

https://dspace.org/

Fault-tolerant control of an unmanned aerial vehicle using optimised neural network

Tshabalala, T

2018

https://hdl.handle.net/10539/25715

Tshabalala, Thando Busisiwe (2018) Fault-tolerant control of an unmanned aerial vehicle using optimised neural network,University of the Witwatersrand, Johannesburg, https://hdl.handle.net/10539/25715

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

FAULT-TOLERANT CONTROL OF AN UNMANNED AERIAL VEHICLE USING OPTIMISED NEURAL NETWORK

T. Tshabalala

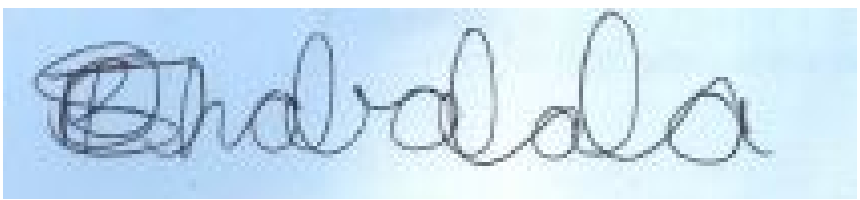
A dissertation submitted to the Faculty of Engineering and the Built Environment, University of the Witwatersrand, Johannesburg, in fulfilment of the requirements for the degree of Master of Science in Engineering.

Johannesburg, May 2018

Declaration

I declare that this dissertation is my own, unaided work, except where otherwise acknowledged. It is being submitted for the degree of Master of Science in Engineering in the University of the Witwatersrand, Johannesburg. It has not been submitted before for any degree or examination at any other university.

Signed this 22nd day of May 2018



T. Tshabalala

Acknowledgements

This research would not have been possible without the support and assistance of many people. I would like to acknowledge and express my gratitude to the following people:

- God, for His divine inspiration and strength.
- Armscor for financial support towards this research.
- Prof. Jimoh O. Pedro for his guidance and sharing his knowledge
- My soul mate half Jean-Patrick Sissing for his support, sanity checks and patience.
- My mother Nobahle Raji, brothers Ntando and Luthando and my family for their continued love and support.

Abstract

An intelligent Fault-Tolerant Control (FTC) system for an Unmanned Aerial Vehicle (UAV) was developed to be capable of tolerating a number of different control actuator faults. The development of the control system, focused on the simulation of the system using a nonlinear flight dynamic model. The general six-degree-of-freedom flight dynamic model made use of available wind tunnel data to improve its accuracy. The system was first controlled using classical Proportional + Integral + Derivative (PID) controllers to test the basic functionality of the control system and to enable isolation of the different parts of the overall control structure. The proposed control strategy consisted of Radial Basis Function Neural Network (RBFNN)-based and Backpropagation Neural Network (BPNN)-based PID controller combined with sequential least squares Control Allocation (CA) algorithm. During normal fault-free operation the Neural Network-based (NN) controllers ensure that the system is able to attain the desired tracking performance. Once failure occurs and the NN-based control system is unable to mitigate the effects of failure, the CA algorithm redistributed the required control effort to the remaining healthy control surfaces. This system played a vital role in minimising the possibility of a fault while necessitating the reconfiguration of the control, guidance or navigation systems in the aircraft by minimising the effects of the simulated fault. The performance of this system was enhanced by optimising the allocation of control effort commanded by the virtual actuators to the physical actuators present on the aircraft. The allocation algorithm made use of secondary and tertiary control effectors. A number of control actuator failures, of varying severity, were modelled including elevator failures; aileron failures; and combined aileron and elevator failures. These failures were also tested in the presence of microburst windshear. The effects of microburst windshear were prevalent within simulations. It was found that the system was able to handle these failures. Though in the case of severe failures, the performance of the system was slightly degraded, and at some times the oscillations previously observed were reduced. The control allocation algorithm was a necessity for the system that operated using classical PID control without the neural network supervision. The proposed system can be effectively used as part of a FTC system.

Scope and Contribution

This research project's novelty exists in:

1. The radial basis function and back propagation neural network control algorithm are applied to both longitudinal and lateral aircraft motion for the purposes of fault tolerant control.
2. A control allocation scheme which may be used with either the neural network or PID controllers for the purposes of fault tolerant control.
3. Design validation was conducted by means of a comparison of different intelligent control schemes previously implemented on the system.

Published Work

Aspects of this thesis have been published in the following references:

- **Pedro, J. O. & Tshabalala, T. B.** (2017). PI-Based Fault Tolerant Control For Fixed-Wing UAVs Using Control Allocation. *Control Conference Africa (CCA 2017).* Johannesburg, South Africa: 7-8 December 2017.
- **Pedro, J. O. & Tshabalala, T. B.** (2017). Fault-Tolerant Control of Fixed-Wing UAV Using GA-Optimised Control Allocation Technique. *11th Asian Control Conference (ASCC 2017).* Gold Coast, Australia: 17-20 December 2017.

Potential publications arising from this work:

- **Tshabalala, T. B. & Pedro, J. O.** (2018). Fuzzy Logic-Based Fault-Tolerant Control For Fixed-Wing UAVs. *15th International Conference on Informatics in Control, Automation and Robotics (ICINCO 2018).* Porto, Portugal: 29-31 July 2018 (accepted for publication).
- **Tshabalala, T. B., & Pedro, J. O.** (2017). Fault-Tolerant Control of Fixed-Wing UAVs Using GA-Optimised Control Allocation Approach in Windy Environments. Automatica, *submitted (under review).*
- **Tshabalala, T. B., & Pedro, J. O.** (2017). RBFNN- and BPNN-Based PID Controllers for Fault-Tolerant Control of Fixed-Wing UAVs. Asian Journal of Control, *submitted (under review).*

- **Tshabalala, T. B., & Pedro, J. O.** (2017). PID-based Fault-Tolerant Control of Fixed-wing UAVs in Microburst Windshear Environment Using Back Propagation Neural Networks. The Aeronautical Journal, *submitted (under review)*.

Contents

Declaration i

Acknowledgements . . . ii
Abstract . . . iii
Scope and Contribution . . . iv
Published Work . . . iv

Contents v

List of Figures ix

List of Tables xiv

Nomenclature xv

Abbreviations xviii

1 Introduction 1

1.1 General Background and Motivation . . . 1
1.2 Problem Statement . . . 3
1.3 Research Objectives . . . 3
1.4 Literature Review . . . 4
1.4.1 UAV Control Schemes . . . 4
1.4.1.1 Current Trend in Fault Tolerant Control Strategies . . . 11
1.4.2 Fault-Tolerant Control Strategies . . . 12
1.4.2.1 Challenges in Fault-Tolerant Control System Design . . . 19
1.4.2.2 Current Trend in Fault-Tolerant Control Strategies . . . 20
1.4.3 Identified Gaps in Literature . . . 22
1.5 Research Scope, Strategy and Methodology . . . 22
1.5.1 Proposed Controller Architecture . . . 23

1.6 Research Contributions . . . 24
1.7 Organisation of Dissertation . . . 24

2 Mathematical and Simulation Model **26**
2.1 Chapter Outline . . . 26
2.2 Ultra Stick Airframe . . . 26
2.3 Flight Dynamic Simulation Methodology . . . 28
2.3.1 Simulation Implementation . . . 29
2.4 Coordinate Systems . . . 30
2.5 Derivation of the Equations of Motion . . . 34
2.6 Model of the Atmosphere . . . 40
2.6.1 Modelling of Wind Effects . . . 41
2.7 Modelling of Forces and Moments . . . 50
2.7.1 Gravitational Forces and Moments . . . 50
2.7.2 Aerodynamic Forces and Moments . . . 52
2.7.3 Thrust Forces and Moments . . . 54
2.8 Kinematic Equations . . . 55
2.9 Navigational Equations . . . 55
2.10 Aerodynamic Coefficients Determination . . . 56
2.11 Propulsion Modelling . . . 56
2.11.1 Propeller . . . 57
2.11.2 Motor . . . 58
2.11.3 Actuator Modelling . . . 59
2.12 Simulation Control Surface Convention . . . 59

3 UAV Controller Development **61**
3.1 Chapter Outline . . . 61
3.2 Overall PID Control Structure . . . 61
3.2.1 Controller Optimization . . . 64
3.2.1.1 Particle Swarm Optimisation . . . 64
3.3 Flight Modes . . . 69
3.4 Control Allocation . . . 70
3.4.1 Control Allocation Algorithm . . . 71
3.4.2 Genetic Algorithm . . . 75
3.4.2.1 Origin of the Species . . . 76
3.4.2.2 Population Selection . . . 76
3.4.2.3 Offspring Generation . . . 77
3.4.2.4 Convergence . . . 78
3.4.3 Control Allocation Algorithm Testing . . . 79

