldsymbol{\Gamma})(\boldsymbol{u}_{fo} - \boldsymbol{u}_n), \tag{4.5}$$

where $\boldsymbol{u}_{fo} \in \Re^m$ is an unknown input, and the system is not biased if and only if $\boldsymbol{u}_{fo} = \boldsymbol{\Theta}$.

This means that in order to identify the faulty actuator, the fault detection and diagnosis (FDD) must detect the presence of the following condition (not mutually exclusive):

$$\boldsymbol{\Gamma} \neq \boldsymbol{I},$$

$$\boldsymbol{u}_{fo} \neq \boldsymbol{\Theta},$$

where $\boldsymbol{I}$ and $\boldsymbol{\Theta}$ are the identity matrix and zero matrix of appropriate dimensions respectively. This chapter does not employ FDD to detect the faults. Only the passive robustness of the SMC is investigated for recovering from faults.

A number of FTCs for rotorcraft have been presented in the literature. The popular ones are H_2 (Paulino et al., 2006; Guerreiro et al., 2018), H_∞ (Gadewadikar et al., 2009; Jeong et al., 2012; Khalesi et al., 2020), and sliding mode controller (He and He, 2018; Halbe and Hajek, 2020; Wang and Zhang, 2018).

Guerreiro et al. (2018) used H_2 technique to control a rotorcraft based on a linear parameter varying (LPV) model representation; which is a combination of affine model systems. In this aircraft, states are measured using LiDAR and only gave relative position of the aircraft. The effectiveness of the controller was validated with simulations and experimental results. Salmah et al. (2013) demonstrated the implementation of H_2 technique, where the controller is supposed to be robust-scheduled to changes among five linearised models in order to cover a large flight envelope. Simulation results showed that the controller is able to stabilise the helicopter in the five different trim states.

H_∞ approach provides a better response in the presence of disturbance than H_2 approach. Gadewadikar et al. (2009) used a static output-feedback H_∞ control to stabilise a rotorcraft in hover when subject to external disturbances. In (Jeong et al., 2012), a loop shaping-based controller is developed using weighted functions and H_∞ approach. This controller is applied to a LTI rotorcraft model and it was able to withstand model parameter variation of up to 50% and $0.1s$ delays in the actuators. Guo et al. (2017) presented a linear matrix inequality (LMI)-based state feedback controller by mixing H_2 and H_∞ techniques with constraints. This controller is used to stabilise a quadrotor with a cable-suspended load. Simulations showed that the hybrid controller performed better than the individual controllers. The major drawback of H_2 and H_∞ approaches has been found to be the reliance on linear models and gain-scheduling.

In the following, the rotorcraft FTC problem is formulated in a form suitable for application of SMC technique.

4.3 Problem Formulation

The rotorcraft state-space equations can be written as shown in Equation 2.54. This system state-space is repeated here for ease of reference:

$$\begin{aligned}
\dot{v} &= vr - wq + s_\theta g + X/m, \\
\dot{u} &= wp - ur + s_\phi c_\theta g + Y/m, \\
\dot{w} &= uq - vp + c_\phi c_\theta g + Z/m,
\end{aligned} \tag{4.6}$$

$$\begin{aligned}
\dot{p} &= \frac{I_{yy} - I_{zz}}{I_{xx}} qr - \frac{I_{xz}}{I_{xx}}(\dot{p} - pq) + \frac{L}{I_{xx}}, \\
\dot{q} &= \frac{I_{xx} - I_{zz}}{I_{yy}} pr + \frac{I_{xz}}{I_{yy}}(r^2 - p^2) + \frac{M}{I_{yy}}, \\
\dot{r} &= \frac{I_{xx} - I_{yy}}{I_{zz}} pq + \frac{I_{xz}}{I_{zz}}(\dot{p} - qr) + \frac{N}{I_{zz}},
\end{aligned} \tag{4.7}$$

$$\begin{aligned}
\tau_f \dot{a_{lc}} &= -a_{lc} - \tau_f q + \left(\frac{\partial a_{lc}}{\partial \mu} \mu_x + \frac{\partial a_{lc}}{\partial \mu_z} \mu_z \right) + K_{lon} \delta_{lon}, \\
\tau_f \dot{b_{ls}} &= -b_{ls} - \tau_f p + \left(\frac{\partial b_{ls}}{\partial \mu_v} \mu_y \right) + K_{lat} \delta_{lat},
\end{aligned} \tag{4.8}$$

$$\begin{aligned}
\dot{x}^I &= c_\theta c_\phi u + (s_\theta c_\phi c_\psi - c_\phi s_\phi) v + (s_\theta c_\phi c_\psi + s_\psi s_\phi) w, \\
\dot{y}^I &= c_\theta c_\psi u + (s_\theta s_\phi s_\psi + c_\theta c_\psi) v + (c_\theta s_\phi s_\psi - c_\psi s_\phi) w, \\
\dot{z}^I &= -s_\theta u + c_\theta s_\psi v + c_\psi c_\theta w,
\end{aligned} \tag{4.9}$$

$$\begin{aligned}
\dot{\phi} &= p + s_\theta t_\theta q + c_\phi t_\theta r, \\
\dot{\theta} &= c_\phi q - s_\phi r, \\
\dot{\psi} &= \frac{s_\phi}{c_\theta} q + \frac{c_\phi}{c_\theta} r,
\end{aligned} \tag{4.10}$$

where $s_\theta = sin(\theta)$, $c_\theta = cos(\theta)$, $s_\phi = sin(\phi)$, $c_\phi = cos(\phi)$, $s_\psi = sin(\psi)$, $c_\psi = cos(\psi)$, and $t_\theta = tan(\theta)$. The difference between Equations 2.54 and 4.10 is that the engine dynamics are assumed to be fast compared to the rotor thrust dynamics and therefore in quasi steady-state when the rotorcraft system is being studied (Iwata, 1996). This assumption is only applied to the model used for controller development. The resulting controller is tested on the rotorcraft with the engine dynamics as shown in Equation 2.54. The rotor speed Ω is regulated as nominal and any deviation from the nominal condition, whether intentional or not, is treated as disturbance. A robust controller should be able to deal with the engine dynamics as model uncertainty.

The Euler angles $[\phi, \theta, \psi]$ depend on the body rates $[p, q, r]$. While the position states $[x, y, z]$ depend on rotorcraft velocity $[u, v, w]$ and Euler angles $[\phi, \theta, \psi]$. Therefore, these

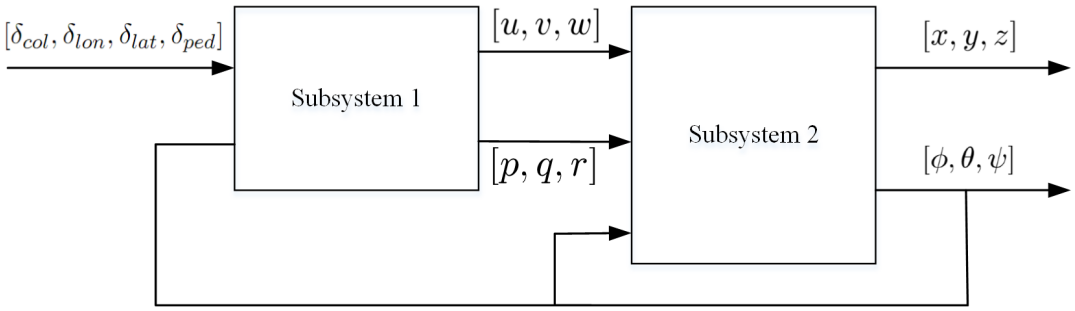


Figure 4.1: The system divided into states that are directly and indirectly influenced by the control input.

derived states that are not directly influenced by the control inputs. Instead, they depend on other states as shown in Figure 4.1.

This means the state-space equation can be split into input-directly dependent states subsystem and subsystem of states variables that are dependent on other states.

The following is the new state equations based on Subsystem 1:

$$\begin{aligned}
\dot{v} &= vr - wq + s_\theta g + X/m, \\
\dot{u} &= wp - ur + s_\phi c_\theta g + Y/m, \\
\dot{w} &= uq - vp + c_\phi c_\theta g + Z/m,
\end{aligned} \tag{4.11}$$

$$\begin{aligned}
\dot{p} &= \frac{I_{yy} - I_{zz}}{I_{xx}} qr - \frac{I_{xz}}{I_{xx}}(\dot{p} - pq) + \frac{L}{I_{xx}}, \\
\dot{q} &= \frac{I_{xx} - I_{zz}}{I_{yy}} pr + \frac{I_{xz}}{I_{yy}}(r^2 - p^2) + \frac{M}{I_{yy}}, \\
\dot{r} &= \frac{I_{xx} - I_{yy}}{I_{zz}} pq + \frac{I_{xz}}{I_{zz}}(\dot{p} - qr) + \frac{N}{I_{zz}},
\end{aligned} \tag{4.12}$$

$$\tau_f \dot{a_{lc}} = -a_{lc} - \tau_f q + \left(\frac{\partial a_{lc}}{\partial \mu} \mu_x + \frac{\partial a}{\partial \mu_z} \mu_z \right) + K_{lon} \delta_{lon},$$

$$\tau_f \dot{b_{ls}} = -b_{ls} - \tau_f p + \left(\frac{\partial b_{ls}}{\partial \mu_v} \mu_y \right) + K_{lat} \delta_{lat}, \tag{4.13}$$

where the reduced state $\boldsymbol{x} \in \mathbb{R}^8 = [u\ v\ w\ p\ q\ r\ a_{lc}\ b_{ls}]^T$. And for input-independent states

based on Subsystem 2:

$$\begin{aligned}
\dot{x}^I &= c_\theta c_\phi u + (s_\theta c_\phi c_\psi - c_\phi s_\phi)v + (s_\theta c_\phi c_\psi + s_\psi s_\phi)w, \\
\dot{y}^I &= c_\theta c_\psi u + (s_\theta s_\phi s_\psi + c_\theta c_\psi)v + (c_\theta s_\phi s_\psi - c_\psi s_\phi)w, \qquad (4.14) \\
\dot{z}^I &= -s_\theta u + c_\theta s_\psi v + c_\psi c_\theta w,
\end{aligned}$$

$$\begin{aligned}
\dot{\phi} &= p + s_\theta t_\theta q + c_\phi t_\theta r, \\
\dot{\theta} &= c_\phi q - s_\phi r, \qquad (4.15) \\
\dot{\psi} &= \frac{s_\phi}{c_\theta} q + \frac{c_\phi}{c_\theta} r.
\end{aligned}$$

After state reduction, the system is still not affine-in-control. In order to apply SMC to the present control problem, there is a need to have a system presented in affine-in-control form. Therefore, the system in Equations 4.11 - 4.13 are transformed into affine-in-control with six states instead of eight. If the flapping angles are considered small and fast - which is the case in hover and low speed steady-forward flight - then Equation 4.13 can be considered as quasi-steady state such that (Khaligh, 2014):

$$a_{lc} = -\tau_f q + a_v + K_{lon}\delta_{lon}, \qquad (4.16)$$

where $a_v = \left(\frac{\partial a_{lc}}{\partial \mu}\mu_x + \frac{\partial a_{lc}}{\partial \mu_z}\mu_z\right)$ and similarly for b_{ls}. Also $\sin a_{lc} \approx a_{lc}$, $\cos a_{lc} \approx 1$, $\sin b_{ls} \approx b_{ls}$ and $\cos b_{ls} \approx 1$. This simplifies the main rotor forces and moments equations as follows:

$$X_M = -T_M a_{lc};\ Y_M = T_M b_{lc};\ Z_M = -T_M; \qquad (4.17)$$

$$L_M = K_\beta b_{ls} - Q_M a_{lc};\ M_M = K_\beta a_{lc} - Q_M b_{ls};\ N_M = -Q_M. \qquad (4.18)$$

The forces in Equation 4.17 can be simplified further by substituting Equation 4.16 into 2.17 and by algebraic manipulation and by neglecting products of control inputs and repeating this operation for all forces and moments, the system is transformed from a 8×1 non-affine system into an 6×1 affine-in-control form in the control inputs as shown:

$$\dot{\boldsymbol{x}}_{6\times 1} = \boldsymbol{f}(\boldsymbol{x})_{6\times 1} + \boldsymbol{g}(\boldsymbol{x})_{6\times 4}\boldsymbol{u}_{4\times 1}, \qquad (4.19)$$

where $\boldsymbol{x} \in \Re^6 = [u\ v\ w\ p\ q\ r]^T$ is the state vector and $\boldsymbol{f}(\boldsymbol{x})$ is a vector of nonlinear equations of state variables and $\boldsymbol{g}(\boldsymbol{x})$ is the input distribution matrix.

Remark 1. *The system in Equation 4.19 is still underactuated; as there are six states to be controlled while there are only four inputs available.*

Equation 4.19 is shown in expanded form in Equations 4.20.

$$
\left[\begin{array}{c} X \\ Y \\ Z \\ \hline L \\ M \\ N \end{array}\right] = \frac{\rho\sigma_M a_M \pi R^4 \Omega^2}{4} \left[\begin{array}{c} (\mu_z - \lambda_0)(\tau_f q - a_v) \\ -(\mu_z - \lambda_0)(\tau_f p - b_v) + \frac{R_T^4 \sigma_T a_T n^2}{R^4 \sigma_M a_M} v \\ (\mu_z - \lambda_0) \\ \hline -(\mu_z - \lambda_0)(\tau_f p - b_v) h_M - K_\beta(\tau_f p - b_v) - \frac{R_T^4 \sigma_T a_T n^2}{R^4 \sigma_M a_M} v h_T \\ -(\mu_z - \lambda_0)(\tau_f q - a_v) h_M \\ 0 \end{array}\right]
$$

$$
+ \frac{\rho\sigma_M a_M \pi R^4 \Omega^2}{4} \left[\begin{array}{cccc} -(\tau_f q - a_v)(\frac{2}{3} + \mu_x^2 + \mu_y^2) & -(\tau_f q - a_v)\mu_y & (\tau_f q - a_v)\mu_x & 0 \\ +2(\mu_z - \lambda_0)\mu_x & +(\mu_z - \lambda_0)(K_{lon} + 1) & & \\ (\tau_f p - b_v)(\frac{2}{3} + \mu_x^2 + \mu_y^2) & (\tau_f p - b_v)\mu_y & -(\tau_f p - b_v)\mu_x & -\frac{R_T^4 \sigma_T a_T n^2}{R^4 \sigma_M a_M}(u^2 + w^2 + \frac{2}{3}) \\ +(\mu_z - \lambda_0)\mu_y & & +(\mu_z - \lambda_0)(K_{lat} + 1) & \\ \frac{2}{3} + \mu_x^2 + \mu_y^2 & -\mu_y & \mu_x & 0 \\ \hline -(\tau_f p - b_v)(\frac{2}{3} + \mu_x^2 + \mu_y^2)h_M & -(\tau_f p - b_v)\mu_y h_M - K_{lat} K_\beta & (\tau_f p - b_v)\mu_x h_M & \frac{R_T^4 \sigma_T a_T n^2}{R^4 \sigma_M a_M}(u^2 + w^2 + \frac{2}{3})h_T \\ +(\mu_z - \lambda_0)\mu_y h_M & +(\mu_z - \lambda_0)K_{lat} h_M & & \\ -(\tau_f q - a_v)(\frac{2}{3} + \mu_x^2 + \mu_y^2)h_M & -(\tau_f q - a_v)\mu_y h_M & (\tau_f q - a_v)\mu_x h_M - K_{lon} K_\beta & 0 \\ +2(\mu_z - \lambda_0)\mu_x h_M & & +(\mu_z - \lambda_0)K_{lon} h_M & \\ R(\mu_z - \lambda_0)\frac{4}{3} & -R(\mu_z - \lambda_0)\mu_y & R(\mu_z - \lambda_0)\mu_x & \frac{R_T^4 \sigma_T a_T n^2}{R^4 \sigma_M a_M}(u^2 + w^2 + \frac{2}{3})l_T \end{array}\right] \boldsymbol{u} \tag{4.20}
$$

4.4 Sliding Mode Control (SMC)

Sliding mode control is a special case of variable structure control (VSC) developed in the USSR (Edwards and Spurgeon, 1998). It uses discontinuous switching control to accommodate the system's matched uncertainties and parameter variations. This ability to be invariant to uncertainties makes the SMC attractive for PFTC in literature.

SMC of the rotorcraft is developed herein, is set up in a similar architecture as it was done for the PID controllers by separating faster dynamics from slower dynamics ones. According to Enns and Keviczky (2006), the reduced state model is created using residualisation of the state equations leaving only the fast dynamics. This way, the system in Equation 4.20 is reduced to a square affine-in-control form:

$$\dot{\boldsymbol{x}}_{4\times1} = \boldsymbol{f}(\boldsymbol{x})_{4\times1} + \boldsymbol{g}(\boldsymbol{x})_{4\times4}\boldsymbol{u}_{4\times1}, \tag{4.21}$$

where $\boldsymbol{x} \in \Re^4 = [w\ p\ q\ r]^T$ is the new state vector and, $\boldsymbol{f}(\boldsymbol{x}) \in \Re^{4\times1}$ and $\boldsymbol{g}(\boldsymbol{x}) \in \Re^{4\times4}$ are similarly reduced.

Assumption 8. *The system in Equation 4.21 is Lipschitz continuous on a compact set $\Omega \subseteq \Re$ (called the flight envelope) such that a solution $\boldsymbol{f}(\boldsymbol{x}_{trim}) = 0$ is unique for the selected $\boldsymbol{x}_{trim} \in \Re^n$.*

To investigate the consequence of uncertainties and disturbances on the developed controller, the system in Equation 4.21 is rewritten as follows:

$$\dot{x}_{4\times1} = \boldsymbol{f}(\boldsymbol{x})_0 + \Delta\boldsymbol{f}(\boldsymbol{x}) + (\boldsymbol{g}(\boldsymbol{x})_0 + \Delta\boldsymbol{g}(\boldsymbol{x}))u_{4\times1} + \boldsymbol{d}(t), \tag{4.22}$$

where $\boldsymbol{f}(\boldsymbol{x})_0$ and $\boldsymbol{g}(\boldsymbol{x})_0$ are the nominal parts of the system. $\Delta\boldsymbol{f}(\boldsymbol{x})$ and $\Delta\boldsymbol{g}(\boldsymbol{x})$ are caused by system parameters variations, such as changes in the rotorcraft's mass. $\boldsymbol{d}(t)$ is the system disturbance due to external influences. The rest of the SMC developments are based on the following assumptions:

Assumption 9. *$\Delta\boldsymbol{f}$ and $\Delta\boldsymbol{g}$ are bounded such that $|\Delta\boldsymbol{f}| > \bar{\boldsymbol{f}}$ and $|\Delta\boldsymbol{g}| > \bar{\boldsymbol{g}}$, where $\bar{\boldsymbol{f}}$ and $\bar{\boldsymbol{g}}$ are vectors of positive constants.*

Assumption 10. *$\boldsymbol{d}(t)$ is globally bounded such that $\boldsymbol{d}(t) > \bar{\boldsymbol{d}}$, where $\bar{\boldsymbol{d}}$ is vector of positive constants.*

For matched uncertainties $\Delta\boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{g}(\boldsymbol{x})\boldsymbol{F}(\boldsymbol{x})\boldsymbol{u}$. Then Equation 4.22 becomes:

$$\dot{x} = \underbrace{\boldsymbol{f}(\boldsymbol{x})_0 + \boldsymbol{g}(\boldsymbol{x})_0 u}_{nominal} + \underbrace{[\boldsymbol{g}(\boldsymbol{x})\boldsymbol{F}(\boldsymbol{x}) + \Delta\boldsymbol{g}(\boldsymbol{x})]\boldsymbol{u}}_{matched\ uncertainty} + \underbrace{d(t)}_{disturbance}. \tag{4.23}$$

Unmatched uncertainties that go beyond the threshold will cause degradation to the controller and the system. Therefore, the uncertainties dealt with in this thesis are only the matched kind.

The architecture of the SMC investigated is shown in Figure 4.2. Included in the figure is the optimisation process followed to fine-tune the controller gains. The x_d and y_d translation are controlled by PID controller. This PID controller will output the desired pitch and roll angles of the rotorcraft in the F_B frame.

The first step in developing the SMC is to design the sliding surface or manifold such that the system has the desired performance when restricted to it, then followed by the development of the control law that forces the system state towards the sliding surface in finite time and keep it there; thereby encouraging the state to slide along the surface. This is achieved by using a discontinuous function chosen carefully in order for the state to be driven to the surface. The forcing of the system to this condition is known as the sliding mode (Perruquetti and Barbot, 2002).

During the sliding phase of the SMC, one sets the desired performance in a sliding surface $\boldsymbol{\sigma} = \boldsymbol{\sigma}(\boldsymbol{x})$, where $\boldsymbol{x}$ is the system state whose performance is desired.

For tracking control problem, the sliding surface σ is designed as a function of the error signal $\boldsymbol{e}$ instead of it being a function of $\boldsymbol{x}$, the state vector. The sliding surface σ is defined as follows:

$$\boldsymbol{\sigma} = \boldsymbol{\lambda e} + \dot{\boldsymbol{e}} \tag{4.24}$$

where $\boldsymbol{e}$, is the error for each system state, is defined as the difference between the current state $\boldsymbol{x}$ and the desired or reference state $\boldsymbol{x}_d$ supplied by commanded trajectory.

$$\boldsymbol{e} = [e_\phi \; e_\theta \; e_\psi \; e_x \; e_y \; e_z]^T \tag{4.25}$$

$$= [\phi - \phi_d \; \theta - \theta_d \; \psi - \psi_d \; x - x_d \; y - y_d \; z - z_d]^T. \tag{4.26}$$

$\boldsymbol{\lambda}$ is a diagonal matrix of positive constants, $\boldsymbol{\lambda} = diag(\lambda_\phi, \lambda_\theta \ldots \lambda_z) > \mathbf{0}$. The positiveness of λ is required by Hurwitz stability condition to ensure that $\boldsymbol{e} \to \mathbf{0}$ as $t \to \infty$ and hence $\boldsymbol{\sigma} = \mathbf{0}$ is attained. The time-derivative of the sliding surface results in:

$$\dot{\boldsymbol{\sigma}} = \lambda \dot{\boldsymbol{e}} + \ddot{\boldsymbol{e}}. \tag{4.27}$$

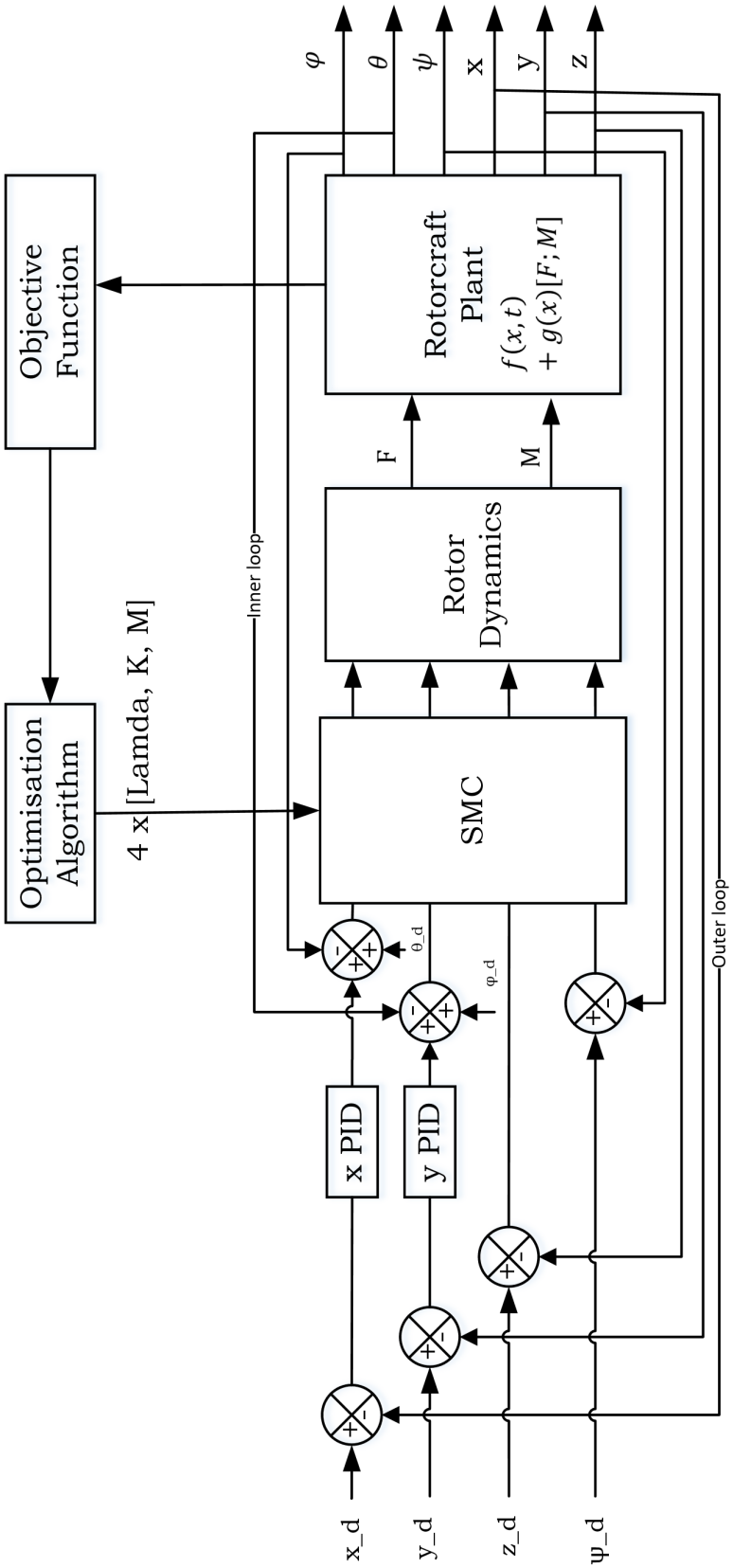


Figure 4.2: The control system architecture of the rotocraft SMC.

When the system in Equation 4.21 is confined to the designed sliding surface, this implies that a steady state has been reached and $\boldsymbol{\sigma}(\boldsymbol{x}) = 0$ and $\dot{\boldsymbol{\sigma}}(\boldsymbol{x}) = 0$. In order to satisfy this condition, a control law is designed. SMC requires the calculation of an equivalent control term which is able to maintain the state trajectory along the sliding surface once it is reached and the switching input which is responsible for driving the system towards the sliding surface. The control law is given as follows:

$$\boldsymbol{u} = \boldsymbol{u}_{eq} + \boldsymbol{u}_{sw}, \tag{4.28}$$

where $\boldsymbol{u}_{eq}$ is dependent on the desired value and the state variables, while $\boldsymbol{u}_{sw}$ is based on the sliding surface.

The time derivative of $\sigma(x)$ is given as follows:

$$\dot{\boldsymbol{\sigma}}(\boldsymbol{x}) = \frac{\partial \boldsymbol{\sigma}}{\partial \boldsymbol{x}}\dot{\boldsymbol{x}} = \frac{\partial \boldsymbol{\sigma}}{\partial \boldsymbol{x}}(\boldsymbol{f}(\boldsymbol{x}) + \boldsymbol{g}(\boldsymbol{x})\boldsymbol{u}) = \boldsymbol{0}. \tag{4.29}$$

Solving Equation 4.29 for the input u that is required for $\dot{\sigma}(x)$ to be equal to zero, is finding the solution of the equivalent control u_{eq}. For the affine-in-control system as Equation 4.21, the equivalent control, u_{eq}, is found when $\dot{\sigma} = 0$, as follows:

$$\boldsymbol{u}_{eq} = -(\boldsymbol{\sigma}(\boldsymbol{x})\boldsymbol{g}(\boldsymbol{x}))^{-1}\boldsymbol{\sigma}(\boldsymbol{x})\boldsymbol{f}(\boldsymbol{x}). \tag{4.30}$$

In order to guarantee that the system state trajectory is driven to the sliding surface $\boldsymbol{\sigma}(\boldsymbol{x}) = 0$, $\boldsymbol{u}_{sw}$ is designed such that $\boldsymbol{\sigma}(\boldsymbol{x})\dot{\boldsymbol{\sigma}}^T(\boldsymbol{x}) \leq -\boldsymbol{\kappa}\left\|\boldsymbol{\sigma}(\boldsymbol{x})\right\|$ is satisfied, where $\boldsymbol{\kappa}$ is the reachability constant vector and the inequality is called the "reachability condition". For guaranteed reachability the following is satisfied.

$$\dot{\boldsymbol{\sigma}}(\boldsymbol{x}) = \frac{\partial \boldsymbol{\sigma}}{\partial \boldsymbol{x}}\dot{\boldsymbol{x}} = \frac{\partial \boldsymbol{\sigma}}{\partial \boldsymbol{x}}(\boldsymbol{f}(\boldsymbol{x}) + \boldsymbol{g}(\boldsymbol{x})\boldsymbol{u}), \tag{4.31}$$

$$\boldsymbol{\sigma}(\boldsymbol{x})\dot{\boldsymbol{\sigma}}^T(\boldsymbol{x}) = \boldsymbol{\sigma}(\boldsymbol{x})\frac{\partial \boldsymbol{\sigma}}{\partial \boldsymbol{x}}(\boldsymbol{f}(\boldsymbol{x}) + \boldsymbol{g}(\boldsymbol{x})\boldsymbol{u}) \leq -\boldsymbol{\kappa}\left\|\boldsymbol{\sigma}(\boldsymbol{x})\right\|. \tag{4.32}$$

From Equation 4.32, the control that guarantees reachability is developed and the reaching phase is asymptotically stable in Lyapunov sense. In the subsections that follow, two different types of SMCs are developed for the rotorcraft control based on the sliding surface presented in Equation 4.24.

4.4.1 Conventional sliding mode control (CSMC)

The CSMC is developed in order to compare its performance to the super-twisting sliding mode control (STSMC). The sliding surface in Equation 4.24 is used in a traditional sense.

When the system has reached the sliding surface, both $\boldsymbol{\sigma} = 0$ and $\dot{\boldsymbol{\sigma}} = 0$. The following is required so that the states are always pointing towards the sliding surface:

$$\dot{\boldsymbol{\sigma}} = \boldsymbol{\lambda}\dot{\boldsymbol{e}} + \ddot{\boldsymbol{e}} = -\boldsymbol{K}sgn(\boldsymbol{\sigma}), \tag{4.33}$$

where $\boldsymbol{K}$ is a positive definite diagonal matrix whose size is determined by the number of controlled variables.

Substituting Equation 4.23 into Equation 4.33.

$$\lambda\dot{\boldsymbol{e}} + \ddot{\boldsymbol{x}} - \ddot{\boldsymbol{x}}_d = -\boldsymbol{K}sgn(\boldsymbol{\sigma}) \tag{4.34}$$

The control input is then found to be:

$$\boldsymbol{u} = \boldsymbol{g}(\boldsymbol{x})^{-1}(\ddot{\boldsymbol{x}}_d - \boldsymbol{f}(\boldsymbol{x}) - \boldsymbol{\lambda}\dot{\boldsymbol{e}} - \boldsymbol{K}sgn(\boldsymbol{\sigma})), \tag{4.35}$$

where $\boldsymbol{g}(\boldsymbol{x})^{-1}$ is nonsingular (Fu et al., 2012). If $\boldsymbol{g}(\boldsymbol{x})$ is not square a Moore-Penrose pseudo-inverse of $\boldsymbol{g}(\boldsymbol{x})$ can be found provided that $\boldsymbol{g}(\boldsymbol{x})\boldsymbol{g}(\boldsymbol{x})^T$ is invertible, using the following matrix multiplication. $\boldsymbol{g}(\boldsymbol{x})^T(\boldsymbol{g}(\boldsymbol{x})\boldsymbol{g}(\boldsymbol{x})^T)^{-1}$.

To prove that the selected sliding surface is attractive and therefore guarantees convergence, we use the candidate Lyapunov function $\boldsymbol{V} = \frac{1}{2}\boldsymbol{\sigma}\boldsymbol{\sigma}^T$. The sufficient condition for attraction is that $\dot{\boldsymbol{V}}$ should be negative definite. Therefore:

$$\dot{\boldsymbol{V}} = \boldsymbol{\sigma}\dot{\boldsymbol{\sigma}}, \tag{4.36}$$

$$= \boldsymbol{\sigma}(-\boldsymbol{K}sgn(\boldsymbol{\sigma})), \tag{4.37}$$

$$= -\boldsymbol{K}|\boldsymbol{\sigma}|, \tag{4.38}$$

that is, for positive definite gain matrix $\boldsymbol{K}$, $\dot{\boldsymbol{V}}$ will be negative definite. Hence, the system will be asymptotically stable.

$$\dot{\boldsymbol{\sigma}} = \boldsymbol{f}(\boldsymbol{x}) + \Delta\boldsymbol{f}(\boldsymbol{x}) + (\boldsymbol{g}(\boldsymbol{x}) + \Delta\boldsymbol{g}(\boldsymbol{x}))\boldsymbol{u} + \boldsymbol{\lambda}\dot{\boldsymbol{e}}, \tag{4.39}$$

substituting 4.35 into 4.39, we get

$$\begin{aligned}\boldsymbol{\sigma}\dot{\boldsymbol{\sigma}} &= \boldsymbol{\sigma}(-\ddot{\boldsymbol{x}}_d + \boldsymbol{f}(\boldsymbol{x}) + \Delta\boldsymbol{f}(\boldsymbol{x}) + (\boldsymbol{g}(\boldsymbol{x}) + \Delta\boldsymbol{g}(\boldsymbol{x}))(\boldsymbol{g}(\boldsymbol{x})^{-1}(\ddot{\boldsymbol{x}}_d - \boldsymbol{f}(\boldsymbol{x}) - \boldsymbol{\lambda}\dot{\boldsymbol{e}} - \boldsymbol{K}sgn(\boldsymbol{\sigma}))) + \boldsymbol{\lambda}\dot{\boldsymbol{e}}), \\ &= \boldsymbol{\sigma}\left(\Delta\boldsymbol{f}(\boldsymbol{x}) - \boldsymbol{K}sgn(\boldsymbol{\sigma}) + \Delta\boldsymbol{g}(\boldsymbol{x})\boldsymbol{g}(\boldsymbol{x})^{-1}(\ddot{\boldsymbol{x}}_d - \boldsymbol{f}(\boldsymbol{x}) - \boldsymbol{\lambda}\dot{\boldsymbol{e}} - \boldsymbol{K}sgn(\boldsymbol{\sigma}))\right).\end{aligned} \tag{4.40}$$

If we define $\ddot{\boldsymbol{x}}_d - \lambda\dot{\boldsymbol{e}} = \boldsymbol{\xi}$, then

$$\begin{aligned}\boldsymbol{\sigma}\{\Delta\boldsymbol{f}(\boldsymbol{x}) + \Delta\boldsymbol{g}(\boldsymbol{x})\boldsymbol{g}(\boldsymbol{x})^{-1}\boldsymbol{\xi} - \Delta\boldsymbol{g}(\boldsymbol{x})\boldsymbol{g}(\boldsymbol{x})^{-1}\boldsymbol{f}(\boldsymbol{x})\} - \boldsymbol{K}sgn(\boldsymbol{\sigma}) - \boldsymbol{K}\Delta\boldsymbol{g}(\boldsymbol{x})\boldsymbol{g}(\boldsymbol{x})^{-1}sgn(\boldsymbol{\sigma}), \\ \leq -\boldsymbol{\kappa}|\boldsymbol{\sigma}|\end{aligned} \tag{4.41}$$

and using the fact that

$$\begin{aligned} \boldsymbol{K}sgn(\boldsymbol{\sigma}) &< \boldsymbol{K}||\boldsymbol{\sigma}||_2, \\ \boldsymbol{K}\Delta\boldsymbol{g}(\boldsymbol{x})\boldsymbol{g}(\boldsymbol{x})^{-1}sgn(\boldsymbol{\sigma}) &\leq \boldsymbol{K}||\boldsymbol{\sigma}||_2 \cdot ||\Delta\boldsymbol{g}(\boldsymbol{x})|| \cdot ||\boldsymbol{g}(\boldsymbol{x})^{-1}sgn(\boldsymbol{\sigma}), || \end{aligned} \tag{4.42}$$

we get

$$\begin{aligned} ||\boldsymbol{\sigma}|| \cdot \{||\Delta\boldsymbol{f}(\boldsymbol{x})|| + ||\boldsymbol{d}|| + ||\Delta\boldsymbol{g}(\boldsymbol{x})|| \cdot ||\boldsymbol{g}(\boldsymbol{x})^{-1}\boldsymbol{\xi}|| - \boldsymbol{K} + \boldsymbol{K}||\Delta\boldsymbol{g}(\boldsymbol{x})|| \cdot ||\boldsymbol{g}(\boldsymbol{x})^{-1}sgn(\boldsymbol{\sigma})||\} \\ \leq -\boldsymbol{\kappa}||\boldsymbol{\sigma}||, \end{aligned} \tag{4.43}$$

$$\begin{aligned} ||\boldsymbol{\sigma}|| \cdot \{\bar{\boldsymbol{f}} + \bar{\boldsymbol{d}} + \bar{\boldsymbol{g}}||\boldsymbol{g}(\boldsymbol{x})^{-1}\boldsymbol{\xi}|| - \boldsymbol{K} + \boldsymbol{K}\bar{\boldsymbol{g}}||\boldsymbol{g}(\boldsymbol{x})^{-1}sgn(\boldsymbol{\sigma})||\} \\ \leq -\boldsymbol{\kappa}||\boldsymbol{\sigma}||, \end{aligned} \tag{4.44}$$

then solve for $\boldsymbol{K}$

$$\boldsymbol{K} = \frac{\bar{\boldsymbol{f}} + \bar{\boldsymbol{d}} + \bar{\boldsymbol{g}}||\boldsymbol{g}(\boldsymbol{x})^{-1}\boldsymbol{\xi}|| + \boldsymbol{\kappa}}{1 - \bar{\boldsymbol{g}}||\boldsymbol{g}(\boldsymbol{x})^{-1}sgn(\boldsymbol{\sigma})||}. \tag{4.45}$$

In order to guarantee that $\boldsymbol{K} > \boldsymbol{0}$, the following sufficient condition is enforced (Chang, 1991):

$$\bar{\boldsymbol{g}}||\boldsymbol{g}(\boldsymbol{x})^{-1}sgn(\boldsymbol{\sigma})|| < 1. \tag{4.46}$$

During the optimisation process that is discussed in the sequel, this condition is always checked so that the numerical experiment avoids the situation where the system dynamics are ill-defined.

4.4.2 Super-twisting sliding mode control (STSMC)

The sliding surface for the STSMC is the same as the one for CSMC. STSMC differs in the type of switching function used (Nagesh and Edwards, 2014). STSMC was developed to control systems with relative degree one in order to reduce chattering. The switching control input is found as follows:

$$\boldsymbol{u}_{sw} = -\boldsymbol{K}\sqrt{|\boldsymbol{\sigma}|}sgn(\boldsymbol{\sigma}) + \boldsymbol{v}, \tag{4.47}$$

$$\dot{\boldsymbol{v}} = -\boldsymbol{M}sgn(\boldsymbol{\sigma}), \tag{4.48}$$

where $\boldsymbol{K}$ and $\boldsymbol{M}$ are positive definite diagonal matrices. In this case u_{sw} is continuous since $\sqrt{|\sigma|}sgn(\sigma)$ is continuous and $v = M\int sgn(\sigma)dt$ is also continuous due to it being buffered by the integral.

If we rewrite

$$\dot{\boldsymbol{\sigma}}(\boldsymbol{x}) = \frac{\partial \boldsymbol{\sigma}}{\partial \boldsymbol{x}}\dot{\boldsymbol{x}} = \frac{\partial \boldsymbol{\sigma}}{\partial \boldsymbol{x}}(\boldsymbol{f}(\boldsymbol{x}) + \boldsymbol{g}(\boldsymbol{x})\boldsymbol{u}) = \boldsymbol{h}(t, \boldsymbol{x}) + \boldsymbol{b}(t, \boldsymbol{x})\boldsymbol{u} \tag{4.49}$$

and there exist positive diagonal matrices of constants $\boldsymbol{C}$, $\boldsymbol{K}_m$, and $\boldsymbol{K}_M$ such that $\boldsymbol{K}_m \leq \boldsymbol{b}(t, \boldsymbol{x}) \leq \boldsymbol{K}_M$. To guarantee the finite robust convergence of the controller parameters $\boldsymbol{K}$ and $\boldsymbol{M}$, the necessary condition that must be satisfied is given by:

$$\frac{2(\boldsymbol{K}_m\boldsymbol{M} + \boldsymbol{C})^2}{\boldsymbol{K}^2\boldsymbol{K}_m^2(\boldsymbol{K}_m\boldsymbol{M} - \boldsymbol{C})} < 1 \tag{4.50}$$

This condition is achieved when $\boldsymbol{K}_m\boldsymbol{M} > \boldsymbol{C}$.

Theorem 1. With $\boldsymbol{K}_m\boldsymbol{M} > \boldsymbol{C}$ and $\boldsymbol{K}$ sufficiently large, the controller in 4.48 guarantees $\boldsymbol{\sigma} = \dot{\boldsymbol{\sigma}} = \boldsymbol{0}$ in system 4.49, which attracts the trajectories in finite time. The control u enters in finite time the segment $[-U_m, U_m]$ and stays there. It never leaves the segment, if the initial value is inside at the beginning.

Proof. See in (Shtessel et al., 2014) □

Therefore, to guarantee stability the values must be $\boldsymbol{M} > \boldsymbol{K}_m^{-1}C$ and $\boldsymbol{K}$ is chosen as:

$$\boldsymbol{K}^2 = \frac{2(\boldsymbol{K}_m\boldsymbol{M} + \boldsymbol{C})^2}{\boldsymbol{K}_m^2(\boldsymbol{K}_m\boldsymbol{M} - \boldsymbol{C})} \tag{4.51}$$

These values can be found analytically in a trivial system. For a coupled MIMO system such as the rotorcraft, however, it is more complex. The computational intelligence optimisation algorithms are used to find these values.

The scheme used to find the switching function of the STSMC is shown in Figure 4.3 (Anderson Azzano et al., 2019).

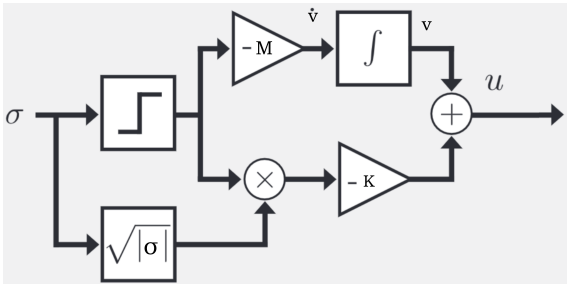


Figure 4.3: The scheme used to find the STSMC switching function.

4.4.3 Dealing with chattering

As mentioned, conventional SMC exhibits chattering due to its discontinuous nature. Chattering is caused by the following (Liu et al., 2020):

- Unmodelled dynamics;
- Digital controller with finite sampling rate.

Without uncertainty the equivalent control would be exact and keep the system state trajectory on the sliding surface. Also with infinite sampling there would be no chattering. However such systems do not exist in real life.

To reduce chattering, instead of the $sgn(\cdot)$ function, we use the $tanh(\cdot)$ function (Liu, 2017). To be used as a substitute, it is shown that the following results must hold:

Lemma 1. *(Liu, 2017) Given a scalar x and a positive scalar ϵ, then the following inequality holds:*

$$x \tanh\left(\frac{x}{\epsilon}\right) \leq \left| x \tanh\left(\frac{x}{\epsilon}\right) \right| \leq |x| \left| \tanh\left(\frac{x}{\epsilon}\right) \right|. \tag{4.52}$$

Then it follows that:

$$x \tanh\left(\frac{x}{\epsilon}\right) = x \frac{e^{\frac{x}{\epsilon}} - e^{-\frac{x}{\epsilon}}}{e^{\frac{x}{\epsilon}} + e^{-\frac{x}{\epsilon}}} = \frac{1}{e^{2\frac{x}{\epsilon}} + 1} x(e^{2\frac{x}{\epsilon}} - 1). \tag{4.53}$$

Since:

$$e^{2\frac{x}{\epsilon}} - 1 \geq 0 \quad \text{if} \quad x \geq 0, \tag{4.54}$$

$$e^{2\frac{x}{\epsilon}} - 1 < 0 \quad \text{if} \quad x < 0, \tag{4.55}$$

then the following is always true:

$$x(e^{2\frac{x}{\epsilon}} - 1) \geq 0, \tag{4.56}$$

therefore:

$$x \tanh\left(\frac{x}{\epsilon}\right) = \frac{1}{e^{2\frac{x}{\epsilon}} + 1} x(e^{2\frac{x}{\epsilon}} - 1) \geq 0. \tag{4.57}$$

Although, the use of $tanh(\cdot)$ reduces chattering in the system, it does not guarantee asymptotic tracking. Depending on the choice of ϵ the controller can be said to provide *uniform ultimate boundedness* instead.