3.5 Neural Network Controller Development . . . 84
3.5.1 System Identification . . . 85
3.5.2 Radial Basis Function Neural Network . . . 86
3.5.3 Backpropagation Neural Network (BPNN) . . . 96
3.5.3.1 Structure of the BPNN-PID controller . . . 96
3.5.4 Network Training . . . 98
3.5.4.1 Scaled Conjugate Gradient Algorithm . . . 98

4 UAV Simulation **105**
4.1 Chapter Outline . . . 105
4.2 Fault Scenarios . . . 105
4.3 Procedures Followed . . . 106
4.3.1 Inner Loop Testing . . . 106
4.3.2 Middle loop testing . . . 107
4.4 Results . . . 108
4.4.1 Inner control loop testing . . . 108
4.4.1.1 Control - No Failures . . . 108
4.4.1.2 Elevator Failure - *0 rad* Deflection . . . 109
4.4.1.3 Elevator Failure - 50% Hardover . . . 112
4.4.1.4 Elevator Failure - 100% Hardover . . . 116
4.4.1.5 Right Aileron Failure - *0 rad* Deflection . . . 120
4.4.1.6 Right Aileron Failure - 50% Hardover . . . 122
4.4.1.7 Right Aileron Failure - 100% Hardover Failure . . . 126
4.4.1.8 Elevator and Aileron Combined Failure - *0 rad* deflection . . . 130
4.4.1.9 Elevator and aileron combined failure - 50% Hardover . . . 133
4.4.1.10 Elevator and Aileron Combined Failure - 100% Hardover . . . 137
4.4.2 Middle Control Loop Simulation . . . 141
4.4.2.1 Control - No Failure . . . 141
4.4.2.2 Elevator Failure - *0 rad* Deflection . . . 146
4.4.2.3 Elevator Failure - 50% Hardover . . . 150
4.4.2.4 Elevator Failure - 100% Hardover . . . 155
4.4.2.5 Right Aileron Failure - *0 rad* Deflection . . . 160
4.4.2.6 Right Aileron Failure - 50% Hardover . . . 165
4.4.2.7 Right Aileron Failure - 100% Hardover . . . 170
4.4.3 Inner Control Loop Testing - Microburst Windshear . . . 175
4.4.3.1 Control - No Failures . . . 175
4.4.3.2 Elevator Failure - *0 rad* Deflection . . . 176
4.4.3.3 Elevator Failure - 50% Hardover . . . 178
4.4.3.4 Right Aileron Failure - *0 rad* Deflection . . . 183

4.4.3.5 Right Aileron Failure - 50% Hardover . . . 185
4.4.3.6 Right Aileron Failure - 100% Hardover . . . 189
4.4.3.7 Elevator and Aileron combined Failure - *0 rad* Deflection . . . 193
4.4.3.8 Elevator and Aileron combined Failure - 50% Hardover . . . 197
4.4.4 Middle Control Loop Simulation - Microburst Windshear . . . 202
4.4.4.1 Control - No Failure . . . 202
4.4.4.2 Elevator Failure - *0 rad* Deflection . . . 207
4.4.4.3 Elevator Failure - 50% Hardover . . . 211
4.4.4.4 Right Aileron Failure - *0 rad* Deflection . . . 217
4.4.4.5 Right Aileron Failure - 50% Hardover . . . 223
4.4.4.6 Right Aileron Failure - 100% Hardover . . . 228

5 Analysis of Results **234**
5.1 Chapter Outline . . . 234
5.2 Research Objectives . . . 234
5.3 Inner Loop Simulations . . . 235
5.3.1 Elevator Failures . . . 235
5.3.2 Aileron Failures . . . 236
5.3.3 Combined Failures . . . 237
5.4 Middle Loop Simulations . . . 238
5.4.1 Control . . . 238
5.4.2 Elevator Failures . . . 239
5.4.3 Aileron Failures . . . 240

6 Conclusions and Recommendations **241**
6.1 Conclusions . . . 241
6.2 Future Recommendations . . . 242

7 References **244**

List of Figures

1.1 MQ-1C Gray Eagle UAV [RP Defense, (2013)]. 1
1.2 Phantom Ray Stealth Unmanned Combat Aerial Vehicle (UCAV) [Mick, (2010)]. 2
1.3 IAI Heron [Defense Industry Daily Staff, (2011)] 2
1.4 Linear Quadratic Gaussian Control Scheme [Huang & Jan, (2014)]. 5
1.5 Closed-Loop PID [Rahimi, Hajighasemi, & Sanaei, (2013)]. 5
1.6 Fuzzy Logic Controller [Rahimi, Hajighasemi, & Sanaei, (2013)]. 8
1.7 Basic Flow of Standard Genetic Algorithm [Feng, Wang, & Qu, (2015)]. . . . 9
1.8 Passive Fault-Tolerant Control Scheme [Sadeghzadeh, (2015)] 13
1.9 Active Fault-Tolerant Control System [Ma, (2011)]. 13
1.10 Gain Scheduling Control Loop [Sadeghzadeh & Zhang, (2013)]. 14
1.11 Model Reference Adaptive Control Structure [Girish, et al., (2015)] 16
1.12 Classification of faults in a controlled system [Tang, (2013)]. 17
1.13 Control Allocation Block Diagram [Sadeghzadeh, (2015)]. 19
1.14 Work Flow Diagram 23
1.15 Proposed Controller Architecture 23
1.16 Structure of the Dissertation 25

2.1 Ultra Stick Aircraft [Paw, (2009)] 27
2.2 High-Level Simulation Methodology 29
2.3 Simplified layout of nonlinear 6-DOF aircraft model 30
2.4 Conventions for earth-fixed coordinate system 31
2.5 Convention for body-fixed coordinate system 32
2.6 Euler angles 33
2.7 Body-fixed reference frames a) aerodynamic $Ox_ay_az_a$ b) arbitrary $Oxyz$ . . . 36
2.8 Forces and moments and their components in system $Oxyz$ 38
2.9 Velocities in the aircraft's plane of symmetry 42
2.10 Velocities on the horizontal plane 44
2.11 Sinusoidal wind field 45
2.12 Angular and gradient wind velocity 46
2.13 Wind gradients in the direction of the aircraft axes 47

2.14 Vortex-ring model [Wang, et al.,(2016)] . . . 48
2.15 Illustrated system references for aircraft . . . 52
2.16 Aircraft body axis forces and moments [Paw, (2009)] . . . 53
2.17 Propeller Performance [Paw, (2009)] . . . 58
2.18 Motor Power Output [Paw, (2009)] . . . 59
2.19 Elevator Deflection Convention . . . 60
2.20 Rudder Deflection Convention . . . 60
2.21 Aileron and Flap Deflection Convention . . . 60

3.1 Block Diagram Showing Control Structure . . . 63
3.2 PSO search mechanism in multidimensional search space [Alam, et al, (2015)] 65
3.3 Flowchart of PSO . . . 67
3.4 Architecture of Optimised PID Controller . . . 68
3.5 Fitness function . . . 69
3.6 Flowchart of active set algorithm . . . 73
3.7 Flowchart of sequential least squares . . . 74
3.8 Flowchart of a continuous GA [Haupt & Haupt, (2004)] . . . 75
3.9 Control deflections required to maintain level flight - Fault-free . . . 81
3.10 Control deflections required to maintain stable flight - No elevator . . . 82
3.11 Control deflections required to maintain stable flight - No aileron . . . 83
3.12 Control deflections required to maintain stable flight - No aileron & flap . . . 84
3.13 System Identification . . . 85
3.14 Neural Network Model [Gibson & Patterson, (2017)] . . . 86
3.15 Radial basis function neural network model [Passino, (2005)] . . . 87
3.16 RBF centres for pitch and roll controllers . . . 90
3.17 RBF centres for velocity controller . . . 91
3.18 RBF centres for altitude controller . . . 91
3.19 RBF centres for heaing controller . . . 92
3.20 RBF mapping for pitch and roll controllers . . . 92
3.21 RBF mapping for velocity controller . . . 93
3.22 RBF mapping for altitude controller . . . 93
3.23 RBF mapping for heading controller . . . 94
3.24 RBF central receptive field unit for pitch and roll controllers . . . 94
3.25 RBF central receptive field unit for velocity controller . . . 95
3.26 RBF central receptive field unit for altitude controller . . . 95
3.27 RBF central receptive field unit for heading controller . . . 96
3.28 BPNN Structure [Che, et al., (2011)] . . . 97
3.29 BPNN-based PID control scheme [Yang, (2014)] . . . 97
3.30 BPNN-based PID control scheme with control allocation . . . 98

4.1 Pitch and bank angles . . . 109
4.2 Pitch and roll inputs . . . 109
4.3 Pitch and bank angles (0 *rad* elevator failure) . . . 110
4.4 Pitch and roll control inputs (0 *rad* elevator failure) . . . 111
4.5 Controller gains 0 *rad* elevator failure . . . 112
4.6 Pitch and bank angles (50% hardover) . . . 113
4.7 Pitch and roll control inputs (50% hardover) . . . 113
4.8 Controllers gains 50% hardover elevator failure . . . 114
4.9 Control surface deflections (50% hardover) . . . 115
4.10 Pitch and bank angles (100% hardover elevator failure) . . . 116
4.11 Pitch and roll control inputs (100% hardover elevator failure) . . . 117
4.12 Controller gains 100% hard over elevator failure . . . 118
4.13 Control surface deflections (100% hardover) . . . 119
4.14 Pitch and bank angles (0 *rad* right aileron deflection failure) . . . 120
4.15 Pitch and roll control inputs (0 *rad* right aileron deflection failure) . . . 121
4.16 Controller gains 0 *rad* right aileron failure . . . 122
4.17 Pitch and bank angles (50% right aileron hardover failure) . . . 123
4.18 Pitch and roll control inputs (50% right aileron hardover failure) . . . 123
4.19 Controller gains with 50% right aileron hardover failure . . . 124
4.20 Control surface deflections (50% hardover right aileron failure) . . . 125
4.21 Pitch and bank angles (100% right aileron hardover failure) . . . 126
4.22 Pitch and roll control inputs (100% right aileron hardover failure) . . . 127
4.23 Controller gains during 100% right aileron hardover failure . . . 128
4.24 Control surface deflections (100% hardover right aileron failure) . . . 129
4.25 Pitch and bank angles (0 *rad* elevator and right aileron deflection failure) . . . 130
4.26 Pitch and roll control inputs (0 *rad* elevator and right aileron deflection failure) 130
4.27 Controller gains 0 *rad* elevator and right aileron failure . . . 131
4.28 Control surface deflections (0 *rad* elevator and right aileron deflection failure) 132
4.29 Pitch and bank angles (50% elevator and right aileron hardover failure) . . . 133
4.30 Pitch and roll control inputs (50% elevator and right aileron hardover failure) 134
4.31 Controller gains 50% elevator and right aileron hardover failure . . . 135
4.32 Control surface deflections (50% elevator and right aileron hardover failure) . 136
4.33 Pitch and bank angles (100% elevator and right aileron hardover failure) . . . 137
4.34 Pitch and roll control inputs (100% elevator and right aileron hardover failure) 138
4.35 Controller gains 100% elevator and right aileron hardover failure . . . 139
4.36 Control surface deflections (100% elevator and right aileron hardover failure) 140
4.37 Pitch, bank, and heading angles (no failures) . . . 142
4.38 Airspeed and altitude . . . 143
4.39 Controller gains - No failure . . . 145

4.40 Pitch, bank, and heading angles (0 *rad* deflection) . . . 146
4.41 Airspeed, and altitude (0 *rad* deflection) . . . 147
4.42 Controller gains 0 *rad* deflection failure . . . 149
4.43 Pitch, bank, and heading angles (50% hardover elevator failure) . . . 150
4.44 Airspeed and altitude (50% hardover elevator failure) . . . 151
4.45 Controller gains 50% hardover elevator failure . . . 153
4.46 Control surface deflections (50% hardover) . . . 154
4.47 Pitch, bank, and heading angles (100% hardover elevator failure) . . . 155
4.48 Airspeed and altitude (100% hardover elevator failure) . . . 156
4.49 Controller gains 100% hardover elevator failure . . . 158
4.50 Control surface deflections (100% hardover) elevator failure . . . 159
4.51 Pitch, bank, and heading angles (0 *rad* right aileron deflection failure) . . . 160
4.52 Airspeed, and altitude (0 *rad* right aileron deflection failure) . . . 161
4.53 Controller gains 0 *rad* right aileron deflection failure . . . 163
4.54 Control surface deflections (0 *rad* right aileron deflection failure) . . . 164
4.55 Pitch, bank, and heading angles (50% hardover right aileron failure) . . . 165
4.56 Airspeed, and altitude (50% hardover right aileron failure) . . . 166
4.57 Controller gains 50% hardover right aileron failure . . . 168
4.58 Control surface deflections (50% hardover right aileron failure) . . . 169
4.59 Pitch, bank, and heading angles (100% hardover right aileron failure) . . . 170
4.60 Airspeed and altitude (100% hardover right aileron failure) . . . 171
4.61 Controller gains 100% hardover right aileron failure . . . 173
4.62 Control surface deflections (100% hardover right aileron failure) . . . 174
4.63 Pitch and bank angles . . . 175
4.64 Pitch and roll controller effort . . . 176
4.65 Pitch and bank angles (0 *rad* elevator failure) . . . 177
4.66 Pitch and roll control inputs (0 *rad* elevator failure) . . . 177
4.67 Controller gains 0 *rad* elevator failure . . . 178
4.68 Altitude, and pitch and bank angles(50% hardover) elevator failure . . . 179
4.69 Pitch and roll control inputs (50% hardover) elevator failure . . . 180
4.70 Controller gains 50% hardover elevator failure . . . 181
4.71 Control surface deflections (50% hardover) elevator failure . . . 182
4.72 Pitch and bank angles (0 *rad* right aileron deflection failure) . . . 183
4.73 Pitch and roll control inputs (0 *rad* right aileron) deflection failure . . . 184
4.74 Controller gains 0 *rad* right aileron failure . . . 185
4.75 Pitch and bank angles (50% right aileron hardover failure) . . . 186
4.76 Pitch and roll control inputs (50% right aileron hardover failure) . . . 186
4.77 Controller gains 50% right aileron hardover failure . . . 187
4.78 Control surface deflections (50% hardover right aileron failure) . . . 188

4.79 Pitch and bank angles (100% right aileron hardover failure) 189
4.80 Pitch and roll control inputs (100% right aileron hardover failure) 190
4.81 Controller gains 100% right aileron hardover failure 191
4.82 Control surface deflections (100% hardover right aileron failure) 192
4.83 Pitch and bank angles (0 *rad* elevator and right aileron deflection failure . . 193
4.84 Pitch and roll control inputs (0 *rad* elevator and right aileron deflection failure) 194
4.85 Controller gains 0 *rad* elevator and right aileron failure 195
4.86 Control surface deflections (0 *rad* elevator and right aileron deflection failure) 196
4.87 Altitude, and pitch and bank angles (50% elevator and right aileron hardover) 198
4.88 Pitch and roll control inputs (50% elevator and right aileron hardover failure) 199
4.89 Controller gains 50% elevator and right aileron hardover failure 200
4.90 Control surface deflections (50% elevator and right aileron hardover failure) . 201
4.91 Pitch, bank, and heading angles 203
4.92 Airspeed and altitude 204
4.93 Controller gains 206
4.94 Pitch, bank, and heading angles (0 *rad* deflection) 207
4.95 Airspeed and altitude (0 *rad* deflection) 208
4.96 Controller gains 0 *rad* deflection 210
4.97 Pitch, bank, and heading angles (50% hardover elevator failure) 211
4.98 Airspeed and altitude (50% hardover elevator failure) 212
4.99 Controller gains 50% hardover elevator failure 214
4.100 Control surface deflections (50% hardover elevator failure) 216
4.101 Pitch, bank, and heading angles (0 *rad* right aileron deflection failure) 217
4.102 Airspeed and altitude (0 *rad* right aileron deflection failure) 218
4.103 Controller gains 0 *rad* right aileron deflection failure 220
4.104 Controller surface deflections (0 *rad* right aileron deflection failure) 222
4.105 Pitch, bank, and heading angles (50% hardover right aileron failure) 223
4.106 Airspeed and altitude (50% hardover right aileron failure) 224
4.107 Controller gains 50% hardover right aileron failure 226
4.108 Control surface deflections (50% hardover right aileron failure) 227
4.109 Pitch, bank, and heading angles (100% hardover right aileron failure) 228
4.110 Airspeed and altitude (100% hardover right aileron failure) 229
4.111 Controller gains 100% hardover right aileron failure 231
4.112 Control surface deflections (100% hardover right aileron failure) 233

List of Tables

1.1 A Brief Overview of Fixed-wing UAV Control 11
1.2 A Brief Overview of Fixed-wing UAV Fault-Tolerant Control 20

2.1 Ultra Stick Parameters as obtained from literature and used in this study . . . 28
2.2 Atmospheric Properties 40
2.3 Ultrastick UAV Aerodynamic Coefficients 56

3.1 Controller PID Gains 64
3.2 Optimised Controller PID Gains 69
3.3 Optimization Parameters 79
3.4 Objective Function Parameters 79
3.5 Control Allocation Parameters 80