The action of chattering using tanh is depicted by Taylor's series expansion as follows:

$$\tanh(\sigma) = \sigma(1 - \sigma^2/3 + (2/15)\sigma^4 + \dots) \quad \text{for} \quad |\sigma| \leq \pi/2. \tag{4.58}$$

Chattering occurs in the vicinity of the sliding surface, $\sigma \approx 0$. Therefore the series can be truncated to be

$$\tanh(\sigma) \approx \sigma. \tag{4.59}$$

According to Maharaj (1994), the smoothing of the sliding surface is defined as follows:

$$\dot{\boldsymbol{\sigma}} = \Delta \boldsymbol{f}(\boldsymbol{x}) + \Delta \boldsymbol{g}(\boldsymbol{x})\boldsymbol{g}(\boldsymbol{x})^{-1}\boldsymbol{\xi} - \Delta \boldsymbol{g}(\boldsymbol{x})\boldsymbol{g}(\boldsymbol{x})^{-1}\boldsymbol{f}(\boldsymbol{x}) - \boldsymbol{K}\boldsymbol{\sigma} - \boldsymbol{K}\Delta \boldsymbol{g}(\boldsymbol{x})\boldsymbol{g}(\boldsymbol{x})^{-1}\boldsymbol{\sigma}, \tag{4.60}$$

$$\dot{\boldsymbol{\sigma}} + \boldsymbol{K}(1 + \Delta \boldsymbol{g}(\boldsymbol{x})\boldsymbol{g}(\boldsymbol{x})^{-1})\boldsymbol{\sigma} = \Delta \boldsymbol{f}(\boldsymbol{x}) + \Delta \boldsymbol{g}(\boldsymbol{x})\boldsymbol{g}(\boldsymbol{x})^{-1}(\boldsymbol{\xi} - \boldsymbol{f}(\boldsymbol{x})). \tag{4.61}$$

This represents a first-order low-pass filter with the uncertainties as inputs and $\boldsymbol{K}(1 + \Delta \boldsymbol{g}(\boldsymbol{x})\boldsymbol{g}(\boldsymbol{x})^{-1})$ defining the cut-off frequency. Edwards and Spurgeon (1998) showed that when the switching control signal is passed through a low-pass filter the equivalent control is recovered. Therefore, using the low-pass filter action of the $tanh(\cdot)$, the equivalent control is used to maintain the system trajectory on the sliding surface, while still being invariant to uncertainties and disturbances.

4.4.4 Controller tuning strategies

The parameters of the SMC controllers discussed above are found using the manual tuning method first, then after by the computational intelligence optimisation algorithms as discussed in Chapter 3. Manual tuning is effective in small and simple system. However, in large, coupled MIMO systems with many DOF, manual tuning becomes tedious and difficult due to the optimisation problem's dimension explosion. For this reason computational intelligence algorithms are used.

The tuning algorithms are used to find the candidate solutions $s_i = \{\lambda,\ K\}$ for CSMC and $s_i = \{\lambda,\ K,\ M\}$ for STSMC respectively. The computed solutions are evaluated according to the following objective function:

$$\begin{aligned} J = \frac{1}{T}\int_0^T \Bigg[&\left(\frac{x_d - x}{x_{max}}\right)^2 + \left(\frac{y_d - y}{y_{max}}\right)^2 + \left(\frac{z_d - z}{z_{max}}\right)^2 + \left(\frac{\theta_d - \theta}{\theta_{max}}\right)^2 + \left(\frac{\phi_d - \phi}{\phi_{max}}\right)^2 \\ &+ \left(\frac{\psi_d - \psi}{\psi_{max}}\right)^2 + \left(\frac{\delta_{col}}{\delta_{col_{max}}}\right)^2 + \left(\frac{\delta_{lon}}{\delta_{lon_{max}}}\right)^2 + \left(\frac{\delta_{lat}}{\delta_{lat_{max}}}\right)^2 + \left(\frac{\delta_{ped}}{\delta_{ped_{max}}}\right)^2 \Bigg] dt. \end{aligned} \tag{4.62}$$

Two optimisation algorithms are investigated in this chapter. These are PSO and ACO. It was seen in the previous chapter that ALO and FA were not effective in the benchmarking exercise.

4.5 Simulation Results and Discussion

A numerical simulation of the rotorcraft system in Figure 4.19 and the SMC controllers for trajectory tracking of z and regulating $\phi,\ \theta,\ \psi$ were designed and implemented in MATLAB®/ Simulink®. A Bogacki-Shampine solver was used with the sampling time of 1 kHz.

4.5.1 SMC parameter optimisation

The following different SMC controller gains optimisation techniques were employed:

1. Manual tuning (MT) method;
2. Particle Swarm Optimisation;
3. Continuous Ant Colony Optimisation.

The reason for choosing these two algorithms is that the ACO outperformed in the previous chapter and the PSO has a reputation for generally performing well across a wide range of optimisation problems (Platt et al., 2018). These techniques were used to find λ, K for the SMC, with eight variables, and λ, K, M for the STSMC parameters, with 12 variables. The first set of non-optimal SMC controllers' gains were found using manual tuning (trial and error) method until the desired performance was attained. The second set of optimised SMC controllers' gains was found using PSO. The final optimised SMC controllers' parameters were found using continuous ACO. These methods were also applied to STSMC tuning.

The values of SMC controllers' gains found using the MT, PSO and ACO techniques are given in Table 4.1. The convergence history of the SMC optimisation is shown in Figure 4.4.

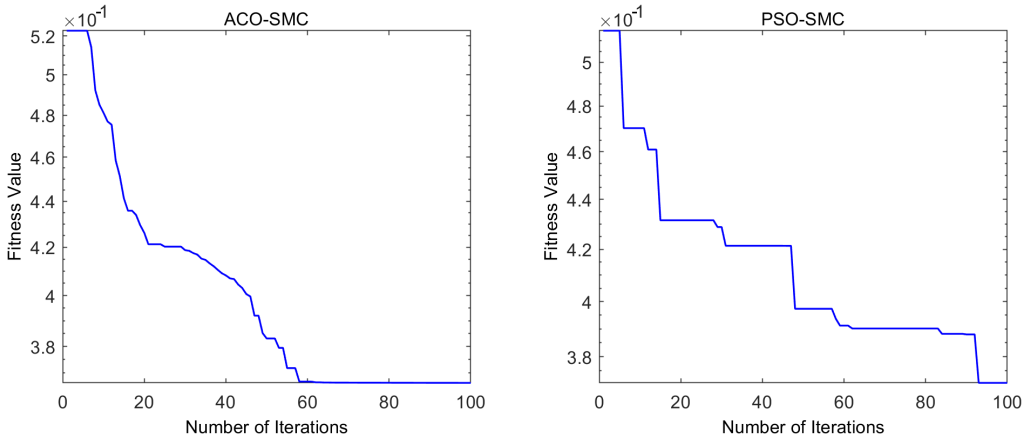


Figure 4.4: SMC convergence history of the fitness value for ACO and PSO over 100 iterations.

Table 4.1: Tuned SMC parameters using the proposed optimisation techniques.

SMC z	λ	K	M
MT	200	20.0	–
PSO	153	13.8	–
ACO	154	12.7	–
SMC ϕ	λ	K	M
MT	5.00	1.00	–
PSO	6.61	4.30	–
ACO	18.0	15.3	–
SMC θ	λ	K	M
MT	5.00	1.00	–
PSO	7.47	1.72	–
ACO	20.6	3.09	–
SMC ψ	λ	K	M
MT	40.0	20.0	–
PSO	35.4	4.18	–
ACO	33.2	14.9	–

For each of the optimisation algorithms, twenty trials were executed and the best result was chosen based on the gains that give the lowest cost function value. Figure 4.5 shows how the system fitness value changes as the iteration increases. Notice that the ACO algorithm converges after 51 iterations, and the PSO converges after 74 iterations for SMC. For STSMC the PSO converges faster than the ACO as shown in Figure 4.5. The optimal gains returned from the STSMC tuning process for each optimisation algorithm are listed in Table 4.2.

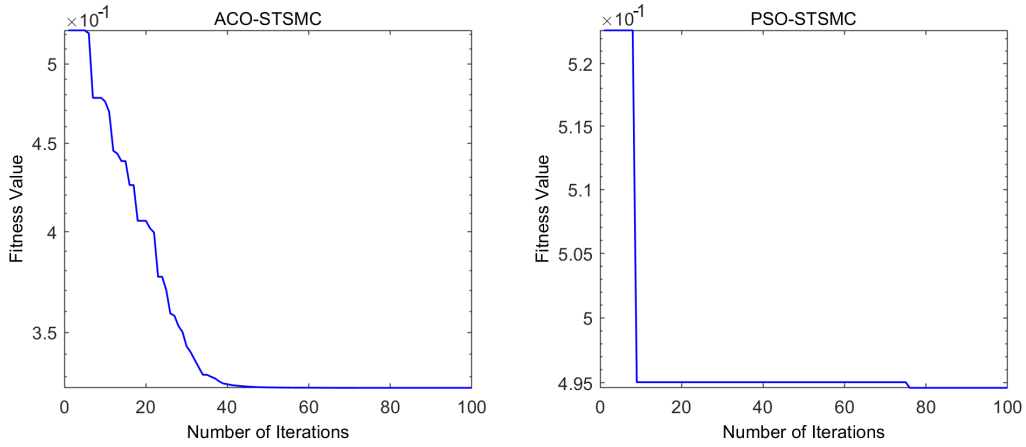


Figure 4.5: STSMC convergence history of the fitness value for ACO and PSO over 100 iterations.

Table 4.2: Tuned super-twisting SMC parameters using the proposed optimisation techniques.

STSMC z	λ	K	M
MT	200	20.0	5.00
PSO	245	40.2	3.37
ACO	116	79.3	17.9
STSMC ϕ	λ	K	M
MT	5.00	2.00	10.0
PSO	4.98	1.22	3.73
ACO	8.45	1.36	6.90
STSMC θ	λ	K	M
MT	5.00	2.00	10.0
PSO	5.33	4.46	2.00
ACO	7.67	3.10	5.96
STSMC ψ	λ	K	M
MT	40.0	20.0	10.0
PSO	27.7	30.5	2.88
ACO	24.9	38.2	11.6

4.5.1.1 Hover

The gain values found for the MT, PSO and the ACO controllers are compared for the rotorcraft in hover to determine the best performing controller. The position tracking graphs shown in Figure 4.6, the SMC tracking for the PSO-based SMC is superior to the others. The manually-tuned SMC has a higher z tracking error than the other methods. However, this is below the specified 20 cm steady-state error.

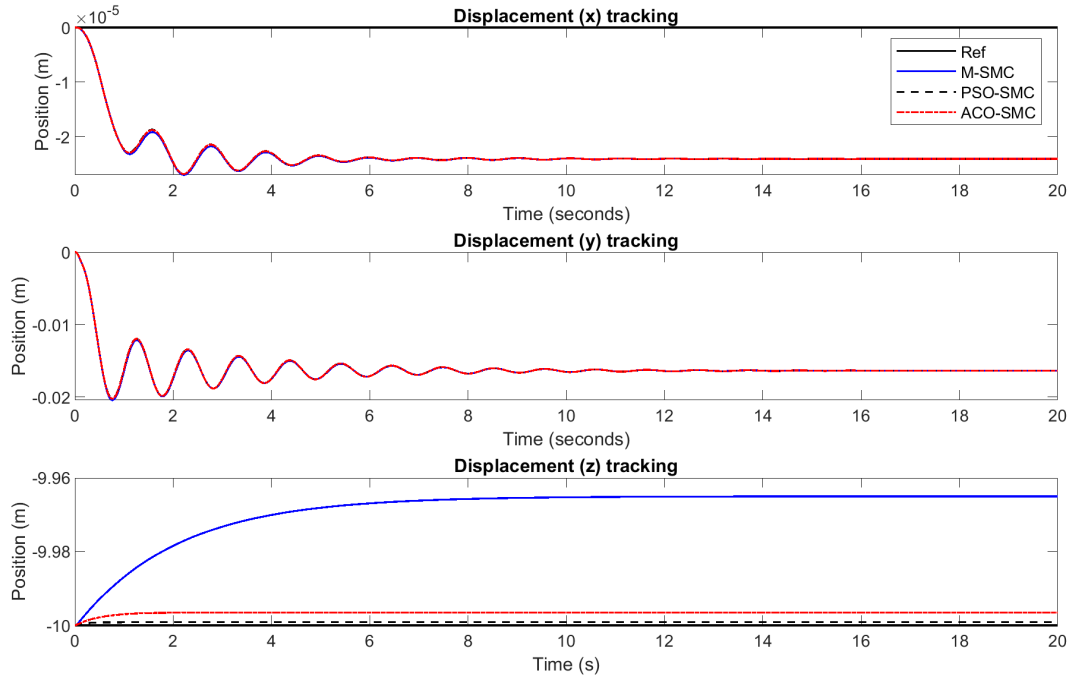


Figure 4.6: Comparing PSO and ACO SMC displacement tracking in hover.

The control effort produced in each axis is shown to be within the specified range as illustrated in Figure 4.8. There is evidence that the manually-tuned SMC applied less collective than the other controllers. This explains why the z tracking error is higher in Figure 4.6.

Although, the PSO-based SMC has the highest pitch error, as shown in Figure 4.7, in the sub-figure, however the effectiveness of produced force is similar for the three methods as depicted in Figure 4.8. Also, both the ACO-based SMC and the PSO-based SMC exhibit transient oscillations before settling.

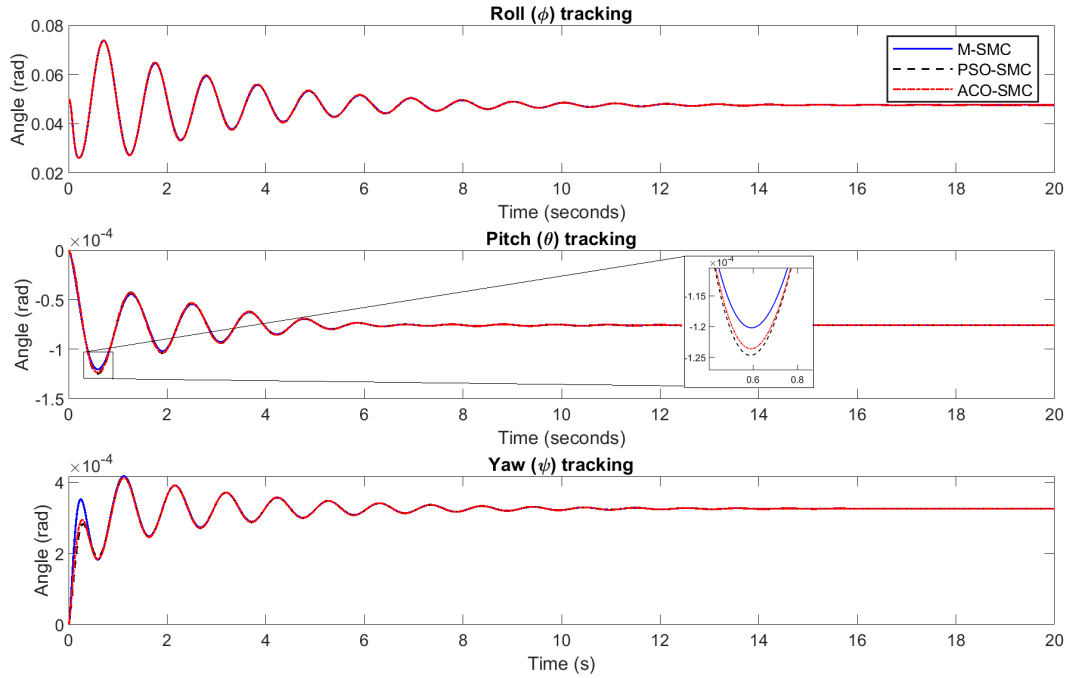


Figure 4.7: Comparing PSO and ACO SMC Euler angles' tracking in hover.

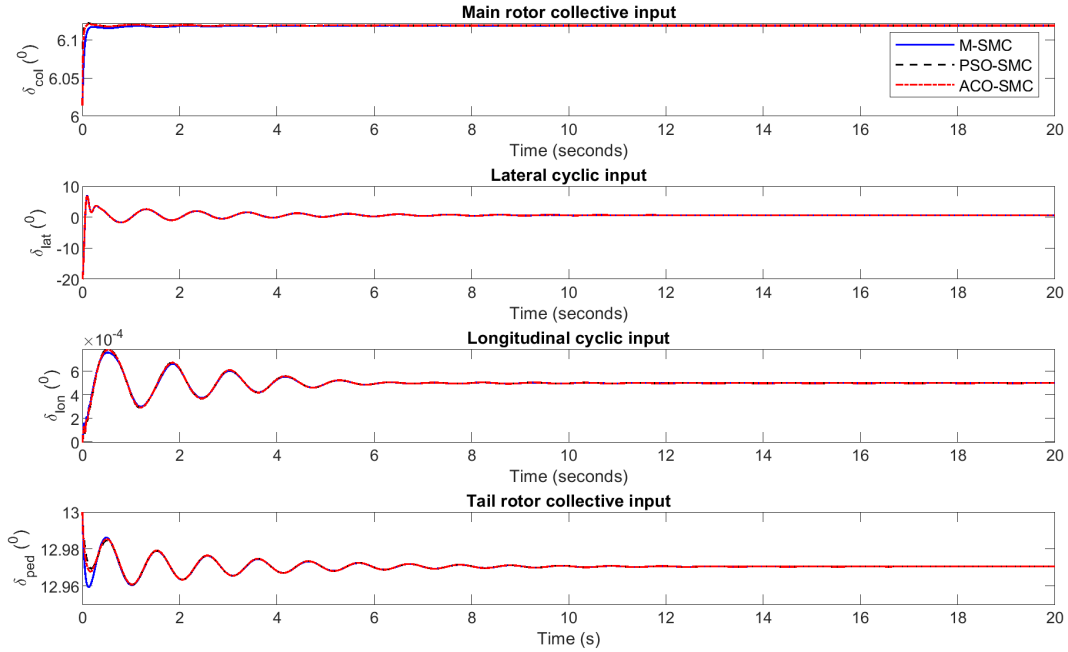


Figure 4.8: Comparing PSO and ACO SMC control inputs for the rotorcraft in hover.

The graphs for the STSMC controllers are shown in Figures 4.9 to 4.11. When comparing the two controller types, it can be observed that while the SMC exhibits steady-state error, the STSMC does not. Additionally, it is notable that the control inputs for both types of controllers do not exhibit chattering and instead present smooth force control. This suggests that the inclusion of *tanh* attenuates the chattering normally found in

conventional SMC. On the other hand, the STSMC is designed to not have chattering

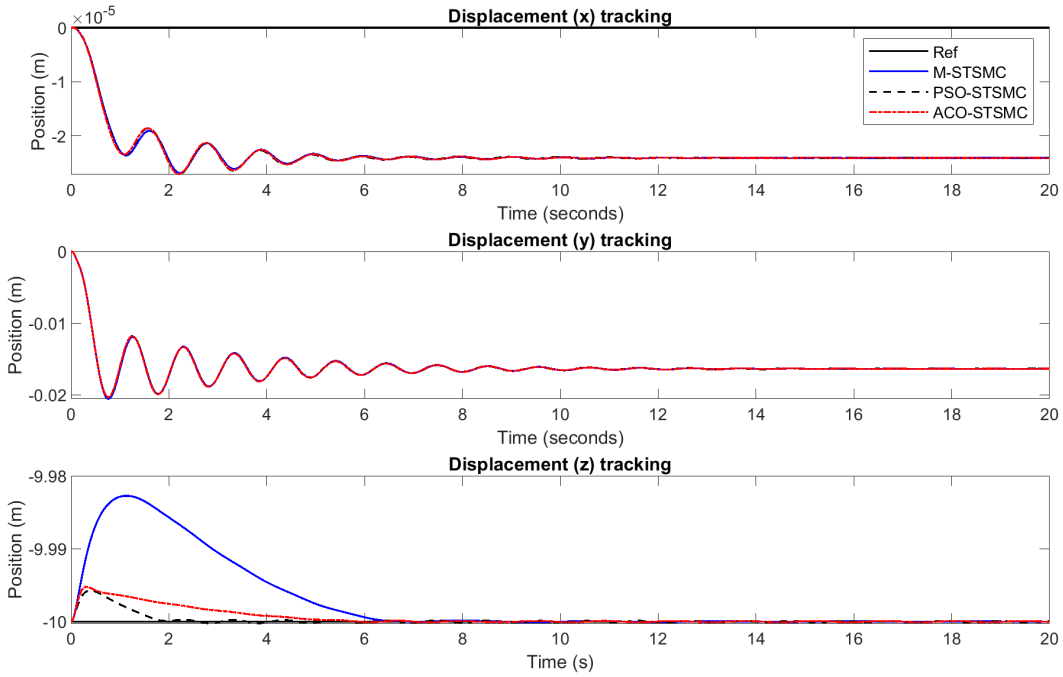


Figure 4.9: Comparing PSO and ACO STSMC displacement tracking in hover.

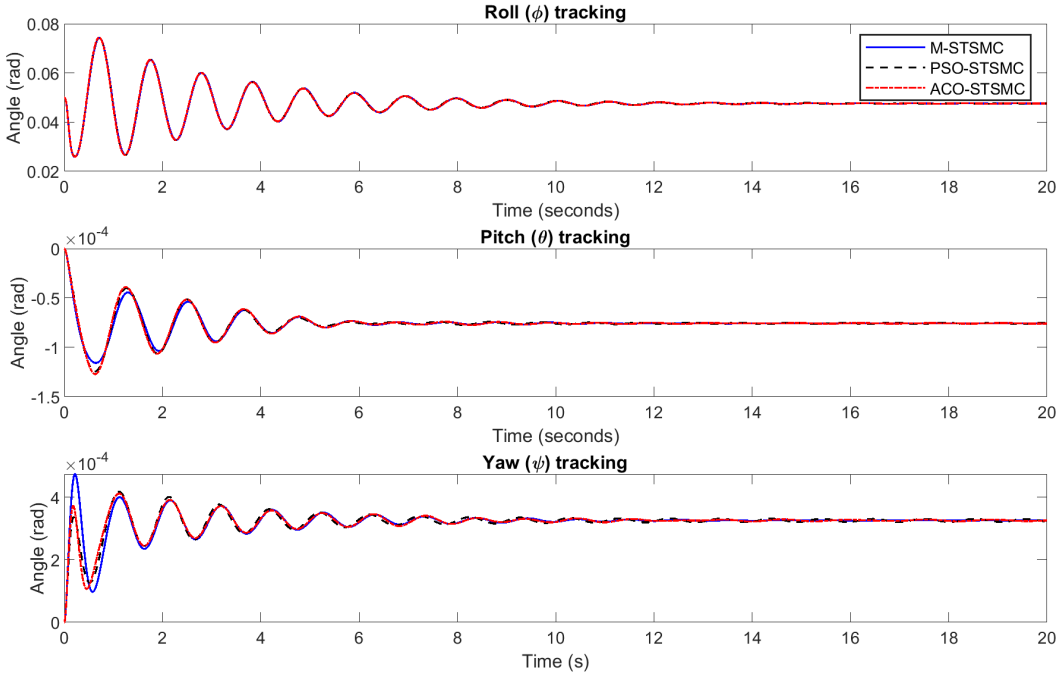


Figure 4.10: Comparing PSO and ACO STSMC Euler angles' tracking in hover.

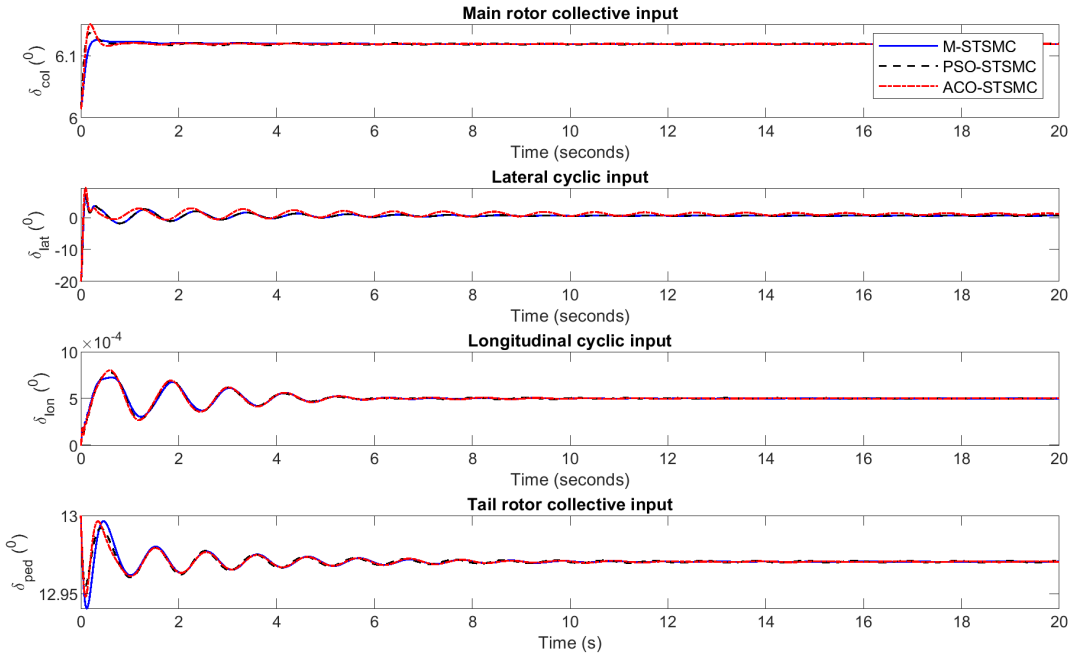


Figure 4.11: Comparing PSO and ACO STSMC control inputs for the rotorcraft in hover.

.

4.5.1.2 Forward flight

The controllers are further compared for the rotorcraft in 10 m/s forward flight to determine the best performing. Unlike the PID controller set presented in the previous chapter, the SMC variants do not require retuning to move from hover to forward flight as can be seen in Figures 4.12-4.14. The position tracking graphs shown in Figure 4.12, the tracking for the PSO-based STSMC is superior to the others. The ACO-based SMC has a higher z tracking error than the other methods. However, this is below the specified 20 cm steady-state error.

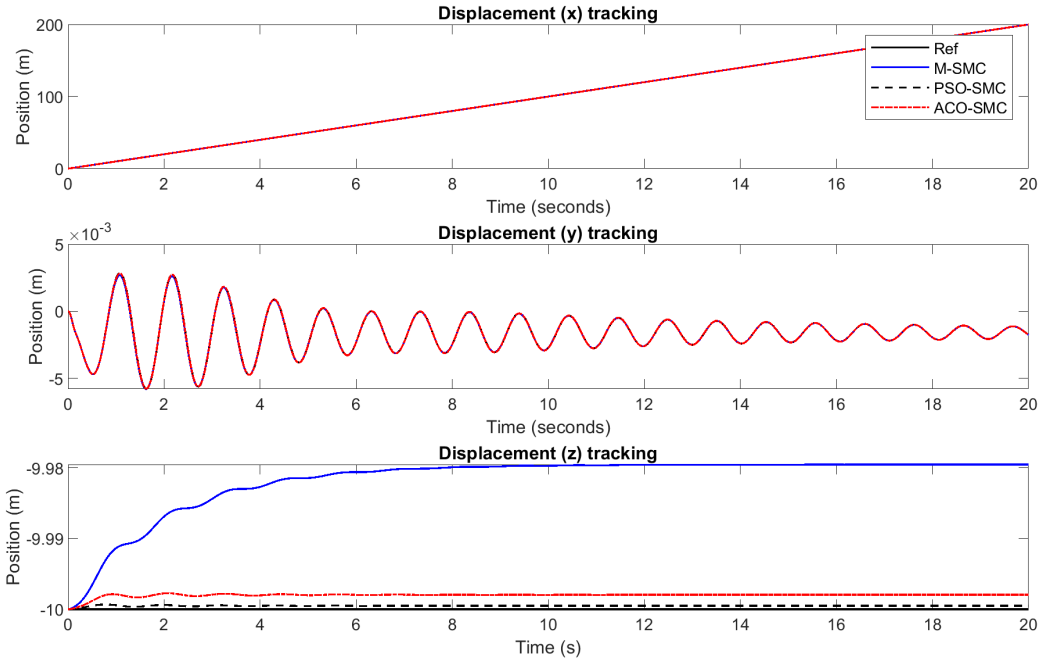


Figure 4.12: Comparing PSO and ACO SMC displacement tracking for the rotorcraft in forward flight.

The control effort produced in each axis is shown to be within the specified range as illustrated in Figure 4.14. It is worth noting that the control input does not have chattering as expected.

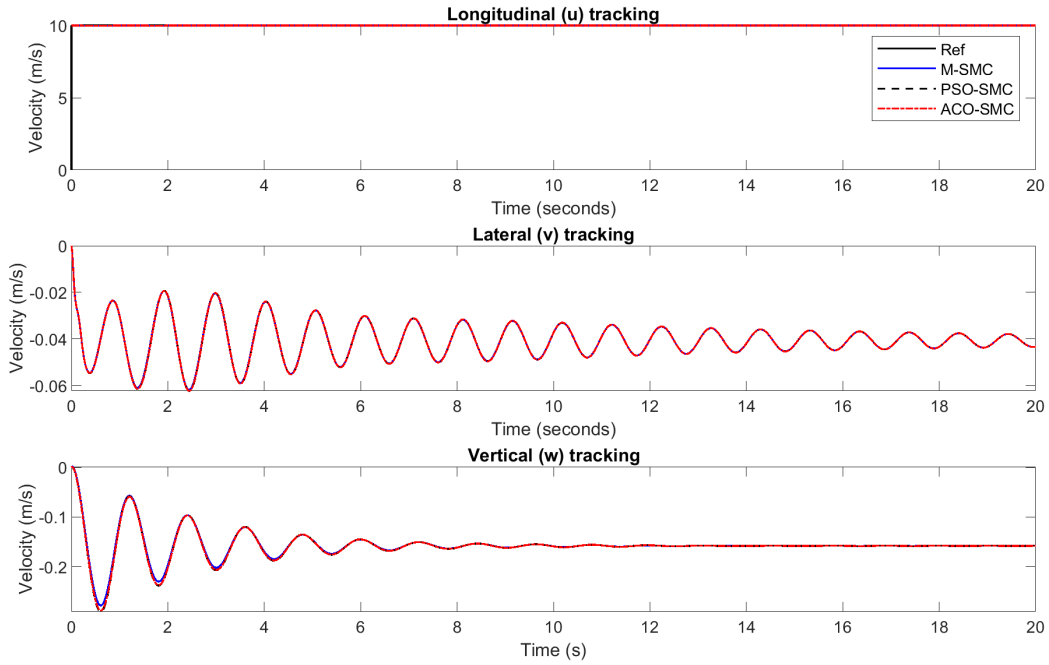


Figure 4.13: Comparing PSO and ACO SMC velocities for the rotorcraft in forward flight.

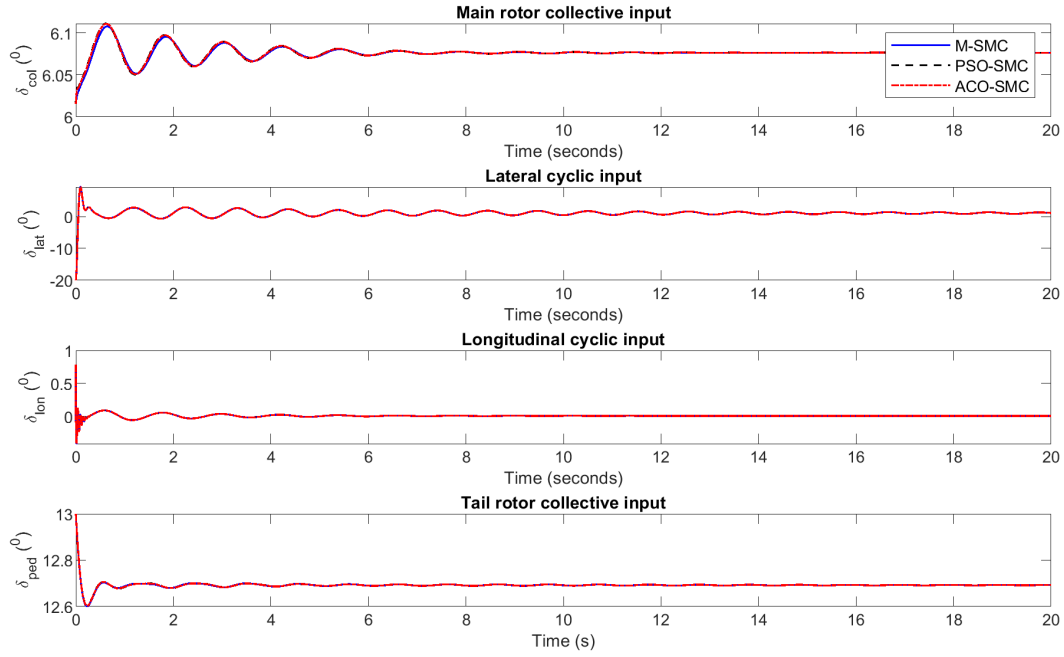


Figure 4.14: Comparing PSO and ACO SMC control inputs for the rotorcraft in forward flight.

The corresponding STSMC outputs for forward flight are shown in Figures 4.15-4.17. The noted difference to SMC in position is that all STSMC position converge to a zero steady-state error even in forward flight.

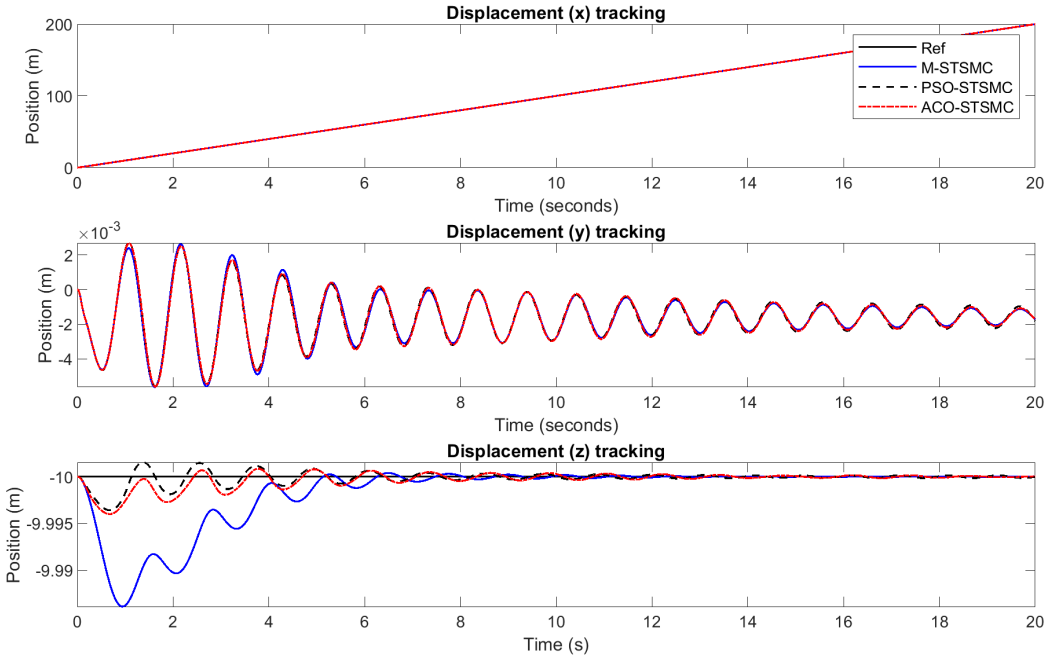


Figure 4.15: Comparing PSO and ACO STSMC displacement tracking for the rotorcraft in forward flight.

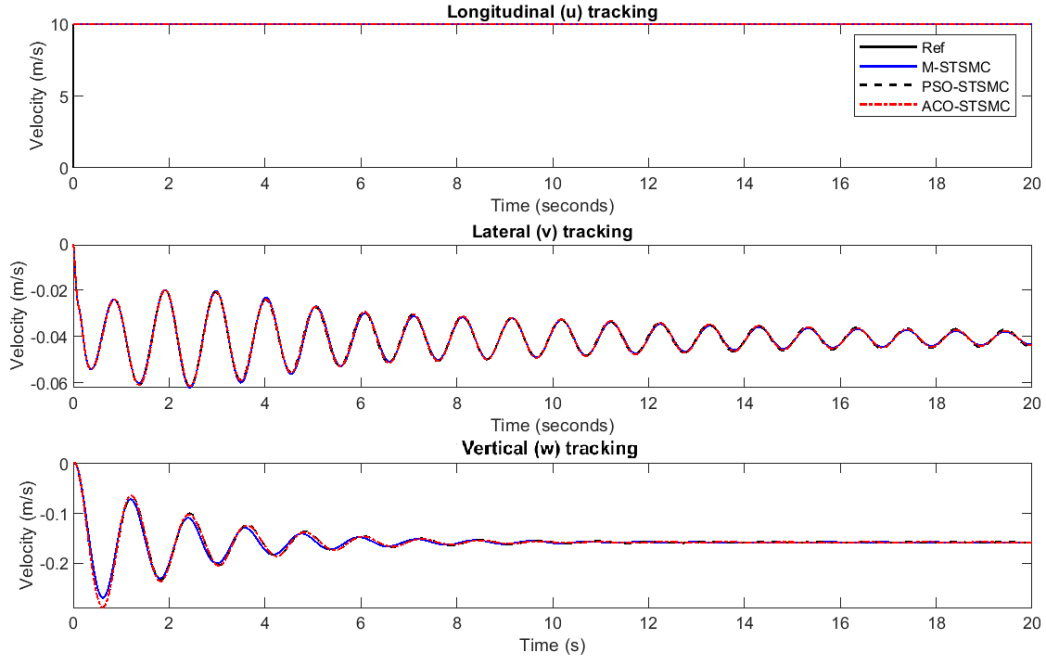


Figure 4.16: Comparing PSO and ACO STSMC velocities for the rotorcraft in forward flight.

.

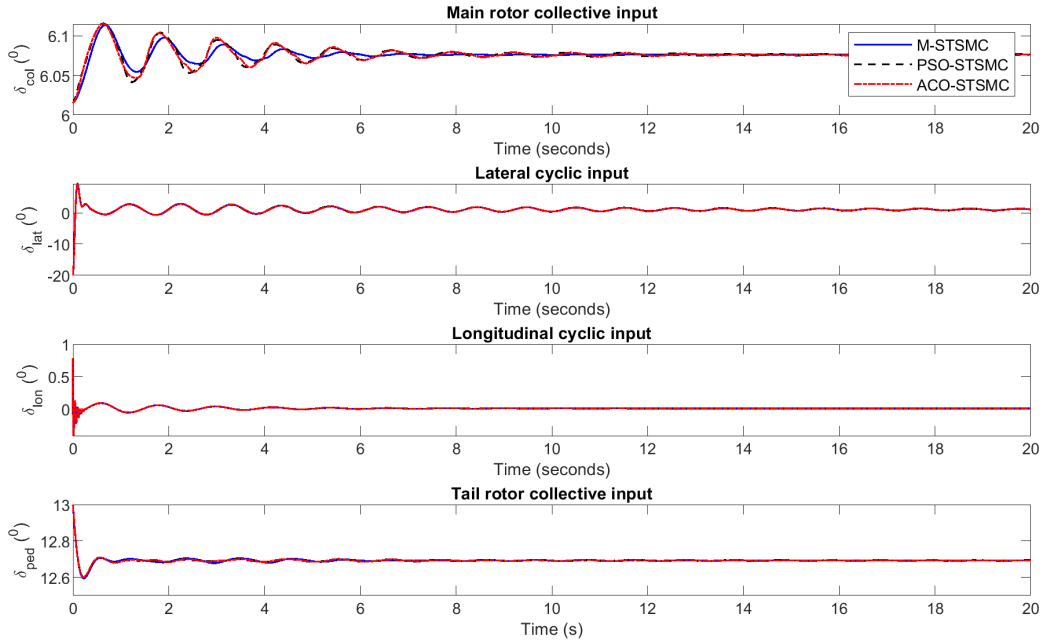


Figure 4.17: Comparing PSO and ACO STSMC control inputs for the rotorcraft in forward flight.

To gain insight into the internal stability behaviour of the SMC and the STSMC controllers developed, the phase-plane portraits are analysed. The z-axis tracking error is utilised. The error is plotted against the error rate. The phase-plane portraits of the hover and forward flight for SMC are shown in Figure 4.18. The corresponding phase-plane portraits for STSMC are shown in Figure 4.19 It can be seen that the SMC controller do not converge to a zero steady state. Instead they remain bounded in a small region. This shows bounded-input bounded-output stability

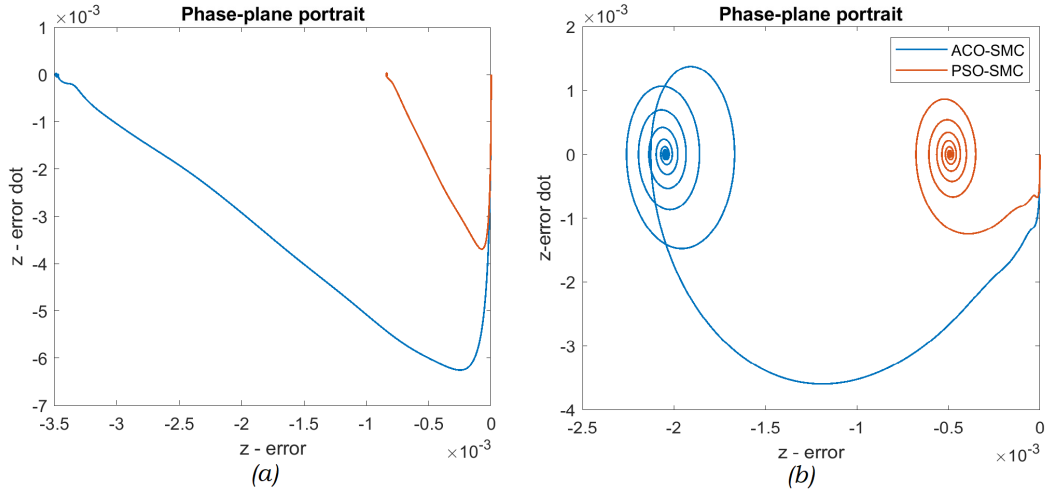


Figure 4.18: SMC phase-plane portraits of the hover (a) and forward flight (b).

On the other hand, the STSMC portrait shows convergence to a zero steady-state error. This shows that when the initial conditions are in a given small area, the error trajectory does not escape a given region and subsequently converges to zero in finite time. The circles shows that the STSMC oscillates before settling at zero. The SMC does not oscillate as can be seen by the linear sliding surface. The non-zero convergence can be attributed to the $tanh()$ which was used for chattering attenuation.

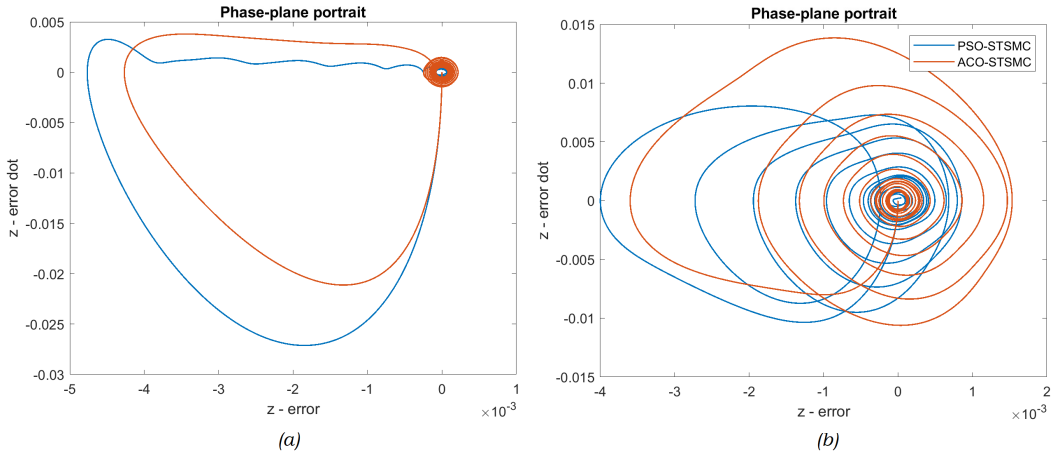


Figure 4.19: STSMC phase-plane portraits of the hover (a) and forward flight (b).

4.5.2 Robustness to parameter variations and external disturbances

The rotorcraft is subjected to change in its parameters and the performance of the designed controllers is analysed. The following parameter variations and disturbances are imposed on the rotorcraft and to see how the respective controllers react:

- The mass of the rotorcraft will be varied from 50, 100 and 200%;
- The moments of inertia will be varied from 50, 100 and 200%;
- The rejection of wind gust of up to $5m/s$;
- The LOE of one of the three actuators by 10, 20 and 40% effectiveness.

SMC is known for its robustness. In the control of the rotorcraft, a number of tests are conducted to evaluate the effectiveness of SMC and STSMC when the system parameters are changed. The PSO-based STSMC was found to be the superior of the developed SMC controllers. When only one controller is tested, the PSO-based STSMC is assumed.

The first test is the change in rotorcraft mass. It can be seen from Figure 4.20 that the controller is able to keep the rotorcraft in stable hover for up to 200% changes in the mass of the aircraft. The control inputs to stabilise the system are shown in Figure 4.21.