4.1 Inner loop simulation test cases 107
4.2 Middle loop simulation test cases 108

Nomenclature

Symbol	Description	Units
α	Angle of attack	rad
β	Sideslip angle	rad
δ_a	Aileron deflection	rad
δ_e	Elevator deflection	rad
δ_f	Flap deflection	rad
δ_r	Rudder deflection	rad
ϕ	Roll angle	rad
γ	Flight path angle	rad
$\gamma_{\alpha r}^{r}$	Three-index Boltzmann multipliers	N/A
μ	Degrees of freedom	N/A
π_μ	Quasi-coordinates	N/A
θ	Pitch angle	rad
θ_T	Engine thrust pitch angle	rad
ρ_0	Air density at sea level	kg/m^3
ω	Angular velocity (subscript refers to system being used)	rad/s
ω_μ	Quasi-velocities	m/s
$\omega(s)$	Instantaneous angular velocity	rad/s
ω_p	Rotation speed of propeller	rad/s
ψ	Yaw angle	rad
ψ_T	Engine thrust yaw angle	rad
ψ_w	Radial velocity vector	m/s
A_{Ref}	Aerodynamic reference from firewall	m
b	Wing span	m
$\bar{c}$	Mean aerodynamic chord	m
C_D	Drag coefficient	N/A
C_L	Lift coefficient	N/A
C_m	Pitching moment coefficient	N/A

C_n	Yawing moment coefficient	N/A
C_p	Power coefficient	N/A
C_r	Rolling moment coefficient	N/A
C_T	Thrust coefficient	N/A
C_y	Side force coefficient	N/A
CG	Centre of gravity	m
D	Downburst column diameter	m
f_h	Intensity of the downdraft	N/A
f_r	Intensity of the horizontal shear	N/A
F_p	Propeller thrust	N
g	Acceleration due to gravity	m/s^2
h	Height above sea level	m
$\widehat{I}_\omega(s)$	Tensor of inertia	N/A
I_{motor}	Moment of inertia of rotating motor body	kgm^2
$I_{propeller}$	Moment of inertia of propeller	kgm^2
I_T	Engine moment of inertia	$kg.m^2$
I_x	Moment of inertia about x-axis	$kg.m^2$
I_y	Moment of inertia about y-axis	$kg.m^2$
I_z	Moment of inertia about z-axis	$kg.m^2$
I_{xy}	Product of inertia about x and y axes	$kg.m^2$
I_{xz}	Product of inertia about x and z axes	$kg.m^2$
I_{yz}	Product of inertia about y and z axes	$kg.m^2$
J	Advance ratio	Nm
L	Environmental Lapse Rate	K/m
L	Moment about x-axis (subscript refers to system being used)	Nm
m	Mass	kg
M	Molar mass	kg/mol
M	Moment about y-axis (subscript refers to system being used)	Nm
N	Moment about z-axis (subscript refers to system being used)	Nm
P	Roll rate	rad/s
P_0	ISA pressure at sea level	Pa

Q	Pitch rate	rad/s
R	Yaw rate	rad/s
R	Radial distance from the microburst center	m
R	Universal gas constant	J/mol
S_{REF}	Wing area	rad
S_x	Static moment of inertia about x body axis	$kg.m^2$
S_y	Static moment of inertia about y body axis	$kg.m^2$
S_z	Static moment of inertia about y body axis	$kg.m^2$
t	Time	s
T^*	Kinetic energy of the system	N/A
T_0	ISA Temperature at sea level	K
T_{motor}	Output torque at motor shaft	Nm
T_p	Propeller torque	Nm
U	Translational velocity of aircraft in the x body axis direction	m/s
V	Linear relative velocity (subscript refers to system being used)	m/s
V	Translational velocity of aircraft in the y body axis direction	m/s
$V(s)$	Velocity of centre of mass with respect to inertial reference frame	m/s
$\overline{W}$	Linear wind velocity vector	m/s
W	Translational velocity of aircraft in the z body axis direction	m/s
W_R	Radial wind component	m/s
X	Force in the x-axis (subscript refers to system being used)	N
Y	Force in the y-axis (subscript refers to system being used)	N
Z	Force in the z-axis (subscript refers to system being used)	N

Abbreviations

Abbreviation	Description
AFTCS	Active fault-tolerant control system
APC	Advanced precision composites
BL	Brushless
BP	Backpropagation
BPNN	Backpropagation neural network
CA	Control allocation
CLAW	Control law
C-MRAC	Combined/Composite model reference adaptive controller
COTS	Commercial-off-the-shelf
DOF	Degree of freedom
ESN	Echo state network
FDD	Fault detection and diagnosis
FDI	Fault detection and identification
FL	Feedback linearisation
FLC	Fuzzy logic controller
FNN	Feedforward neural networks
FTC	Fault-tolerant control
FTCS	Fault-tolerant control system
GA	Genetic algorithm
GS	Gain-scheduled
GS-PID	Gain-scheduled proportional, Integral & derivative
H	H infinity
IAE	Integral absolute error
ID	Identification
ITAE	Integral time absolute error
ISA	International Standard Atmosphere

LMS	Least mean squares
LQG	Linear quadratic gaussian
LQR	Linear quadratic regulator
MIT	Massachusetts Institute of Technology
M-MRAC	Modified model reference adaptive controller
MPC	Model predictive control
MRAC	Model reference adaptive controller
NMPC	Nonlinear model predictive controller
NN	Neural network
PFTCS	Passive fault-tolerant control system
PID	Proportional, Integral & derivative
PSO	Particle swarm optimisation
RA	Re-allocation
RC	Radio-controlled
RBF	Radial based function
RBFNN	Radial based function neural network
RNN	Recurrent neural networks
RWNN	Recurrent wavelet neural network
SCG	Scaled conjugate gradient
SHL	Single hidden layer
SMC	Sliding mode control
UAV	Unmanned aerial vehicle
ZN	Ziegler-Nichols

1 Introduction

1.1 General Background and Motivation

An unmanned aerial vehicle (UAV) can be defined as a vehicle that uses aerodynamic forces and moments to produce lift and flies autonomously or remotely. The first autopilot was designed by the Sperry Corporation in 1912. Since then, much research has been performed on aircraft control and many different flight control methods have been applied, thus increasing UAV's capability [Sadeghzadeh, (2015)], [Higgs, (2014)].

UAVs are often favoured for missions that are too dull, dirty, or dangerous for manned aircraft. In military applications, such as surveillance and reconnaissance, suppression of enemy air defence and high-value asset recovery scenarios carried out by UAVs can be considered. UAVs are also used in a limited but growing number of civil applications, such as fire-fighting, surveillance of pipelines, telecommunication relay (hot spots), mining, crop dusting, power line monitoring, and etc. [Sadeghzadeh, (2015)]. Figures 1.1 to 1.3 show some examples of operational fixed-wing UAVs.



Figure 1.1: MQ-1C Gray Eagle UAV [RP Defense, (2013)].



Figure 1.2: Phantom Ray Stealth Unmanned Combat Aerial Vehicle (UCAV) [Mick, (2010)].



Figure 1.3: IAI Heron [Defense Industry Daily Staff, (2011)]

In the aviation industry, the safety of aircraft passengers and environmental effects have always been an important issue. In history, a number of aircraft disasters occurred due to damaged or faulty aircraft, resulting in the aircraft becoming unstable or uncontrollable during flight. The occurrence of these types of accidents led to increasing interest in the emerging field of fault-tolerant control systems (FTCS) and fault detection and diagnosis (FDD). It has been suggested that a successful FDD system can assist crew members to detect and point out faults and failures with detailed information and solutions timeously to mitigate the problems of the aircraft during the initial stage following a failure. Where the crew encounters problems with the aircraft, and the FDD system points out the faults and failures in detail, and gives the solution timely, the crew may be able to respond appropriately. In the presence of a FDD system, the degree of danger of the faults or failures could be decreased, the probability of the safety of aircraft increased, and the lives of civilians protected [Khan, (2016)].

In order to reduce the risk to human life, both emerging military and civilian applications promote autonomous UAV use. The majority of the mission scenarios of interest, such as persistent surveillance, are inherently long-duration and require coordination of multiple co-operating UAVs in order to achieve the mission objectives. The importance of UAV flight fault-tolerant control, is currently a challenging topic of research [Sadeghzadeh, (2015)].

FTC is able to reduce the costs associated with lost UAVs as well as ensure that with the number of increased UAVs in operation a balance is maintained with safety. FTC systems enhance the performance of UAVs by increasing the safety and reliability of UAVs (these may result in the ability to operate manned and unmanned aircraft in the same airspace).

1.2 Problem Statement

This study is aimed at developing and simulating an optimised RBFNN-based FTC system for a fixed-wing UAV. The performance of this optimised FTC system will be compared to a PID controller and a backpropagation neural network (BPNN)-based controller. The proposed system is expected to address the issue of control actuator failure, partial loss of a control surface, and minor structural damage.

This problem may be solved with an intelligent controller that utilizes an optimised neural network. Neural networks replicate the learning process of the human brain. As in the case of the human brain, they may be trained to minimize the effect of the presented faults.

Due to the high control demand associated with a UAV, there are various types of faults which may occur such as actuator, sensor, processor and power supply failures. The type of fault which is the focus of this study is that of control actuator failure. This failure may be due to a malfunction of the servomotor, an oversight in assembly, or the disconnection of the servo in flight. Another area of concern is in the loss of a control surface, due to flutter, combat or other activities which require an extremely high level of manoeuvrability. These will be the focus of this research, however the tolerance of uncertainty within the aircraft structure will also be considered.

1.3 Research Objectives

The main objective of this research is to develop and simulate an intelligent fault-tolerant control system for a UAV. This objective will be implemented by the following sub-objectives:

- A mathematical model of a UAV will be implemented during simulation.
- An optimised NN controller will be created, which is tolerant to (but not limited to) the following faults:
  - actuator malfunction;
  - partial loss of control surface;
  - structural damage; and
  - aircraft model uncertainties.
- A comparison to other intelligent control technologies applied to UAVs of similar architecture and purpose will be investigated.

1.4 Literature Review

This section presents literature on various control schemes that have been used for UAVs.

1.4.1 UAV Control Schemes

On the subject of linear control schemes, the Linear Quadratic Regulator (LQR) is an optimal control approach based on closed-loop optimal control with linear state feedback or output feedback [Rahimi, Hajighasemi, & Sanaei, (2013)]. A LQR is used in [Valavanis, et al., (2008)], where a LQR-based path-following algorithm is presented and the control effort is computed using optimal control theory. This framework allows the aircraft to determine the minimal control effort required to minimise the cross-track error as well as cross-track error rate, and ensures that the UAV is close to the path through the use of a defined boundary. Figure 1.4 illustrates a Linear Quadratic Gaussian (LQG) control scheme.

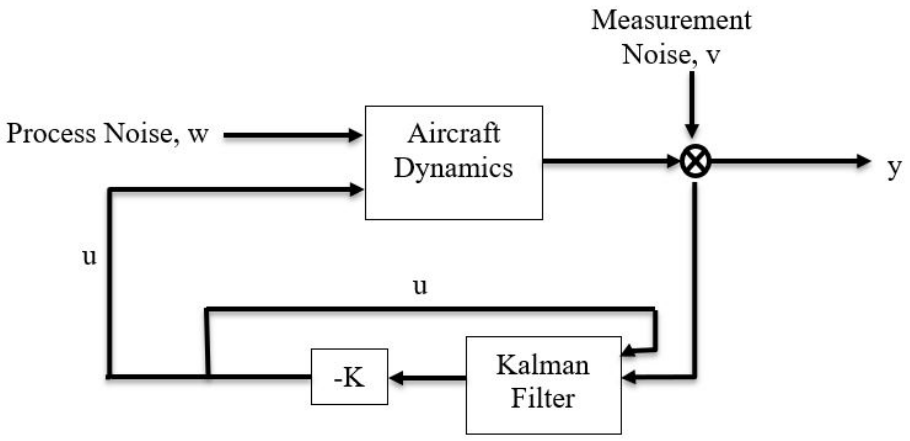


Figure 1.4: Linear Quadratic Gaussian Control Scheme [Huang & Jan, (2014)].

In comparison to the LQG control scheme, a proportional, integral, and derivative (PID) controller is a generic control loop feedback mechanism commonly used controllers by the industry due to its availability and widespread research. The controller attempts to minimize the process error by continually adjusting the inputs [Rahimi, Hajighasemi, & Sanaei, (2013)]. [Ghazzawi, (2011)] proposed a mathematical-based framework for designing robust PID controllers for a class of higher-order systems. The proposed strategy combines the deadbeat step response, the multi-step design system and model reduction techniques in a one-procedure design. The resulting control system improves time-response dynamic properties, ensures robustness against large parameter uncertainties, and presents a tuning tool with a reduced number of tuning parameters. Figure 1.5 illustrates a PID closed loop system.

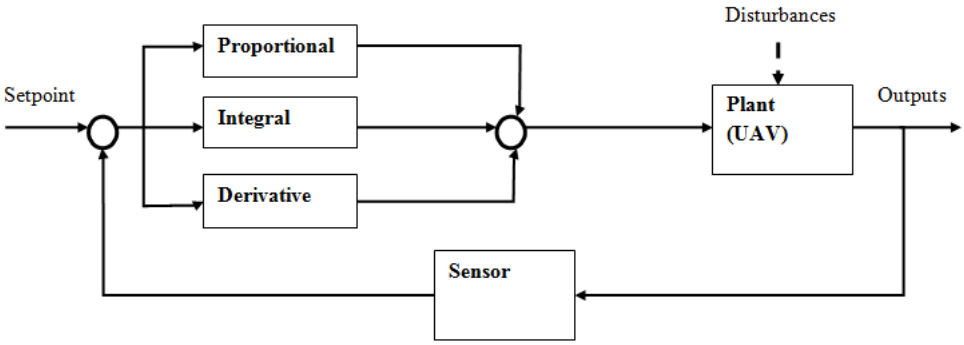


Figure 1.5: Closed-Loop PID [Rahimi, Hajighasemi, & Sanaei, (2013)].

In non-linear intelligent control schemes, sliding mode control (SMC) consists of a two-step procedure. The first step is the description of the control objective in terms of a state-space

surface; namely the sliding surface and the second step is controller design. In the first step the surface choice must be developed such that the system trajectories satisfy the performance specifications when the sliding variable lies on the sliding surface. In the second step the definition of a control action steers the state trajectories onto the sliding surface after a finite transient time [Norton, Khoo, & Kouzani, (2015)], [Hervas, et al.,(2014)].

Other forms of non-linear control include feedback linearisation, which transforms a nonlinear system into an equivalent linear system so that the system may be handled easily. Feedback linearisation (FL) is one of the most popular control methods for nonlinear systems [Voos, (2009)]. [Lee, Kim, & Sastry, (2009)] presents a FL controller that operates on a feedback linearisation method for integrated x-y-z control. Feedback linearisation controllers can be directly applied to nonlinear dynamics without linear approximations. They simplify the equation of system dynamics for the FL controller in order to avoid complex calculations involving repeated differentiation.

To increase the control capability one may use intelligent control schemes. A 'learning system' may be defined as one which possesses the capability to improve its performance over time by interaction with its environment. A learning control system is designed to ensure that a 'learning controller' has the ability to improve the performance of the closed-loop system by generating command inputs to the plant and using feedback information from the plant [Lin & Chung, (2014)].

A 'learning system' is a system that exhibits all of the following characteristics [Lin & Chung, (2014)]:

- Learning;
- Adaptability;
- Robustness across problem domains;
- Improving efficiency (over time and/or space);
- Information compression (data to knowledge); and
- Extrapolated reasoning.

The major categories under intelligent control are:

1. Neural network-based control schemes;
2. Fuzzy logic-based control schemes; and

3. Evolutionary algorithm-based control schemes.

Neural networks (NN) are composed of simple elements operating in parallel. These elements are inspired by biological systems. The neural network function is determined largely by the connections between elements [Lin & Chung, (2014)].

Neural networks can be classified as feedforward neural networks (FNNs) and recurrent neural networks (RNNs). A FNN is capable of approximating arbitrary continuous functions accurately. However, FNN is a static mapping and is unable to represent a dynamic mapping without the aid of tapped delays. RNNs, which comprise of both feedforward and feedback connections, have better capabilities than the FNNs [Dadian, Bhandari, & Raheja, (2016)].

RNNs have a feedback loop that allows them to capture past information of the networks and adapts rapidly to sudden changes of the control environment. RNNs have the ability to deal with time-varying input or output through their own natural temporal operation. For this ability, the structure of the neural network is simplified. A robust recurrent neural network controller is derived by assuming that the bound of residual approximation error is known in [Dadian, Bhandari, & Raheja, (2016)]. The application of an echo state network (ESN) for the nonlinear control of a fixed-wing UAV was also presented. The data required for the network training was generated using a validated flight dynamics model of the UAV. The ESN was used for both offline and online training. The offline training realized the inversion required for the feedback linearisation. The online training was used to reduce the inversion error that results due to modelling deficiencies. The data required for the online training is obtained in real-time from the FlightGear flight simulator while the UAV was flying in the simulation. The FlightGear model of the UAV was based on the flight validated nonlinear model of the UAV.

In comparison to NNs, a fuzzy logic controller (FLC) is mutually exclusive to conventional dynamic model-based controllers, for which a mathematical model is not required in order to control the plant. It is applicable to both linear and nonlinear systems. Fuzzy control is a control method of applying expert knowledge to control a plant without having detailed information about the plant. A FLC has three main components [Yadav & Gaur, (2014)]; [Rahimi, Hajighasemi, & Sanaei, (2013)]:

1. Fuzzifier, which converts the input signal into a fuzzy signal;
2. Fuzzy interface engine, which processes the fuzzified signal using decision rules; and
3. Defuzzifier, which converts the fuzzy controller output signal to a signal used for the control input signal to the system model.

Figure 1.6 shows a simplistic overview of the fuzzy controller.