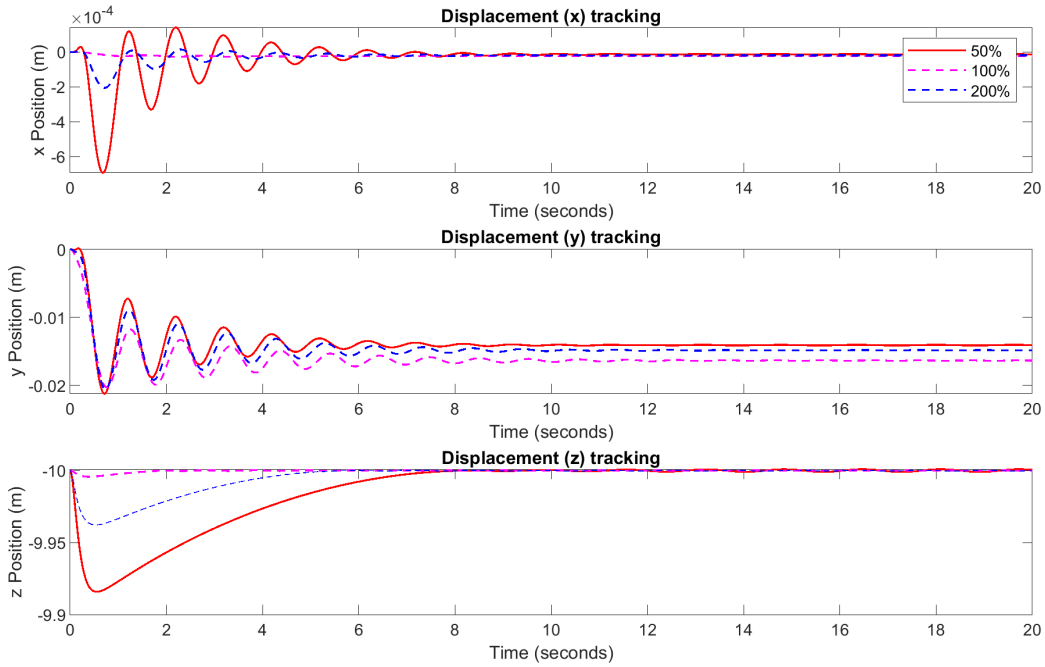


Figure 4.20: Comparing robustness to mass variations for position in hover.

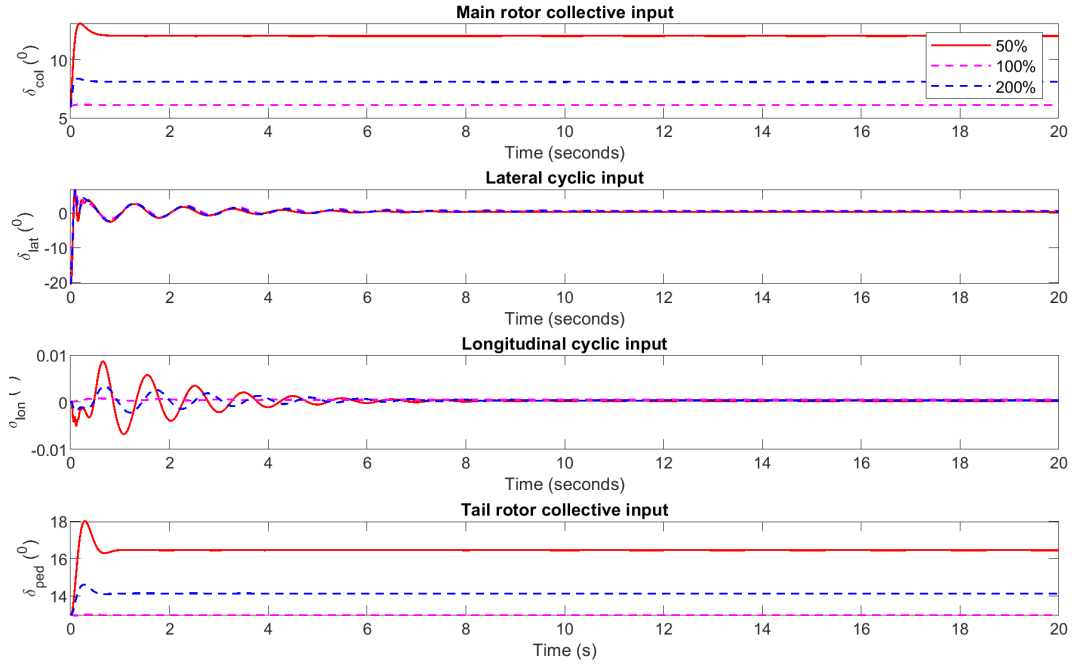


Figure 4.21: Comparing robustness to mass variations control inputs in hover.

The second test is to vary the inertia tensor of the rotorcraft. Figures 4.22 and 4.23 show

that the controller stuggles to stabilise the rotor but succeeds. The larger the variations the more oscillation is experienced, especially as seen in the longitudinal and lateral axes.

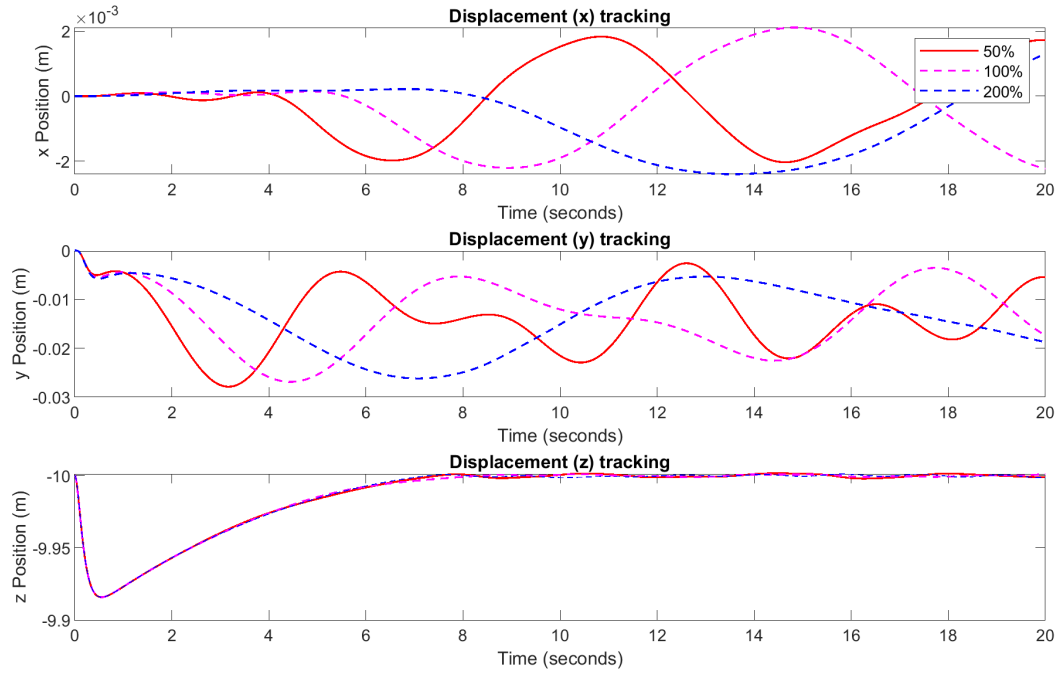


Figure 4.22: Comparing robustness to inertia tensor variations for position in hover.

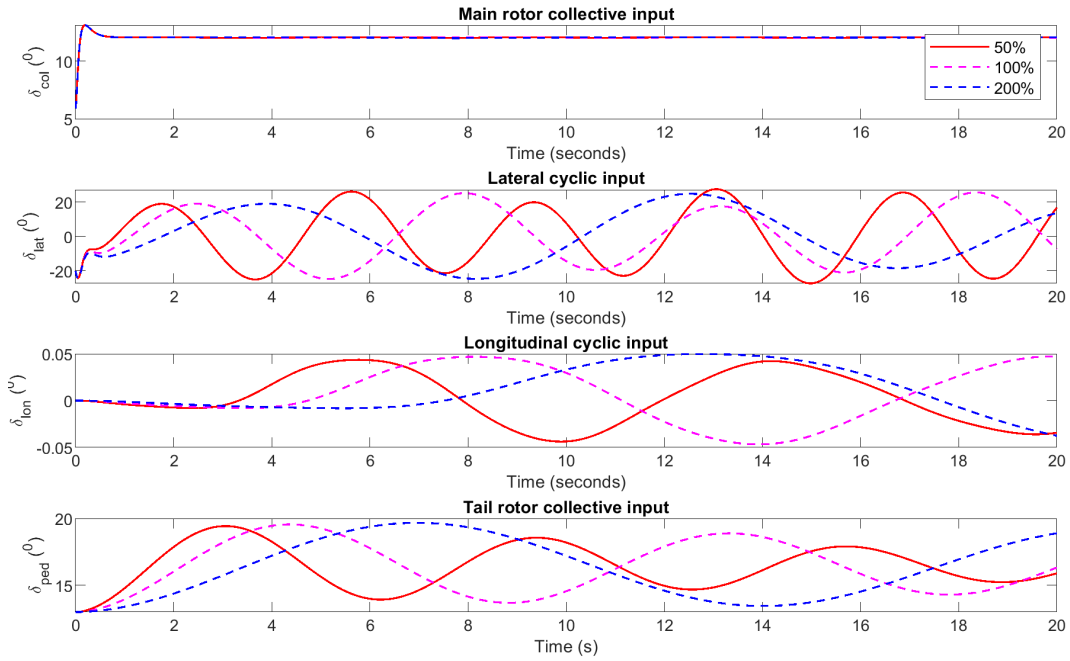


Figure 4.23: Comparing robustness to inertia tensor variations control inputs in hover.

The third test is the rotorcraft's response to gust winds. A gust is applied at 7s. Figures 4.24 and 4.25 show the position of the rotorcraft in hover and the control efforts applied to keep the rotorcraft in its hovering position during this disturbance. The controller is able to reject the wind gust and maintain the rotorcraft within hover specification.

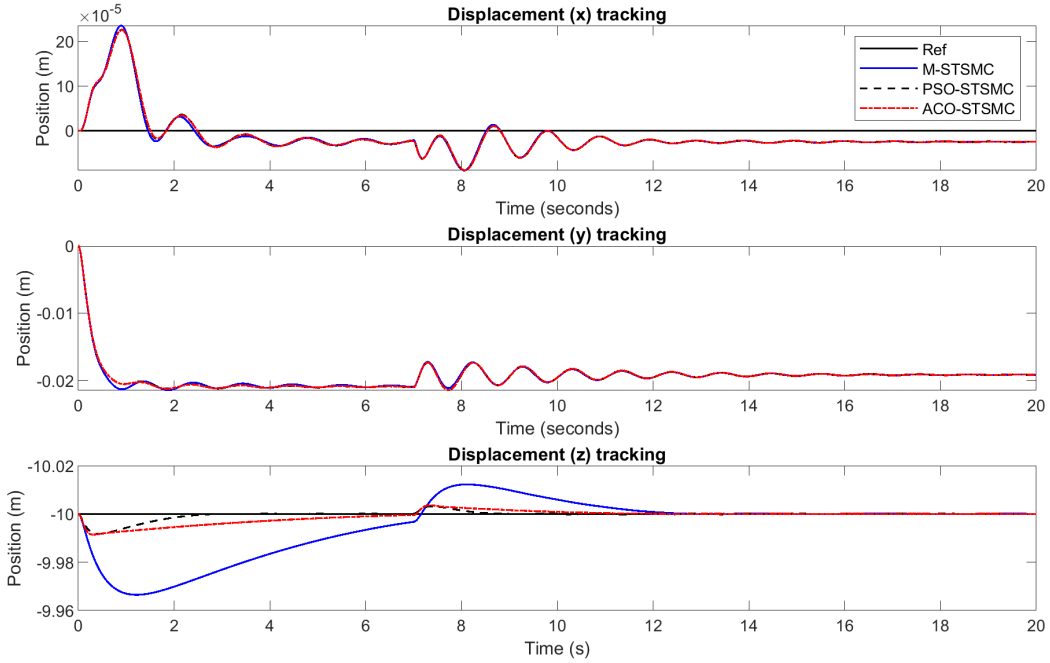


Figure 4.24: Comparing robustness to gust disturbance position in hover.

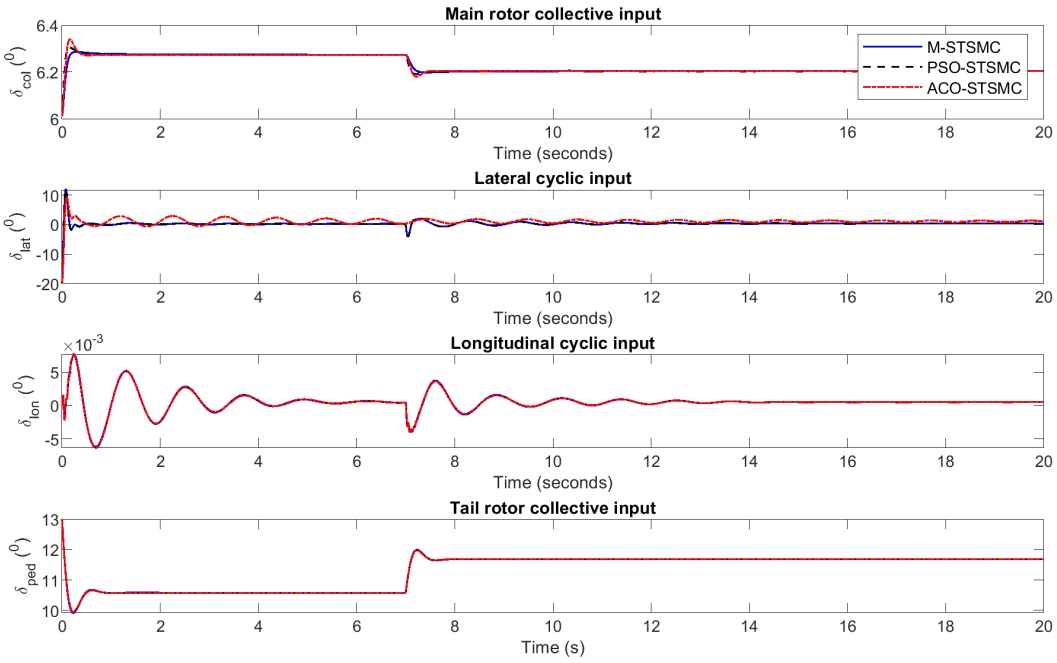


Figure 4.25: Comparing robustness to gust disturbance control inputs in hover.

Figures 4.26 and 4.27 show the position of the rotorcraft in forward flight and the control efforts applied to keep the rotorcraft in its steady forward flight during the gust disturbance.

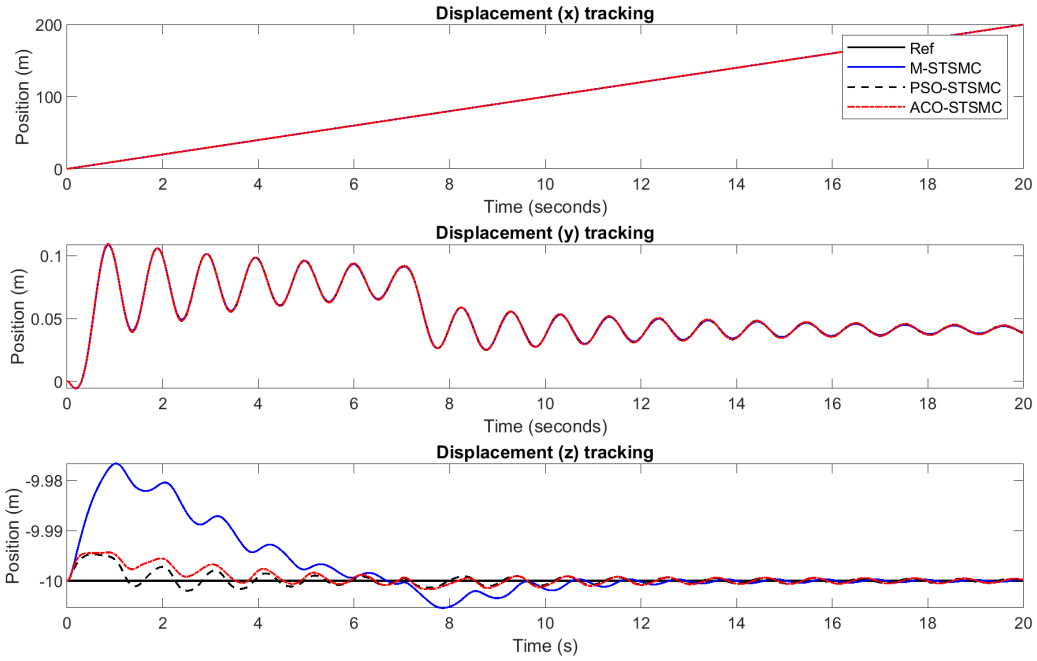


Figure 4.26: Comparing robustness to gust disturbance position in forward flight.

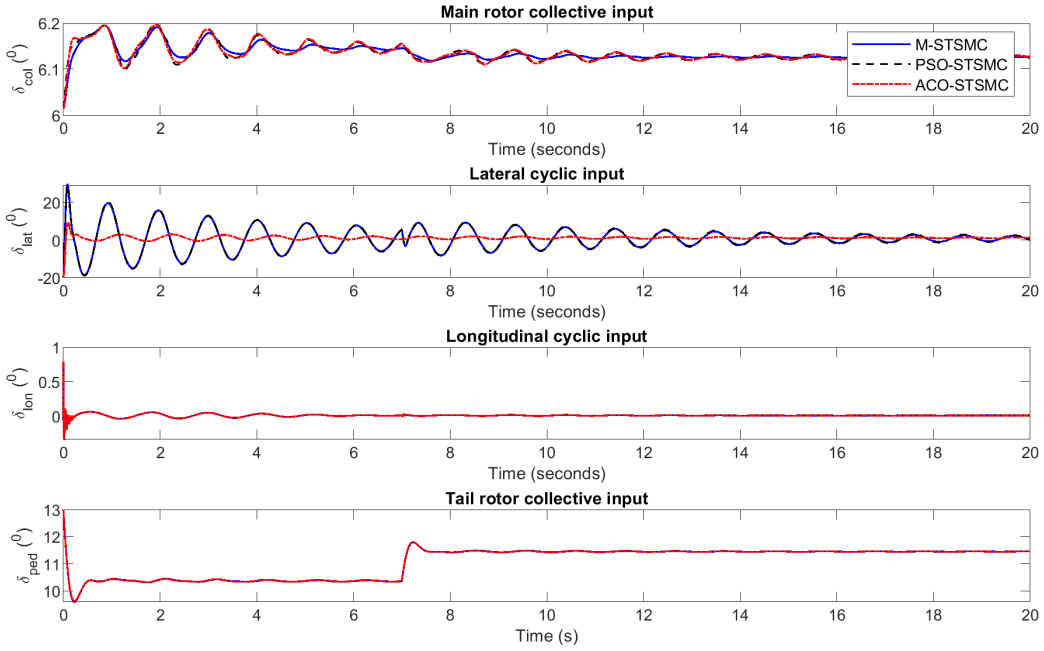


Figure 4.27: Comparing robustness to gust disturbance control inputs in forward flight.

4.5.3 Actuator fault tolerance

The attraction of SMC is that it is useful for passive fault tolerance. In this section the rotorcraft is subjected to actuator faults. The faults are limited to the the forward pitch actuator which will result in a reduced effectiveness on the pitch control channel. The fault is injected at 7 s during level and steady flight. The PSO-based STSMC was used for fault tolerance.

Figure 4.28 shows the rotorcraft response in position when subjected to 10%, 20% and 40% reductions in effectiveness. A disturbance can be seen after 7 s, but the controller does well to recover. The inputs used in order to recover from the fault are shown in Figure 4.29. It is seen that more effort is required in the δ_{lon} in order to retain control of the helicopter. The greater the error, the more control input is required.

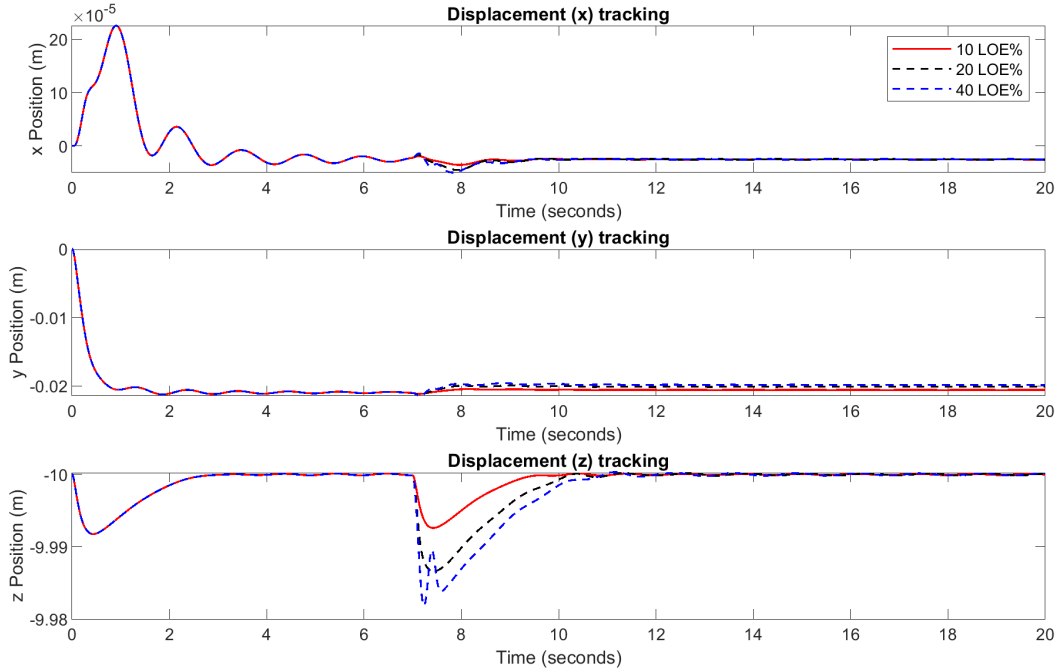


Figure 4.28: Comparing robustness of the RUAV position for different levels of loss-of-effectiveness.

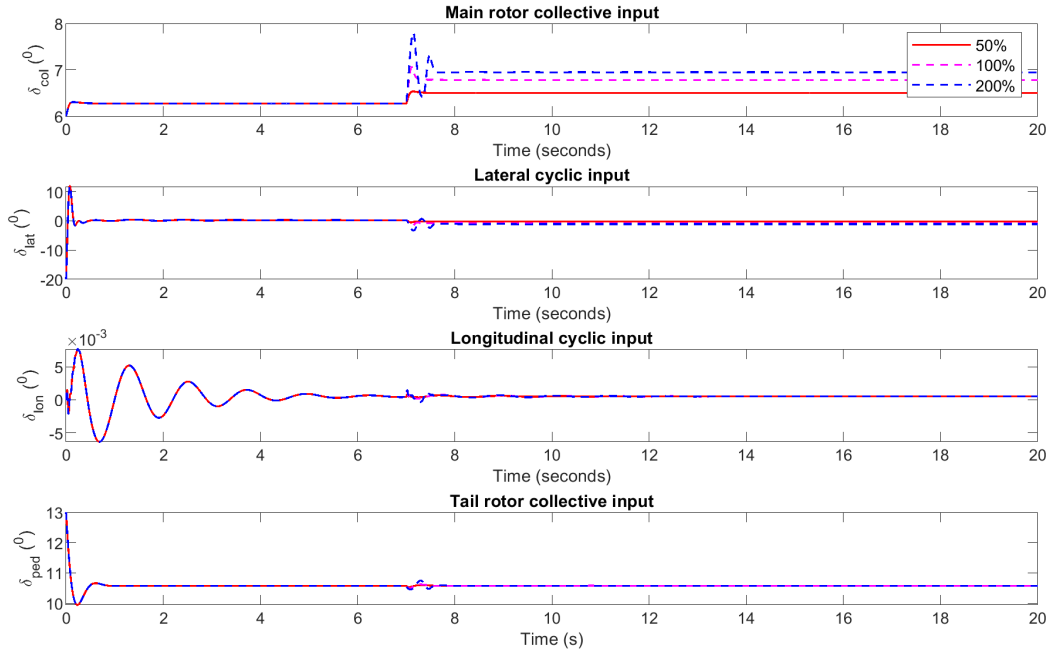


Figure 4.29: Comparing robustness of the RUAV control inputs for different levels of loss-of-effectiveness.

4.5.4 Comparing the two optimisation algorithms

To further test whether the difference between the tuning conducted using PSO and ACO algorithms is statistically significant or not, the optimisation data is subjected to statistical tests (Coffin and Saltzman, 2000). The results of these tests are summarised in Tables 4.3 and 4.4, and are discussed. The null-hypothesis is presented stating that "there is no difference between the SMC gains tuned using PSO and the ones tuned using ACO in tuning the respective controllers." The SMC F-score, t-score and p-value are 1.13×10^{-5}, 0.065 and 0.029 respectively. This means the performance of PSO is significantly better than that of ACO in this exercise.

Table 4.3: Statistical test for comparing PSO and ACO used to tune the SMC.

	PSO-SMC	ACO-SMC
Mean	1.48×10^{-3}	1.52×10^{-3}
SD	8.1×10^{-5}	2.6×10^{-5}
F-score	1.13×10^{-5}	
t-test score	0.065	
p-value	0.029	

The summary of comparing PSO and ACO tuning for the STSMC is shown is Table 4.4. The STSMC F-score, t-score and p-value are 8.7×10^{-2}, 0.147 and 0.075 respectively.

PSO is better than ACO, but not significantly. Overall, PSO does provide better tuning when compared to ACO, however, the null-hypothesis can not be rejected due to the p-value being greater than 0.05.

Table 4.4: Statistical test for comparing PSO and ACO used to tune the STSMC.

	PSO-STSMC	ACO-STSMC
Mean	1.34×10^{-3}	1.41×10^{-3}
SD	2.2×10^{-5}	3.2×10^{-5}
F-score	8.7×10^{-2}	
t-test score	0.147	
p-value	0.075	

It was shown that the PSO perform significantly better than the ACO. Therefore, only the PSO-based SMC and STSMC are compared in Table 4.5. The null-hypothesis is stated as follows: "there is no difference between the performance of the optimised-SMC and optimised-STSMC." The F-score of 0.06687, t-score of 0.000142 and p-value of 0.000229 means that the performance of the two optimised controllers are significantly different. Therefore the null-hypothesis is rejected.

Table 4.5: Statistical test computation for comparing conventional SMC and STSMC.

	SMC vs. STSMC
F-score	0.06687
t-test score	0.000142
p-value	0.000229

At this point it should be pointed out that the statistical analysis results presented only apply to the optimisation problem under investigation. The rejection of the hypothesis does not imply that PSO is generally better performing than ACO. Mirjalili and Dong (2020) studied this dilemma on what is referred to as the "no-free lunch Theorem". The authors found that optimisation algorithms cannot be classified as generally better or worse than others. When comparing these algorithms, one that performs best in one task can be the worst performing in another task. They recommend evaluating a number of algorithms for each problem of interest.

4.6 Chapter Summary

During literature review it was found that most controllers in the field are not properly tuned. A simulation-based research was conducted to determine the optimal SMC pa-

rameters for the rotorcraft. The performance of a SMC and STSMC controllers designed for a rotorcraft UAV were presented in this chapter. The objective was to investigate the application of PSO and ACO algorithms to the two types of sliding mode controller for gains tuning. These optimisation techniques were benchmarked against manually-tuned SMC controllers. There are 8 parameters for CSMC and 12 for STSMC optimisation problem. A detailed stability analysis using the Lyapunov method was presented. In addition to that, the smoothing of the control action with the $sign()$ function replaced by $tanh()$ function was derived and shown to represent a first-order low-pass filter. These results mathematically showed the mechanism of how the chattering is eliminated.

The designed CSMC and STSMC controllers were evaluated by for the control of x, y and z in hover and in forward flight. Simulation results showed that the PSO-based STSMC controller performed better than the ACO-tuned STSMC in terms of pitch tracking error. In terms of translation tracking, the ACO results are comparable to the PSO. Even though the results of PSO-based STSMC and ACO-tuned STSMC are comparable, the ACO algorithm converges faster than the PSO optimisation. Statistical tests show that the PSO-based STSMC controllers perform significantly better that the other controllers. It was also shown that the PSO-based system is robust and bounded stable for actuator loss-of-effectiveness of up to 40%. The effectiveness of the optimised STSMC over the conventional SMC for the rotorcraft control at hand was demonstrated using statistical tests.

Chapter 5

Active Fault-Tolerant Control of Rotorcraft for Integrated Flight-Propulsion Control by Application of Dynamic Neural Network-based Feedback Linearisation

In the previous chapters we presented PID and SMC controllers of the medium-scale rotorcraft. This chapter presents the development of a Dynamic Neural Network-based fault-tolerant controller. Two popular indirect methods based on DNN are presented that is feedback linearisation and active fault-tolerant control using DNN as the fault detection and diagnosis method.

5.1 Active Fault-Tolerant Control (AFTC)

Actuators are the link between the user-input commands, the system control and the desired rotorcraft response or physical action. When there are no faults in the system, this link is carried out to completion without issues (Zhong et al., 2018). However, during an actuator fault the system can react unexpectedly to what is commanded and result in rotorcraft instability. This could lead eventually to failure of the mission or adversely to the loss of the rotorcraft and, worst of all, loss of life. As demonstrated in the previous chapter, small to medium faults can be handled by a robust passive

fault-tolerant controller such as the SMC. More severe faults leading to total disablement of system function require AFTC which is capable of detecting the presence of such a fault, isolating its location and estimating its magnitude. This chapter presents the development of intelligent AFTC with the ability to handle swashplate actuator faults while distinguishing them from deliberate changes in rotor speed. The changes in the rotor speed are treated as a system disturbance.

One of the major reasons for failure of most controllers is the lack of proper gain-tuning. Whether this has an impact on the controller's ability to handle faults is still an open research topic. Computational intelligence techniques have been put forward to overcome this problem by a number of authors (Ramalakshmi et al., 2013; Mpanza and Pedro, 2021). Ramalakshmi et al. (2013) used PSO algorithm and its modified form to tune PID controllers of a 2-DOF model helicopter. These were compared to GA-based PID controllers. Through simulation, it was shown that the modified PSO-tuned parameters produced better command tracking and less steady-state error. In (Mpanza and Pedro, 2021), the authors compared the tuning of PID controller gains of a full 6-DOF rotorcraft based on PSO, ACO and CS. The tuning was done for hover and 10 m/s forward flight. With computational simulation results, the optimisation algorithms were shown to give better performance. Rodríguez-Molina et al. (2020) reviewed the application of multi-objective optimisation (MOO) for tuning controllers and showed that MOO can handle the trade-offs among conflicting control performance objectives. Zhou and Zhang (2019) compared a non-dominated sorting genetic algorithm II (NSGA-II) to the multi-objective evolutionary algorithm (MOEA) for tuning PID controllers applied to an aerial manipulator.

The nonlinear nature of rotorcraft has motivated authors to propose application of different nonlinear controllers, such as: feedback linearisation (FBL) (Lee et al., 2009), sliding mode control (Nonaka and Sugizaki, 2011; Mokhtari and Cherki, 2015), and neural network (NN)-based controllers. The idea of feedback linearisation is to cancel out systems' nonlinearity, thereby rendering the rotorcraft linear in its input-output relation. Lee et al. (2009) applied FBL and compared it to SMC on a quadrotor. Since FBL requires a number of derivatives of the output to expose the input, it is highly sensitive to noise.

NN-based system identification and controllers have seen a resurgence lately, mainly due to availability of compact and affordable processing platforms. NNs are able to approx-

imate any arbitrary nonlinear system given adequate input-output data (Isermann and Münchhof, 2010). Numerous efforts on system identification have been applied on both the multi-layer perceptron (MLP) and radial basis function (RBF) to model complex systems (Majhi and Panda, 2011; Ugalde et al., 2015). Even though NN are able to model complex systems, this comes at a cost of large number of hidden nodes and the weights to be trained. NNs also lack the ability to model dynamic data due to their inability to preserve historic knowledge (Adeyemi et al., 2018). As a result, they fail to incorporate the temporal dimension of a system such as the rotorcraft under investigation (Shoaib et al., 2016).

Dynamic NN are more suitable for dynamic systems such as rotorcraft; due to their inherent ability to handle time-series data (Garces et al., 2003). NN is applied as direct adaptive control, i.e., used as the actual system controller or indirect adaptive, used for system identification and control (Suprijono and Kusumoputro, 2017). DNN in combination with FBL has been applied successfully in multi-input and multi-output systems (Pedro et al., 2018; Naimi et al., 2020). In (Pedro et al., 2018), the DNN system identification model of a suspension system and the accompanying PID gains were tuned using the PSO algorithm. Naimi et al. (2020), applied quasi-Newton optimisation to a multi-input single-output pressurised water reactor. Both these reported methods are single-objective optimisation. However, none has been applied to rotorcraft system and none use multi-objective optimisation to tune the controller gains.

Other intelligent control strategies have been used successfully for system identification such as the Takagi-Sugeno (T-S) fuzzy system that provides an intuitive rule-based representation of a nonlinear system (Liu, 2018; Singh and Bhushan, 2020). The T–S fuzzy system is suitable for the piecewise linear control system representation which limits its application to rotorcraft control.

5.1.1 Fault Detection and Isolation (FDI) Model

The research presented here is focused on the controller part of the active FTC. However, for completeness this section presents a brief introduction into a useful FDI mechanism. According to Blanke et al. (2006), FDI has been studied since the 70s. For the requirements of an AFTC, two system blocks are required: fault detection and diagnosis/identification, and fault reconfiguration. Fault detection measures the signals of

interest and compares them with the ideal model signals. If enough deviation is detected this information is sent to the control system to be reconfigured. Control reconfiguration alters the controller in order to account for the fault detected.

Desirable properties of an FDI system are (Blanke et al., 2006; Zheng et al., 2016; Zolghadri et al., 2014):

1. Early detection;
2. High sensitivity (detection rate) and performance (low false alarm); and
3. Robustness to noise and other disturbances.

The strategy to adopt and the level of performance that can be recovered depends on the following (Noura et al., 2009):

1. The complexity of the process under investigation;
2. The availability of redundancy in the system - be it hardware or analytical;
3. The severity of the estimated fault; and
4. The level of performance required of the system

Nyulászi et al. (2018) used multi-models based on ARX for FDI on an aircraft turbojet engine. Data was collected from a small turbojet engine using multiple sensors. This input-output data was then used to train the ARX model. An expert voting system was used to detect system deviations from nominal. This arrangement was successful in detecting engine fault in a laboratory setup. Taimoor and Aijun (2020) used a MLPNN for model-based FDI of aircraft actuators and sensors of a Boeing 747-100 aircraft. The authors added an extended Kalman filter (EKF) to adapt the weights of the MLPNN online to improve the precision of the fault detection of the sensors and actuators. This method was validated on a Boeing 747/200 nonlinear model and was found to be better and quicker than traditional MLPNN. Amozegar and Khorasani (2016) compared DNNs, to MLPNN and RBFNN for identifying faults in a gas turbine engine. The ensemble of the three models was also created. While RBFNN performed best as a single model, the ensemble outperformed all three strategies in terms of accuracy and reliable performance.

The preceding are examples of model-based fault detection. There is also a large literature that reports on filters/observer-based fault detection. Bateman et al. (2011) applied EKF to the FDI problem on a F-16 model. This was compared to the robust Kalman filter (RKF). The latter was found to be more effective in detecting actuator/control surfaces faults. An improved robust adaptive Kalman filter (RAKF) was used to detect faults on a fixed-wing UAV (Hajiyev and Soken, 2013) and the implementation compared to optimal Kalman filter (OKF). The RAKF proved to be effective even under harsh environmental conditions. Pérez et al. (2018) implemented third-order sliding mode observer to generate a residual for fault detection. The STSMC had a better detection rate than the continuous-twisting SMC (CTSMC).

As effective as they are, filter-based methods are computationally expensive when compared to model-based FDIs. Therefore for this study, the DNN-based FDI is used to find actuator fault on the swashplate. The rest of this chapter discussed dynamic neural networks and its application to both FBL and FDI.

5.2 DNN-based Feedback Linearisation

Since the system in Equation 2.54 is non-affine-in-control, this means that FBL by the analytical methods is impossible. A DNN, just like a static NN, possesses the property of universal approximation. Unlike static NN, a DNN also has dynamic memory and it is more suitable for approximating dynamic systems such as rotorcraft. A DNN can model any plant and be expressed in the following differential equation (Garces et al., 2003):

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{\Theta}), \tag{5.1}$$

$$\boldsymbol{y} = \boldsymbol{h}(\boldsymbol{x}, \boldsymbol{\theta}), \tag{5.2}$$

where $\boldsymbol{x} \in \Re^n$, $\boldsymbol{u} \in \Re^m$ and $\boldsymbol{y} \in \Re^p$ are the state, input and output vectors respectively. The parameter $\boldsymbol{\Theta}$ is a tunable network architecture variable, $\boldsymbol{f}(\cdot)$ represents the network structure and $\boldsymbol{h}(\cdot)$ maps the states to the output. The state is also known as the dynamic units of order n. The DNN can be written as (Garces et al., 2003):

$$\dot{x}_i = -\beta_i x_i + \sum_{j=1}^{n} \omega_{ij} \sigma(x_i) + \sum_{j=1}^{m} \gamma_{ij} u_j, \tag{5.3}$$

where β_i, ω_{ij} and γ_{ij} are adjustable parameters, x_i is the activation state of unit i and u_j is the j^{th} network input. The network output corresponds to the first p states, $\boldsymbol{x}^o$. The

remainder, $n-p$ states, are considered hidden, $\boldsymbol{x}^h$: such that $\boldsymbol{x} = [\boldsymbol{x}^o \ \boldsymbol{x}^h]^T$. In a vector form, Equation 5.3 can be expressed as (Naimi et al., 2020):

$$\begin{aligned} \dot{\boldsymbol{x}} &= -\boldsymbol{\beta}x + \boldsymbol{\omega}\boldsymbol{\sigma}(\boldsymbol{x}) + \boldsymbol{\gamma}\boldsymbol{u}, \\ \hat{\boldsymbol{y}} &= \boldsymbol{C}\boldsymbol{x}, \end{aligned} \tag{5.4}$$

where $\boldsymbol{\beta} \in \Re^{n\times n}$, $\boldsymbol{\omega} \in \Re^{n\times n}$, $\boldsymbol{\gamma} \in \Re^{n\times m}$, $\boldsymbol{\sigma}(\boldsymbol{x}) = [\sigma(x_1), \ldots, \sigma(x_n)]^T$ and $\sigma(\cdot)$ is a sigmoidal activation function: $\sigma(\cdot) = \tanh(\cdot)$, chosen for network stability, and $\boldsymbol{C} = [\boldsymbol{I}_{p\times p} \ \boldsymbol{O}_{p\times(n-p)}]$ is the output matrix.

The development of a DNN nonlinear mapping requires experimental data $\boldsymbol{Z}^N = [u(k), y(k)]$, with $k = 1, \ldots, N$ and N is the size of the dataset. Here $u(k)$ and $y(k)$ are the measured input-output pairs in the dataset. The training problem is therefore:

$$argmin_{\boldsymbol{\Theta}} \ \ F_N(\boldsymbol{\Theta}, \boldsymbol{Z}^N), \tag{5.5}$$

where F_N is the objective function and $\boldsymbol{\Theta}$ is the parameter vector to be adjusted in order to minimise the objective function.

$$\boldsymbol{\Theta} = \begin{bmatrix} \boldsymbol{\beta}_{n\times 1} \\ vec(\boldsymbol{\gamma}_{n\times m}) \\ vec(\boldsymbol{\omega}_{n\times n}) \end{bmatrix}. \tag{5.6}$$

The error function that was chosen for this investigation is the RMSE as shown:

$$RMSE = \frac{1}{N}\sqrt{\sum_{i=1}^{N}(y-\hat{y})^2}, \tag{5.7}$$

where y is the target and $\hat{y}$ is the prediction.

In the following, ant-based multi-objective optimisation techniques are used for finding $\boldsymbol{\Theta}$ of Equation 5.5 and for optimising the rotorcraft performance as discussed.

5.2.1 Multi-objective optimisation

To solve the problem stated above, we formulate the rest of the investigation as an optimisation problem. This section discusses the formulation of multi-objective optimisation and act as a supplement to the computational intelligence optimisation algorithms already discussed in Chapter 3.

MOO is about finding an optimal solution to a problem with two or more objective functions. As shown in Equation 3.6, the control optimisation considered can be treated

as two-objective functions, while the DNN parameters estimation is treated as a single-objective function; according to Equation 5.7.

There are two prevalent approaches for dealing with multi-objective optimisation problems (MOOP). Firstly, the MOOP can be converted to a single-objective function and solved as such (Gomez et al., 2020). The shortcomings of this method is that it may be difficult to find suitable weights for aggregation which means an optimal solution is never found. The second approach is to keep the separate objective functions. This means a Pareto optimal set of the problem must be found (Gomez et al., 2020).

The MOOP is formulated as follows:

$$\text{argmim}_x \; \boldsymbol{F}(\boldsymbol{x}) = \{f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), \ldots, f_J(\boldsymbol{x})\}\,, \tag{5.8}$$

where $\boldsymbol{x}$ is a candidate solution and given $g_i(\boldsymbol{x}) \geq 0$ and $h_i(\boldsymbol{x}) = 0$, the inequality and equality constraints respectively. The subscript J is the total number of objective functions. Since $\boldsymbol{F}(x)$ is a vector of real-valued functions, simple relational operators cannot be applied to compare fitness of each solution. The definitions of optimality used for comparison are as follows (Mirjalili and Dong, 2020):

1. **Pareto dominance:** Given two candidate solutions $\boldsymbol{x} = (x_1, x_2, \ldots, x_d)$ and $\boldsymbol{y} = (y_1, y_2, \ldots, y_d)$, where d is the dimension of the solution. In this case $\boldsymbol{x}$ is said to dominate $\boldsymbol{y}$, that is to say $(\boldsymbol{x} \prec \boldsymbol{y})$, if and only if:

$$\forall i \in [1, 2, \ldots, d] \;\; f(x_i) \leq f(y_i) \wedge \exists i \in [1, 2, \ldots, J] : \quad f_i(\boldsymbol{x}) < f_i(\boldsymbol{y}). \tag{5.9}$$

2. **Pareto optimality:** A solution $\boldsymbol{x} \in \boldsymbol{X}$ is called Pareto optimality if and only if:

$$\{\nexists \boldsymbol{y} \in \boldsymbol{X} | \boldsymbol{y} \prec \boldsymbol{x}\}\,. \tag{5.10}$$

3. **Pareto optimal set:** The set of all Pareto optimal solutions is called Pareto optimal set:

$$P_s = \{\boldsymbol{x}, \boldsymbol{y} \in \boldsymbol{X} | \nexists \boldsymbol{y} \prec \boldsymbol{x}\}\,. \tag{5.11}$$

4. **Pareto front:** A set containing the values of objective functions of the Pareto solution set:

$$P_f = \{f(\boldsymbol{x}) | \boldsymbol{x} \in P_s\}\,. \tag{5.12}$$

To solve a MOOP, the Pareto optimal set must be obtained and then evaluated to give a Pareto front the results which are used to determine the best trade-off between given objective functions.

The general framework for meta-heuristic MOO is as follows:

- start by initialising multiple random candidate solutions;
- compare solutions using Pareto dominance;
- non-dominant solutions are stored to be improved later;
- improve non-dominant solutions for future iterations. The improvement methodology is the part where these algorithms differ.

Two meta-heuristic based-algorithms are compared for training the DNN and also finding the PID gains for the feedback-linearised system. The first is ant colony optimisation and the second is the antlion optimisation. Both algorithms are discussed in the next subsection.

5.3 Methodology

DNN is used to identify the rotorcraft in a formulation suitable for FBL. In order to apply DNNFBL control to the present problem, the first step involves system identification. The process followed in system identification is shown in Figure 5.1 and detailed here:

1. identify the physical model from which the data is to be collected;
2. excite the physical model, collect and pre-process data;
3. select the model architecture by:
   (a) selecting the number of hidden layers;
   (b) selecting the optimal number of nodes in each hidden layer;
   (c) selecting the appropriate transfer functions.
4. train the DNN using training data;
5. validate the DNN using validation data;

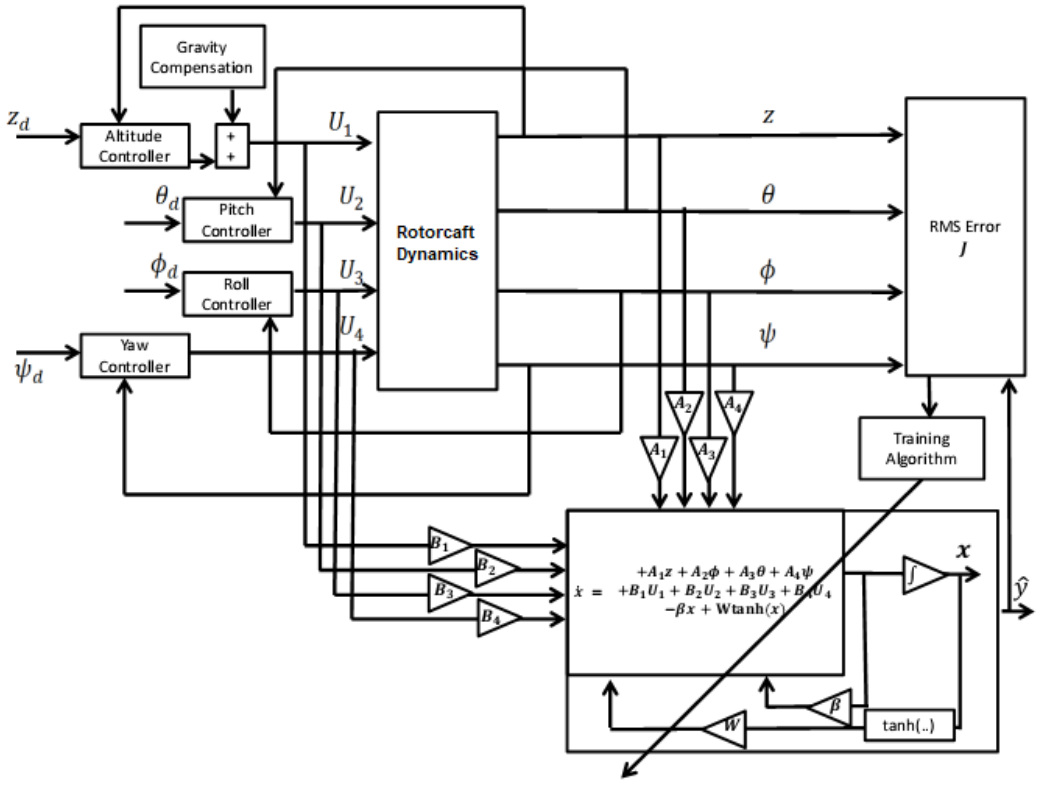


Figure 5.1: Schematic of the DNN-based system identification process for the rotorcraft system.