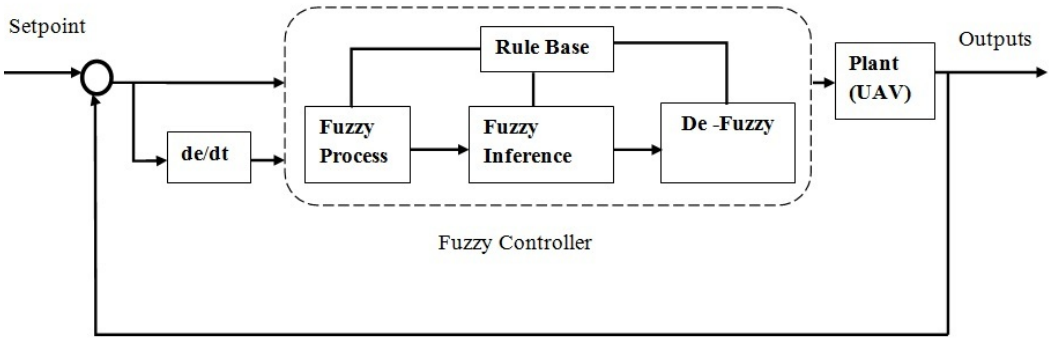


Figure 1.6: Fuzzy Logic Controller [Rahimi, Hajighasemi, & Sanaei, (2013)].

Another form of intelligent control is the Model Predictive Control (MPC) technique, which is based on the idea of computing an optimal sequence of control signals that optimise a certain cost function, possibly with state and control constraints. Once the first inputs are applied and the aircraft moves, the prediction and optimization steps are again repeated, and a new control sequence is obtained, thus closing the loop. [Gavilan, Vazquez, & Camacho, (2015)] implemented a path-following guidance strategy based on MPC which computes optimised values of airspeed, flight path angle, and bank angle. These are commanded to a low-level attitude control system in order to make the UAV follow a precomputed reference trajectory (given by straight flight segments). The nonlinear optimisation is implemented by iteratively solving a set of linearised problems around a nominal control sequence. In addition to this, the algorithm is initialized by a complementary low-level guidance law.

In comparison to the previous form of MPC, [Garcia & Keshmiri, (2011)] present a Nonlinear Model Predictive Controller (NMPC) that is designed to perform guidance, navigation, and control of a fixed-wing UAV in waypoint trajectory tracking. The inherent nonlinear nature of the aircraft together with the limited desired operation ranges of both control and state variables make the use of an on-line control law computation more appropriate. The MPC theory presented allows an optimal on-line solution, when sufficient on-board computational power is available. Using nonlinear guidance laws for both the lateral and longitudinal planes (such as extended Kalman filter estimations) and a complete nonlinear model of the UAV, the NMPC generates on-line optimal or suboptimal control signals to keep the UAV tracking the desired trajectory.

To optimise a system one may utilise a genetic algorithm (GA) which simulates the genetic and evolutionary process of organisms in the natural environment and forms a kind of global optimisation probability search algorithm. Genetic algorithm is a group of type operation as the operation is performed on all individuals in the group as an object. Heredity, selection, recombination, and mutation, are the four main operating operators of genetic algorithm. They form the so-called genetic operation [Fiuzy, Haddadnia, & Mashhadi, (2013)]. Currently, genetic algorithm has been widely used in PID control. Because the genetic algorithm itself has a strong global optimisation capability and robustness characteristic making it suitable for complex nonlinear control systems [Feng, Wang, & Qu, (2015)]. Figure 1.7 illustrates the basic flow of a GA.

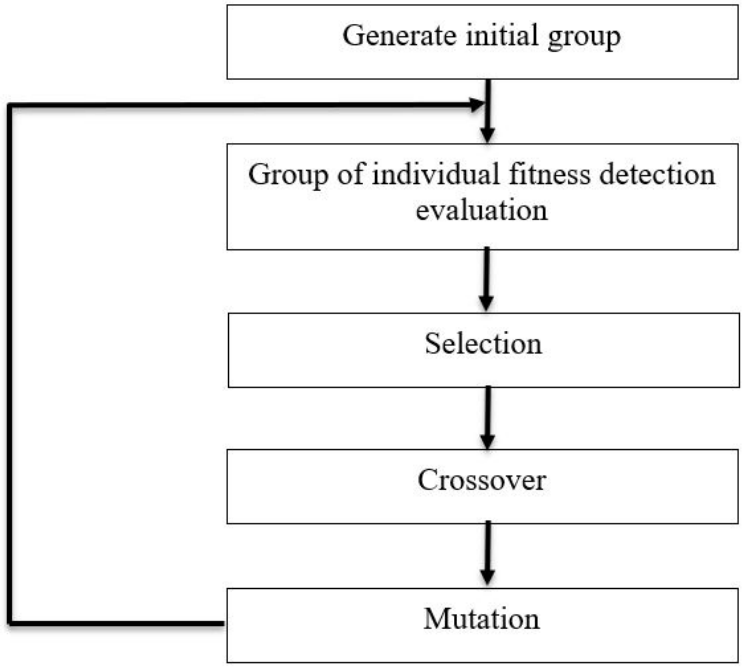


Figure 1.7: Basic Flow of Standard Genetic Algorithm [Feng, Wang, & Qu, (2015)].

One may also use adaptive control such as in [Valavanis, et al., (2008)], whereby the NN selected for adaptive control were the radial basis function (RBF) and single hidden layer (SHL) neural network. Theoretically, the neural network can model all bounded and piecewise continuous nonlinear dynamics present in the UAV as well as all uncertainties in the plant, provided the appropriate control law and NN structure have been selected, and the neurons are properly tuned. The performance characteristics of a longitudinal flight controller were analysed when the SHL and RBF neural architectures were used in a model reference adaptive controller. The NN served as the adaptive element responsible for approximating the inversion error resulting from unmodelled higher-order dynamics in the aircraft plant model. When the adaptive element was correctly used, the Model Reference Adaptive Controller (MRAC)

effectively provided precise and accurate responses to commanded changes that were both sudden and large.

In [Lin & Chung, (2014)] an intelligent control system was developed by using a recurrent wavelet neural network (RWNN). The developed RWNN was used to mimic an ideal controller. In addition to this the adaptive tuning laws of RWNN were derived based on the sliding-mode approach. The developed RWNN control system was applied to UAV motion control for achieving desired trajectory tracking.

To improve the capability of a neural network one may introduce deep learning into the network. Deep learning is built around a hypothesis that a deep, hierarchical model can be exponentially more efficient at representing some functions than a shallow one. It has been shown that a deep sum-product network may require exponentially less units to represent the same function compared to a shallow sum-product network [Pascanu, et al., (2014)]. The depth is defined in the case of feedforward neural networks as having multiple nonlinear layers between inputs and outputs. At each time step, when computation is carried out by a RNN there are certain transitions which are not deep. The different ways in which a RNN, may be made deep is by adding a nonlinear layer between these transitions. The hidden-to-hidden transition may be made deeper by having one or more intermediate nonlinear layers between two consecutive hidden states. At the same time, the hidden-to-output function can be made deeper, by plugging multiple intermediate nonlinear layers between the hidden state and the output [Pascanu, et al., (2014)].

1.4.1.1 Current Trend in Fault Tolerant Control Strategies

Table 1.1 presents a brief overview of the current trend in fixed-wing UAV control.

Table 1.1: A Brief Overview of Fixed-wing UAV Control

Classification of Control	Control Method	A Few References in Literature	Integrated with Intelligent Control
Classical Control	PID	(Rahimi, Hajighasemi, & Sanaei, 2013)	(Gripp & Sampaio, 2014)
Optional Control	LQR	(Rahimi, Hajighasemi, & Sanaei, 2013)	
Robust Control	H_∞	(Akyurek, Gunes, Elbir, Kasnakoglu, & Kaynak, 2017)	
Nonlinear Control	Feedback Linearisation	(Zhou, Yin, Wang, & Wang, 2014)	
	Lyapunov	(Girish, Emilio, Jonathan, & Hugh, 2015)	
	Sliding Mode	(Norton, Khoo, & Kouzani, 2015)	
	Backstepping	(Girish, Emilio, Jonathan, & Hugh, 2015)	
	Dynamic Inversion	(Siyang, Guang-Xun, Quan, & Kai-Yuan, 2014)	
Intelligent Control	Neural Network Based	(Lin & Chung, 2014)	(Valavanis, Oh, & Piegl, 2008)
	Fuzzy Logic Based	(Yadav & Gaur, 2014), (Rahimi, Hajighasemi, & Sanaei, 2013), (Norton, et al., 2015)	(Gomez & Jamshidi, 2011)
	Predictive Control	(Viudez-Moreiras, Martin, & Martin-Sanchez, 2014), (Gavilan, Vazquez, & Camacho, 2015), (Garcia & Keshmiri, 2011)	
	Evolution Algorithm Based	(Girish, et al., 2015)	

1.4.2 Fault-Tolerant Control Strategies

A FTC system (FTCS) is defined as a control system that possesses the ability to accommodate system component failures automatically. They are capable of maintaining overall system stability and acceptable performance in the event of such system component failures. FTCS are also known as self-repairing, reconfigurable or self-designing control systems [Ma, (2011)]. Existing FTCS can be divided into two groups [Ma, (2011)], passive FTCS (PFTCS), which do not include controller reconfiguration; and Active FTCS (AFTCS), which integrates controller reconfiguration based on the outputs of a fault detection and diagnosis (FDD) module. The difference between PFTCS and AFTCS is that the controllers in PFTCS are designed to be robust against a class of presumed faults, whilst the controllers in AFTCS can be reconstructed on-line in case of faults occurring in the system. AFTCS is dependent on a FDD process to monitor system performance, to detect and identify faults in the systems [Ma, (2011)]. Fixed-wing UAVs are useful testbeds for fault-tolerant flight control research, due to their flight characteristics and stability factors. Fixed-wing UAVs are able to fly at higher speeds and altitudes, and their flight range and endurance are more than the ones for rotary-wing UAVs. They have lower noise emission and have a higher payload to weight ratio than their rotary-wing counter parts. In addition to this they have some redundant control surfaces already built into their design [Sadeghzadeh, (2015)].

In PFTCS, deviations of the plant parameters from their reference values or deviations of the actuators from their expected positions may be adequately compensated by a fixed robust feedback controller. PFTCSs are designed to tolerate a certain class of component faults without the need of on-line fault information. For such systems, the anticipated level of faults and the controller's structure and its parameters are constant during the normal and faulty case operations. However, if these deviations become excessively large and exceed the robustness limits, the controlled plant can become unstable. In addition, if deviations occur due to sensor inaccuracies, inevitable deviations from the reference command signals will occur [Tang, (2013)], [Sadeghzadeh, (2015)]. A PFTCS is shown in Figure 1.8.

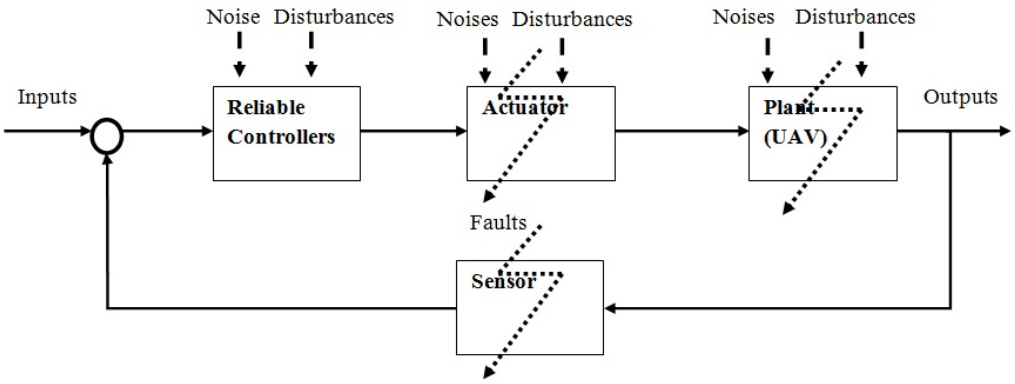


Figure 1.8: Passive Fault-Tolerant Control Scheme [Sadeghzadeh, (2015)]

On the other hand, an active (reconfigurable) FTCS is capable of detecting faults in the system (which are more challenging to detect than failures) and is able to adequately compensate for failures (which is more difficult to accommodate in addition to faults). An active fault-tolerant controller usually contains a separate module which is an fault detection and diagnosis (FDD) system that monitors the operation of the system. The FDD system provides information to a supervisory module about the magnitude of the fault, failure or damage. Once the fault is detected, located and identified (i.e., the type and magnitude of the fault), appropriate actions may be taken using the fault-tolerant controller to handle the irregularity in the system [Ma, (2011)], [Sadeghzadeh, (2015)]. Figure 1.9 illustrates an AFTCS.

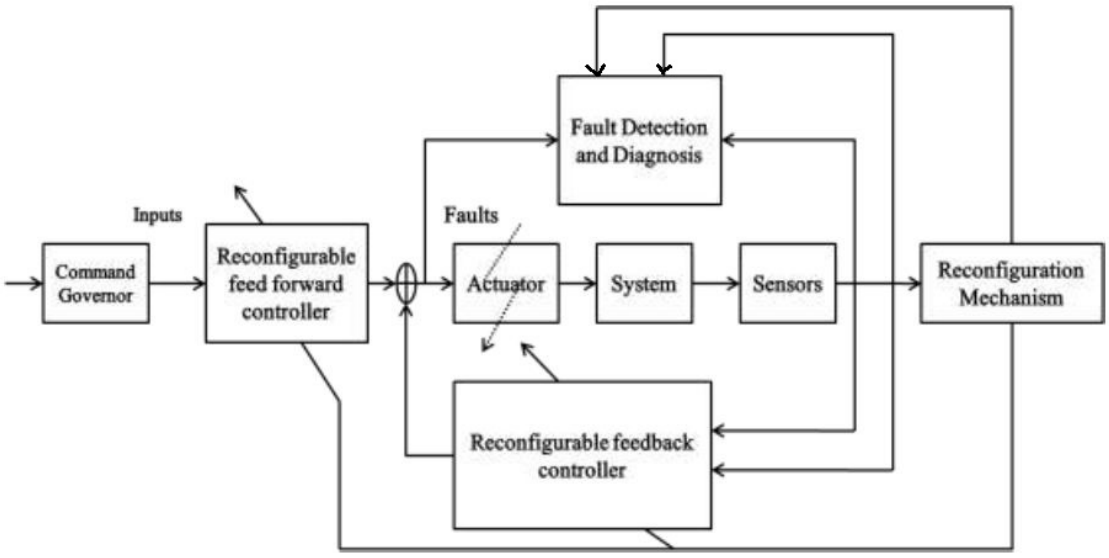


Figure 1.9: Active Fault-Tolerant Control System [Ma, (2011)].

One may find in a number of control systems that the UAV dynamics change with operating conditions. If the plants dynamic parameters vary without re-tuning the controller, the control system may become unstable or sluggish. This issue with variable process dynamics can be solved by utilizing a gain-scheduled (GS) technique (GS-PID). If the change of dynamics in a system/process with the operating condition is known, it is possible to change the parameters of the controller by monitoring the operating conditions of the process. This approach is called gain-scheduling as the scheme was initially utilised to accommodate changes in process gain [Milhim, Zhang, & Rabbath, (2012)], [Sadeghzadeh & Zhang, (2013)].

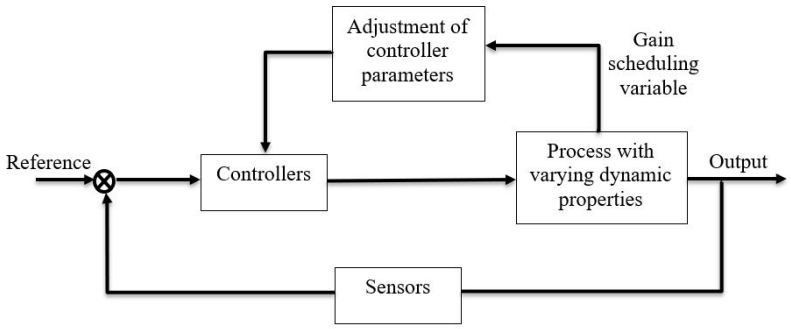


Figure 1.10: Gain Scheduling Control Loop [Sadeghzadeh & Zhang, (2013)].

There are different auto-tuning techniques for PID controllers (i. e., Ziegler-Nichols (ZN), loop shaping method, analytical tuning methods, pole placement, dominant pole placement, step and frequency response) of which the relay-based self-oscillating technique is one of the most commonly used techniques. The self-tuning algorithm gives the controller the capability to measure the system response to determine effective settings for PID control parameters [Gautam & Ha, (2013)], [Sadeghzadeh, (2015)].

As control requirement becomes more demanding, one may implement fuzzy logic control. In the case of fault tolerance, fuzzy logic control was implemented with control allocation in [Higgs, (2014)]. The model reference fuzzy logic adaptation algorithm allows the autopilot system to become intelligent. PID controllers were assigned to control pitch, roll, airspeed, altitude and heading. The adaptation algorithm adjusted the gains of each PID controller based on the feedback from the aircraft to ensure that the response of the aircraft matches the defined ideal model. [Hafez & Kamel, (2016)] proposed a decision algorithm based on a fuzzy logic controller. The cooperative UAVs would make their decision to reconfigure into a new geometric pattern based on the output of a decentralized fuzzy logic controller. Fuzzy logic control (FLC) has the ability to manage ill-defined processes that are characterised with poor knowledge of their underlying dynamics.