6. Once an acceptable model is achieved, then it is used in FBL. If not accepted, return to 2.

As it has been mentioned in the previous chapters, the rotorcraft under investigation only exists as a conceptual design. The model presented in Chapter 2 was developed to be as closed to the design as possible. As a result, this model was used to extract the input-output data that is used in the sequel. It might, therefore, be confusing as to why a new model is created when one from Chapter 2 is available and already has been validated. As presented, in the end of that chapter, the model in non-affine-in-control and not feedback linearisable. The model developed here is to make the aircraft feedback linearisable using DNN approximation.

In this numerical experimentation, a MATLAB® physical model of the rotorcraft is used instead of actual flight test. The aim of the exercise is to determine if the DNN can embody the dynamics of the rotorcraft model. If the dynamics are captured with reasonable accuracy, then, without any loss of generalisation, the effectiveness of the method followed

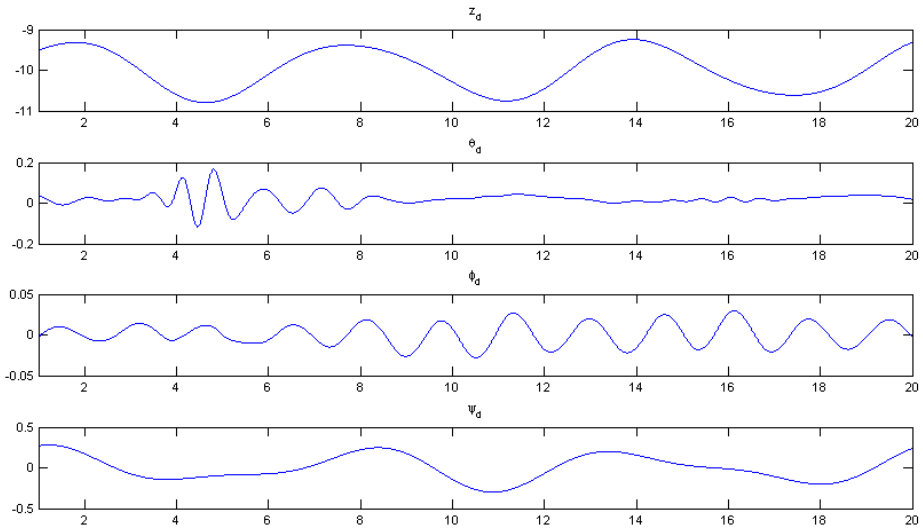


Figure 5.2: Reference signal used to excite the closed-loop rotorcraft.

for system identification can capture rotorcraft dynamics from actual flight data.

Since a rotorcraft is open-loop unstable, it is not possible to collect input-output data without crashing. In this numerical experimentation, system identification is performed on a closed-loop rotorcraft system. According to Miliken (1947): *"It would appear that the optimum input in a given case is that which best excites the frequency range of interest, and hence the harmonic content of the input should be examined before the test to ensure that it is suitable."* (Jategaonkar, 2005). Therefore, to excite the rotorcraft effectively and safely, reference inputs are modified in order to inject test data. The test data is based on a Fourier series as follows:

$$x_d = a_o + \sum_{i=1}^{N} a_i \sin(2\pi f_i t), \tag{5.13}$$

where a_i and f_i are the amplitude and the frequency of each signal harmonics. The index i up to N, is the number of harmonics included. In this investigation it is chosen to be two. The reference x_d represents z, ϕ, θ and ψ respectively. These are the desired values sent to the PID controllers. Figure 5.2 shows the data collected from the closed-loop simulations.

When performing system identification the interest is in the actual inputs that go into the rotorcraft. These are shown in Figure 5.3. Notice that the data have been normalised between $[0,\ 1]$ to ensure that all the observations in the data set are of the same order of magnitude to simplify and hasten the DNN training process. This normalisation is

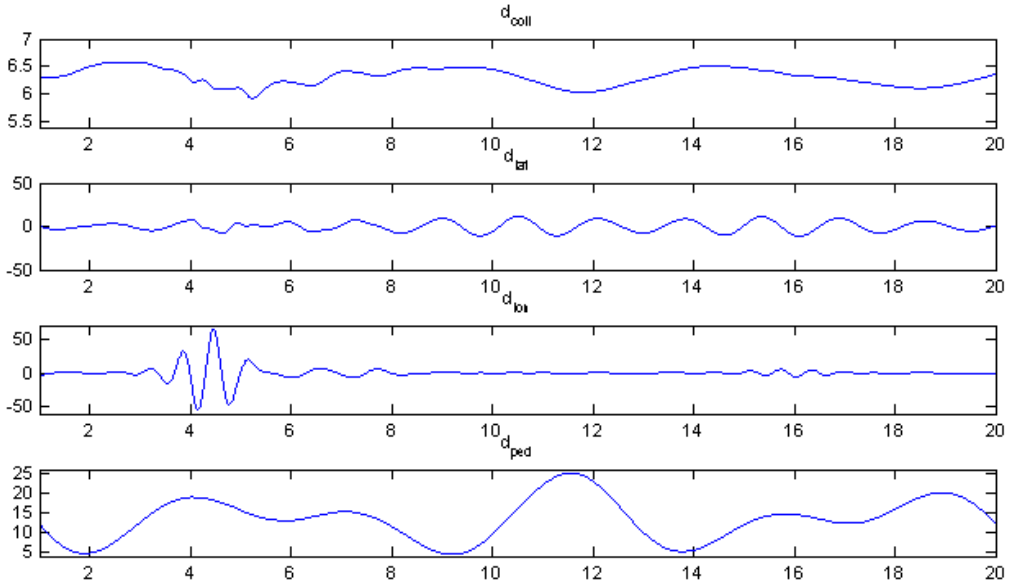


Figure 5.3: A sample of the system input signals applied to the rotorcraft for DNN system identification.

achieved by making use of the following relationship:

$$x_{norm} = \frac{x - x_{min}}{x_{max} - x_{min}},$$

where x_{norm} is the normalised observation, x_{min} and x_{max} are the smallest and the largest observations in the data set respectively.

It is essential to select a suitable state vector such that DNNFBL is possible. It was found that the rotorcraft rates of change capture meaningful relationships of the rotorcraft dynamics. Therefore, the state vector was identified as $[w,\ p,\ q,\ r]^T$, which is set as the output of the DNN. Therefore, the identified DNN model has 4 inputs, 4 outputs and 4 delayed outputs are also applied as inputs to the DNN. In Figure 5.1, the $\boldsymbol{\gamma} = [\boldsymbol{A}\ \boldsymbol{B}]$ is used for input distribution matrix $\boldsymbol{A}$ and delayed output distribution matrix $\boldsymbol{B}$.

The second part of the training is to select a suitable number of hidden units. Since the output is equal to 4 items, the minimum number of the DNN hidden units is 4. The structure of the DNN was selected using the pruning method. The hidden units are varied from 4 to 15 until the topology with the lowest validation error is found. Through numerical experimentation, the best number of states were found to be 4 and the relative degree vector is $[1, 1, 1, 1]^T$.

Lastly, the system represented by a DNN is combined with FBL to cancel out nonlinearities. Even though the rotorcraft system is not affine-in-control, the DNN approximation

makes it possible for the rotorcraft to be treated as affine-in-control as follows (Garces et al., 2003):

$$\begin{aligned} \boldsymbol{f}(\boldsymbol{x}) &= -\boldsymbol{\beta}x + \boldsymbol{\omega}\boldsymbol{\sigma}(\boldsymbol{x}), \\ \boldsymbol{g}_j(\boldsymbol{x}) &= [\gamma_{1j}, \ldots, \gamma_{Nj}]^T, \\ \boldsymbol{h}(\boldsymbol{x}) &= \boldsymbol{C}_n\boldsymbol{x}. \end{aligned} \tag{5.14}$$

Since each output has a relative degree of 1, the system is feedback linearisable. Therefore, the control law can found by differentiating the output function until it is an explicit function of the input. The following applies:

$$\begin{aligned} \dot{\hat{y}}_i &= \frac{\partial \hat{y}_i}{\partial t} = \frac{\partial \hat{h}_i(\boldsymbol{x})}{\partial \boldsymbol{x}}\dot{\boldsymbol{x}} \\ &= \frac{\partial \hat{h}_i}{\partial \boldsymbol{x}}\left[-\beta_i x_i + \sum_{j=1}^{n}\omega_{ij}\sigma(x_j) + g_i(\boldsymbol{x})\boldsymbol{u}\right]. \end{aligned} \tag{5.15}$$

This leads to the system expressed as follows:

$$\hat{\boldsymbol{y}} = \hat{\boldsymbol{A}}(\boldsymbol{x}) + \hat{\boldsymbol{B}}(\boldsymbol{x})\boldsymbol{u},$$

where $\hat{\boldsymbol{A}}$ and $\hat{\boldsymbol{B}}$ are defined as follows:

$$\hat{\boldsymbol{A}}(\boldsymbol{x}) = \left[\sum_{k=0}^{r_1}\lambda_{1k}L_f^k h_1(\boldsymbol{x}) \;\; \sum_{k=0}^{r_2}\lambda_{2k}L_f^k h_2(\boldsymbol{x}) \;\; \ldots \;\; \sum_{k=0}^{r_m}\lambda_{mk}L_f^k h_m(\boldsymbol{x})\right]^T, \tag{5.16}$$

and

$$\hat{\boldsymbol{B}}(\boldsymbol{x}) = \begin{bmatrix} L_{g_1}L_f^{r_1-1}h_1(\boldsymbol{x}) & L_{g_2}L_f^{r_1-1}h_1(\boldsymbol{x}) & \ldots & L_{gm}L_f^{r1-1}h_1(\boldsymbol{x}) \\ L_{g_1}L_f^{r_2-1}h_2(\boldsymbol{x}) & L_{g_2}L_f^{r_2-1}h_2(\boldsymbol{x}) & \ldots & L_{gm}L_f^{r2-1}h_2(\boldsymbol{x}) \\ \vdots & \vdots & \ddots & \vdots \\ L_{g1}L_f^{r_m-1}h_m(\boldsymbol{x}) & L_{g2}L_f^{r_m-1}h_m(\boldsymbol{x}) & \ldots & L_{gm}L_f^{rm-1}h_m(\boldsymbol{x}) \end{bmatrix}, \tag{5.17}$$

where $(r_1,\ r_2,\ \ldots,\ r_m)$ are the relative degrees in each variable in the model given by Equation 5.14 and $\lambda_{ik}s$ are tunable design parameters. Noting that the determinant $|\hat{\boldsymbol{B}}(\boldsymbol{x})| \neq 0\ \forall\ \boldsymbol{x}$ and where $L_f^{r_i}h_i(x)$ and $L_{g_i}L_f^{r_i-1}h_i$ are the corresponding Lie derivatives. The matrix $\hat{\boldsymbol{B}}(\boldsymbol{x})$ is called the 'decoupling matrix" of Equation 5.14.

Theorem 2. The system define in Equation 5.14 is statically decouplabled on a subset $\boldsymbol{Z}_0$ of $\Re^n$ if and only if the $rank(\hat{\boldsymbol{B}}(\boldsymbol{x})) = m$, $\forall \boldsymbol{x} \in \boldsymbol{Z}_0$.

Proof. See in (Garces et al., 2003) □

The transformed state is:

$$\hat{\boldsymbol{z}} = \Psi[\boldsymbol{x}^o \ \boldsymbol{x}^h]^T$$

where $\hat{\boldsymbol{y}} = \boldsymbol{x}^o$ is the predicted network output and $\boldsymbol{x}^h$ is the hidden dynamics which in this case $\boldsymbol{x}^h = 0$. The transformed system can be written as:

$$\dot{\boldsymbol{x}}^o = \hat{\boldsymbol{A}}\boldsymbol{x}^o + \hat{\boldsymbol{B}}\boldsymbol{v}, \tag{5.18}$$

$\boldsymbol{v}$ is found by inverse relationship:

$$\boldsymbol{v} = \hat{\boldsymbol{P}}(\boldsymbol{x}) + \hat{\boldsymbol{Q}}(\boldsymbol{x})\boldsymbol{u}, \tag{5.19}$$

where $\hat{\boldsymbol{P}} = -\hat{\boldsymbol{B}}(\boldsymbol{x})^{-1}\hat{\boldsymbol{A}}(\boldsymbol{x})$ and $\hat{\boldsymbol{Q}} = -\hat{\boldsymbol{B}}(\boldsymbol{x})^{-1}$. The virtual control for each channel is found to be:

$$v_j = \sum_{k=0}^{r_j} \lambda_{jk} \frac{d^k \hat{y}_j}{dt^k}, \text{ for } j = 1 \dots m.$$

As can be seen in Equation 5.18, this controller is only related to the observable state. In order to stabilise the hidden state, more analysis is required. It has been shown that each variable was developed with relative degree 1. Therefore there are no hidden dynamics in the DNNs that were used for FBL.

For each of the four closed-loop controllers, the system is solved for $[\lambda_{0_j} \ \lambda_{1_j}]$, where j is the index of the controlled variable. The system architecture used in this investigation is shown in Figure 5.4. The identified linearised system is then augmented with the PID controllers for the four system variables. The PID gains $[K_{P_j} \ K_{I_j} \ K_{D_j}]$ are found through the MOO algorithms as discussed previously.

Similar to all the control methods presented in the previous chapters, the stability analysis of the DNN-based controller is required. This enables the developed controller to be tested whether it will always results in bounded output for every bounded input presented to the system. Even though a helicopter is inherently unstable, and this is also expected in the model if the dynamics of the rotorcraft are captured correctly. The stability analysis is, therefore, a method of testing that the modelling instrument does not introduce unexpected behaviour into the behaviour of the rotorcraft. The conditions that must be met to ensure stability of the DNNFBL controller are listed in (Garces et al., 2003; Pedro et al., 2018):

1. The activation function $\sigma(x)$ used must be continuously differentiable. *The function*

used is the hyperbolic tangent - this function is differentiable.

$$y = \tanh(x), \tag{5.20}$$

$$\frac{dy}{dx} = \frac{1}{\cosh^2(x)}. \tag{5.21}$$

2. $0 \leq d\sigma(x)/dx \leq 1$. *The derivative of the hyperbolic tangent function is within the stated range.*

3. $\sigma(x)$ is bounded. *The hyperbolic tangent function is bounded* $\{\sigma(x) \in \Re : -1 \leq \sigma(x) \leq 1\}$.

4. Given $u \in \Re$, there is a symmetric positive definite solution P to the following Lyapunov equation:

$$-\beta^T \boldsymbol{P} - \boldsymbol{P}\beta = -\mu \boldsymbol{I}, \tag{5.22}$$

where μ is the scaling factor chosen to be 1, and $\boldsymbol{I}$ is the identity matrix of appropriate size. *The matrix β is found through computational intelligence algorithm meets this condition.*

5. The following inequality must be satisfied:

$$\|\mathbf{W}\|^2 \leq \frac{\mu - w\|\boldsymbol{P}\|}{\|\boldsymbol{P}\|}, \quad \|\boldsymbol{P}\| < \frac{\mu}{2}. \tag{5.23}$$

The matrix $\boldsymbol{W}$ is found through computational intelligence algorithm meets this condition.

For the present DNN controller design, the hyperbolic tangent function is selected as the DNN activation function. This directly satisfies the first three conditions. During the optimisation of weights, Equation 5.22 and the inequalities in 5.23 are enforced to guarantee stability of the DNN models derived from training.

5.4 Controller Implementation

5.4.1 System performance specifications

The objective of the present chapter is to develop nonlinear controller for the rotorcraft. A benchmarking PID controller is developed for purpose of comparison. A controller

based on DNN combined with FBL is developed, analysed and then compared to PID controller. The controller parameters are found using ant-based optimisation algorithms. These are elaborated upon in the next subsections.

In order to develop a suitable DNNFBL controller, two objective function are used. The first one is defined as follows:

$$J_1 = \frac{1}{T}\int_0^T \left[\left(\frac{x_d - x}{x_{max}}\right)^2 + \left(\frac{y_d - y}{y_{max}}\right)^2 + \left(\frac{z_d - z}{z_{max}}\right)^2 + \left(\frac{\theta_d - \theta}{\theta_{max}}\right)^2 + \left(\frac{\phi_d - \phi}{\phi_{max}}\right)^2 + \left(\frac{\psi_d - \psi}{\psi_{max}}\right)^2\right] dt. \tag{5.24}$$

This objective function is to make sure that the tracking error is minimised. The second objective function ensures that the actuators are not over-stressed and stay within limits and it is defined as follows:

$$J_2 = \frac{1}{T}\int_0^T \left[\left(\frac{\delta_{col}}{\delta_{col_{max}}}\right)^2 + \left(\frac{\delta_{lon}}{\delta_{lon_{max}}}\right)^2 + \left(\frac{\delta_{lat}}{\delta_{lat_{max}}}\right)^2 + \left(\frac{\delta_{ped}}{\delta_{ped_{max}}}\right)^2\right] dt. \tag{5.25}$$

These two objective functions differ from the one given in Chapter 3, Equation 3.6 in that Equations 5.24 and 5.25 are separated in a manner suitable for MOO.

5.4.2 PID controller augmentation

The PID controllers developed in Chapter 3 are used for benchmarking the proposed DNN controller. In later sections, the PID controller is combined with DNN in a FBL framework.

5.5 Simulation Results and Discussion

This section presents the numerical simulation of the rotorcraft system shown in Figure 5.4, covering system identification, the development of the PID and DNNFBL-based PID controllers for trajectory tracking of z and ϕ, θ, ψ. The experiment design and implementation is conducted in the MATLAB® / Simulink® environment. A Bogacki-Shampine solver is used with the sampling time of 1 kHz. Single-objective ALO and ACO are employed for system identification. Multi-objective optimisation ALO and ACO are used for PID controller gains tuning in addition to the manual tuning (MT) method.

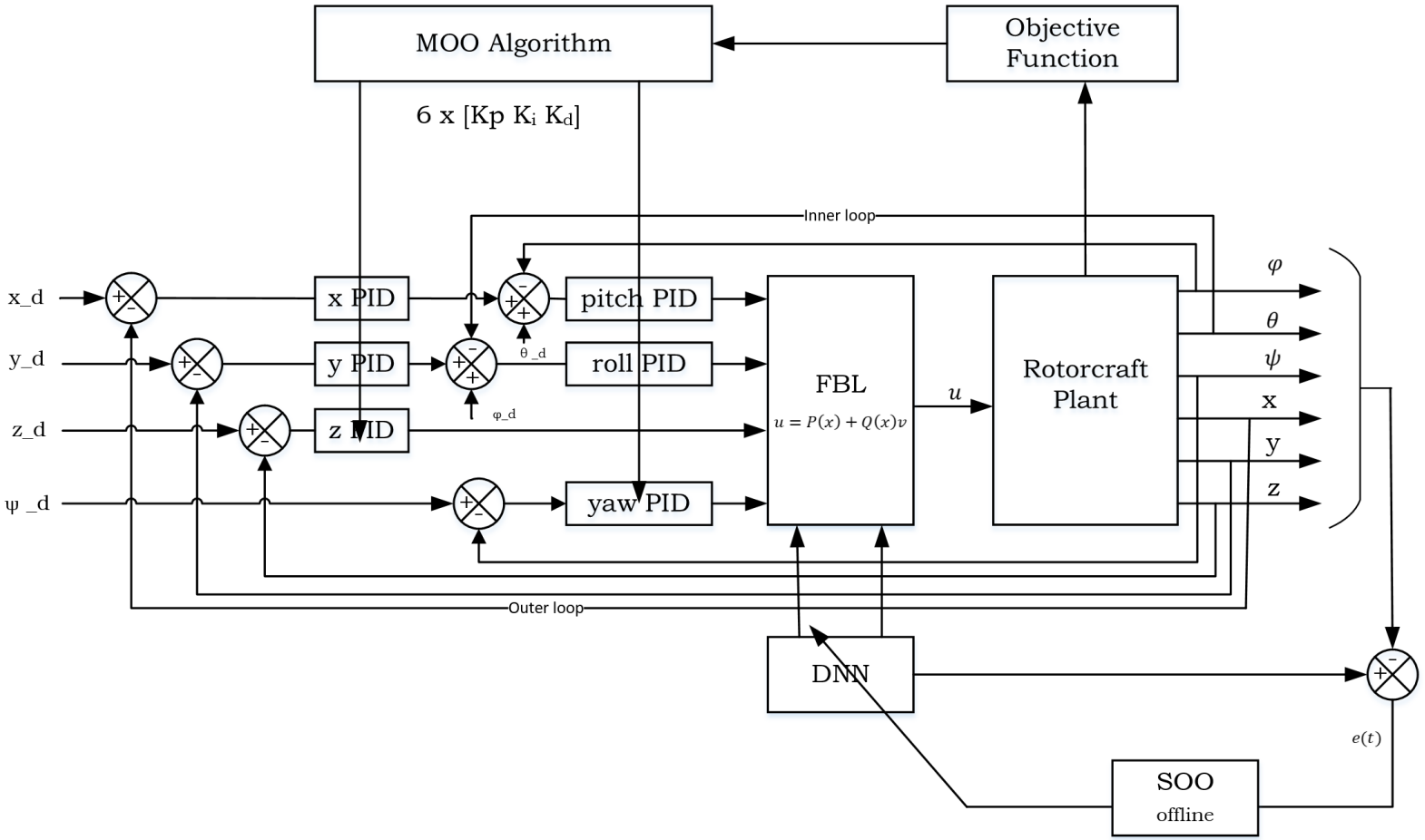


Figure 5.4: The DNN-based input-output feedback linearisation (DNNFBL) architecture with off-line SOO algorithm for DNN identification and MOO algorithm for PID gains tuning.

Two models were created for the rotorcraft system identification: the ALO-based DNN and ACO-based DNN. Each model has $\theta \in \Re^{50\times 1}$, a total of 50 parameters to be tuned. The ACO algorithm parameters chosen are are shown in Table 5.1.

Table 5.1: The setup parameters for the ACO algorithm in tuning DNN models.

Parameters	**Value**
number of ants, m	100
number of archives, k	180
maximum iteration, i	100
forgetting constant, ζ	0.8
pheromone constant, q	0.05

The parameters for the ALO algorithm algorithm parameters chosen are are shown in Table 5.2.

Table 5.2: The setup parameters for the ALO algorithm in tuning DNN models.

Parameters	**Value**
Search agents, k	100
Archive length, T	100

The fitness evolution of the selected DNN models for ACO and ALO are shown in Figure 5.5. The ACO shows faster convergence at around 30 iterations while the ALO converged at 90 iterations. The ACO has a validation error of 0.108 while the ALO has a validation error of 0.116

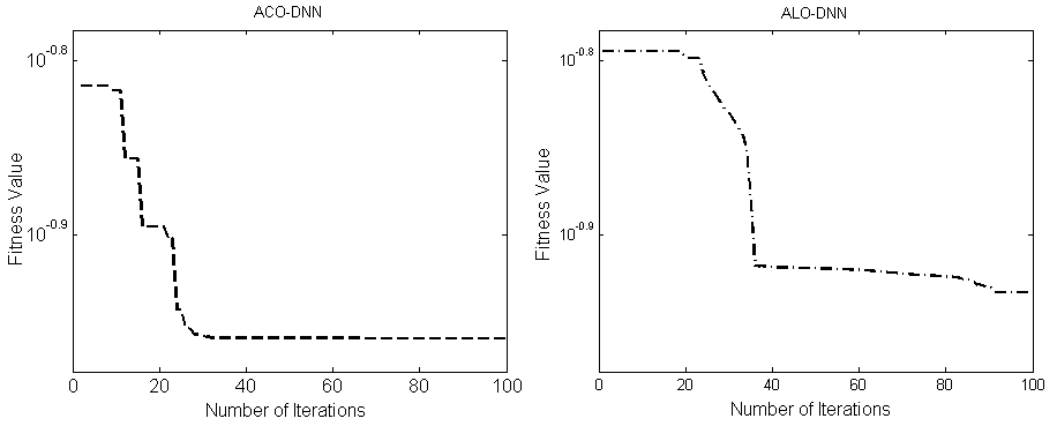


Figure 5.5: The system identification system fitness changes over the number of iterations.

Figure 5.6 shows how the trained DNN approximate the data from the real system. The DNN is able to predict the system's output well which means it has successfully learned

the rotorcraft dynamics.

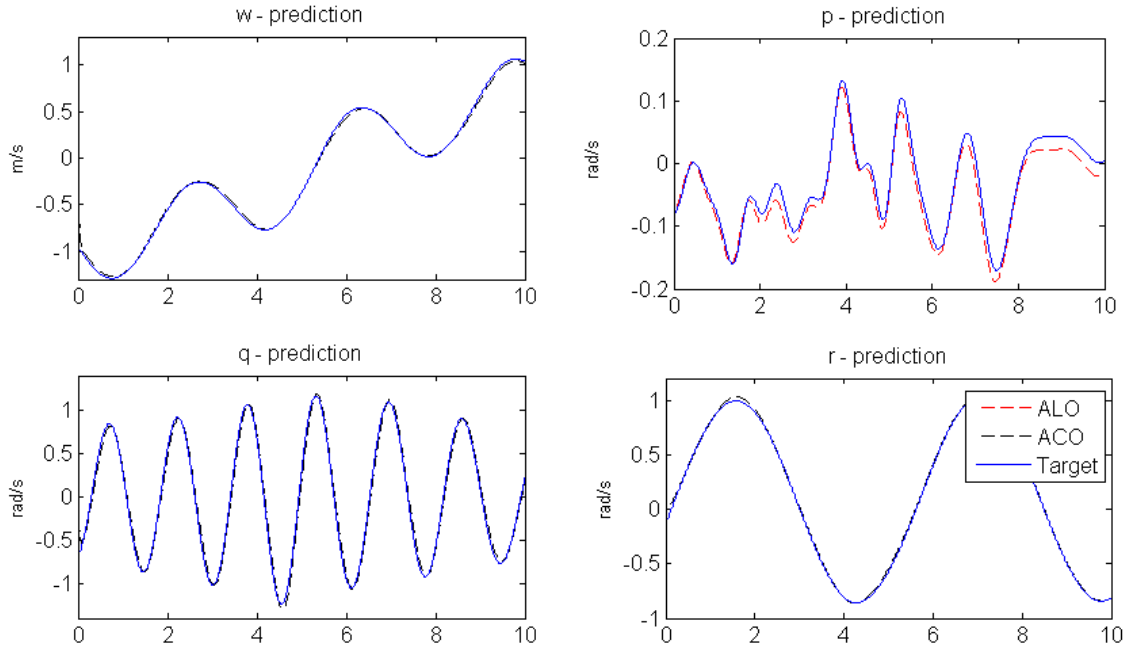


Figure 5.6: DNN predictions vs validation data collected.

The algorithms are further used to tune $[K_{P_j}\ K_{I_j}\ K_{D_j}]$ for the six PID controllers, with a total of 16 variables. The values of PID controllers' gains found using the MT, MOACO and MOALO techniques are given in Table 5.3. The corresponding MOALO and MOACO Pareto fronts are as shown in Figure 5.7. 15 trials were executed and the best result was selected based on the candidate solution that gives the best Pareto optimal set.

Table 5.3: The PID controller gains found using multi-objective optimisation algorithms.

PID gains	Roll	Pitch	Yaw	Altitude	Lon	Lat
MT K_p	60.0000	50.0000	25.0000	-15.0000	8.0000	-35.0000
K_i	0	0	-50.0000	25.0000	60.0000	-30.0000
K_d	0	1.0000	25.0000	12.0000	-3.0000	6.0000
ALO K_p	65.4128	43.8105	17.3546	-9.2113	14.7463	-41.4408
K_i	0	0	-49.7323	20.1482	47.6738	-28.0746
K_d	5.8625	1.6045	7.1407	9.6753	-7.5070	25.7663
ACO K_p	59.4986	51.4500	15.2475	-13.6239	12.0153	-1.0600
K_i	0	0	-54.5593	15.6975	55.0661	-34.7223
K_d	2.8425	0.5375	8.7758	3.5232	-4.0871	31.6719

Figures 5.8 and 5.9 show the performance of the tuned PID controllers in displacement tracking, velocity of the rotorcraft, the Euler angles and the control effort produced in each axis. All the controllers meet the performance specifications. However, the ACO-tuned

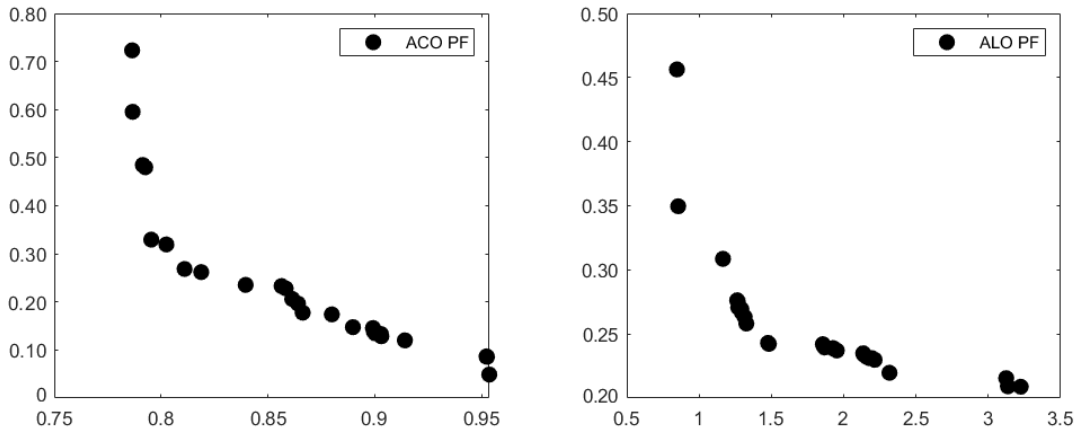


Figure 5.7: PID controller system Pareto front over the number of iterations.

controllers outperform the other ALO-tuned controllers.

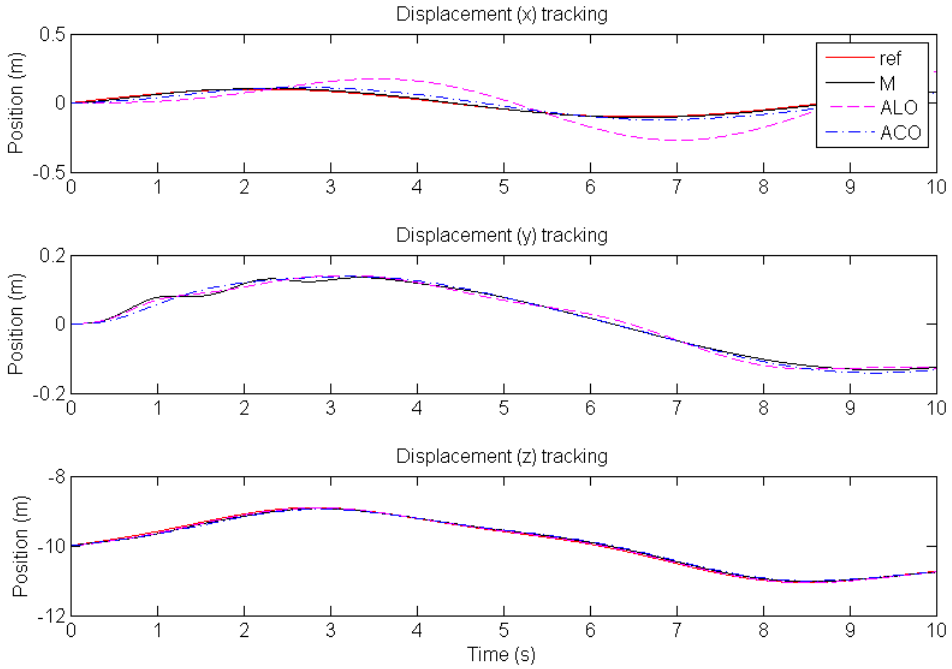


Figure 5.8: Comparing MOALO and MOACO PID-DNNFBL displacement tracking.

The best controller from the PID optimisation simulation is compared to the DNNFBL optimisation in order to contrast the performance of the two control strategies. The MOACO and MOALO were used to optimise the DNNFBL gains $[\lambda_{0j}\ \lambda_{1j}]$ and the associated PID controller gains $[K_{P_j}\ K_{I_j}\ K_{D_j}]$. A total of 24 parameters were tuned. The values of DNNFBL and PID controllers' gains found using the MOACO and MOALO techniques are given in Table 5.4. Figure 5.10 shows the associated Pareto front. Fifteen

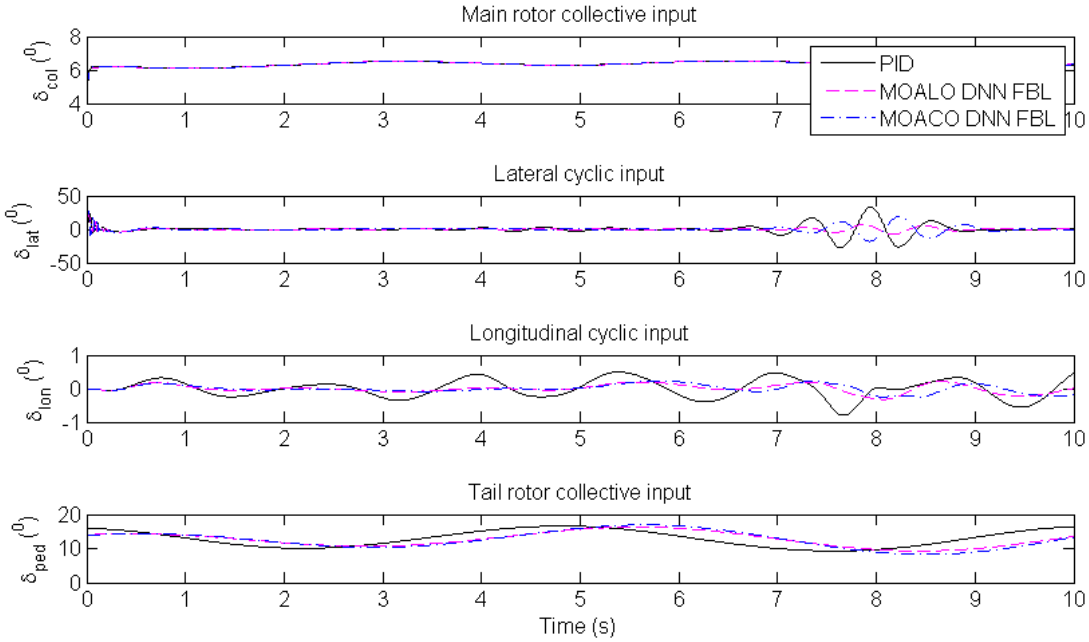


Figure 5.9: Comparing ALO and ACO PID control inputs.

trials were executed and the best result was selected based on the candidate solution that gives the best Pareto optimal set.

Table 5.4: The PID controller gains and the DNNFBL parameters found using multi-objective optimisation algorithms

PID gains		Roll	Pitch	Yaw	Altitude	Lon	Lat
ALO	K_p	38.4524	800.1077	129.9790	750.1017	-648.8120	93.7630
	K_i	0	0	-870.6659	946.4286	-34.3106	308.1582
	K_d	333.6327	100.3127	-893.0022	-265.7071	60.1670	-319.2285
	λ_0	-208.9696	-125.4109	-670.3806	947.6190	-	-
	λ_1	635.0942	-525.4168	23.2759	7.1495	-	-
ACO	K_p	491.0457	399.8292	-266.5365	11.9192	154.8345	-39.6656
	K_i	0	0	474.1616	5.9218	83.1601	393.813
	K_d	696.32	767.9171	252.6947	-276.9483	334.0301	28.0775
	λ_0	-463.4413	717.6914	414.8464	147.7619	-	-
	λ_1	890.0248	111.8247	166.4946	7.1495	-	-

All the points on the Pareto front are considered optimal. In this experiment, the selected Pareto point is closest to the origin. This is the point that has middle of the ground compromise between the tracking error and minimising the control effort. Table 5.5 gives the values of the points selected for each controller. The number of Pareto optimal gains to choose from are also shown. Looking at the values, it is apparent that the MOACO-based PID solution dominates the MOALO-based PID controller solution.

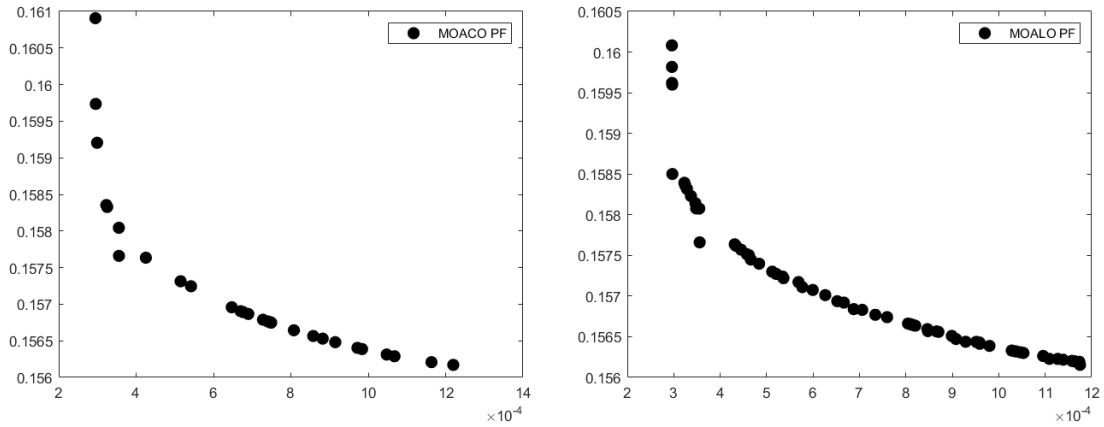


Figure 5.10: PID-DNNFBL system Pareto front over the number of iterations.

Table 5.5: The Pareto fronts for MOO-based PID controller gains and the FBL parameters found using optimisation. n-d means number of non-dominated solutions.

	J_1	J_2	n-d solutions
MOACO-PID	0.759	0.343	37
MOALO-PID	0.832	0.351	34
MOACO-DNNFBL	3.56×10^{-4}	0.1577	50
MOALO-DNNFBL	2.969×10^{-4}	0.1585	100

The controller was tested in tracking sinusoidal z and ψ while ϕ and θ are to be regulated at zero. The tracking performance is shown in Figure 5.11, the velocity is shown in Figure 5.12, and the Euler angles are compared in Figure 5.13. The control effort produced in each axis is shown to be within the specified range as illustrated in Figure 5.14.

The MOALO-based DNNFBL has the lowest tracking error, however the effectiveness of the produced force is similar for the MOACO methods. Also, both the MOALO-based DNNFBL and MOACO-based DNNFBL do not have transient oscillations before settling, while the ACO-based DNNFBL does. This means that the DNNFBL offers smoother control action when compared to the PID controllers.

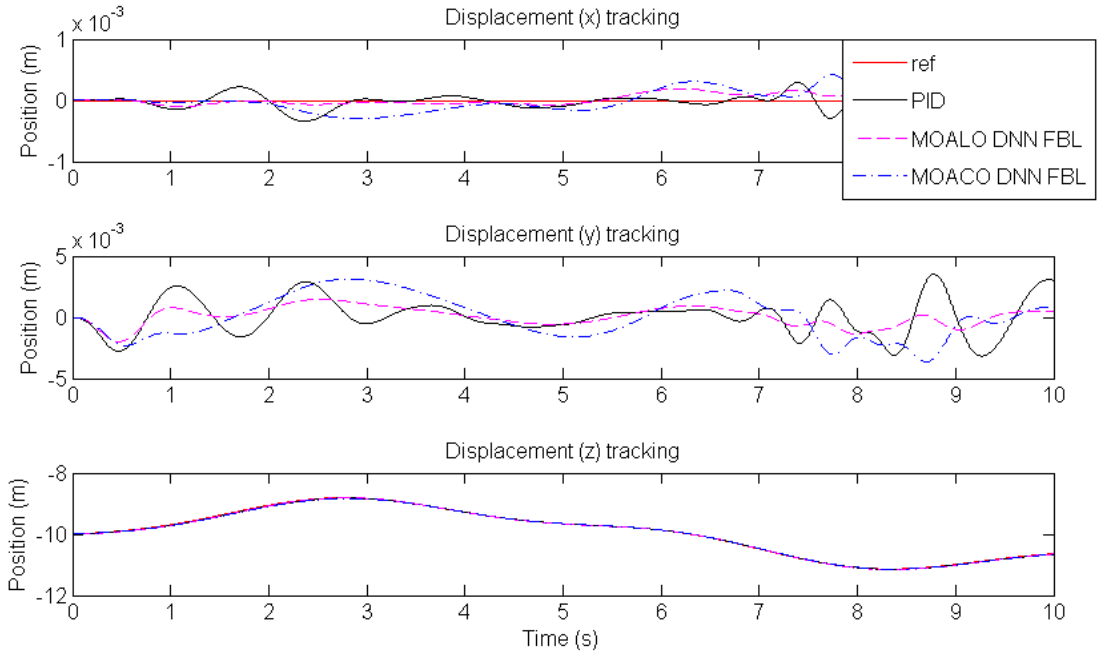


Figure 5.11: Comparing ALO and ACO PID-DNNFBL displacement tracking.

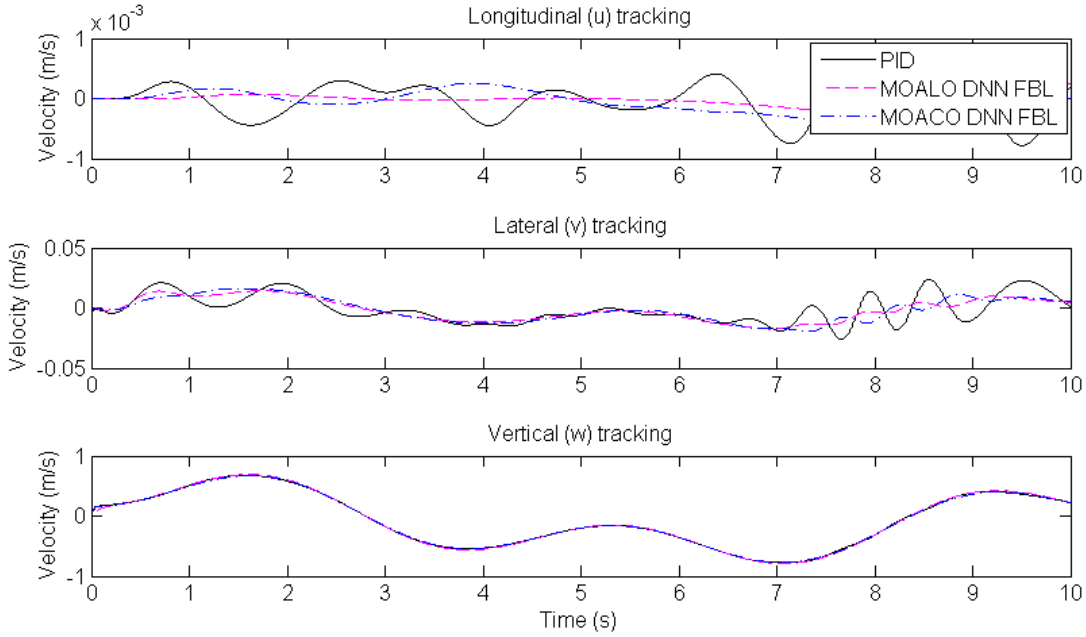


Figure 5.12: Comparing ALO and ACO PID-DNNFBL velocities.

.

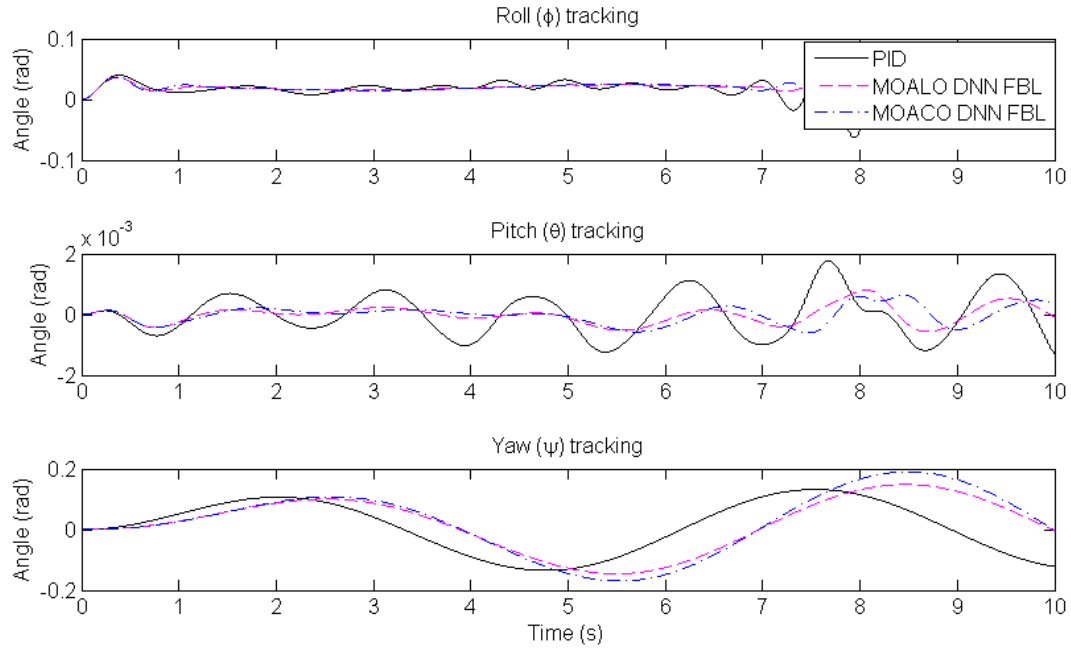


Figure 5.13: Comparing ALO and ACO PID-DNNFBL Euler angles.

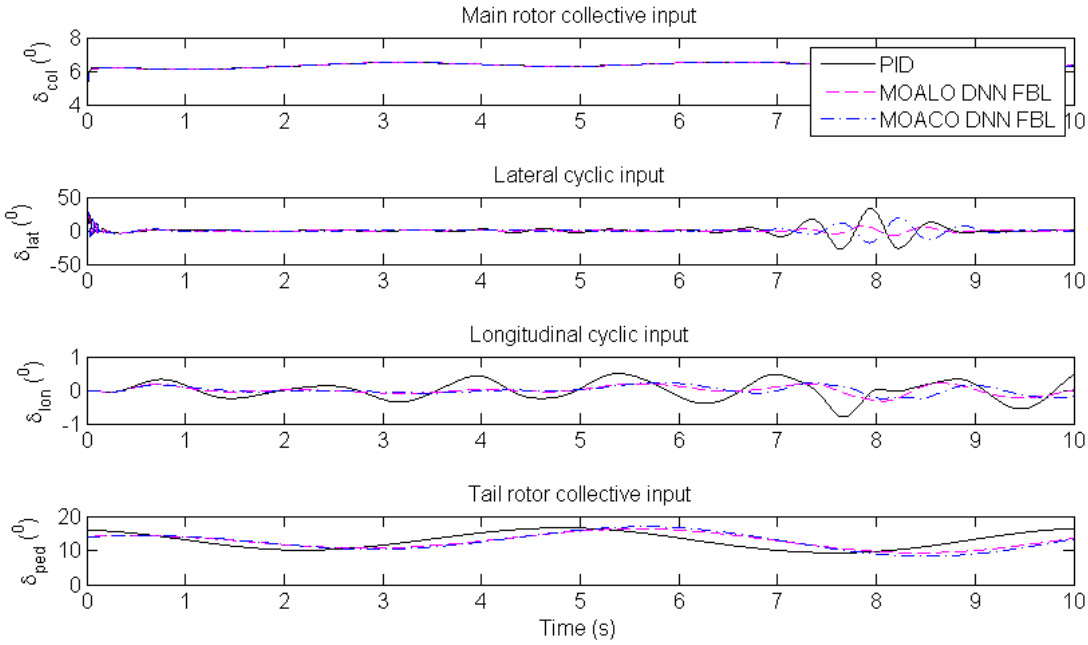


Figure 5.14: Comparing ALO and ACO PID-DNNFBL control inputs.

.

5.6 DNN, Fault Diagnosis and Isolation

A DNN is used to detect and identify actuator fault online. The fault identified are then used to reconfigure the control algorithms presented in the preceding section. The FDI is applied to the electromechanical swashplate as detailed in Section 2.7.1. The dynamic model of the swashplate is shown in Figure 2.3. The swashplate dynamics are identified using the DNN and used to identify the controller gains online.

FDI using DNN is one of the many ways to perform fault detection and isolation. The concept works by creating a bank of models that are used to generate a residual and using this residual to detect which of the actuators in the system has a fault. The following process is followed in creating DNN-based FDI:

- Collect input-output data from the healthy system using the swashplate model (in the next chapter, the data is collected from the experimental rig presented);
- Pre-process the data by partitioning it into training, validation and testing data. Scaling the data is also performed;
- Configure the DNN structure in terms of number of inputs, output and hidden nodes;
- Initialise the architecture of the network;
- Train the network using the collected data and the optimisation algorithm. In this part of research, ACO and GA are used as optimisation algorithms;
- Validate the trained model;
- Install the model into the AFTC as an FDI component.

Figure 5.15 shows the architecture of the integrated FDI which introduces the stages described above. The DNN is used online to create a residual signal based on the system inputs and the expected outputs. The residual evaluator and the identified threshold are used to detect the presence of a fault in the swashplate (Ding, 2008). Once a fault is identified, integrated flight-propulsion is activated to accommodate the fault.

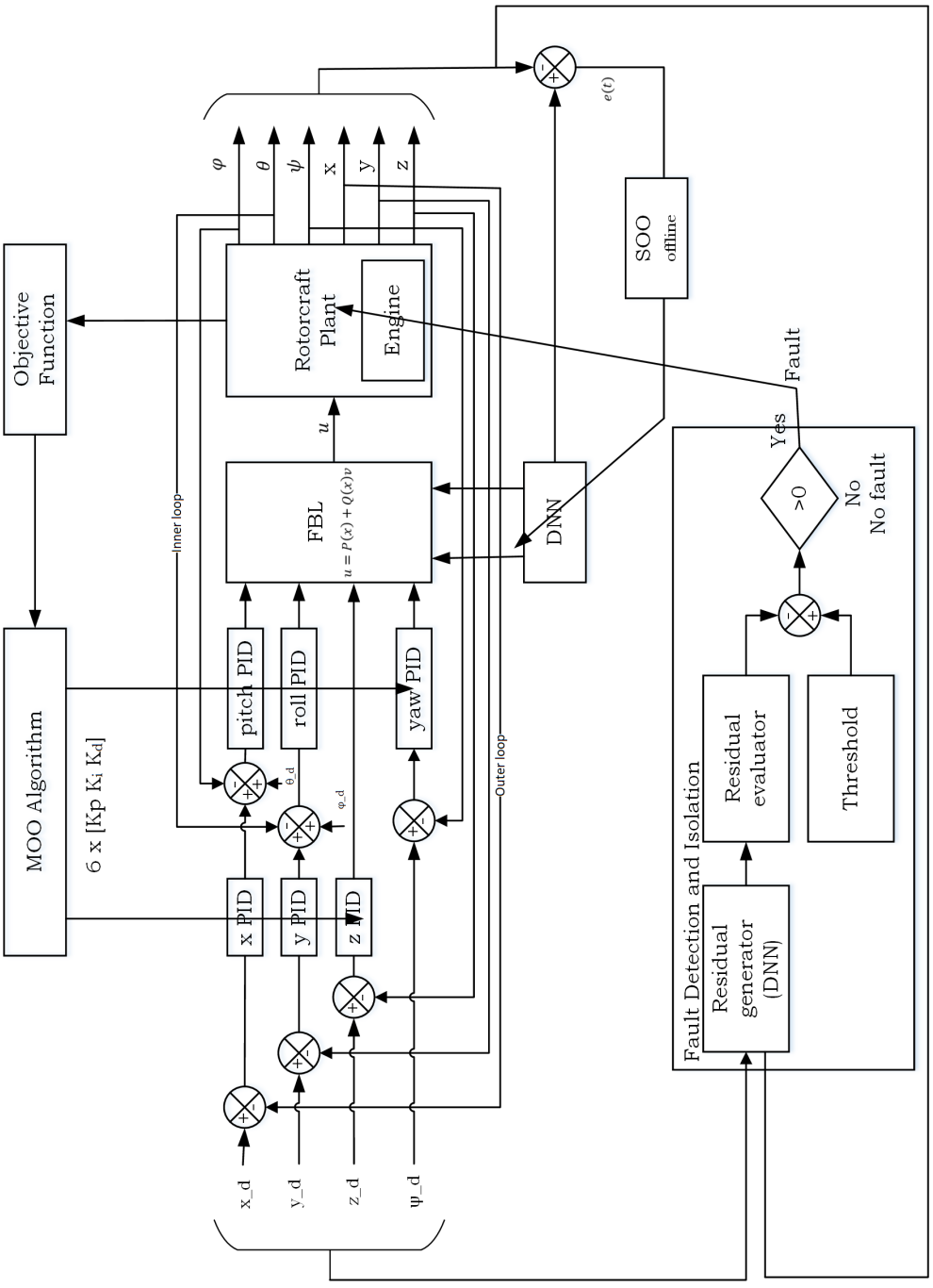


Figure 5.15: The DNN-based AFTC with the FDI and the control reconfiguration to use the engine control as a redundant control variable.

The three inputs to the swashplate are the desired positions of the actuators, while the outputs were chosen to be the vertical displacement, the pitch and the roll orientation of the swashplate. The selected inputs must be able to excite all the modes of interest in order to fully characterise the swashplate. This confirms Resnikoff conundrum of fault detection and maintenance which stated clearly that *"in order to collect failure data, there must be equipment failures"* (Crespo Márquez et al., 2016). The collected data uses polyharmonic signal generated using Equation 5.13 for the collective, lateral and longitudinal cyclic input. These inputs are chosen as follows:

1. Collective $a_1 = 1.0$, $a_2 = 0.7$, $w_{n1} = 0.1$, $w_{n2} = 0.7$.
2. Cyclic lat $a_1 = \pi/2$, $a_2 = \pi/15$, $w_{n1} = 0.7$, $w_{n2} = 1.15$.
3. Cyclic long $a_1 = \pi/2$, $a_2 = \pi/15$, $w_{n1} = 0.5$, $w_{n2} = 1.05$.

These are the inputs to the swashplate. The system identification is performed on the actuators taking into consideration the voltage input and the actuator travel as the output. The data was collected in a MATLAB simulation at $1kHz$ sampling rate. 75000 input-output data pairs were recorded and split into training, validation and testing data with 25000 cases in each.

The first step is to determine the number of hidden nodes in the DNN. The graph used for selection of the number is shown in Figures 5.16. Comparison of the history evolution of the RMS as the function of number of iteration/generation is shown in Figure 5.16.The training set had the lowest RMS with one state. However the validation showed that the best network structure is to have three states, that is, one output and two hidden states.

The training and validation of the identified DNN is shown in the Figure 5.17. It is seen that the DNN prediction data correlates well with the set measured from the actuator. The training and validation errors are 4.32×10^{-4} and 0.22 respectively.

After the training process, the newly trained DNN is included in the AFTC design. Three DNN models are used, one for each actuator.

5.6.1 Residual generation

The DNN models are used to determine the actuator positions when there is no fault present in the swashplate. When a fault occurs in the swashplate, the value of the output

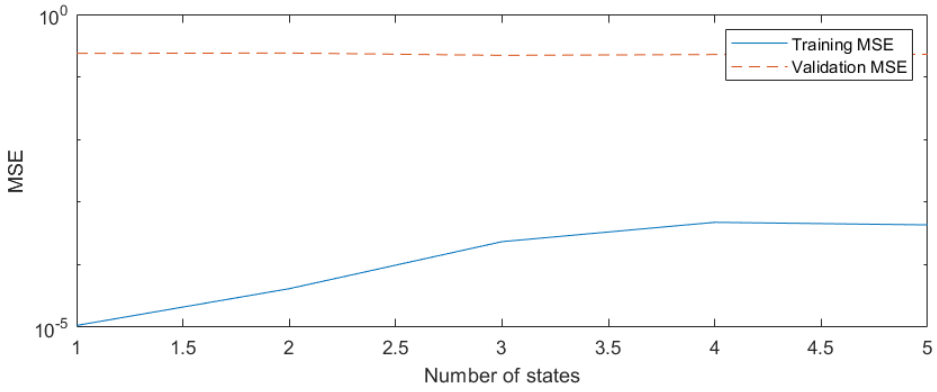


Figure 5.16: The determination of the number of states for the training of the DNN.

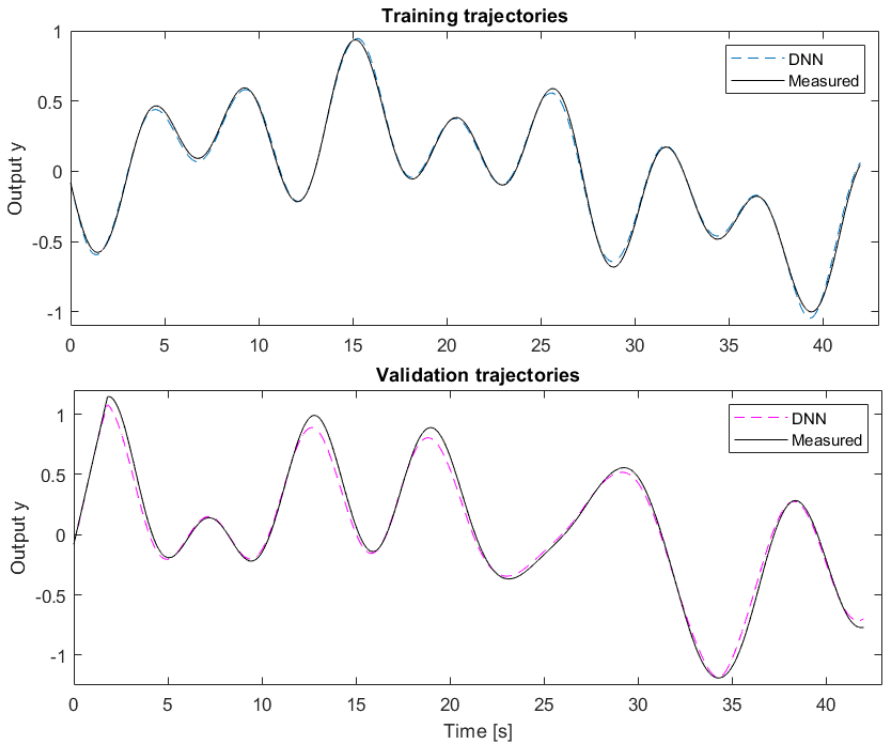


Figure 5.17: The training and validation of the DNN for actuator identification.

from the actual actuator will differ compared to the prediction made by the DNN.

Let the real position value of the actuator be z_i and the predicted value be $\hat{z}_i$, where $i = a, b, c$ for each actuator positions. Then the residuals are calculated as follows (Guezmil et al., 2018):

$$F_a = |z_a - \hat{z}_a|$$

$$F_b = |z_b - \hat{z}_b|$$

$$F_c = |z_c - \hat{z}_c|$$

The faults are detected and identified according to the binary diagnostic matrix shown in Table 5.6 (Korbicz et al., 2004).

Table 5.6: The use of residuals to identify faults in actuators a, b and c. μ_a, μ_b and μ_c are the fault thresholds.

F_a F_b F_c	Residual identified	Fault condition
$F_a < \mu_a$ $F_b < \mu_b$ $F_c < \mu_c$	0 0 0 0	No fault
$F_a > \mu_a$ $F_b < \mu_b$ $F_c < \mu_c$	0 1 0 0	Actuator A fault
$F_a < \mu_a$ $F_b > \mu_b$ $F_c < \mu_c$	0 0 1 0	Actuator B fault
$F_a < \mu_a$ $F_b < \mu_b$ $F_c > \mu_c$	0 0 0 1	Actuator C fault

In order to find the values of μ, a number of simulations were conducted. Taking into consideration the inherent robustness of the employed controllers, the residual threshold was set as the point where the controlled system started to fail. This value was found to be $\mu = 0.5$ for all the actuators. This occurs at the 35% loss-of-effectiveness mark.

5.6.2 Controller reconfiguration

For the actuator fault detected by the DNN, if there is LOE, the actuator will move by a fraction of what is required by the flight controller. When the actuator is stuck in place there will be no movement in response to the voltage input. Since the swashplate has three DOF, (x, ϕ, θ), one DOF is lost when one actuator has a fault. The following conditions have been chosen to respond to the respective actuator failures.

Sacrificing z in Table 5.7 allows us to apply the control of the rotor speed in order to make up for the loss of control in this DOF. This control is achieved by the fact that increasing or decreasing Ω results in an equivalent increase in thrust as given by Equation

Table 5.7: The response determined in order to react to a fault in a single swashplate actuator.

Actuator failure	Control maintained	Control sacrificed
Actuator A	ϕ and θ	z
Actuator B	ϕ and θ	z
Actuator C	ϕ and θ	z

2.14, repeated here for ease of reference.

$$dT_M = \rho\pi R^4 \Omega^2 C_T dr$$

With this equivalent relationship in mind, the allocation of control to the swashplate for each fault detected is reorganised as follows:

Actuator A failure:

$$\begin{bmatrix} z \\ z_b \\ z_c \end{bmatrix} = \begin{bmatrix} 1 & +\cos(\chi) & -\sin(\chi) \\ 1 & -\frac{\sqrt{3}}{2}\sin(\chi) + \frac{1}{2}\cos(\chi) & -\frac{\sqrt{3}}{2}\cos(\chi) - \frac{1}{2}\sin(\chi)) \\ 1 & \frac{\sqrt{3}}{2}\sin(\chi) + \frac{1}{2}\cos(\chi) & \frac{\sqrt{3}}{2}\cos(\chi) - \frac{1}{2}\sin(\chi) \end{bmatrix} \cdot \begin{bmatrix} z_a \\ \theta \\ \phi \end{bmatrix} \tag{5.26}$$

Actuator B failure:

$$\begin{bmatrix} z_a \\ z \\ z_c \end{bmatrix} = \begin{bmatrix} 1 & -\cos(\chi) & \sin(\chi) \\ 1 & +\frac{\sqrt{3}}{2}\sin(\chi) - \frac{1}{2}\cos(\chi) & +\frac{\sqrt{3}}{2}\cos(\chi) + \frac{1}{2}\sin(\chi)) \\ 1 & \frac{\sqrt{3}}{2}\sin(\chi) + \frac{1}{2}\cos(\chi) & \frac{\sqrt{3}}{2}\cos(\chi) - \frac{1}{2}\sin(\chi) \end{bmatrix} \cdot \begin{bmatrix} z_b \\ \theta \\ \phi \end{bmatrix} \tag{5.27}$$

Actuator C failure:

$$\begin{bmatrix} z_a \\ z_b \\ z \end{bmatrix} = \begin{bmatrix} 1 & -\cos(\chi) & \sin(\chi) \\ 1 & -\frac{\sqrt{3}}{2}\sin(\chi) + \frac{1}{2}\cos(\chi) & -\frac{\sqrt{3}}{2}\cos(\chi) - \frac{1}{2}\sin(\chi)) \\ 1 & -\frac{\sqrt{3}}{2}\sin(\chi) - \frac{1}{2}\cos(\chi) & -\frac{\sqrt{3}}{2}\cos(\chi) + \frac{1}{2}\sin(\chi) \end{bmatrix} \cdot \begin{bmatrix} z_c \\ \theta \\ \phi \end{bmatrix} \tag{5.28}$$

Notice in each of the Equations 5.26, 5.27 and 5.28 that the stuck actuator position is used as the reference input. Since the z position is not controlled by the swashplate, it is floating depending on the selected ϕ and θ. From the controller point of view the difference between the desired and actual value, $\Delta z = z - z_d$, will remain. This Δz is sent to the propulsion controller so as to replicate the effect of the desired thrust. This is known as "analytical repair" (Blanke et al., 2016).

Since the rotor speed is regulated at Ω_{ref}, the new engine speed is requested as follows (Enns and Si, 2003):

$$\Omega_r = \Omega_{ref} + \frac{\partial \Omega}{\partial z} \Delta z \tag{5.29}$$

where, $\frac{\partial \Omega}{\partial z}$ is the derivative found by numerical experimentation. It was found to be 3.15 $rad/s/mm$. This way, an AFTC is able to achieve analytical redundancy using rotor speed control.

A 50% LOE fault was injected at $7s$ and the residual generator was inspected. The residual generated from the DNN is shown in Figure 5.18. The FDI was able to identify the fault on actuator A. The residual $r(t)$ is shown to increase from time zero and then levels off after three seconds. This is consistent with the expectation during rotorcraft transients. At time $t = 7s$, the difference is sustained and exceeds the threshold. For this reason, the reconfiguration was selected to deal with the loss of actuator A as given by Equation 5.26.

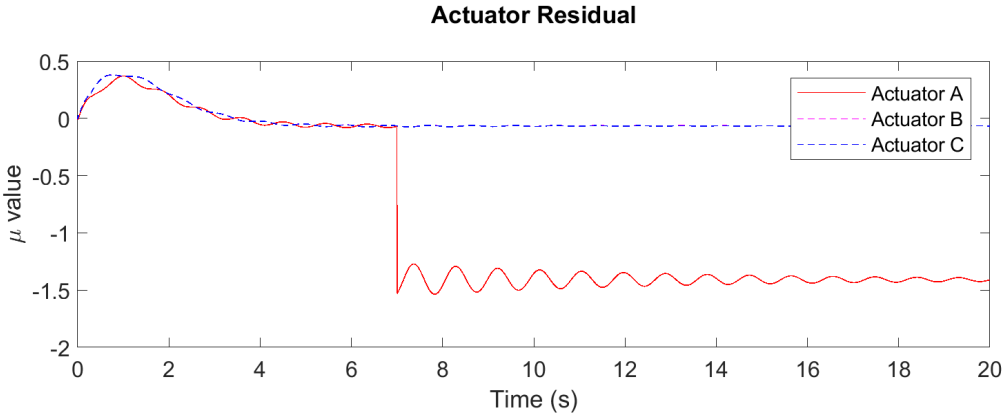


Figure 5.18: Residuals generated for a 50% LOE on actuator A.

In the figure, between zero and two seconds there is a departure from zero residual for all three actuators. This happens when the system is undergoing transient. However this residual is below the threshold. At the point where the fault is injected ($7s$), the residual on actuator A is shown to exceed the threshold and a fault is detected. The two points contrast trade-off between the false alarms that can occur, such as during transients and high positive fault detection rate.

The DNNFBL combined with DNN-FDI is able to control the rotorcraft and the performance is similar to the fault-free actuator response. Figure 5.19 shows the rotorcraft response in hover. When the actuator fails at $7s$, it can be seen that there is a loss of height. When the fault is detected and corrected, the height settles in a new steady state. The difference in height is detected in the range of $1mm$. This is below the resolution of any aircraft height instrument. Therefore, the FTC response is without loss of height.

The control signal produced by the DNNFBL input to the rotorcraft are shown in Figure 5.20. It can be seen that the collective is increased. This is to make up for the LOE in the actuator. This extra collective results in Δz which is detected by the residual generator. Once the fault is detected and using the reconfiguration discussed, this information is passed to the engine controlled.

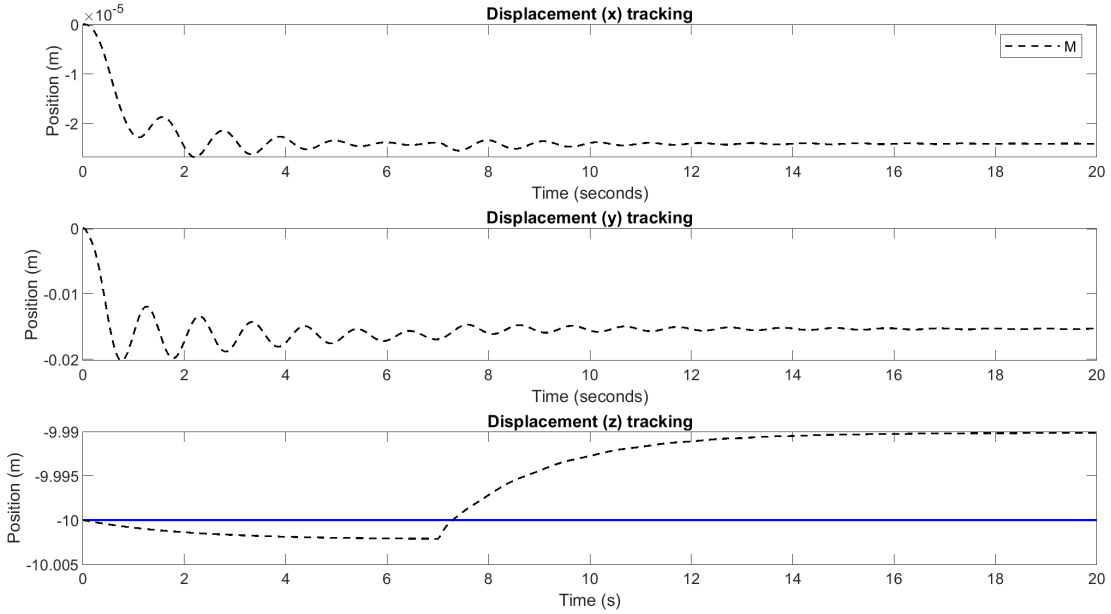


Figure 5.19: Rotorcraft position response under the DNNFBL controller for a 50% LOE on actuator A.

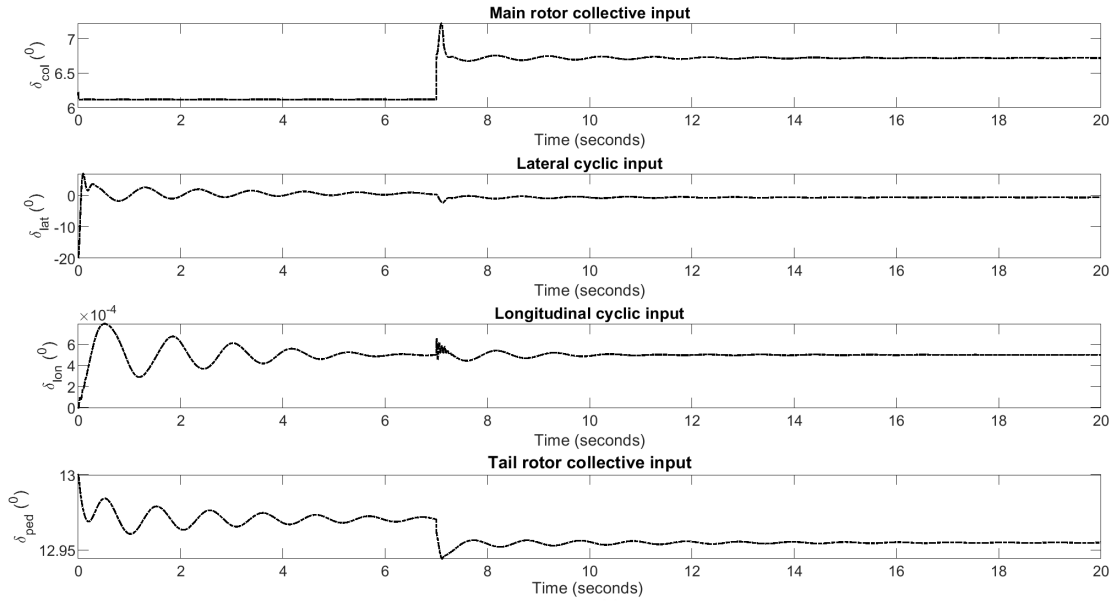


Figure 5.20: Rotorcraft control inputs under the DNNFBL controller for a 50% LOE on actuator A.

Figure 5.21 shows the residual detecting the fault in the actuator. The two other actuators are without faults. The response of the rotor speed to the reconfigured control is also shown. Since the desired change in rotor speed is directly related to the residual, it can be seen in that the graphs are similar.

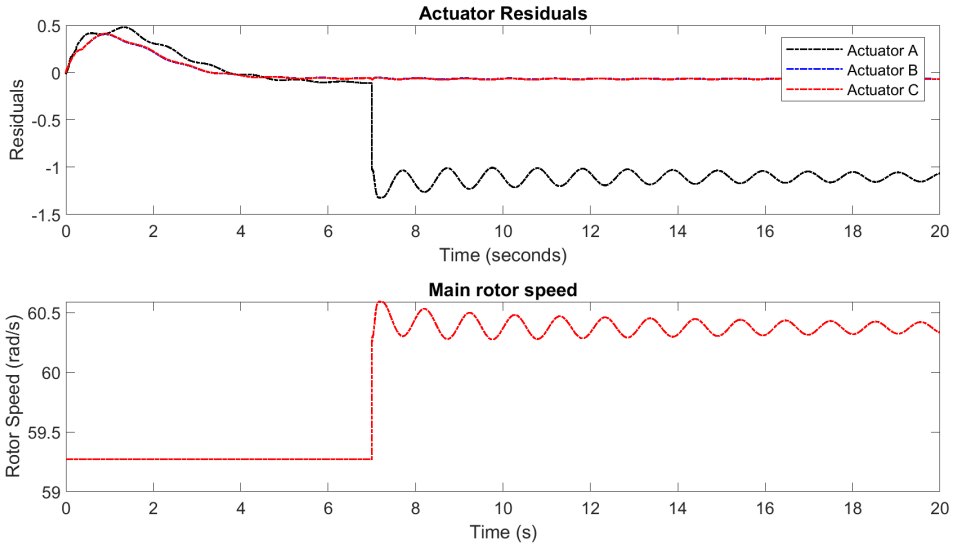


Figure 5.21: Residual generated for a 50% LOE on actuator A for the rotorcraft in hover.

The experiment is repeated for the rotorcraft in steady forward flight and when tracking as sinusoidal height reference. Figure 5.22 shows the rotorcraft response during a fault and the associated control inputs to keep the rotorcraft in stable forward flight are shown in Figure 5.23. There is a noticeable spike in the control input during the fault transients. The controls settles after a few seconds. The residual and reconfiguration control of the engine are shown in Figure 5.24.

The tracking is shown in Figure 5.25. The tracking control is effective even with the loss of effectiveness there is no detected height difference in the z direction. The x and y direction are also not affected.

The integrated flight-propulsion controller is able to keep the rotorcraft stable after an occurrence of an actuator fault and the presence of a gust wind. Figures 5.26 and 5.27 show the time history of the rotorcraft with a fault injected followed by a sudden gust wind in the y direction at $13s$. It can be seen that the rotorcraft is disturbed, however it does settle into a stable hover. The actuator residuals and the rotor speed are shown in Figure 5.28

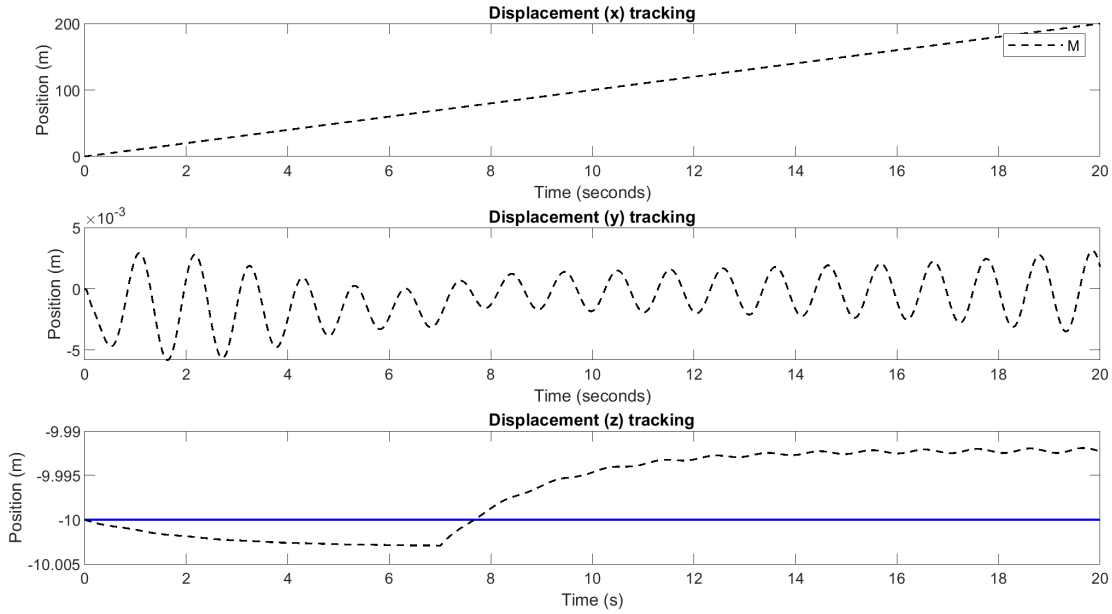


Figure 5.22: Rotorcraft position response under the DNNFBL controller for a 50% LOE on actuator A in forward flight.

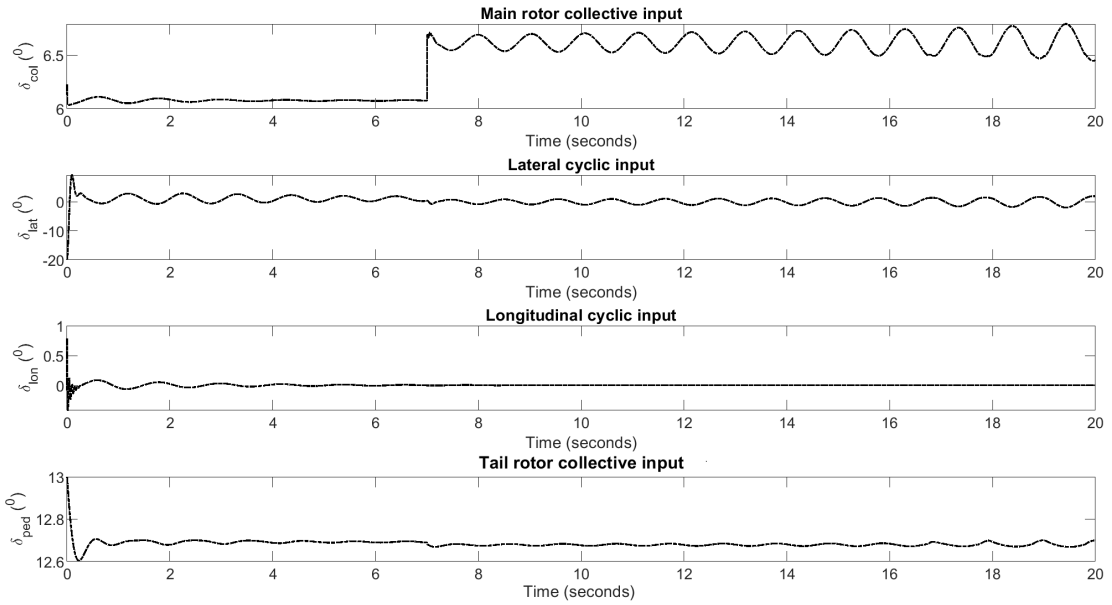


Figure 5.23: Rotorcraft control inputs under the DNNFBL controller for a 50% LOE on actuator A in forward flight.

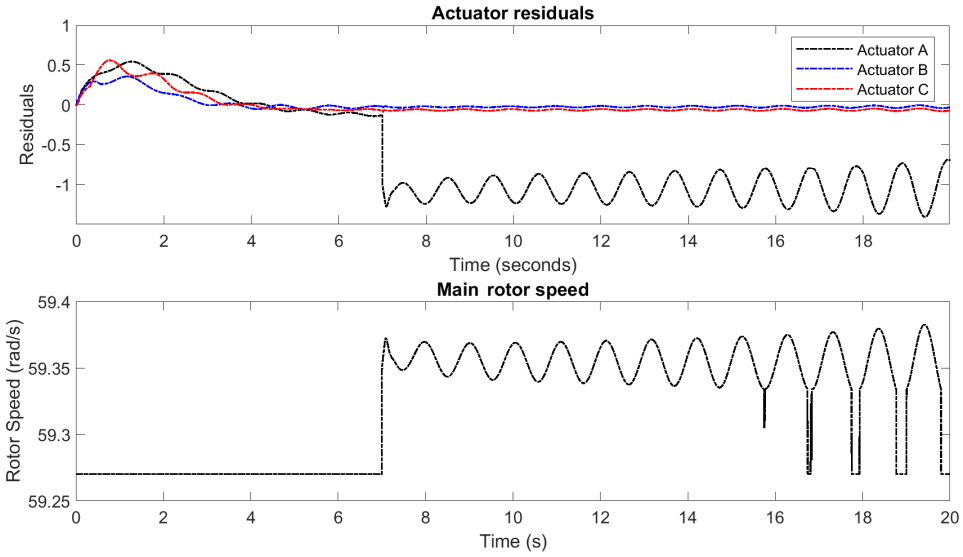


Figure 5.24: Residual generated for a 50% LOE on actuator A for the rotorcraft in steady forward flight.

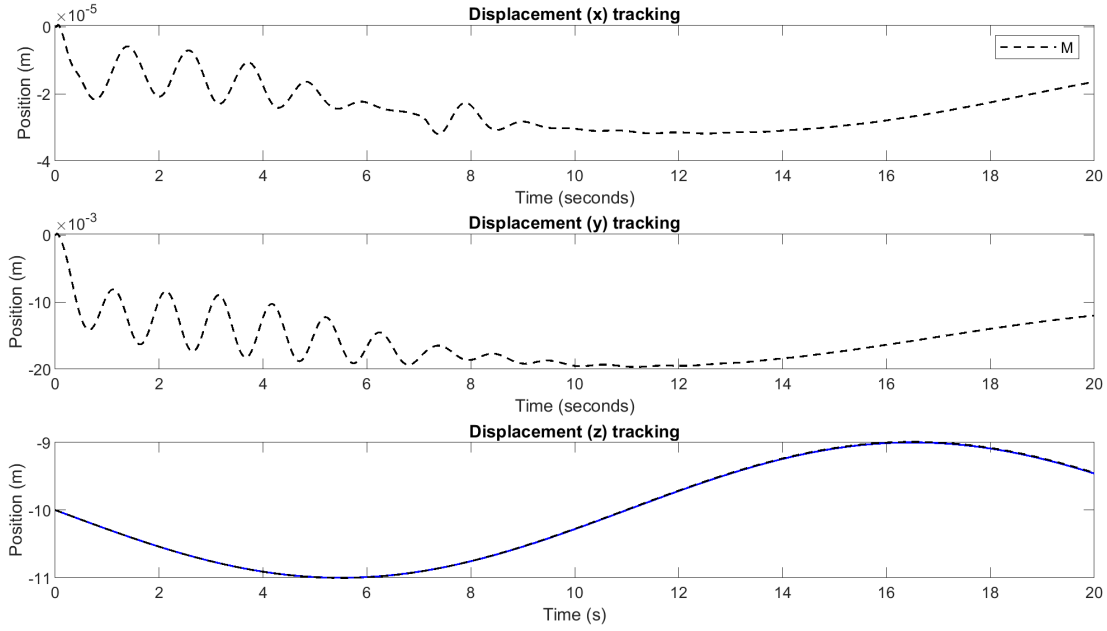


Figure 5.25: Rotorcraft position response under the DNNFBL controller for a 50% LOE on actuator A.

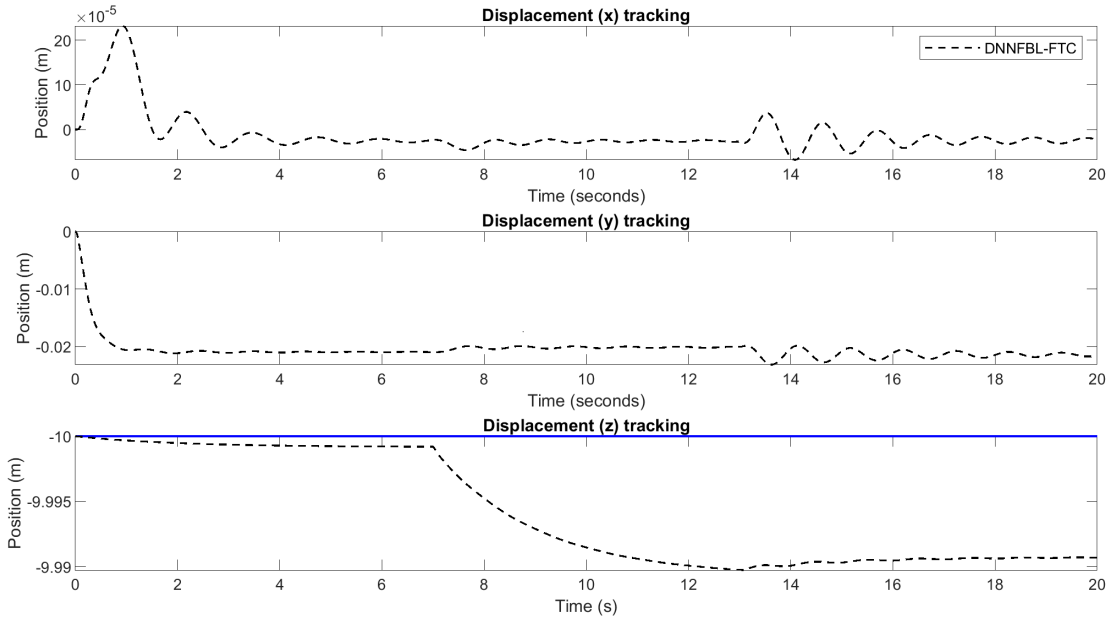


Figure 5.26: Rotorcraft position response under the DNNFBL controller for a 50% LOE on actuator A and the presence of $5m/s$ gust wind.

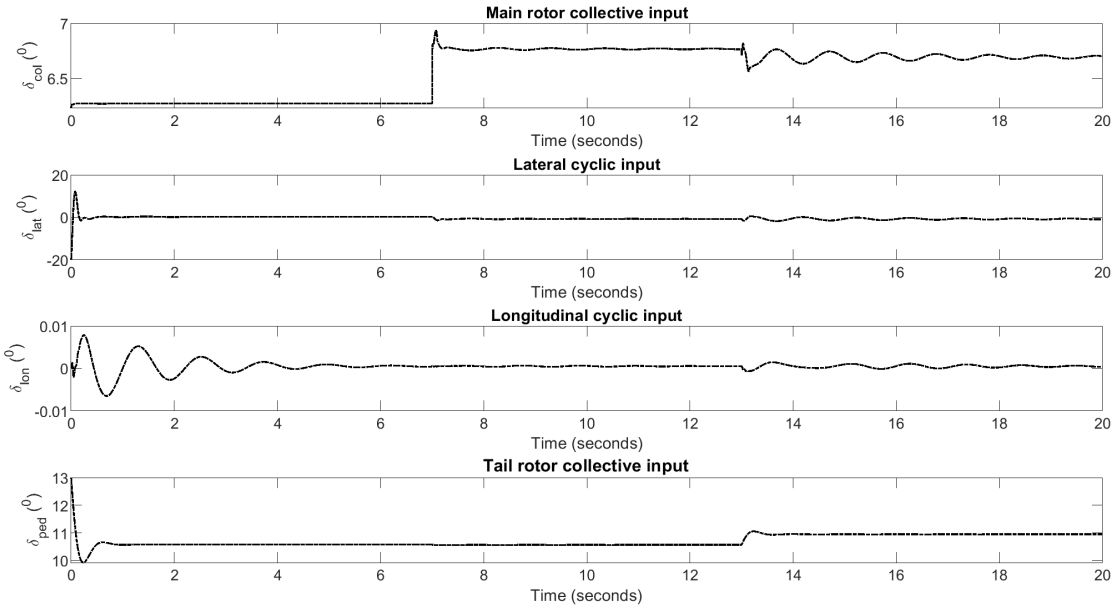


Figure 5.27: Rotorcraft control inputs under the DNNFBL controller for a 50% LOE on actuator A and the presence of $5m/s$ gust wind.

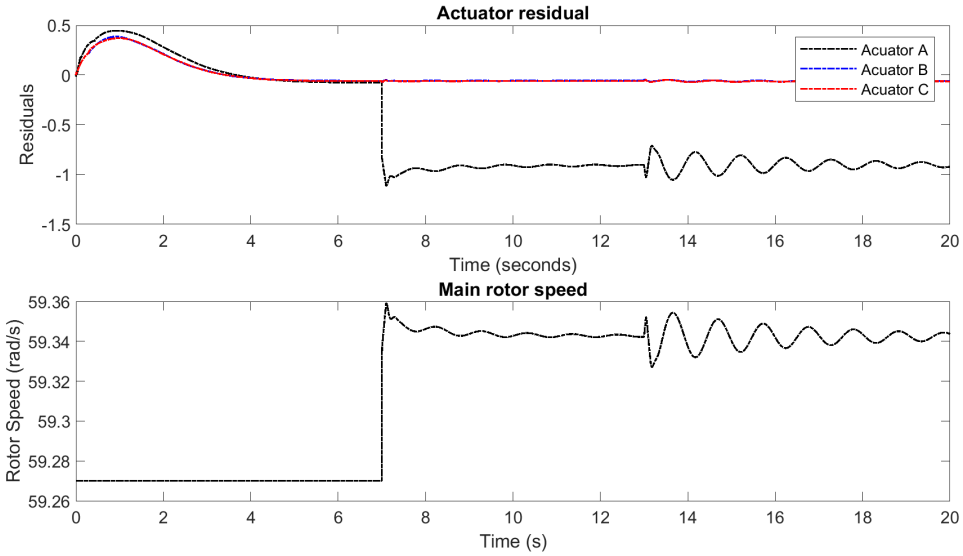


Figure 5.28: Residual generated for a 50% LOE on actuator A for the rotorcraft in the presence of $5m/s$ gust wind.

5.7 Chapter Summary

This chapter has presented the AFTC based on DNN and applied the strategy on a rotorcraft system. The results presented herein show improved controller performance to system's parameters variations, actuator faults and variable rotor speeds.

A closed-loop data collection and system identification for a rotorcraft UAV using DNN and FBL was presented. In order to do that, a mathematical model derivation and simulation of the system were presented. The main objective of this investigation was to evaluate the application of MOALO and MOACO for the PID-DNNFBL hybrid system controller gains optimisation. There are 16 and 24 parameters that were used for the two optimisation problems. The designed PID and PID-DNNFBL controllers were evaluated by tracking x, y, z and ψ.

Simulation results showed that the MOALO-based DNNFBL controller performed better than the PID controller and the MOACO-based DNNFBL in terms of pitch and roll tracking errors. In the translation tracking channel the MOALO results are comparable to the MOACO. Even though the results of MOALO-based DNNFBL and MOACO-based DNNFBL are comparable.

A FDI system using DNN identification was also developed to be incorporated to an AFTC strategy. Using control reconfiguration and integrating it with the variable rotor

speed control, the three actuator faults were successfully accounted for and the rotorcraft regain stability. The integrated system proved to be resilient to actuator faults in hover, steady forward flight and when tracking as sinusoidal signal. It was seen that the stability of the closed-loop control can be affected by the reconfiguration of the system. This was more noticeable in forward flight.

Chapter 6

Hardware-in-the-Loop Simulation

This chapter presents the hardware-in-the-loop simulations. The hardware that was developed is the swashplate mechanism used to control the collective, lateral and longitudinal cyclic. Actuators models are replaced with electromechanical actuators to determine how the proposed control techniques perform to real fault for experimental validation. Three different schemes are compared: the ideal actuator, the modelled actuator and the real actuator incorporated in hardware-in-the-loop. The control algorithms developed previously: the PID, STSMC and DNNFBL and compared. This chapter concludes by discussing software certification issues surrounding the developed algorithms

6.1 Introduction

The nonlinear model of the RUAV was discussed in Chapter 2. The flight control methods developed and results presented in Chapters 3, 4 and 5 were under the assumption that the swashplate actuators are ideal, i.e., they respond instantaneously to controller commands. However, real-life actuators have a finite transient period which results in a lag between the desired response sent by the flight controller and the actual response of the aircraft.

In this chapter, the response of the flight controllers that were identified and optimised in the previous chapters are compared to the response of the rotorcraft with actuator dynamics. In order to compare the actuator models to the real actuators, the optimal PID gains are used going forward, i.e., the PSO gains for hover and GA gains for forward flight. For the PFTC the STSMC optimised by PSO is used. For the DNNFBL it was

found that the MOACO tuned the DNNFBL better than its counterparts. Table 6.1 summarises the flight control methods to be validated using HILS.

Table 6.1: The summary of the flight control methods compared in this chapter.

Controller	Gain optimisation	Identifier
PID Hover	PSO	PSO-PID
PID Forward	GA	GA-PID
STSMC	PSO	PSO-STSMC
DNNFBL	MOACO	MOACO-DNNFBL

The experimental investigation includes the following:

1. Rotorcraft without actuator dynamics (as presented in the previous chapters);
2. Rotorcraft with modelled electromechanical actuator dynamics: analytical and system identification models;
3. Rotorcraft with electromechanical actuator dynamics.

Once the model and the real actuator have been compared and the response found satisfactory, then the PID, SMC and DNNFBL controllers are implemented for hardware-in-the-loop simulation (HILS). The description of the experiment setup is elaborated first, followed by the system identification of actuator and concludes with the control application and the execution of the experiment and results discussion.

6.2 Experiment Setup Description

6.2.1 Apparatus

The setup of the system is shown in Figure 6.1. The description of each item is given below.

Physical construction

- 1) The mechanical structure used to conduct the experiment is made up of aluminium square tubes of 25×25 mm. They form the reinforcement area and the base for holding the power electronics;

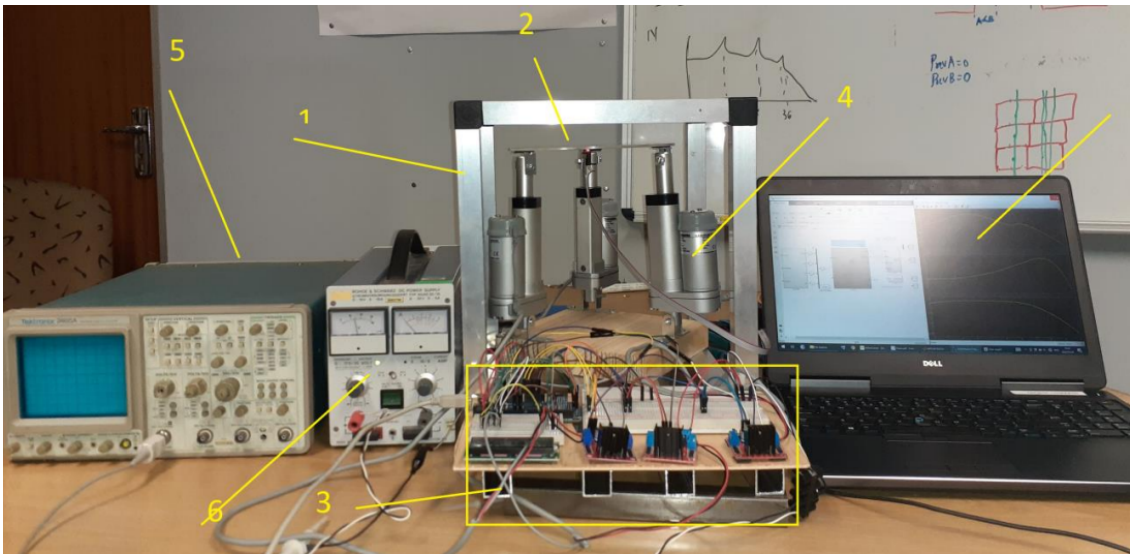


Figure 6.1: The experimental setup.

- 2) Makeshift swashplate is the structure made up of Plexiglass and its purpose is to support the fixing of the rate and angular sensors;
- 3) The base is constructed from wood and it is where the actuators are rigidly fixed. It also offers electrical insulation.

Electromechanical actuator

- Linear actuator: The actuator used is the Transmotec DLA-24-20-50-HS2-IP65 shown in Figure 6.1 as number 4. The internal of the actuator is shown in Figure 6.2.
- Power electronics: Arduino Mega 2560, the actuator is driven by a bi-directional H-bridge - the L298N motor driver board module. A pulse-width-modulated (PWM) signal is sent to the H-bridge to control the speed of the actuator. This group is labelled 3 in Figure 6.1.

Power supply and measurements instruments

1. High-power DC supply;
2. Oscilloscope with 4 input channels;
3. Digital multimeter.

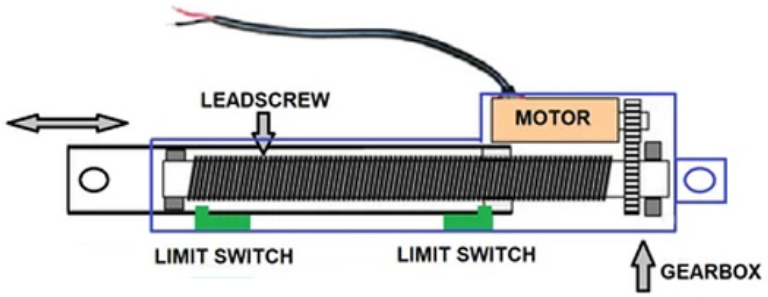


Figure 6.2: Actuator internal.

Computers

1. Simulation computer with MATLAB®/Simulink®, V-Realm® and Arduino® IDE v1.8.19, i7-6560U CPU 2.2 GHz and 8.00 GB RAM.
2. Visualisation computer with X-Plane® 10 and Flight Gear® 2020, i7-6820HQ CPU 2.7 GHz RAM and 16 GB RAM.
3. 10/100Mbps Desktop Switch for networking the two computers.

6.3 Methodology

The method followed in conducting the experiment is shown in Figure 6.3. The actuators are controlled from an Arduino Mega 2560. This micro-controller was chosen due to its versatility. It has a four PWM channels. These are enough for the three actuators used in this study.

The first part of the study was to connect the Hall-effect sensor and make sure that the position feedback works as specified in the manufacturer's manual. The Hall-effect uses magnetic field sensor to create a pulse when a magnet passes. The displacement of the actuator is measured by counting the number of pulse that passes from a specified reference point. Figure 6.4 shows the test results for the actuator full-scale deflection. The full-scale deflection of each actuator was measured as 305 pulses. This converted to 50 mm. The manufacturer gave an error of ± 1 mm. In this experiment, the measurement of full deflection has a standard deviation of 3.5 pulses. This is equal to 0.5341 mm, which is within specification of the steady-state error. The velocity was measured at $12mm/s$. An attempt to increase this velocity resulted in a higher error in pulse counting. A trade-off must be struck between speed and accuracy.

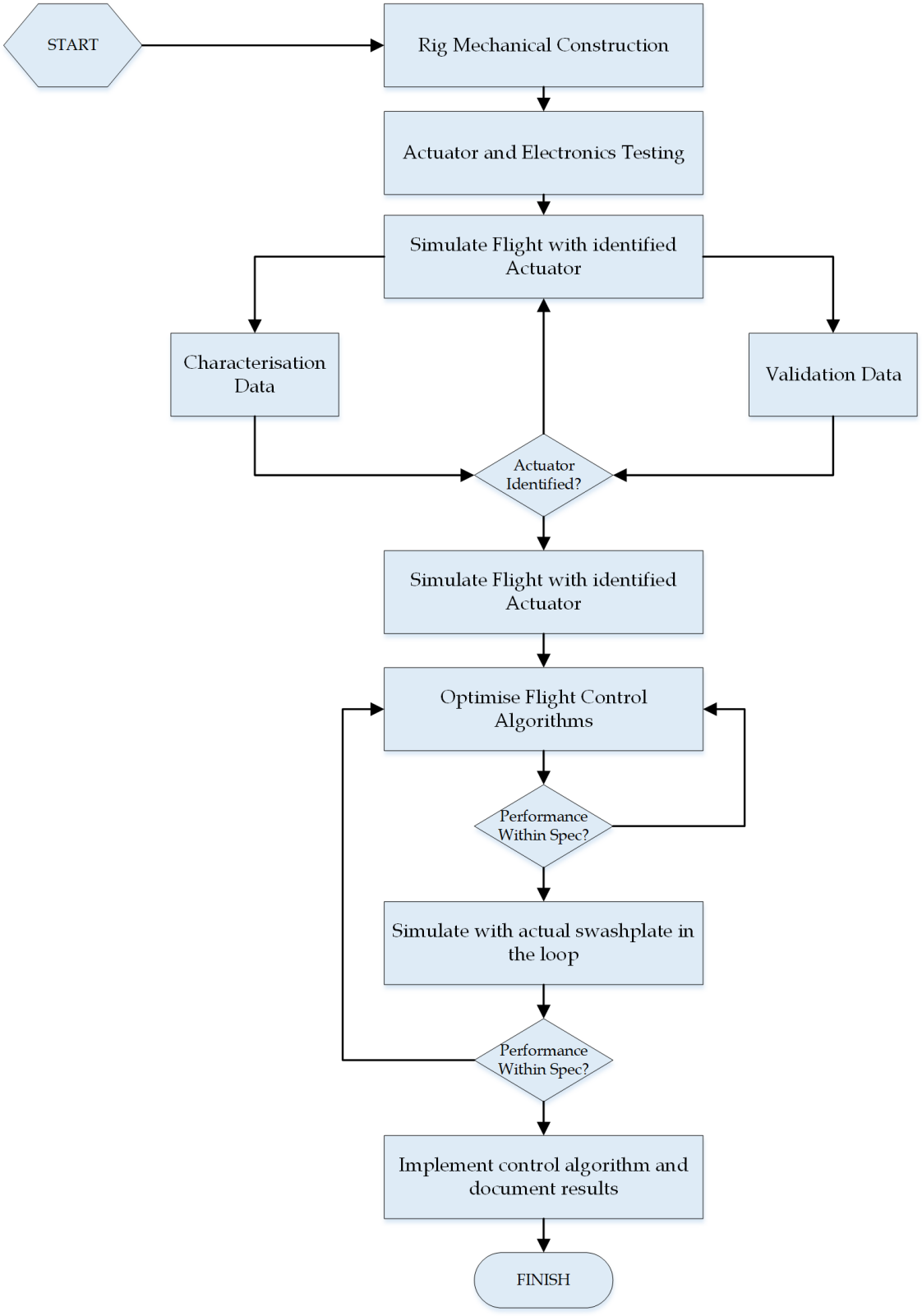


Figure 6.3: Flowchart of the methodology followed in conducting the HILS experiment.

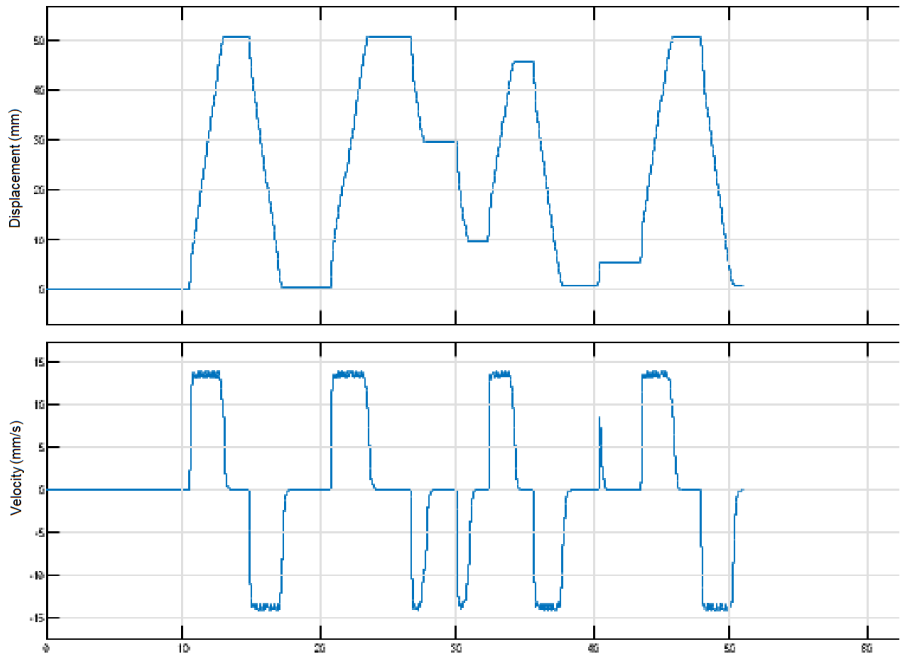


Figure 6.4: Actuator full-scale deflection test.

The Arduino was configured to transmit and receive data using the USB serial protocol built into the micro-controller. The serial communication channel was configured at 115200 baud. This was chosen to avoid any latency that might be induced as low transfer frequency.

The data was read in Simulink using the Instrument Control Toolbox^{TM} serial send/receive library. The data was collected to the MATLAB® workspace. This data was used for system identification of the actuators. System identification was conducted using predicted error minimisation (PEM) using the Levenberg-Marquardt algorithm. The model for using PEM in transfer function is as follows:

$$T(s) = \frac{Y(s)}{V(s)} = \frac{133}{0.1s^2 + 120s} \tag{6.1}$$

where $Y(s)$ is the displacement and $V(s)$ is the input voltage in Laplace transform.

The actuator model discussed in Chapter 2 was refined using the measured experimental data. A different set of data was collected to validate the models recovered. The comparison of the model and the experiment is shown in Figure 6.5.

The model demonstrates a match with the real data collected from the actuator. The difference in delay between the two results are attributed to communication delay loop from Simulink to Arduino to Actuators and the return path.

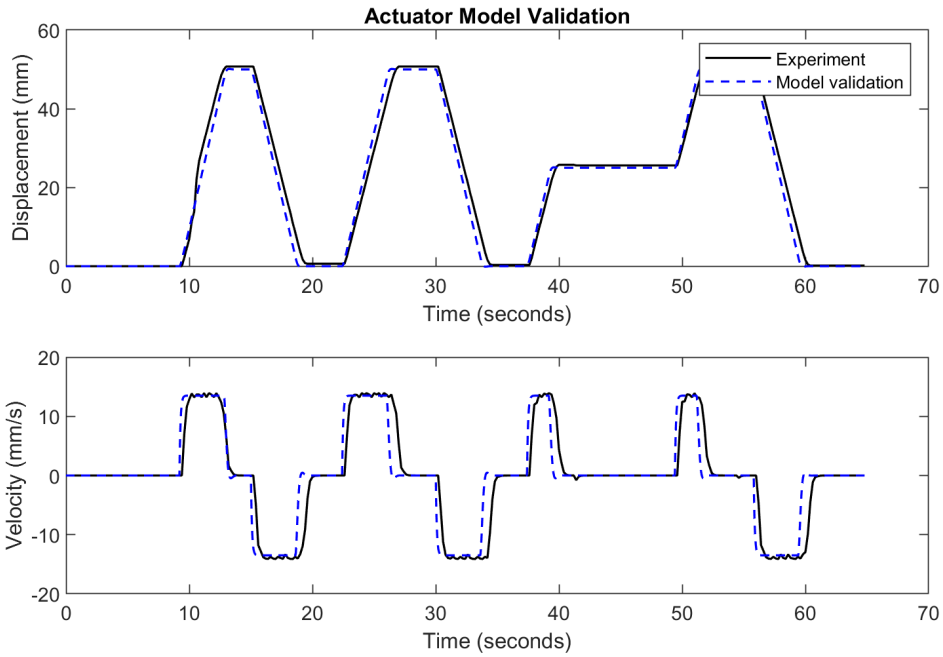


Figure 6.5: Actuator model validation with full-scale deflection data.

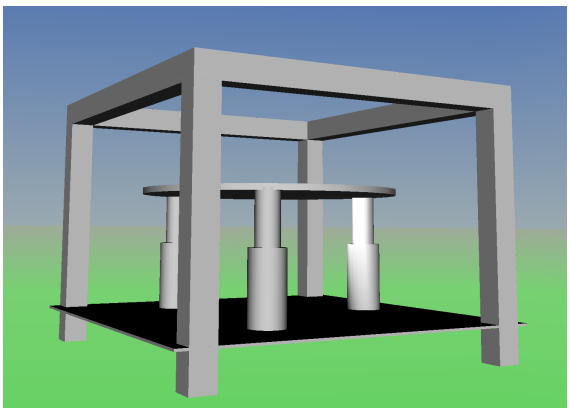


Figure 6.6: The digital twin of the hardware rig created in V-Realm® and integrated in Simulink®.

The characterised model is incorporated into a swashplate digital twin shown in Figure 6.6. The digital twin is created in V-Realm and connects directly to MATLAB®/Simulink®. The use of a digital twin assists in rapid prototyping actuator controllers. Once the controllers are ready, they are uploaded onto the micro-controller to drive the hardware actuators.

6.4 Sensors and Data Collection Limitations

The control of the swashplate depends on the sensor information used to close the loop. The variables of interest are z vertical displacement, the pitch θ and the roll ϕ of the swashplate. These variables cannot be measured directly. Unlike the experimental rig described here, in an actual rotor-head the rotor mast goes directly through the swashplate and the exact height cannot be measured. The same applies to the angles. In the test rig the following provisions have been made.

1. The vertical displacement can be inferred by taking the average position of the actuators, z_i for $i = a, b, c$;
2. The angular orientation of the swashplate can be derived trigonometrically using the same actuator position information;
3. In addition, the accelerometer is used for $\ddot{z}$ and the height position is found by integration; and
4. The gyroscope is used to find the $\dot{\phi}$ and $\dot{\theta}$ from which the orientation of the swashplate is derived.

6.5 Actuator Control Design

The swashplate has three actuators that are independently controllable. This part of the experiment is to make sure that the actuator are properly controlled and their reaction is fast enough for the FCS. The PID controller is compared to the SMC for actuator control. In keeping with the other controllers presented, the optimal gains were found using the Cuckoo Search optimisation algorithm. The parameters that are used to setup the CS algorithm are shown in Table 6.2 (Mpanza and Pedro, 2020).

Table 6.2: CS setup parameters for PID and SMC tuning for the swashplate actuators.

Parameters	**Value**
Number of nests, m	20
Number of birds, k	30
Discovery rate, P_a	0.25
Maximum iteration, i	100

The convergences of the training is shown in Figures 6.7 and 6.8.

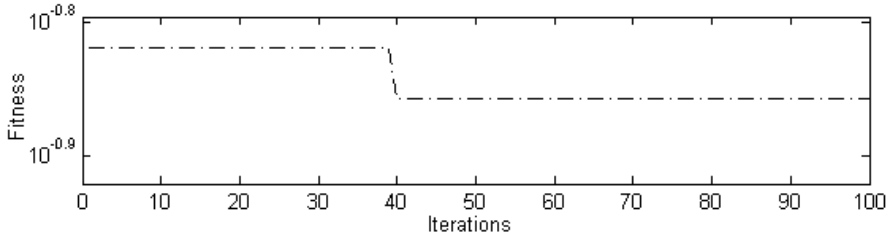


Figure 6.7: Fitness values of changes over time using CS-based PID for the swashplate actuators.

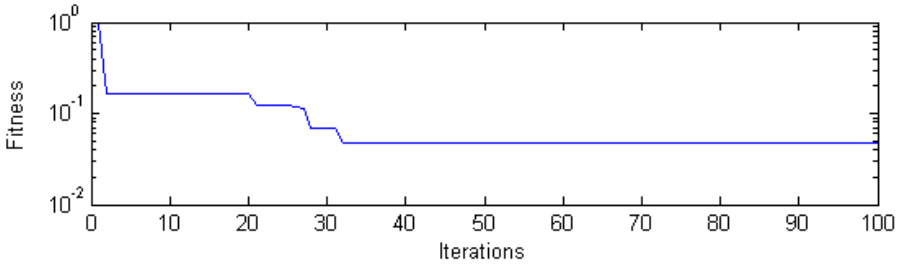


Figure 6.8: Fitness values changes over time using CS-based SMC for the swashplate actuators.

The swashplate system was design and tested using a step reference signal at $t = 0$ for vertical displacement, at $t = 1.5$ for the roll angle and at $t = 3$ for the pitch angle. The comparison graphs of the CS-tuned PID controller and the CS-tuned SMC results are shown in Figure 6.9. Both controllers show smooth and accurate tracking the desired step signal. The steady-state error for vertical displacement is 1.99 mm, which is within the error resolution of the actual actuator as shown prior. The steady-state error for pitch, θ, and roll, ϕ, are 4.71×10^{-4} rad and 5.5×10^{-4} rad respectively. Figure 6.10 shows the errors in each of the EMA controlled by CS-tuned SMC. All three errors show asymptotic convergence to a neighbourhood of zero.

It is noticeable in the z tracking of MT-PID controlled system that the achieved steady-state error is outside the specifications. The PID controller shows improved steady-state

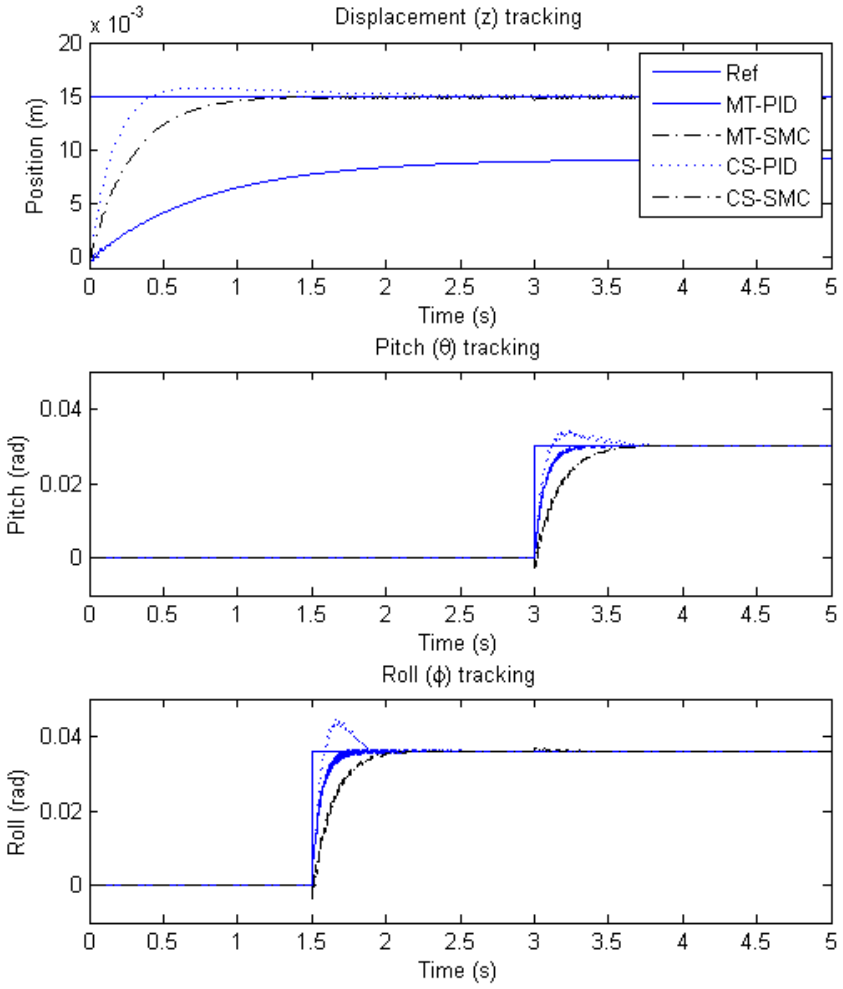


Figure 6.9: CS-based PID and SMC controlled swashplate tracking response for the desired reference signals.

performance when optimised using CS for gains tuning. This, however, resulted in a large overshoot for ϕ and θ of $14,33\%$ and 23.33% respectively. Due to this limitation of the PID controller, only SMC tuned by CS is discussed for the remainder of this thesis.

The robustness of the CS-based SMC in handling parameter variations was evaluated. The model swashplate actuators was subjected to changes in the mass of the swashplate by reducing and increasing by up to 80%. The controller was able to handle these changes without noticeable change in the performance. These robustness results are depicted in Figure 6.11. This invariance property of the SMC is upheld and the developed controller will be robust to system wear and ageing, and new parts with production defects.

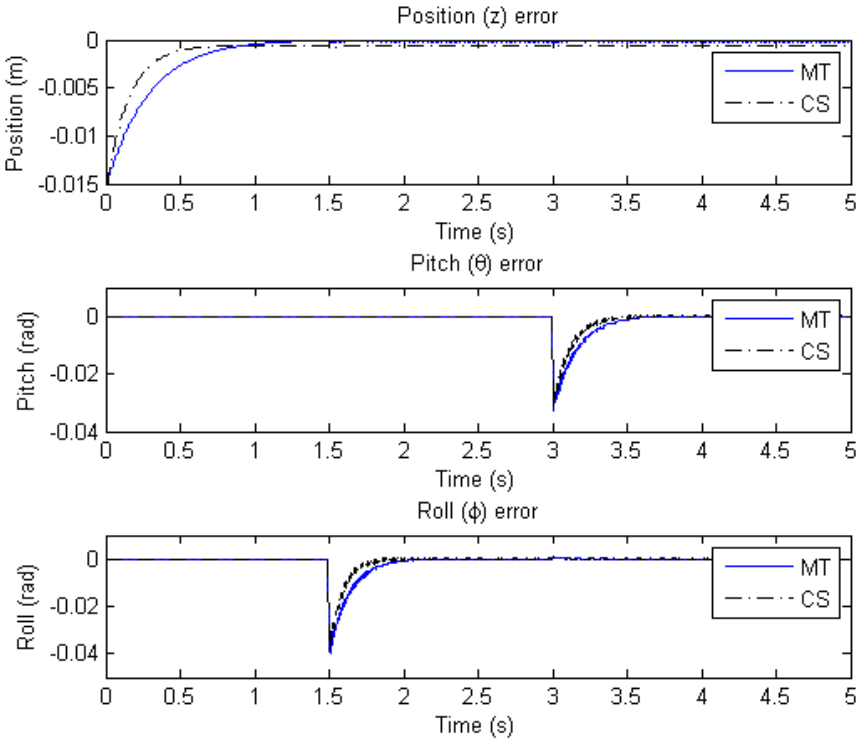


Figure 6.10: CS-based SMC swashplate errors command for the desired reference signals.

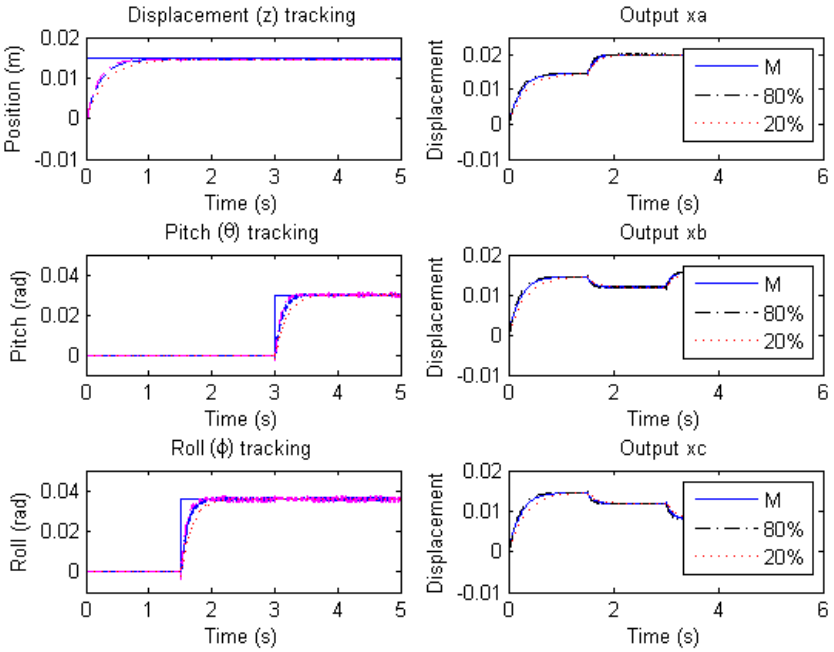


Figure 6.11: Robustness testing with parameter variations.

6.6 Comparing the Rotorcraft Response with Actuator Dynamics

The model identified and the actuator controllers presented in the prequel are used in the simulation. The simulation for PID, SMC and DNNFBL are repeated here. To compare the response of the actuators to the flight control, the ideal actuators' reference input from the controllers is sent through to the actual actuator without the corresponding output being sent to the rotorcraft dynamics. This way, it is possible to see how the real actuator response compares to the flight control request. Figure 6.12 shows the comparison of these response for the three actuators. The response of the actuators does not track the request. The delay in rise time has affected the position response such that it is out of phase and it does not reach the peak position that is required.

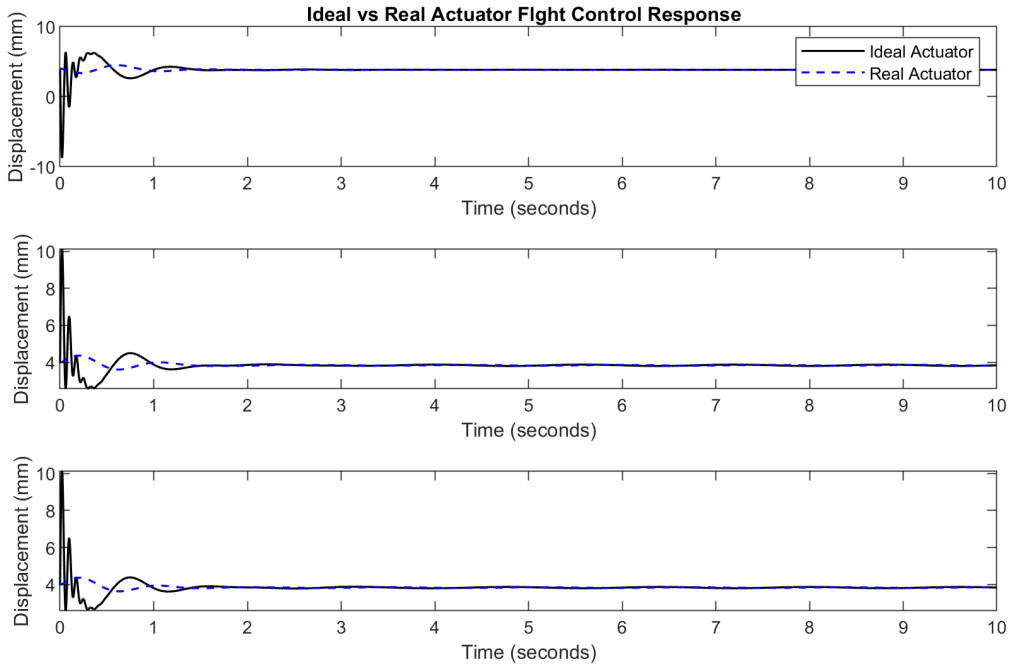


Figure 6.12: Ideal compared to real actuator response.

The real actuators are evaluated for rotorcraft in hover trim. The real actuators' outputs are allowed to affect the aircraft dynamic using the controllers tuned in Chapter 3, the results are shown in Figure 6.13. The position response of the aircraft is shown in Figure 6.14.

From Figures 6.13 and 6.14 it is apparent that the actuators fail to meet the demand of the controller in maintaining the rotorcraft in hover equilibrium. This shows that the PID controller developed are affected by the parameter changes in the actuator system. The

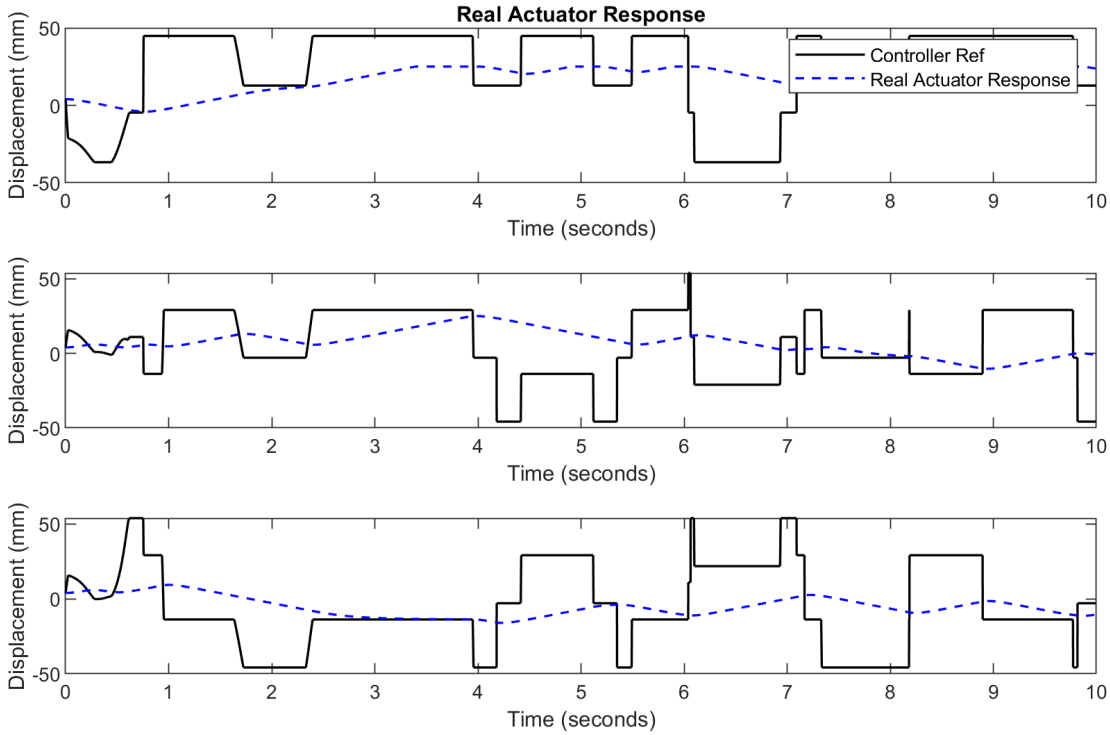


Figure 6.13: The response of the real actuators when affecting flight dynamics.

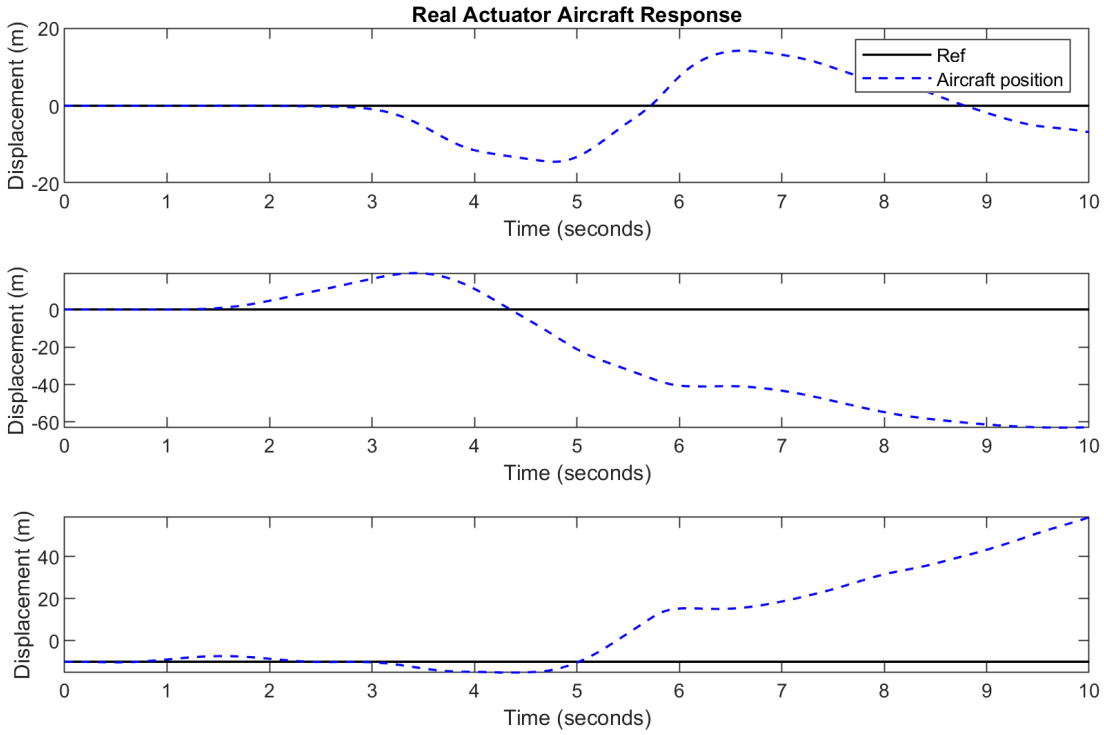


Figure 6.14: Rotorcraft response with real actuators under PID control.

real actuators have a finite response time, while the response time of the ideal actuator is infinite. This means the PID controllers have to be retuned to align with the new system dynamics.

6.6.1 Controller gains retuning

The PID controller gains are retuned with the resulting values given in Table 6.3:

Table 6.3: The new PID controller gains found after retuning using ACO algorithm.

PID gains	Roll	Pitch	Yaw	Altitude	Long	Lat
K_p	7.000	150.000	90.606	-75.000	3.0000	-3.000
K_i	0.000	0.000	0.000	0.000	0.000	0.000
K_d	-1.200	1.000	-10.000	27.000	-1.0000	-2.000

The response of the rotorcraft after tuning using the ACO is shown in Figures 6.15. This shows the effectiveness of the computational intelligence methods in recalibrating the control system for changes in the parameter. The optimisation tools do not have to know the exact nature of the parameter changes in order to make the correct prediction of the gains required to control the rotorcraft. The requirements is that the model of the system is available. This was shown in the previous section.

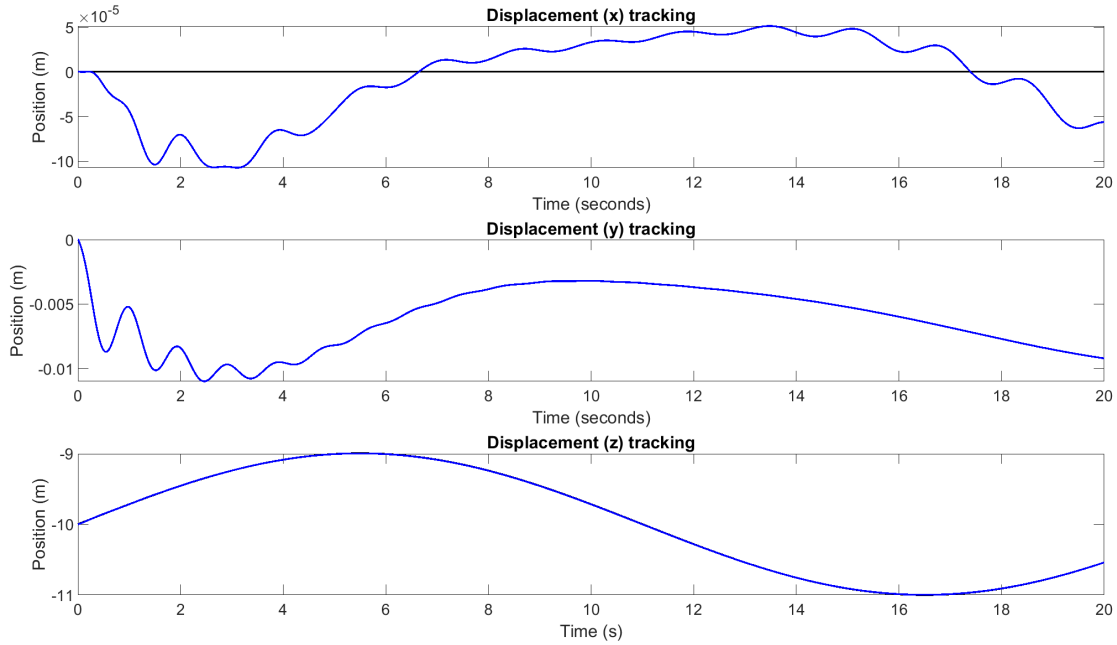


Figure 6.15: Rotorcraft position response with real actuator models under the retuned PID controller.

6.6.2 PID HILS response

The new controller is applied to hardware-in-the-loop actuators. The responses of the aircraft in HILS are shown in Figures 6.16 and 6.17. The control inputs to the rotorcraft are shown in Figure 6.18.

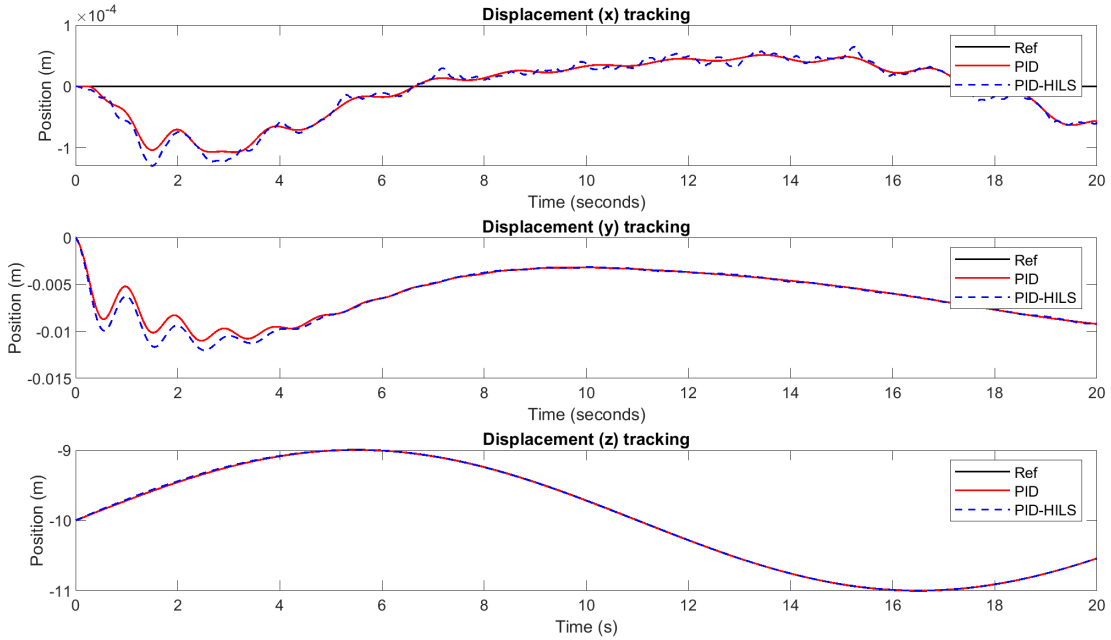


Figure 6.16: Rotorcraft position response with HILS actuators under the retuned PID controller.

Comparing the modelled actuator response to the HILS response, it is evident that the controller developed with the actuator models is able to control the HILS actuator and the response of the rotorcraft is within the specifications. This means that at a selected equilibrium point the optimised PID is able to control the real-actuated rotorcraft. The RMS for the model is 5.3069×10^{-3} and for the real actuator 5.8185×10^{-3}.

When the system is disturbed from equilibrium, such as when transitioning from hover to forward flight, the PID controller fails to control the rotorcraft as shown in Figure 6.19. This is consistent with the results presented in Chapter 3 with ideal actuators.

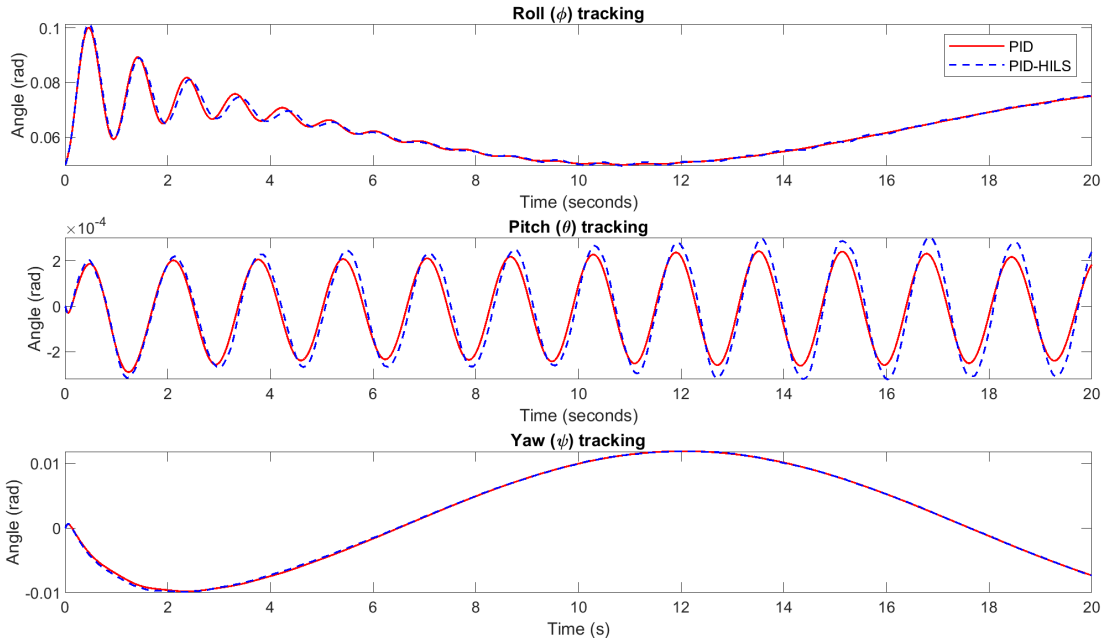


Figure 6.17: Rotorcraft Euler angles response with HILS actuators under the retuned PID controller.

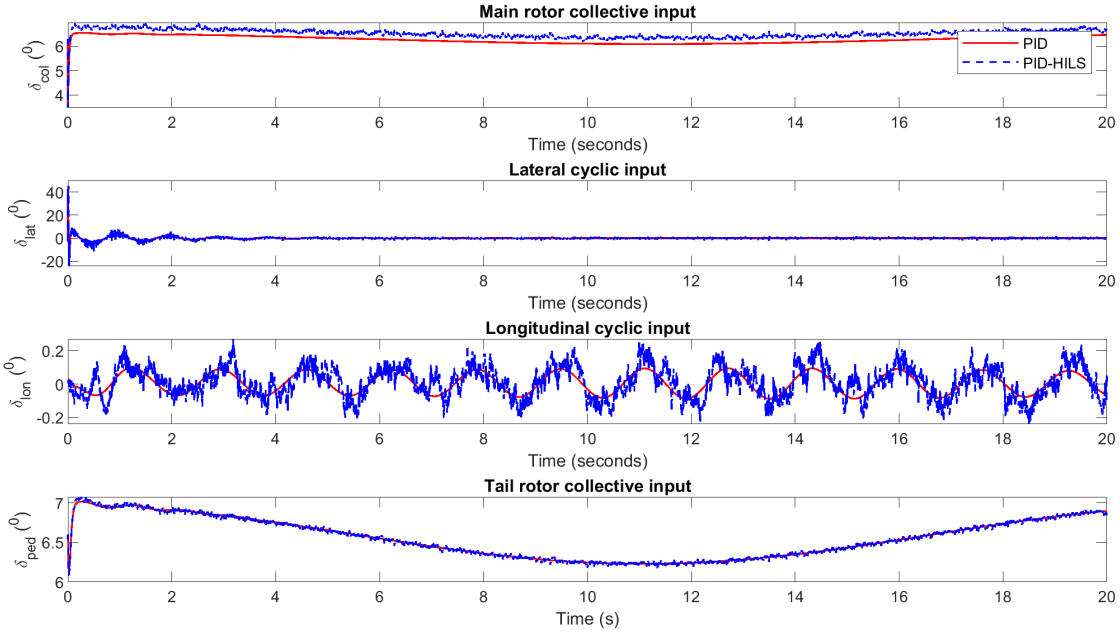


Figure 6.18: Rotorcraft control inputs with HILS actuators under the retuned PID controller.

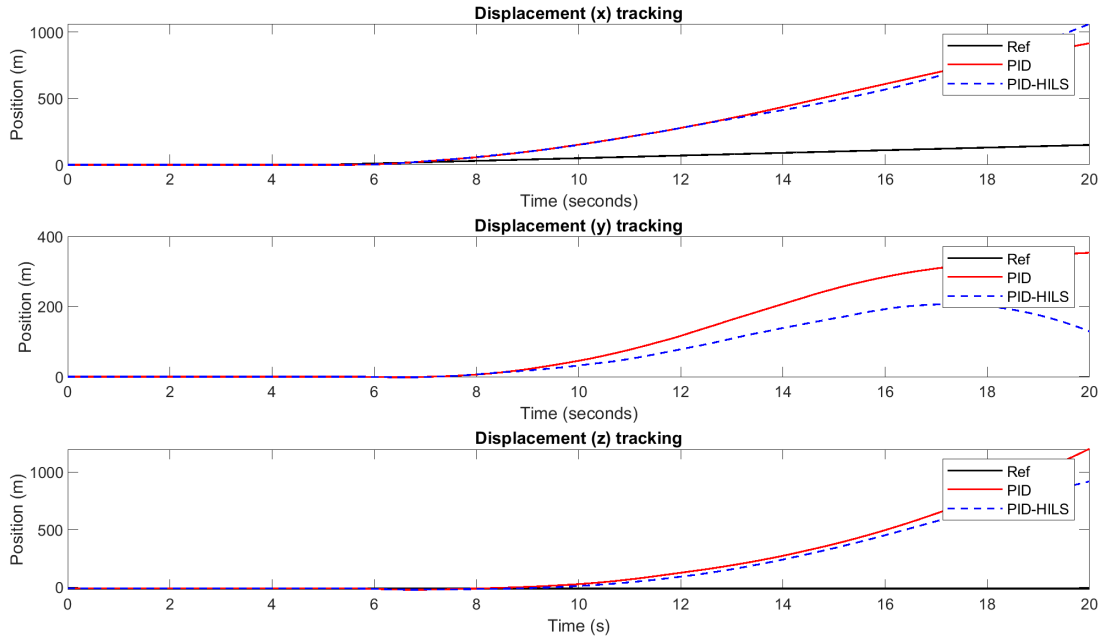


Figure 6.19: Rotorcraft position response with HILS actuators under PID controller transitioning from hover to forward flight.

.

6.7 Robust SMC Response to Real Actuators

As it was seen with the PID controller, the SM controllers developed in Chapter 4 are tested with the real actuator. The STSMC is applied to the HILS actuator in order to verify that it can control the real system. The results are shown in Figures 6.20 and 6.21. It is noted that the STSMC is able to control the rotorcraft without the need for retuning. This is an exemplary display of the STSMC's robustness.

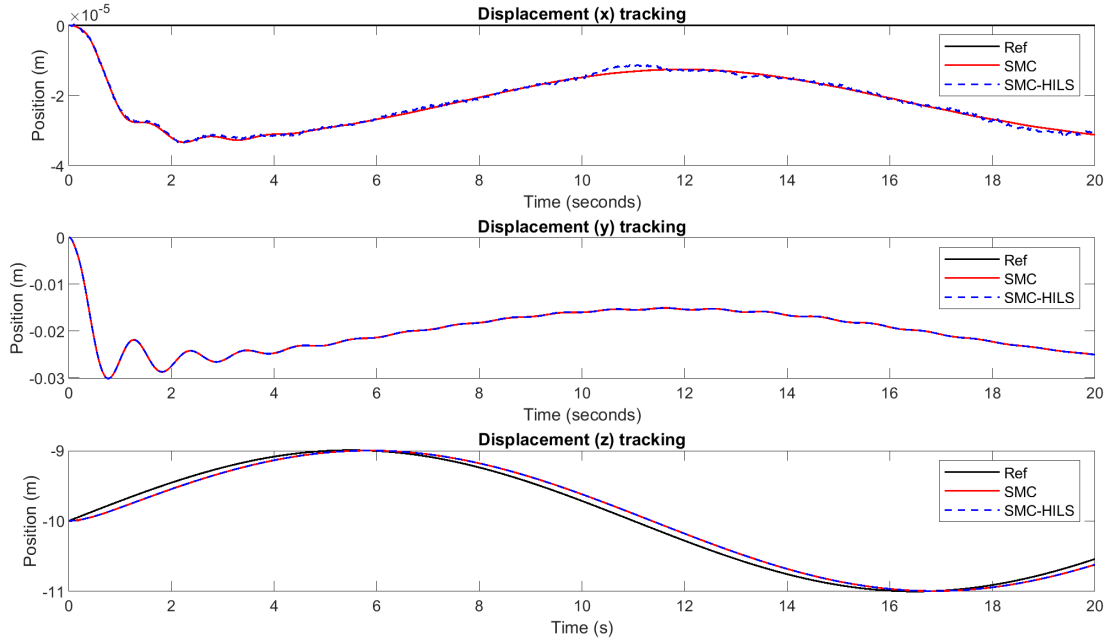


Figure 6.20: Rotorcraft position response with real actuator models under the STSMC.

The STSMC is able to control the HILS model and closely related to the model actuator response. Since the SMC is prone to chattering, the control inputs are shown in Figure 6.22.

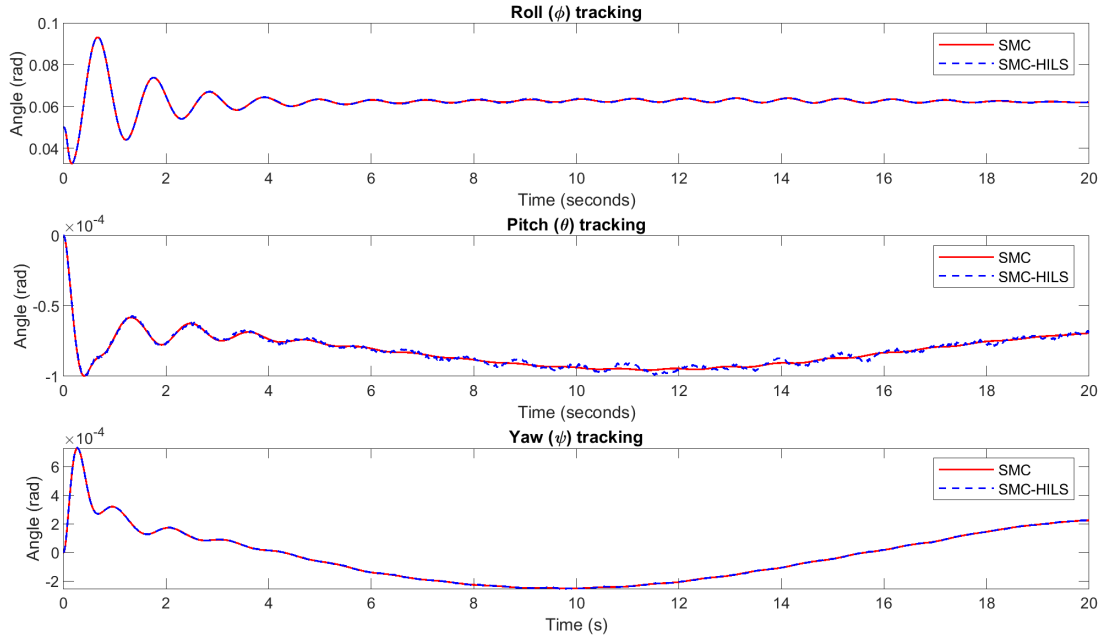


Figure 6.21: Rotorcraft Euler angles response with real actuator models under the STSMC.

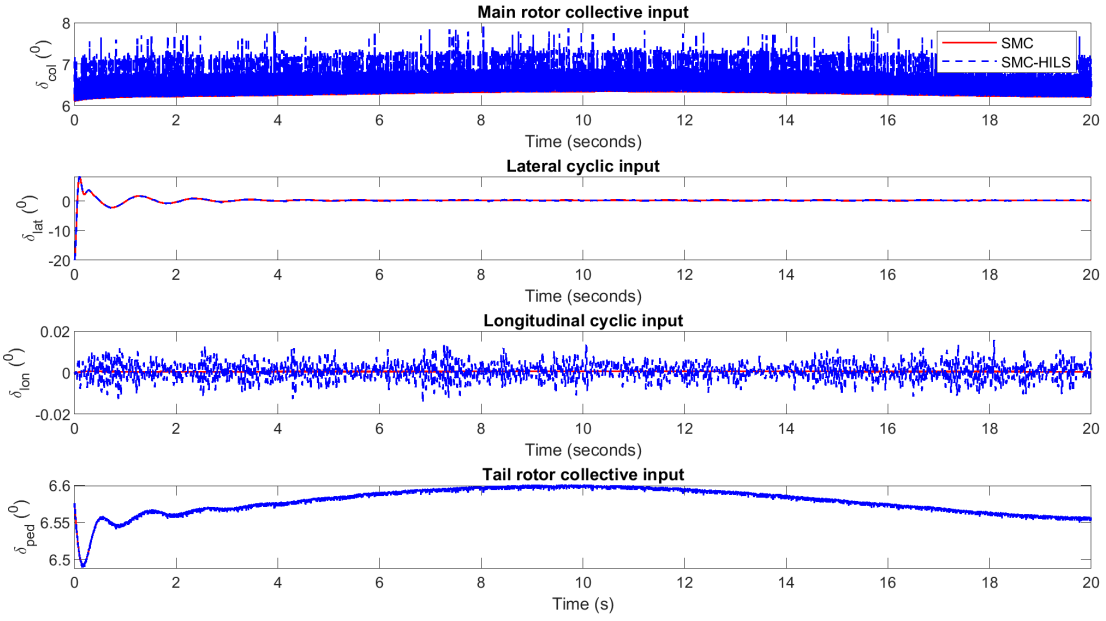


Figure 6.22: Rotorcraft control inputs with HILS actuators under the STSMC.

6.8 Active DNN Response to Real Actuators

The next part of the experiment is to use the DNNFBL controller that was presented in Chapter 5. The MOACO-based DNNFBL is used in this study.

The DNN is then applied to the HILS actuator in order to verify that it can control the real system. The results are shown in Figure 6.23 and 6.24. The DNN is able to control the HILS model and the performance is similar to the model actuator response. Forward flight position response is shown in Figure 6.26. The control inputs produced by the DNN are shown in Figures 6.25 and 6.27.

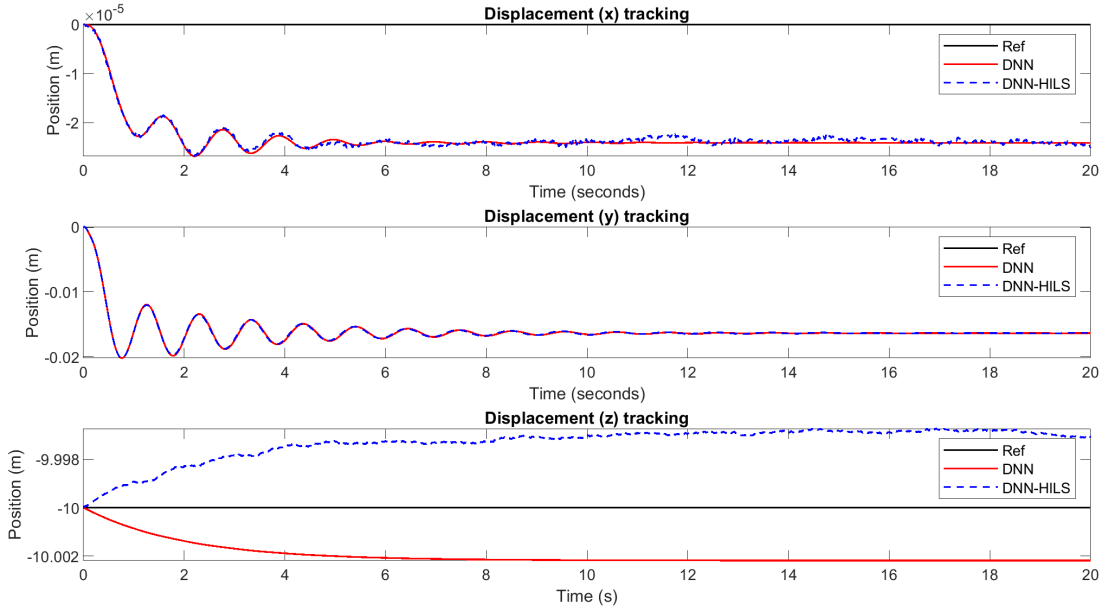


Figure 6.23: Rotorcraft position response with real actuator models under the DNNFBL.

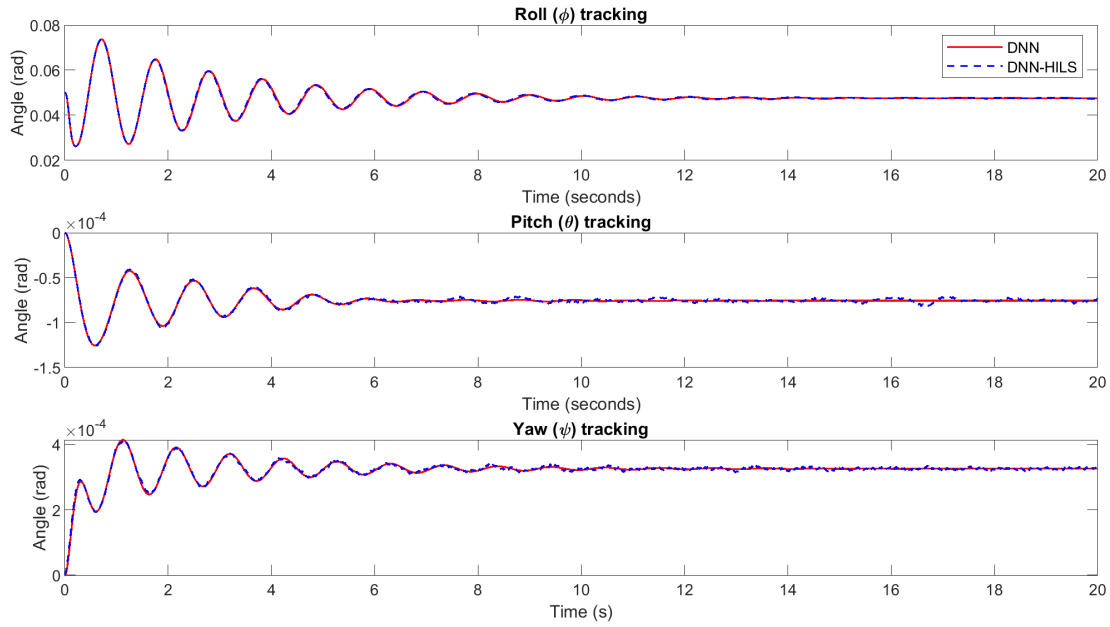


Figure 6.24: Rotorcraft Euler angles response with real actuator models under the DNNFBL.

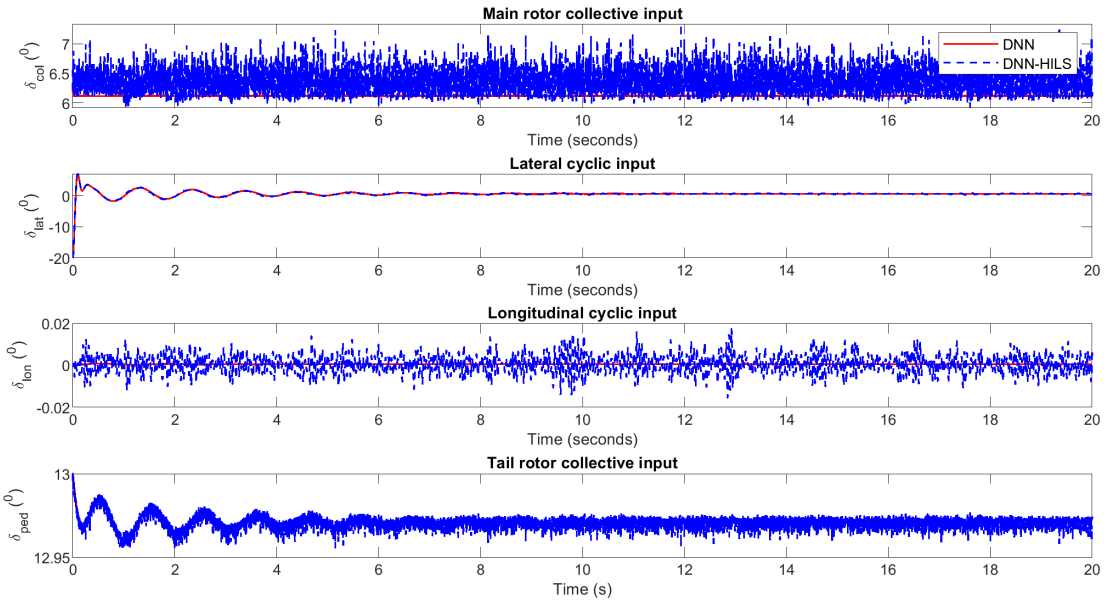


Figure 6.25: Rotorcraft control inputs with HILS actuators under the DNNFBL.

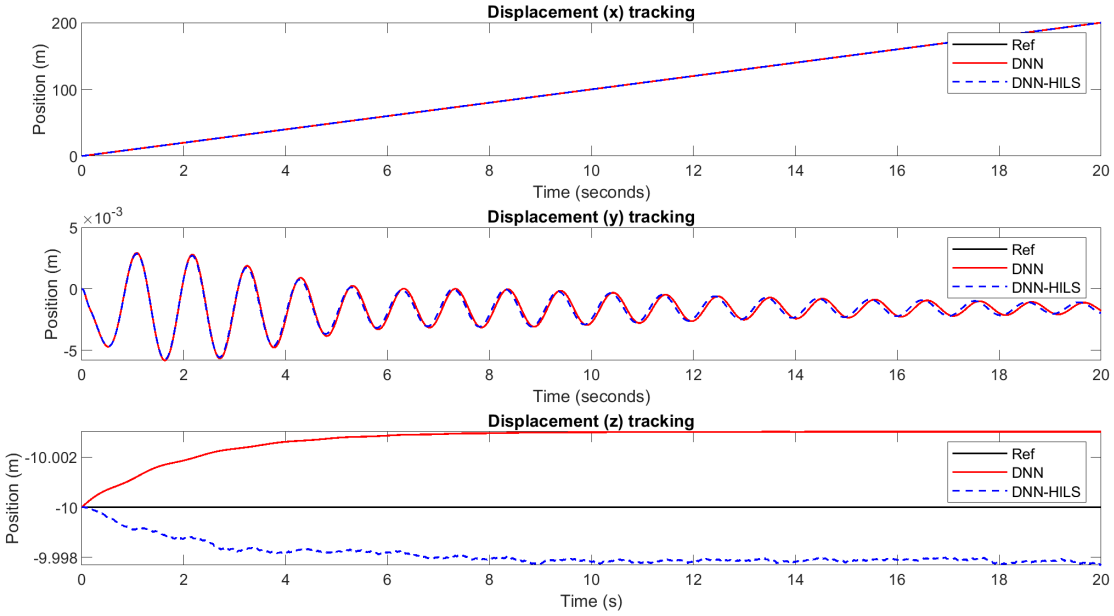


Figure 6.26: Rotorcraft position response with real actuator models under the DNNFBL in forward flight.

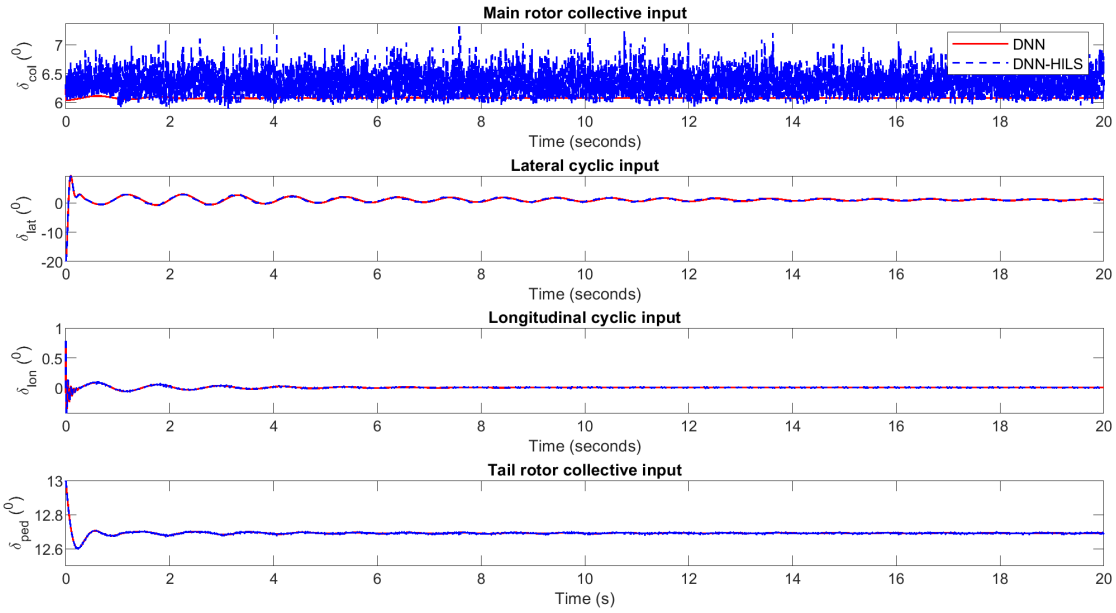


Figure 6.27: Rotorcraft control inputs with HILS actuators under the DNNFBL in forward flight.

6.9 Comparing the Responses of the Proposed Controllers

The error from the rotorcraft controlled by PID, STSMC and DNNFBL was recorded. The summary of the performance of the three controllers based on the performance specification for hover and $10m/s$ forward flight are shown in Table 6.4.

Table 6.4: Comparing the HIL simulation results for the three controllers.

Performance conditions	PID	STSMC	DNNFBL
Hover	0.33	0.34	0.31
Forward	4.14	0.54	0.28

From the table, it is seen that the PID has a lower error in hover than the STSMC. This is due to the fact that PID controller is tuned specifically for the selected flight conditions. For the STSMC and the DNNFBL, the same controller is used for hover and forward flight. To cover a larger range results in the limitation and sacrificing performance when a specific condition is investigated. However, when the full envelope is considered, STSMC and DNNFBL perform better.

6.10 HILS Fault Tolerance

The DNNFBL controller is applied to the fault tolerance of the LOE of the actuator. The bank of DNN models are used for residual generation. The residual and the fault decisions are sent to the reconfiguration controller. When a fault is identified, the swashplate switching mechanism is reconfigured and the engine speed is used as one of the controller commands.

The fault injection is coded into the Arduino micro-controller system. The PWM that is sent to the H-bridge is multiplied by a factor less than 1 to simulate a loss-of-effectiveness. A fault LOE of 50% is injected into actuator A at $7s$ and the residual is monitored in Figure 6.28. Since the residual exceeds the threshold, the fault is identified and the control is reconfigured. Figure 6.28 shows the signal sent to the engine and the resulting rotor speed.

Figures 6.29 and 6.30 illustrate the position and Euler angles' responses of the rotorcraft when subjected to a fault. The corresponding control inputs can be seen in Figure 6.31. The DNNFBL HILS response has a higher steady-state error. However, this is still within

performance specifications. Therefore, the system with DNNFBL recovers successfully in HILS.

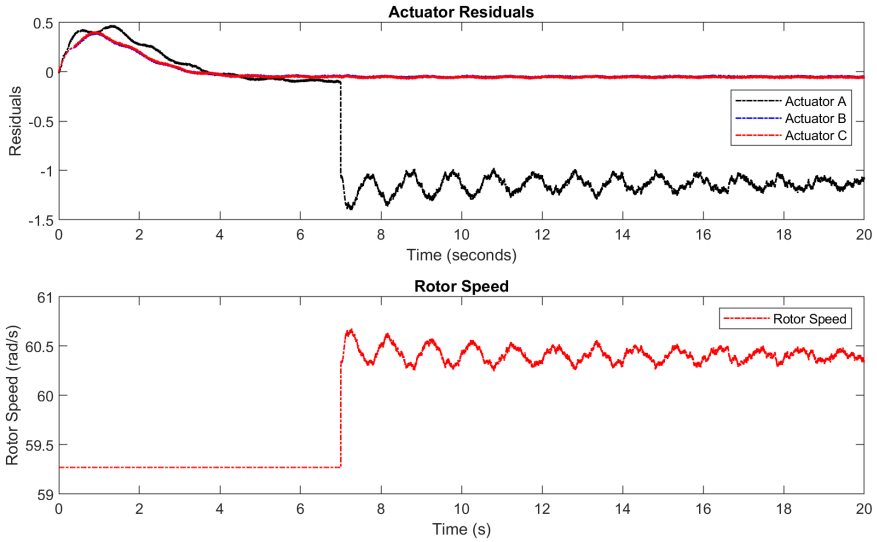


Figure 6.28: Residual and rotor speed with HILS actuators under the DNNFBL for a LOE of 50%.

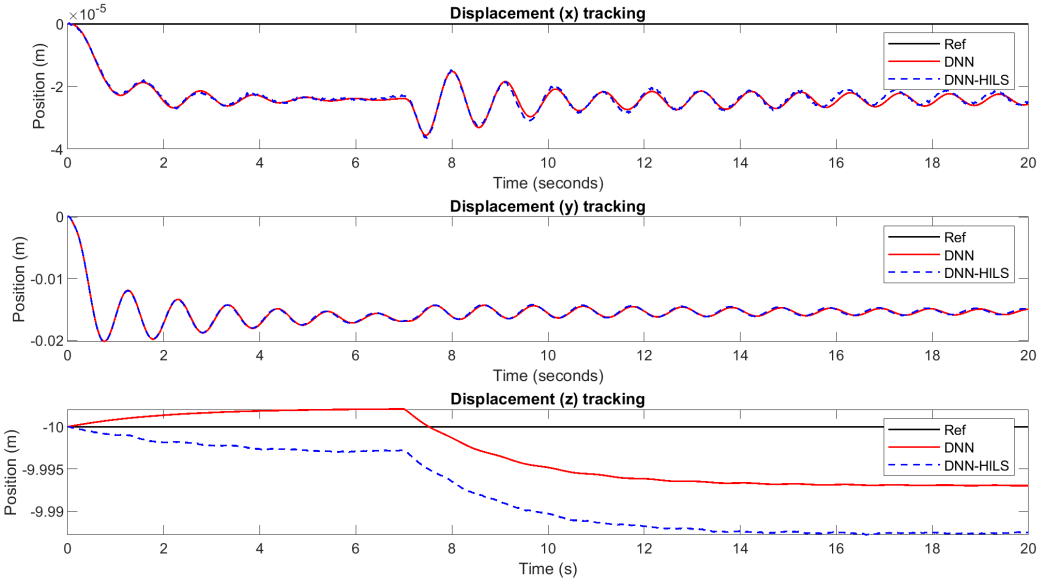


Figure 6.29: Rotorcraft position response with real actuator models under the DNNFBL for a LOE of 50%.

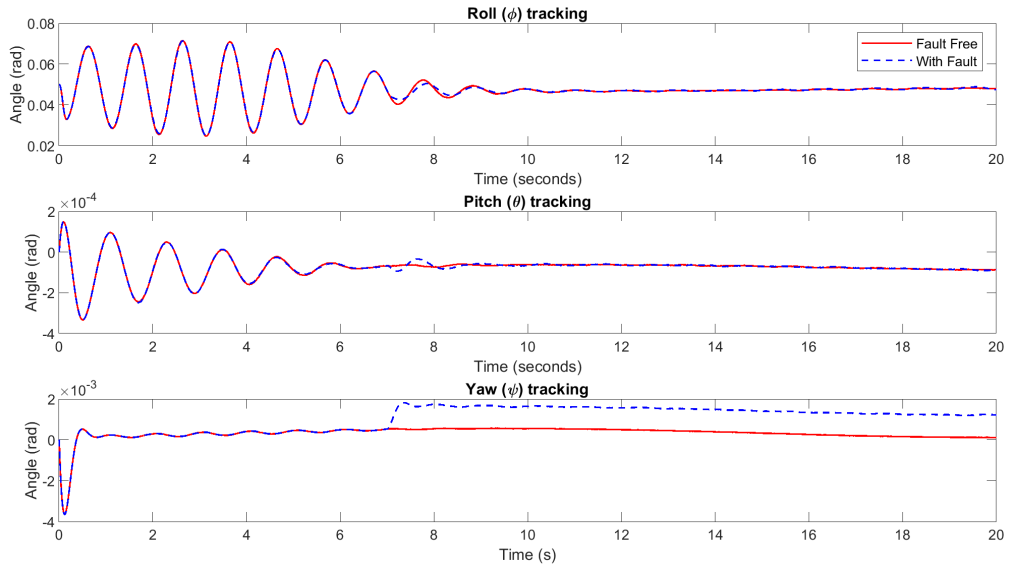


Figure 6.30: Rotorcraft Euler angles response with HILS actuators under the DNNFBL for tracking with LOE of 50%.

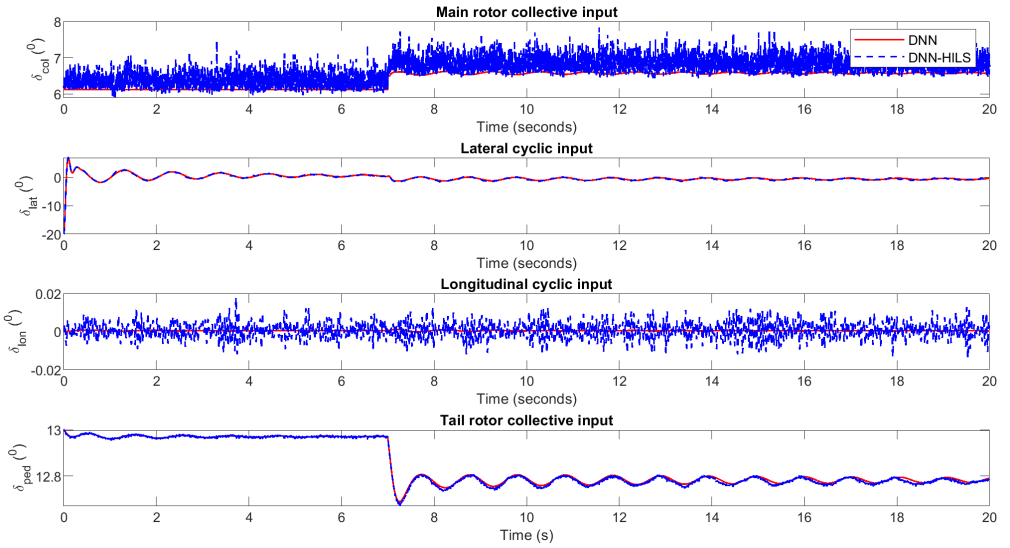


Figure 6.31: Rotorcraft control inputs with HILS actuators under the DNNFBL for tracking with LOE of 50%.

The DNNFBL system is also able to control the HILS setup signal when it is tracking a height sinusoidal reference and the performance is similar to the model actuator response. The performance of a fault-free case is compared with the performance of an actuator LOE of 50%.

The sinusoidal position tracking responses of a fault-free rotorcraft compared to a faulty rotorcraft are illustrated in Figure 6.32, while the corresponding control inputs that control the rotorcraft are presented in Figure 6.33.

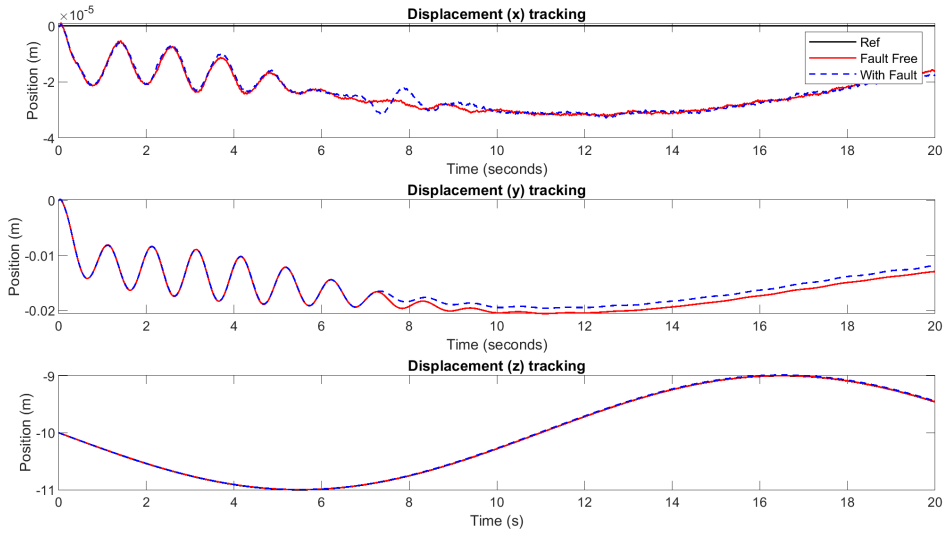


Figure 6.32: Rotorcraft position response with real actuator models under the DNNFBL for tracking with LOE of 50%.

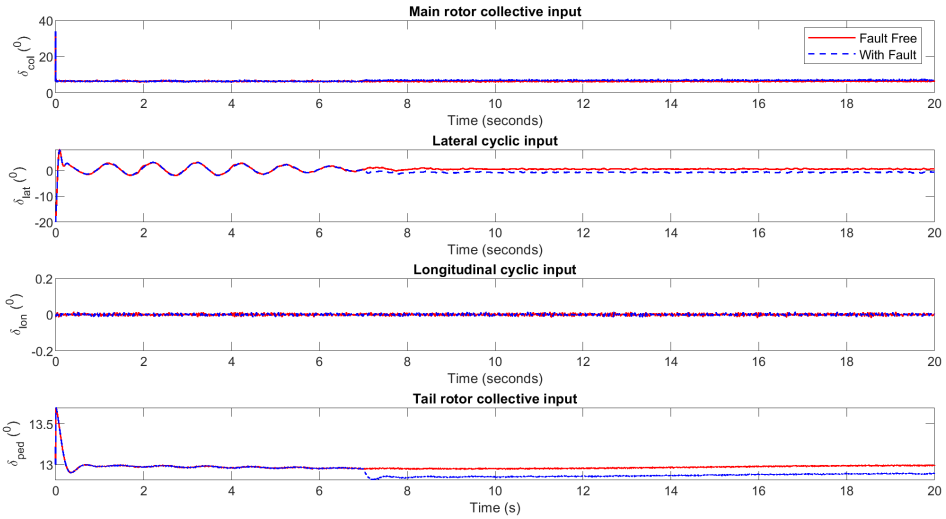


Figure 6.33: Rotorcraft control inputs with HILS actuators under the DNNFBL for tracking with LOE of 50%.

These experiments confirmed that a rotorcraft UAV that employs a rotor speed variation technique can demonstrate control actuator fault tolerance by the application of an integrated flight-propulsion control and computational intelligence techniques.

6.11 Chapter Summary

This chapter presented control of the rotorcraft with hardware-in-the-loop simulations of the swashplate and the electromechanical actuators. The hardware of the swashplate mechanism was developed and tested with the control of the collective, lateral and longitudinal cyclic. The control algorithms presented in previous respective chapters were implemented in the HILS setup. Actuators models were replaced with real hardware actuators to determine how the proposed control techniques perform to real fault for experimental validation.

The PID controller had to be retuned in order to be compatible with the HILS. The STSMC and the DNNFBL did not need retuning. The performance of the PID at the selected trim was better than the SMC and DNNFBL. The SMC handled actuator faults well. The DNNFBL with the aid of the DNN FDD was able to handle more severe actuator faults. This was done with the incorporation of engine control.

This chapter demonstrated that the inclusion of propulsion control to the flight control can assist a faulty swashplate in continuing to control the rotorcraft. The limitation of the proposed method is that it caters for a single actuator fault in the main rotor swashplate. Also, the results of this chapter only compare the two fault-tolerant techniques, further research is necessary to establish the software acceptance levels required for RTCA DO-178 in order for these techniques to be used on the rotorcraft.

Chapter 7

Conclusion and Recommendations for Future Work

This concluding chapter of this thesis summarises the approach that was followed and the results that have been presented so far. Based on these results, further work on this topic is recommended.

7.1 Conclusions

This thesis has presented the results for the design and development of a medium-scale, single-rotor rotorcraft UAVs nonlinear dynamic model to be used for model-based design of FTC variable rotor speed applications in a flight-propulsion combination. The need arose from the requirement for a cost effective, reliable and environmentally friendly rotorcraft. The developed model includes the swashplate actuator dynamics and the engine propulsion model.

Most rotorcraft controller design treat actuator and the engine propulsion system components as linear first-order systems. This thesis introduced the application of electromechanical actuators in the swashplate. This has not previously been applied in rotorcraft UAV models. The inclusion of nonlinear actuator dynamics allows us to simulate the actuator faults with increased fidelity, which enables the development of robust integrated flight-propulsion controllers that are a true reflection of the real system to be controlled. Thereby reducing the time and effort required to transition from simulated results to real systems.

The loss of a rotorcraft system is undesirable, due to loss of property and life that can result. Reliability is even more important in unmanned platforms where there is no pilot to handle system errors and environmental uncertainties. IFPC was proposed in this thesis for the RUAV in order to handle the deliberate variations in rotor speed and for its exploitation as a redundant thrust control input in the case of a single swashplate actuator fault.

The presented integrated RUAV system model was first controlled by using proportional-integral-derivative (PID) controllers in 6 degrees-of-freedom (DOF). This was performed to benchmark the controllers and have a starting point for the comparison of the fault-tolerant controllers that followed. However, the PID controllers present here were enhanced as the first contribution of this thesis. It became apparent that most PID currently in application are not properly tuned. It proved to be difficult tuning a multi-input multi-output system with four inputs and six outputs. This thesis endeavoured to investigate if this problem can be remedied. To find the optimal controller gains for the rotorcraft at hover and forward speeds, computational intelligent optimisation techniques are used, these are:

1. ant-based ant colony and antlion optimisation;
2. flight-based cuckoo search and firefly optimisation algorithms;
3. particle swarm optimisation; and
4. genetic algorithm.

These techniques were compared. Continuous ant colony and cuckoo search optimisation had not been used in a similar study. Both methods proved to be suitable for the task. It was noticeable, also, that the performance of the PID, even the optimal one, deteriorated away from the trim conditions with limited robustness.

Since rotorcraft UAV can operate well beyond the control of a remote pilot, it becomes important that the system is able to tolerate a some level of system malfunction before heading back to the base. For fault tolerance, we presented the passive fault-tolerant control (PFTC) based on sliding mode controller (SMC). Conventional SMC and super-twisting SMC were compared in terms of robustness to handle loss-of-effectiveness (LOE) fault in one of the three swashplate actuators and the deviation of the rotor speed from

nominal. The parameters of these three controllers were tuned using computational intelligence algorithms as was used in the PID controllers case. In order to apply this robust control, the system had to be transformed from a complex non-affine-in-control system in to an affine-in-control representation. The controllers proved to be invariant to the rotor speed changes and maintain the rotorcraft control. Chattering was removed by using the hyperbolic tangent function. The SMC-based controllers proved robust to actuator LOE faults. However, they fail to recover the rotorcraft from total actuator failure. This was shown using hypothesis testing that the performance results were significantly different. The smoothing of the control action was derived mathematical and showed how the *tanh* function is able to filter out chattering when compared to the *sgn* function.

Active fault-tolerant control (AFTC) scheme based on dynamic neural networks (DNN) is employed. RUAV system identification using DNN is combined with feedback linearisation (FBL). Multi-objective optimisation algorithms are used to find the controller and FBL gains. The application of MOO in tuning rotorcraft control design was presented for the first time in this thesis. The MOO focused on Antlion and Ant colony optimisation, both of which proved suitable for finding near optimal parameters for he DNNFBL. The ACO was found to be better than the ALO.

DNN is also used for fault detection and diagnosis (FDD) for the AFTC. This applied indirect DNNFBL control strategy proved to be more suitable for IFPC and was able to recover from severe actuator faults and rejected rotor speed variations. The effectiveness of the proposed control strategies is evaluated in hardware-in-the-loop simulations using an experimental swashplate rig of three electromechanical actuators. The experimental results validated the developed and simulated control strategies.

The aim of this research was to improve the knowledge base by introducing flight-propulsion-based design through thorough investigation of variable rotor speed. The control strategies will be used to study the application to RUAV design and trade-offs between performance, reliability and safety. This project proposed the use of computational intelligence FTC and this was compared to conventional controllers.

7.2 Recommendations for Future Work

1. The study of the rotorcraft and the actuator dynamics presented in the thesis were based on a combination of simulated results and limited experimental results on the hardware-in-the-loop bases. This was done for the development phase of the RUAV under investigation. It is recommended that as the RUAV evolves and more of the experimental system becomes available, the experiment be expanded to include as many real-life dynamics as possible. This includes, but should not be limited, the real propulsion system, the rotor hub and the effect of the coupling gearbox both in the main and the tail rotors. The controller developed here can then be evaluated for a wider applicability. This way the limiting assumptions listed in Chapter 2, that guided the development of our proposed systems can be validated.

2. In Chapter 3 a number of optimisation algorithms were presented and benchmarked. These were used to find PID controller gains. This process involved only the optimisation using a single objective. Future test can be conducted to investigate if multi-objectives optimisation offers better tuning results when compared to the single objective one.

3. The robustness of the conventional SMC and STSMC were based on the assumption that uncertainty and disturbance are below a certain threshold. However, this threshold is not known *a priori*. Adaptive control has been shown to yield a good approximation of these unknown functions. A better understanding can be gained by investigating the incorporation of this adaptation with the SMC results presented in this thesis.

4. For future investigation, it is recommended that an intelligent fault detection and identification agent is used in conjunction with the DNNFBL for AFTC of the rotorcraft. This setup can be compared to the robust methods that currently prevail in literature, such as high-order sliding mode controller.

5. In Chapter 2 it was shown the the actuator can be positioned in an optimal way, by making $\gamma \neq 0$. The study of this optimisation with the inclusion of AFTC should be conducted to gain insight in whether there is an optimal position for the actuator that maximises the effectiveness of the two actuators should one fail.

Bibliography

E. Abdulhamitbilal and E. M. Jafarov. Gain scheduled automatic flight control systems design for a light commercial helicopter model. *WSEAS Transactions on Systems and Control*, 6(12):440–455, 2011. ISSN 1598-6446.

O. Adeyemi, I. Grove, S. Peets, Y. Domun, and T. Norton. Dynamic neural network modelling of soil moisture content for predictive irrigation scheduling. *Sensors*, 18(10): 3408, 2018.

N. Administrator. NASA Awards Historic Green Aviation Prize, June 2013. URL `http://www.nasa.gov/home/hqnews/2011/oct/HQ_11-334_GFC_Winners.html`.

ADS-33E. Aeronautical design standard, performance specification, handling qualities requirements for military rotorcraft. *US Army Aviation and Missile Command*, 2000.

B. Ahmed and H. R. Pota. Flight control of a rotary wing UAV using adaptive backstepping. In *Control and Automation, ICCA 2009. IEEE International Conference on*, pages 1780–1785. IEEE, 2009.

A. A. L. Al-Jodah, B. Shirinzadeh, J. Pinskier, M. Ghafarian, T. K. Das, Y. Tian, and D. Zhang. Antlion optimized robust control approach for micropositioning trajectory tracking tasks. *IEEE Access*, 8:220889–220907, 2020.

A. Alkamachi and E. Erçelebi. Modelling and genetic algorithm based-PID control of H-shaped racing quadcopter. *Arabian Journal for Science and Engineering*, 42(7): 2777–2786, 2017.

J. Alvarenga, N. I. Vitzitaios, K. P. Valavanis, and M. J. Rutherford. *Survey of Rotorcraft Navigation and Control*. University of Denver - Unpublished, 2014.

H. Alwi, C. Edwards, and C. P. Tan. *Fault Detection and Fault-Tolerant Control Using Sliding Modes*. Springer, 2011.

A. A. Aly. PID parameters optimization using genetic algorithm technique for electrohydraulic servo control system. *Intelligent Control and Automation*, 2(02):69–76, 2011.

S. Amini and A. A. Akbari. Attitude control of unmanned aerial vehicle based on sliding mode technique with parameter estimation. *ADMT Journal*, 10(2):49–60, 2017.

M. Amozegar and K. Khorasani. An ensemble of dynamic neural network identifiers for fault detection and isolation of gas turbine engines. *Neural Networks*, 76:106–121, 2016.

H. Amri, R. Feil, M. Hajek, and M. Weigand. Possibilities and difficulties for rotorcraft using variable transmission drive trains. *CEAS Aeronautical Journal*, 7(2):333–344, 2016.

J. L. Anderson Azzano, J. J. Moré, and P. F. Puleston. Design and stability analysis of a super-twisting controller for a PS-FBC–based fuel cell module. *Advanced Control for Applications: Engineering and Industrial Systems*, 1(1):e19, 2019.

K. J. Åström and B. Wittenmark. *Adaptive Control*. Addison-Wesley Longman Publishing Co., Inc., New York, 1989.

R. Austin. *Unmanned aircraft systems: UAVs design, development and deployment*, volume 54. John Wiley & Sons, Chichester, 2011.

R. K. Barnhart, D. M. Marshall, and E. Shappee. *Introduction to unmanned aircraft systems*. CRC Press, Boca Raton, 2021.

F. Bateman, H. Noura, and M. Ouladsine. Fault diagnosis and fault-tolerant control strategy for the aerosonde UAV. *IEEE Transactions on Aerospace and Electronic Systems*, 47(3):2119–2137, 2011.

C. P. Bechlioulis and G. A. Rovithakis. Prescribed performance adaptive control for multi-input multi-output affine in the control nonlinear systems. *IEEE Transactions on Automatic Control*, 55(5):1220–1226, 2010.

O. Bendjeghaba, S. I. Boushaki, and N. Zemmour. Firefly algorithm for optimal tuning of PID controller parameters. In *4th International Conference on Power Engineering, Energy and Electrical Drives*, pages 1293–1296. IEEE, 2013.

L. Bilyaletdinova and A. Steblinkin. Simulation of direct drive electromechanical actuator with ballscrew. *Procedia Engineering*, 176:85–95, 2017.

R. Bindu and M. K. Namboothiripad. Tuning of PID controller for DC servo motor using genetic algorithm. *International Journal of Emerging Technology and Advanced Engineering*, 2(3):310–314, 2012.

C. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, New York, 2003.

M. Blanke, M. Kinnaert, J. Lunze, M. Staroswiecki, and J. Schröder. *Diagnosis and fault-tolerant control*, volume 2. Springer, 2006.

M. Blanke, M. Kinnaert, J. Lunze, M. Staroswiecki, and J. Schröder. *Diagnosis and fault-tolerant control*, volume 3. Springer, Heidelberg, 2016.

C. Böhm, C. Brommer, A. Hardt-Stremayr, and S. Weiss. Combined system identification and state estimation for a quadrotor UAV. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 585–591. IEEE, 2021.

A. Boubakir, F. Boudjema, and S. Labiod. A neuro-fuzzy-sliding mode controller using nonlinear sliding surface to the coupled tanks system. *International Journal of Automation and Computing*, 06(1):72–80, 2009.

H. Boubertakh. Knowledge-based ant colony optimization method to design fuzzy proportional integral derivative controllers. *Journal of Computer and Systems Sciences International*, 56(4):681–700, 2017.

G. M. Bowen-Davies and H. Yeo. Update on UH-60A rotor performance and loads correlation at high advance ratios using RCAS. In *58th AIAA/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*, page 0861, 2017.

A. R. S. Bramwell, D. Balmford, and G. Done. *Bramwell's helicopter dynamics*. Butterworth-Heinemann, Oxford, 2001.

A. Budiyono, T. Sudiyanto, and H. Lesmana. Global linear modeling of small scale helicopter. In *Intelligent unmanned systems: Theory and applications*, pages 27–62. Springer-Verlag, Berlin, Heidelberg, 2009.

A. Budiyono, I. E. Putro, K. Yoon, G. B. Raharja, and G. Kim. Real-time hardware simulation of a small-scale helicopter dynamics. *Aircraft Engineering and Aerospace Technology*, 82(6):360–371, 2010.

G. Cai, B. M. Chen, and T. H. Lee. *Unmanned Rotorcraft Systems*. Springer-Verlag, London, 2011.

B. S. Caldwell and M. E. Splitt. GA pilot information and weather technology and the cockpit: Fixed wing and rotorcraft issues. In *AIAA AVIATION 2021 FORUM*, page 2955, 2021.

M. Y. Chachou, Z. Liu, Z. Zhou, A. Benchalal, and C. Zerfaoui. Modeling and control of a simulated flight of a mini helicopter using MATLAB/SIMULINK. In *International Conference on Computer, Communications and Information Technology*, pages 216–223. Atlantis Press, 2014.

L.-W. Chang. A MIMO sliding control with a first-order plus integral sliding condition. *Automatica*, 27(5):853–858, 1991.

R. Chen, Y. Yuan, and D. Thomson. A review of mathematical modelling techniques for advanced rotorcraft configurations. *Progress in Aerospace Sciences*, 120:100681, 2021.

Z. Chen and J. Huang. *Stabilization and Regulation of Nonlinear Systems: A Robust and Adaptive Approach*. Springer, Cham, Heidelberg, New York, Dordrecht, London, 2015.

C. Chi, X. Yan, R. Chen, and P. Li. Analysis of low-speed height-velocity diagram of a variable-speed-rotor helicopter in one-engine-failure. *Aerospace Science and Technology*, 91:310–320, 2019.

M. Coffin and M. J. Saltzman. Statistical analysis of computational tests of algorithms and heuristics. *INFORMS Journal on Computing*, 12(1):24–44, 2000.

A. Crespo Márquez, P. Moreu de León, A. Sola Rosique, and J. F. Gómez Fernández. Criticality analysis for maintenance purposes: A study for complex in-service engineering assets. *Quality and Reliability Engineering International*, 32(2):519–533, 2016.

J. Dai, G. Wu, Y. Wu, and G. Zhu. Helicopter trim research based on hybrid genetic algorithm. In *2008 7th World Congress on Intelligent Control and Automation*, pages 2007–2011. IEEE, 2008.

A. Demirel. Investigating trim algorithms for a utility helicopter. Master's thesis, Middle East Technical University, 2019.

Denel-Dynamics. Denel Dynamics Seeker 400 UAS. (Last accessed 04 July 2022), 2018. URL `http://www.deneldynamics.co.za/products/uas/reconnaissance-systems/seeker-400-uas`.

S. X. Ding. *Model-based fault diagnosis techniques: design schemes, algorithms, and tools*. Springer Science & Business Media, Berlin Heidelberg, 2008.

M. Dorigo. *Optimization, Learning and Natural Algorithms*. PhD thesis, Politecnico di Milano, Milano, 1992.

G. R. Drozeski. *A fault-tolerant control architecture for unmanned aerial vehicles*. PhD thesis, Georgia Institute of Technology, 2005.

G. R. Drozeski, B. Saha, and G. J. Vachtsevanos. A fault detection and reconfigurable control architecture for unmanned aerial vehicles. In *Aerospace Conference, 2005 IEEE*, pages 1–9. IEEE, 2005.

G. J. J. Ducard. *Fault-Tolerant Flight Control and Guidance System for a Small Unmanned Aerial Vehicle*. PhD thesis, ETH Zurich, 2007.

G. J. J. Ducard. *Fault-tolerant Flight Control and Guidance Systems: Practical methods for small unmanned aerial vehicles*. Springer, Dordrecht, Heidelberg, London, New York, 2009.

Z. T. Dydek. *Adaptive control of Unmanned Aerial Systems*. PhD thesis, Massachusetts Institute of Technology, 2010.

R. Eberhart and J. Kennedy. A new optimizer using particle swarm theory. In *MHS'95. Proceedings of the Sixth International Symposium on Micro Machine and Human Science*, pages 39–43. IEEE, 1995.

C. Edwards and S. Spurgeon. *Sliding mode control: theory and applications*. CRC Press, London, 1998.

C. Edwards, T. Lombaerts, and H. Smaili. *Fault Tolerant Flight Control: A Benchmark Challenge*. Springer-Verlag, Berlin, 2010.

N. El Gmili, M. Mjahed, A. El Kari, and H. Ayad. Particle swarm optimization and cuckoo search-based approaches for quadrotor control and trajectory tracking. *Applied Sciences*, 9(8):1719, 2019.

A. P. Engelbrecht. *Computational Intelligence: An Introduction*. John Wiley & Son, Ltd, West Sussex, 2007.

D. Enns and T. Keviczky. Dynamic inversion based flight control for autonomous RMAX helicopter. In *American Control Conference*, pages 3916–3923. IEEE, 2006.

R. Enns and J. Si. Helicopter flight-control reconfiguration for main rotor actuator failures. *Journal of Guidance Control and Dynamics*, 26(4):572–584, 2003.

J. Eyerman. Unmanned aircraft and the human element: Public perceptions and first responder concerns. *Institute for Homeland Security Solutions, Applied Research*, 2013.

S. G. Fabri and V. Kadirkamanathan. *Functional Adaptive Control: An Intelligent Systems Approach*. Springer-Verlag, London, 2001.

P. Fahlstrom and T. Gleason. *Introduction to UAV systems*. John Wiley & Sons, Chichester, 2012.

S.-K. S. Fan and C.-H. Jen. An enhanced partial search to particle swarm optimization for unconstrained optimization. *Mathematics*, 7(4):357, 2019.

J. Fu, W.-h. Chen, and Q.-x. Wu. Chattering-free sliding mode control with unidirectional auxiliary surfaces for miniature helicopters. *International Journal of Intelligent Computing and Cybernetics*, 5(3):421–438, 2012.

J. Gadewadikar, F. L. Lewis, K. Subbarao, K. Peng, and B. M. Chen. H-infinity static output-feedback control for rotorcraft. *Journal of Intelligent and Robotic Systems*, 54 (4):629–646, 2009.

F. R. Garces, V. M. Becerra, C. Kambhampati, and K. Warwick. *Strategies for Feedback Linearisation: a dynamic neural network approach*. Springer, London, 2003.

V. Gavrilets. *Autonomous aerobatic maneuvering of miniature helicopters*. PhD thesis, Massachusetts Institute of Technology, 2003.

B. Godbolt, N. I. Vitzilaios, and A. F. Lynch. Experimental validation of a helicopter autopilot design using model-based PID control. *Journal of Intelligent & Robotic Systems*, 70(1-4):385–399, 2013.

V. Gomez, N. Gomez, J. Rodas, E. Paiva, M. Saad, and R. Gregor. Pareto optimal PID tuning for Px4-based unmanned aerial vehicles by using a multi-objective particle swarm optimization algorithm. *Aerospace*, 7(6):71, 2020.

P. Goupil and A. Marcos. Advanced diagnosis for sustainable flight guidance and control: The European ADDSAFE project. Technical report, SAE technical paper, 2011.

D. R. Green and C. Gómez. From radio-controlled model aircraft to drones. In *Unmanned Aerial Remote Sensing: UAS for Environmental Applications*, pages 13–34. CRC Press, 2020.

B. J. Guerreiro, C. Silvestre, R. Cunha, and D. Cabecinhas. LiDAR-based Control of Autonomous Rotorcraft for the Inspection of Pierlike Structures. *IEEE Transactions on Control Systems Technology*, 26(4):1430–1438, 2018.

A. Guezmil, H. Berriri, R. Pusca, A. Sakly, R. Romary, and M. F. Mimouni. Unknown input observer-based fault diagnosis in induction machine: Theory and experiment. *Electric Power Components and Systems*, 46(11-12):1351–1363, 2018.

İ. Gümüşboğa and A. İftar. Aircraft trim analysis by particle swarm optimization. *Journal of Aeronautics and Space Technologies*, 12(2):185–196, 2019.

M. Guo, Y. Su, and D. Gu. Mixed H_2/H_∞ tracking control with constraints for single quadcopter carrying a cable-suspended payload. *IFAC-PapersOnLine*, 50:4869–4874, 2017.

W. Guo. Flight control design for rotorcraft with variable rotor speed. PhD Thesis, Pennsylvania State University, 2009.

C. Hajiyev and H. E. Soken. Robust adaptive Kalman filter for estimation of UAV dynamics in the presence of sensor/actuator faults. *Aerospace Science and Technology*, 28(1):376–383, 2013.

O. Halbe and M. Hajek. Robust helicopter sliding mode control for enhanced handling and trajectory following. *Journal of Guidance, Control, and Dynamics*, 43(10):1805–1821, 2020.

O. Halbe, T. Mehling, M. Hajek, and M. Vrdoljak. Synthesis and piloted evaluation of advanced rotorcraft response types using robust sliding mode control. *Journal of the American Helicopter Society*, 66(4):1–20, 2021.

C. Hall. Low noise engine design for the silent aircraft initiative. *Aeronautical Journal*, 113(1147):599–607, 2009.

M. T. Hamayun, C. Edwards, and H. Alwi. A fault tolerant control allocation scheme with output integral sliding modes. *Automatica*, 49(6):1830–1837, 2013.

M. T. Hamayun, C. Edwards, and H. Alwi. Design and analysis of an integral sliding mode fault tolerant control scheme. In *Fault Tolerant Control Schemes Using Integral Sliding Modes*, pages 39–61. Springer, 2016.

M. He and J. He. Extended state observer-based robust backstepping sliding mode control for a small-size helicopter. *IEEE Access*, 6:33480–33488, 2018.

Y. He and J. Han. Acceleration-feedback-enhanced robust control of an unmanned helicopter. *Journal of Guidance, Control, and Dynamics*, 33(4):1236, 2010.

J. J. Hopfield. Artificial neural networks. *IEEE Circuits and Devices Magazine*, 4(5): 3–10, 1988.

R. Isermann and M. Münchhof. *Identification of Dynamic Systems: An introduction with applications*. Springer Science & Business Media, 2010.

T. Iwata. *Integrated flight/propulsion control with variable rotor speed command for rotorcrafts*. PhD thesis, Stanford University, 1996.

R. V. Jategaonkar. Flight vehicle system identification-engineering utility. *Journal of Aircraft*, 42(1):11–11, 2005.

J. Javadi-Maghaddam and A. Bagheri. An adaptive neuro-fuzzy sliding mode based genetic algorithm control system for under water remotely operated vehicle. *Expert Systems with Applications*, 37:647–660, 2010.

D.-Y. Jeong, T. Kang, H. R. Dharmayanda, and A. Budiyono. H-infinity attitude control system design for a small-scale autonomous helicopter with nonlinear dynamics and uncertainties. *Journal of Aerospace Engineering*, 25(4):501–518, 2012.

T. Jiang, D. Lin, and T. Song. Novel integral sliding mode control for small-scale unmanned helicopters. *Journal of the Franklin Institute*, 356(5):2668–2689, 2019.

Q. Jin, L. Qi, B. Jiang, and Q. Wang. Novel improved cuckoo search for PID controller design. *Transactions of the Institute of Measurement and Control*, 37(6):721–731, 2015.

S. John and J. O. Pedro. Neural network-based adaptive feedback linearization control of antilock braking system. *International Journal of Artificial Intelligence*, 10(A13): 21–40, 2013.

W. Johnson. *Helicopter Theory*. Dover Publication, Inc., 1980.

W. Johnson. *Rotorcraft aeromechanics*, volume 36. Cambridge University Press, 2013.

W. Johnson. NDARC-NASA design and analysis of rotorcraft. Technical report, NASA, 2015.

M. H. Khalesi, H. Salarieh, and M. Saadat Foumani. System identification and robust attitude control of an unmanned helicopter using novel low-cost flight control system. *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering*, 234(5):634–645, 2020.

S. P. Khaligh. *Control-oriented modeling and system identification for nonlinear trajectory tracking control of a small-scale unmanned helicopter*. PhD thesis, University of Alberta (Canada), 2014.

A. H. Khan, Z. Weiguo, Z. H. Khan, and S. Jingping. Optimized reconfigurable control design for aircraft using genetic algorithm. *Research Journal of Applied Science, Engineering and Technology*, 6(24):4653–4662, 2013.

A. N. Khizer, D. Yaping, S. A. Ali, and X. Xiangyang. Stable hovering flight for a small unmanned helicopter using fuzzy control. *Mathematical Problems in Engineering*, 2014:1–17, 2014. doi: DOI:10.1016/S0020-0255(02)00197-4. http://dx.doi.org//10.1155/2014/208253.

H. J. Kim, D. H. Shim, and S. Sastry. Nonlinear model predictive tracking control for rotorcraft-based unmanned aerial vehicles. In *American Control Conference, 2002. Proceedings of the 2002*, volume 5, pages 3576–3581. IEEE, 2002.

K.-S. Kim, K.-J. Lee, and Y. Kim. Reconfigurable flight control system design using direct adaptive method. *Journal of Guidance, Control, and Dynamics*, 26(4):543–550, 2003.

S. K. Kim and D. M. Tilbury. Mathematical modeling and experimental identification of an unmanned helicopter robot with flybar dynamics. *Journal of Robotic Systems*, 21 (3):95–116, 2004.

E. Kolman and M. Margaliot. *Knowledge-Based Neurocomputing: A Fuzzy Logic Approach*. Springer, Berlin, 2009.

J. Korbicz, J. M. Koscielny, Z. Kowalczuk, and W. Cholewa. *Fault diagnosis: models, artificial intelligence, applications*. Springer Science & Business Media, 2004.

R. Kruse, C. Moewes, C. Borgelt, M. Steinbrecher, F. Klawonn, and P. Held. *Computational Intelligence: A Methodological Introduction*. Springer, London, 2013.

S. Kurnaz, O. Cetin, and O. Kaynak. Adaptive neuro-fuzzy inference system based autonomous flight control of unmanned air vehicles. *Expert Systems with Applications*, 37:1229–1234, 2010.

J. C. Lagarias, J. A. Reeds, M. H. Wright, and P. E. Wright. Convergence properties of the Nelder–Mead simplex method in low dimensions. *SIAM Journal on Optimization*, 9(1):112–147, 1998.

A. Lapedes and R. Farber. How neural nets work. In *Neural Information Processing Systems*, 1987.

D. M. Layton. *Helicopter Performance*. Monterey, California: Naval Postgraduate School, 1983.

D. Lee, H. J. Kim, and S. Sastry. Feedback linearization vs. adaptive sliding mode control for a quadrotor helicopter. *International Journal of Control, Automation and Systems*, 7(3):419–428, 2009.

C. A. Leke. *Computational Intelligence Techniques for High-Dimensional Missing Data Estimation*. Phd thesis, University of Johannesburg (South Africa), 2017.

C. A. Leke and T. Marwala. *Deep learning and missing data in engineering systems*. Springer, 2019.

J. Liu. *Sliding mode control using MATLAB*. Academic Press, London, 2017.

J. Liu. *Intelligent control design and Matlab simulation*. Tsinghua University Press and Springer Nature, Beijing and Singapore, 2018.

J. Liu, Y. Gao, Y. Yin, J. Wang, W. Luo, and G. Sun. *Sliding mode control methodology in the applications of industrial power systems*. Springer International Publishing, 2020.

Y. Ma, L. Lin, and F. Qi. Fault diagnosis of the motor of electro-mechanical transmission of the high speed rotorcraft. In *Proceedings of the 11th International Conference on Modelling, Identification and Control (ICMIC2019)*, pages 359–371. Springer, 2020.

D. Y. Maharaj. *The Application of Nonlinear Control Theory to Robust Helicopter Flight Control*. PhD thesis, Imperial College London (University of London), 1994.

B. Majhi and G. Panda. Robust identification of nonlinear complex systems using low complexity ANN and particle swarm optimization technique. *Expert Systems with Applications*, 38(1):321–333, 2011.

B. Mettler, M. B. Tischler, and T. Kanade. System identification of small-size unmanned helicopter dynamics. In *Annual Forum Proceedings- American Helicopter Society*, volume 2, pages 1706–1717, 1999.

B. Mettler, M. B. Tischler, and T. Kanade. System identification modeling of a small-scale unmanned rotorcraft for flight control design. *Journal of the American Helicopter Society*, 47(1):50–63, 2002.

W. Miliken. Progress in dynamic stability and control research. *Journal of the Aeronautical Sciences*, 14(9):493–519, 1947.

MilitaryFactory. Northrop Grumman MQ-8 Fire Scout. (Last accessed 1 Jul 2018), 2008. URL `https://www.militaryfactory.com/aircraft/detail.asp?aircraft_id=889`.

S. Mirjalili. The ant lion optimizer. *Advances in Engineering Software*, 83:80–98, 2015.

S. Mirjalili and J. S. Dong. *Multi-objective optimization using artificial intelligence techniques*. Springer, 2020.

M. A. Mohd Basri, A. R. Husain, and K. A. Danapalasingam. A hybrid optimal backstepping and adaptive fuzzy control for autonomous quadrotor helicopter with time-varying disturbance. *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering*, 229(12):2178–2195, 2015.

M. R. Mokhtari and B. Cherki. Sliding mode control for a small coaxial rotorcraft UAV. In *2015 3rd International Conference on Control, Engineering & Information Technology (CEIT)*, pages 1–6. IEEE, 2015.

L. J. Mpanza. *A rough set approach to bushings fault detection.* Master's thesis, University of Johannesburg (South Africa), 2012.

L. J. Mpanza and J. O. Pedro. Sliding mode control of a 2-DOF hydraulic servo system. In *6th International Carpathian Control Conference (ICCC2015)*, pages 321–327. IEEE, 2015.

L. J. Mpanza and J. O. Pedro. Nature-inspired optimization algorithms for sliding mode control parameters tuning for autonomous quadrotor. In *2019 IEEE Conference on Control Technology and Applications (CCTA)*, pages 1087–1092. IEEE, 2019.

L. J. Mpanza and J. O. Pedro. Sliding mode control of an electromechanical actuator swashplate using cuckoo search optimisation algorithm. In *2020 3rd International Conference on Control and Robots (ICCR)*, pages 204–210. IEEE, 2020.

L. J. Mpanza and J. O. Pedro. Optimised Tuning of a PID-Based Flight Controller for a Medium-Scale Rotorcraft. *Algorithms*, 14(6):178, 2021.

A. M. Munna, A. Ferdous, M. N. Islam, and A. Azad. Analysis of stability and control of helicopter flight dynamics through mathematical modeling in Matlab. In *2020 IEEE Region 10 Symposium (TENSYMP)*, pages 722–725. IEEE, 2020.

R. Murray-Smith and T. Johansen. *Multiple Model Approaches to Nonlinear Modelling and Control.* CRC Press, New York, 2020.

I. Nagesh and C. Edwards. A multivariable super-twisting sliding mode approach. *Automatica*, 50(3):984–988, 2014.

A. Naimi, J. Deng, S. Shimjith, and A. J. Arul. Dynamic neural network-based feedback linearization control of a pressurized water reactor. In *2020 13th International Conference on Developments in eSystems Engineering (DeSE)*, pages 228–232. IEEE, 2020.

Z. Nejati and A. Faraji. Actuator fault detection and isolation for helicopter unmanned aerial vehicle in the presence of disturbance. *International Journal of Engineering*, 34 (3):676–681, 2021.

A. Nemati, M. Haddad Zarif, and M. Fateh. Helicopter adaptive control with parameter estimation based on feedback linearization. *Engineering and Technology International Journal of Mechanical and Mechatronics Engineering*, 1:247–252, 2007.

N. T. Nguyen. Model-reference adaptive control. In *Model-Reference Adaptive Control: A Primer*, pages 83–123. Springer, Cham, 2018.

K. Nonaka and H. Sugizaki. Integral sliding mode altitude control for a small model helicopter with ground effect compensation. In *Proceedings of the 2011 American Control Conference*, pages 202–207. IEEE, 2011.

H. Noura, D. Theilliol, J.-C. Ponsart, and A. Chamseddine. *Fault-tolerant control systems: Design and practical applications*. Springer, London, 2009.

L. Nyulászi, R. Andoga, P. Butka, L. Főző, R. Kovacs, and T. Moravec. Fault detection and isolation of an aircraft turbojet engine using a multi-sensor network and multiple model approach. *Acta Polytechnica Hungarica*, 15(2):189–209, 2018.

A. O'dwyer. *Handbook of PI and PID controller tuning rules*. World Scientific, London, 2009.

K. Ogata. *Modern control engineering*. Prentice hall, New Jersey, 2010.

M. O. Okwu and L. K. Tartibu. *Metaheuristic optimization: Nature-inspired algorithms swarm and computational intelligence, theory and applications*, volume 927. Springer Nature, 2020.

J. Oliveira, P. M. Oliveira, J. Boaventura-Cunha, and T. Pinho. Evaluation of hunting-based optimizers for a quadrotor sliding mode flight controller. *Robotics*, 9(2):22, 2020.

G. G. Padfield. *Helicopter Flight Dynamics, 2nd Edition*. Wiley-Blackwell, Washington, 2008.

N. Paulino, C. Silvestre, and R. Cunha. Affine parameter-dependent preview control for rotorcraft terrain following flight. *Journal of Guidance, Control, and Dynamics*, 29(6): 1350–1359, 2006.

M. Pavel, P. Shanthakumaran, O. Stroosma, Q. Chu, M. Wolfe, and H. Cazemier. Development of advanced flight control laws for the AH-64 apache helicopter: Sketches from the work of TU Delft-Boeing Project in SIMONA Simulator. In *72nd Annual Forum of the American Helicopter Society: Leveraging Emerging Technologies for Future Capabilities*. AHS, 2016.

J. O. Pedro and O. Dahunsi. Neural network based feedback linearization control of a servo-hydraulic vehicle suspension system. *International Journal of Applied Mathematics and Computer Science*, 21(1):137–147, 2011.

J. O. Pedro and P. Kantue. Online aerodynamic parameter estimation of a miniature unmanned helicopter using radial basis function neural networks. In *Control Conference (ASCC), 2011 8th Asian*, pages 1170–1175. IEEE, 2011.

J. O. Pedro, M. Dangor, and P. J. Kala. Differential evolution-based PID control of a quadrotor system for hovering application. In *Evolutionary Computation (CEC), 2016 IEEE Congress on*, pages 2791–2798. IEEE, 2016.

J. O. Pedro, M. Dangor, O. A. Dahunsi, and M. M. Ali. Dynamic neural network-based feedback linearization control of full-car suspensions using PSO. *Applied Soft Computing*, 70:723–736, 2018.

U. Pérez, E. Capello, E. Punta, J. Perea, and L. Fridman. Fault detection and isolation for a 3-dof helicopter with sliding mode strategies. In *2018 15th International Workshop on Variable Structure Systems (VSS)*, pages 279–284. IEEE, 2018.

W. Perruquetti and J.-P. Barbot. *Sliding mode control in engineering*. CRC Press, 2002.

C. Phillips, C. L. Karr, and G. Walker. Helicopter flight control with fuzzy logic and genetic algorithms. *Engineering Applications of Artificial Intelligence*, 9(2):175–184, 1996.

G. M. Platt, X.-S. Yang, and A. J. S. Neto. *Computational Intelligence, Optimization and Inverse Problems with Applications in Engineering*. Springer, 2018.

G. L. Plett. Adaptive inverse control of linear and nonlinear systems using dynamic neural networks. *Neural Networks, IEEE Transactions on*, 14(2):360–376, 2003.

S. Pourrezaei Khaligh. *Control-oriented modeling and system identification for nonlinear trajectory tracking control of a small-scale unmanned helicopter*. PhD thesis, University of Alberta, 2014.

K. Premkumar and B. V. Manijandan. Adaptive Neuro-Fuzzy Inference System based Speed Controller for Brushless DC Motor. *Neurocomputing*, 138:260–270, 2014.

T. K. Priyambodo, A. E. Putra, and A. Dharmawan. Optimizing control based on ant colony logic for quadrotor stabilization. In *International Conference on Aerospace Electronics and Remote Sensing Technology (ICARES)*, pages 1–4. IEEE, 2015.

R. Prouty. *Helicopter aerodynamics*. PJS Publication, Boston, MA, 1984.

R. W. Prouty. *Helicopter performance, stability, and control*. Krieger Publishing Company, Melbourne, FL, 1995.

I. E. Putro, A. Budiyono, K.-J. Yoon, and D. Kim. Modeling of unmanned small scale rotorcraft based on neural network identification. In *Robotics and Biomimetics, 2008. ROBIO 2008. IEEE International Conference on*, pages 1938–1943. IEEE, 2009.

A. Ramalakshmi, P. Manoharan, and P. Deepamangai. PID tuning and control for 2-DOF helicopter using particle swarm optimization. In *International Conference on Swarm, Evolutionary, and Memetic Computing*, pages 662–672. Springer, 2013.

J. R. Raol, G. Gopalratnam, and B. Twala. *Nonlinear Filtering: Concepts and Engineering Applications*. CRC Press, 2017.

I. A. Raptis and K. P. Valavanis. *Linear and nonlinear control of small-scale unmanned helicopters*, volume 45. Springer Science & Business Media, 2010.

B. Ratner. *Statistical and Machine-Learning Data Mining*. Second Edition. CRC Press, Taylor & Francis Group, 2011.

B. Ren, S. Ge, C. Chen, C.-H. Fua, and T. Lee. *Modeling, Control and Coordination of Helicopter Systems*. Springer, 2012.

J. H. Richter. *Reconfigurable Control of Nonlinear Dynamical Systems: A Fault-Hiding Approach*. Springer-Verlag, Berlin, 2011.

A. Rodríguez-Molina, E. Mezura-Montes, M. G. Villarreal-Cervantes, and M. Aldape-Pérez. Multi-objective meta-heuristic optimization in intelligent control: A survey on the controller tuning problem. *Applied Soft Computing*, 93:106342, 2020.

K. Rudin, G. J. Ducard, and R. Y. Siegwart. Active fault-tolerant control with imperfect fault detection information: Applications to UAVs. *IEEE Transactions on Aerospace and Electronic Systems*, 56(4):2792–2805, 2020.

M. Sababha, M. Zohdy, and M. Kafafy. Robust pole placement using firefly algorithm. *International Journal of Electrical & Computer Engineering (2088-8708)*, 9(2), 2019.

I. Sadeghzadeh. *Fault Tolerant Flight Control of Unmanned Aerial Vehicles*. PhD thesis, Concordia University, 2015.

A. Saez, M. Manzo, and M. Ciarcià. Mars drone configurations and approaches to rotor design: A review. In *ASME International Mechanical Engineering Congress and Exposition*, volume 85611, page V07AT07A046. American Society of Mechanical Engineers, 2021.

M. Saffarian and F. Fahimi. A comprehensive kinematic analysis of a model helicopter's actuating mechanism. In *46th AIAA Aerospace Sciences Meeting and Exhibit*, pages 133:1–25, Reno, Nevada, 2008.

S. Salmah, N. Y. Megawati, E. Joelianto, and A. Budiyono. Control of autonomous helicopter models with robust H2-type switched linear controller. *International Journal of Applied Mathematics and Statistics (IJAMAS)*, 35(5):137–148, 2013.

M. K. Samal, S. Anavatti, and M. Garratt. Neural network based system identification for autonomous flight of an Eagle helicopter. In *Proceedings of the 17th World Congress of the International Federation of Automatic Control*, pages 6–11, 2008.

E. N. Sanchez, H. M. Becerra, and C. M. Velez. Combining fuzzy, PID and regulation control for an autonomous mini-helicopter. *Information Sciences*, 177(10):1999–2022, 2007.

J. M. Seddon and S. Newman. *Basic helicopter aerodynamics*. Blackwell Science, London, 2 edition, 2002.

A. Shafiekhani, M. J. Mahjoob, and M. Akraminia. Design and implementation of an adaptive critic-based neuro-fuzzy controller on an unmanned bicycle. *Mechatronics*, 28:115–123, 2015.

N. G. Shakev and A. V. Topalov. Continuous Sliding Mode Control of a Quadrotor. In *Recent Advance in Sliding Modes: From Control to Intelligent Mechatronics*, pages 441–458. Springer International Publishing, 2015.

Y. T. Shi, Q. Kou, D. H. Sun, Z. Li, S. Qiao, and Y. Hou. H_∞ fault tolerant control of wind turbine system with actuator faults. *World Congress*, 19(1):5838–5843, 2014.

D. H. Shim, H. J. Kim, and S. Sastry. Control system design for rotorcraft-based unmanned aerial vehicles using time-domain system identification. In *Proceedings of the 2000. IEEE International Conference on Control Applications. Conference Proceedings (Cat. No. 00CH37162)*, pages 808–813. IEEE, 2000.

M. Shoaib, A. Y. Shamseldin, B. W. Melville, and M. M. Khan. A comparison between wavelet based static and dynamic neural network approaches for runoff prediction. *Journal of hydrology*, 535:211–225, 2016.

Y. Shtessel, C. Edwards, L. Fridman, A. Levant, et al. *Sliding mode control and observation*, volume 10. Springer, New York, 2014.

R. Singh and B. Bhushan. A novel fault classification-based fault-tolerant control for two degree of freedom helicopter systems. *International Journal of Adaptive Control and Signal Processing*, 34(8):1080–1104, 2020.

H. Sira-Ramirez, M. Zribim, and S. Ahmad. Dynamical sliding mode control approach for vertical flight regulation in helicopters. *IEEE Proceedings-Control Theory and Applications*, 141(1):19–24, 1994.

J.-J. E. Slotine and W. Li. *Applied Nonlinear Control*. Prentice-Hall, Inc, 1991.

M. Smaili, J. Breeman, T. Lombaerts, and O. Stroosma. A simulation benchmark for aircraft survivability assessment. In *Proceedings of the International Congress of Aeronautical Sciences*. IEEE, 2008.

K. Socha and M. Dorigo. Ant colony optimization for continuous domains. *European Journal of Operational Research*, 185(7):1155–1173, 2008.

C. K. Solem. Using fly-by-wire technology in future models of the UH-60 and other rotary wing aircraft. Technical report, NASA, 2011.

M. I. Solihin, L. F. Tack, and M. L. Kean. Tuning of PID controller using particle swarm optimization (PSO). In *Proceeding of the International Conference on Advanced Science, Engineering and Information Technology*, volume 1, pages 458–461, 2011.

S. Sridhar, R. Kumar, G. Gupta, M. Kumar, and K. Cohen. Nonlinear control of a novel class of tilt-rotor quadcopters using sliding mode method: Theory and hardware implementation. *Journal of Aerospace Engineering*, 35(3):04022017, 2022.

H. Suprijono and B. Kusumoputro. Direct inverse control based on neural network for unmanned small helicopter attitude and altitude control. *Journal of Telecommunication, Electronic and Computer Engineering (JTEC)*, 9(2-2):99–102, 2017.

M. Taimoor and L. Aijun. Adaptive strategy for fault detection, isolation and reconstruction of aircraft actuators and sensors. *Journal of Intelligent & Fuzzy Systems*, 38(4): 4993–5012, 2020.

I. B. Tijani, R. Akmeliawati, A. Legowo, and A. Budiyono. Nonlinear identification of a small scale unmanned helicopter using optimized NARX network with multi-objective differential evolution. *Engineering Applications of Artificial Intelligence*, 33:99–115, 2014.

M. B. Tischler and R. K. Remple. *Aircraft and Rotorcraft System Identification: Engineering Methods with Flight Test Examples*. AIAA Education Series. American Institute of Aeronautics and Astronautics, 2012. ISBN 9781600868207.

H. M. R. Ugalde, J.-C. Carmona, J. Reyes-Reyes, V. M. Alvarado, and J. Mantilla. Computational cost improvement of neural network models in black box nonlinear system identification. *Neurocomputing*, 166:96–108, 2015.

V. Utkin, A. Poznyak, Y. Orlov, and A. Polyakov. Conventional and high order sliding mode control. *Journal of the Franklin Institute*, 357(15):10244–10261, 2020.

K. P. Valavanis and G. J. Vachtsevanos. *Handbook of Unmanned Aerial Vehicles*. Springer-Verlag, Dordrecht, 2015.

P. Van Overschee and B. De Moor. RAPID: the end of heuristic PID tuning. *IFAC Proceedings Volumes*, 33(4):595–600, 2000.

J. Velagic and N. Osmic. Design and implementation of fuzzy logic controllers for helicopter elevation and azimuth controls. In *Control and Fault-Tolerant Systems (SysTol), 2010 Conference on*, pages 311–316, Nice, France, 2010. IEEE.

R. Venkataraman, P. Bauer, P. Seiler, and B. Vanek. Comparison of fault detection and isolation methods for a small unmanned aircraft. *Control Engineering Practice*, 84: 365–376, 2019.

C. Venkatesan. *Fundamentals of helicopter dynamics*. CRC Press, Boca Raton, 2014.

B. Wang and Y. Zhang. An adaptive fault-tolerant sliding mode control allocation scheme for multirotor helicopter subject to simultaneous actuator faults. *IEEE Transactions on Industrial Electronics*, 65(5):4227–4236, 2018.

S. Wang, J. Zhang, Q. Zhang, and C. Pei. An innovative fuzzy backstepping sliding mode controller for a tri-rotor unmanned aerial vehicle. *Microsystem Technologies*, 23(12): 5621–5630, 2017.

T. Wang, Y. Zhang, J. Qiu, and H. Gao. Adaptive fuzzy backstepping control for a class of nonlinear systems with sampled and delayed measurements. *IEEE Transactions on Fuzzy Systems*, 23(2):302–312, April 2015. ISSN 1063-6706.

W. Wang, D. Li, and C. Liu. Helicopter flight simulation trim in the coordinated turn with the hybrid genetic algorithm. *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering*, 233(3):1159–1168, 2019.

Y. Wang, C. Jiang, Y. Zhang, and H. Wang. Helicopter safe landing trajectory after main rotor actuator failures. *Applied Sciences*, 10(8):2917, 2020.

J. Williams, A. Davis, and P. Drake. *Condition-based maintenance and machine diagnostics*. Chapman & Hall, London, 1994.

G. X. Xu. Nonlinear fault-tolerant guidance and control for damaged aircraft. MSc Thesis, University of Toronto, 2011.

X.-S. Yang. *Cuckoo search and firefly algorithm: theory and applications*, volume 516. Springer Cham, Switzerland, 2013.

X.-S. Yang and S. Deb. Cuckoo search via Lévy flights. In *Nature & Biologically Inspired Computing, 2009. NaBIC 2009. World Congress on*, pages 210–214. IEEE, 2009.

X.-S. Yang and X. He. Firefly algorithm: recent advances and applications. *International Journal of Swarm Intelligence*, 1(1):36–50, 2013.

X. Yao, Y. Liu, K.-H. Liang, and G. Lin. Fast evolution algorithms. *Advances in Evolutionary Computing Theory and Application*, pages 45–94, 2003.

B. Yu. *Fault-Tolerant Control with Applications to Aircraft Using Linear Quadratic Design Framework*. PhD thesis, Concordia University, 2016.

G.-R. Yu. Nonlinear optimal control of helicopter using fuzzy gain scheduling. In *Systems, Man and Cybernetics, 2005 IEEE International Conference on*, volume 4, pages 3313–3318. IEEE, 2005.

D. Yuke, L. Jun, and X. Yiming. Mechanical modeling and simulation for the take-off and landing performance calculation of low-cost UAVs. In *Proceedings of the 5th China Aeronautical Science and Technology Conference*, pages 1–14. Springer, 2022.

L. A. Zadeh. Fuzzy sets. *Information and Control*, 8(3):338 – 353, 1965. ISSN 0019-9958. doi: http://dx.doi.org/10.1016/S0019-9958(65)90241-X. URL `http://www.sciencedirect.com/science/article/pii/S001999586590241X`.

C. Zang, H. Xin, and J. Driscoll. Development and validation of an engineering simulation model in FLIGHTLAB with customized modeling enhancements. In *73rd American Helicopter Society Forum, 9-11 May*, Fort Worth, Texas, US, 2017.

H. Zhang, N. Xu, G. Zong, and A. F. Alkhateeb. Adaptive fuzzy hierarchical sliding mode control of uncertain under-actuated switched nonlinear systems with actuator faults. *International Journal of Systems Science*, 52(8):1499–1514, 2021.

Y. Zhang and A. Chamseddine. Fault tolerant flight control techniques with application to a quadrotor UAV testbed. *Automatic Flight Control Systems - Latest Developments*, pages 119 – 140, 2012. http://dx.doi.org//10.5772/38918.

Y. Zhang and J. Jiang. Bibliographical review on reconfigurable fault-tolerant control systems. *Annual Reviews in Control*, 32(2):229–252, 2008.

W. Zheng, D. Hu, and J. Wang. Fault localization analysis based on deep neural network. *Mathematical Problems in Engineering*, 2016, 2016.

Y. Zhong, Y. Zhang, W. Zhang, J. Zuo, and H. Zhan. Robust actuator fault detection and diagnosis for a quadrotor UAV with external disturbances. *IEEE Access*, 6:48169–48180, 2018.

J. Zhou and C. Wen. *Adaptive backstepping control of uncertain systems: nonsmooth nonlinearities, interactions or time-variations*. Springer-Verlag, Berlin, 2008.

X. Zhou and X. Zhang. Multi-objective-optimization-based control parameters auto-tuning for aerial manipulators. *International Journal of Advanced Robotic Systems*, 16 (1):1729881419828071, 2019.

G. Zogopoulos-Papaliakos, G. C. Karras, and K. J. Kyriakopoulos. A fault-tolerant control scheme for fixed-wing UAVs with flight envelope awareness. *Journal of Intelligent & Robotic Systems*, 102(2):1–33, 2021.

A. Zolghadri, D. Henry, J. Cieslak, D. Efimov, and P. Goupil. *Fault Diagnosis and Fault-Tolerant Control and Guidance for Aerospace Vehicles*. Springer-Verlag, London, 2014.