Other implemented methods include model reference adaptive control. In MRAC systems, the plant has a known structure, but the parameters are not known and the reference model specifies the desired response to the external command. The controller is parametrized and provides tracking while the adaptation mechanism is used to adjust parameters in the control law. In an MRAC controller, some parameters need to be optimized based on the system response for achieving the required output. The output of the reference model is considered as the ideal output of the system and the adjustment mechanism adjusts (forces) the controller parameters to match the process output with the ideal reference model output [Sadeghzadeh, (2015)]; [Girish, et al., (2015)].

There are various manners in which MRAC is implemented such as:

- Massachusetts Institute of Technology (MIT) rule: MIT rule is one of the early approaches to MRAC, its concept was to specify the desired command-to-output performance of a servotracking system using a differential or a difference equation (the reference model) that would define the ideal response of the system due to external commands. It has the disadvantage of not being able to guarantee closed-loop system stability [Lavertsky & Wise, (2013)].

- MRAC design based on Lyapunov stability theory: In order to generate MRAC based on Lyapunov stability theory the differential equation of the error, which is the difference between the output of the model and the output of the system, should be derived, this contains the adjustable parameters [Sadeghzadeh, (2015)];

- Model-following MRAC: This is based upon matching the response of a UAV to that of a reference model, or "model." In the case of perfect model following the reference model must have the same number of states as that of the UAV being controlled [Kaufman, (1998)];

- Hyperstability and passivity theory: This is applied to nonlinear or uncertain plants whose plant dynamics are so fast that identification or on-line computations make other approaches infeasible.

- Modified MRAC (M-MRAC): This may be implemented to improve the robustness of the MRAC. An example of this is the σ-modification scheme. The σ-modification scheme does not require any prior information on the system disturbance upper bounds. The adaptive law with the σ-modification is where s is a strictly positive constant. In essence, this modification adds damping to the ideal adaptive law [Lavertsky & Wise, (2013)].

- Adaptive PID Using Fuzzy Logic: This is the use of fuzzy logic supervision on PID controllers.

- Other approaches include Error model, Combined/Composite MRAC (C-MRAC) or Fuzzy Model Reference Learning Control [Jiang, et al, (2010)], [Sadeghzadeh, (2015)].

Figure 1.11 illustrates an MRAC structure.

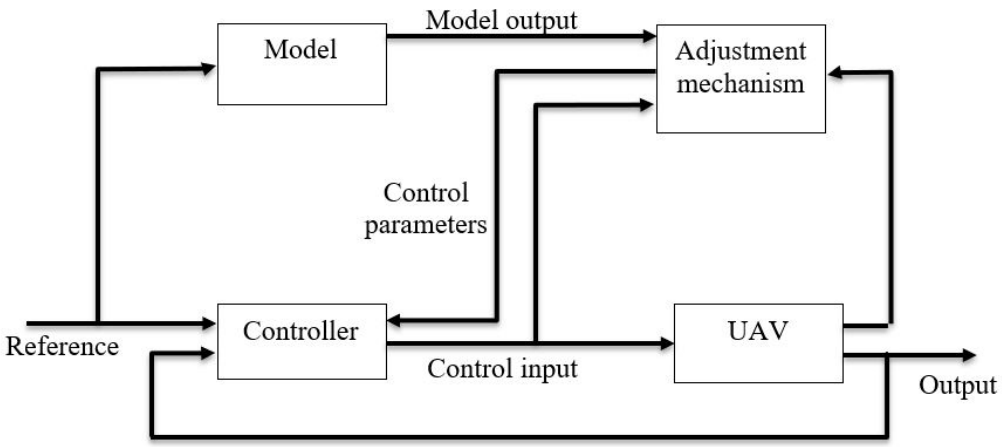


Figure 1.11: Model Reference Adaptive Control Structure [Girish, et al., (2015)]

To successfully implement FTC one must employ a fault detection and diagnosis system. [Khan, (2016)] stated that, one of the biggest challenges to designing fault tolerance is incorporating an appropriate fault detection and identification (FDI) unit. FDI is a very mature area of study and provides a number of powerful quantitative and qualitative modelling tools regarding any given fault. The FDI module is an integral part of the active FTC system and can mean the difference between complete system failure and system recovery. FDI is the most difficult aspect of FTC and due to the excessive costs involved and increase in system weight, hardware redundancy and full self-diagnosis equipment are not always viable solutions, particularly for UAVs. The role of the FDI system has been identified as the ability to gather information about the variations occurring in the system parameters (or in the system operating point) due to faults. There have been many kinds of FDI approaches developed namely: quantitative, qualitative and knowledge-based [Tang, (2013)]; [Khan, (2016)]. Figure 1.12 depicts some classification of faults in a controlled system.

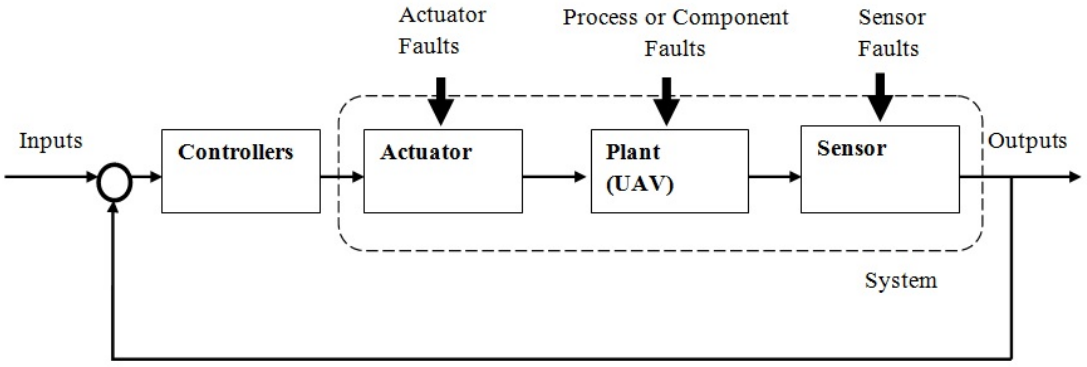


Figure 1.12: Classification of faults in a controlled system [Tang, (2013)].

The FTC system may also undergo system identification. For the specific case of UAVs, the process of system identification (ID) begins with designing the input signals. The pilot or autopilot performs manoeuvres designed to excite the UAVs dynamics. The signals given to the control surfaces and actuators are recorded. The actual deflection of the control surfaces are also be recorded. Various sensors record the current state of the UAV; acceleration, velocities, angular velocities, positions, aerodynamic angles, and angles relative to the earth's surface. From this training data a model of the UAV dynamics can be identified. If the model structure is already known then its parameters can be identified. This is referred to as parameter estimation, which is a more specific type of system ID. The model's response is validated using test data. This validation uses inputs to a set of test data separate from the training data used to generate the model. If the test data and identified model match, then the model can be used [Drozeski, (2005)].

There are five main elements to system ID [Hoffer, et al., (2014)]:

1. Input signals;
2. Data collection;
3. Selection of the model structure;
4. Selection of the system ID method; and
5. Optimization of the model using system ID method, model structure, and test data.

Model-based approaches are also utilised in FTC. In [Ma, (2011)], the Kalman filter is utilised for FDD. The Kalman filter is an efficient recursive filter that estimates the state of a linear dynamic system from a series of noisy measurements. In theory, the Kalman filter is an

estimator for a linear-quadratic problem, which is the problem of estimating the instantaneous states of a linear dynamic system perturbed by white noise using the measurements linearly related to the state and corrupted by white noise. As a typical efficient recursive filter, the Kalman filter is often used to estimate the states of a dynamic system under white noise disturbances. The Kalman filter can be applied to both linear and nonlinear systems [Zhang, (2015)].

The Kalman filter may be iterated between two steps as follows [Ma, (2011)]:

1. Time Update (Prediction), where the filter produces a time update at the current time step, based on the previous estimated states before obtaining any new measurements at the next time step using the system model; and
2. Measurement Update (Correction), whereby once the new measurements information arrives at the new time step, the filter will produce a more accurate state estimate. This is very close to the true value by combining the new measurements information with the result generated in the previous time update procedure based on an optimal adjustment.

An alternative to this control strategy may be reconfigurable control. Reconfigurable control takes different forms such as the control allocation and re-allocation (CA/RA) of one control objective to various control effectors [Qian, Jiang, & Xu, (2012)]. Control allocation may be applied to actuator faults in the determination of control deflections to create a desired moment about a particular axis. Reconfigurable control is generally accomplished by placing the control allocation algorithm between the controller and the actuators. Thus no change to the control law is required. Instead, the output of the control law (CLAW) is distributed through the control allocation mechanism to all of the control effectors [Higgs, (2014)]. For fault-tolerance purpose, the system uses the control allocation block when the system operates in fault-free conditions and the control re-allocation block at the time of fault occurrence in the actuators. Different constrained optimisation-based algorithms such as least-squares, linear programming and quadratic programming can be implemented for CA/RA purpose. The pseudo-inverse, fixed-point, direct control allocation and weighted least-squares are just few algorithms used [Sadeghzadeh, (2015)].

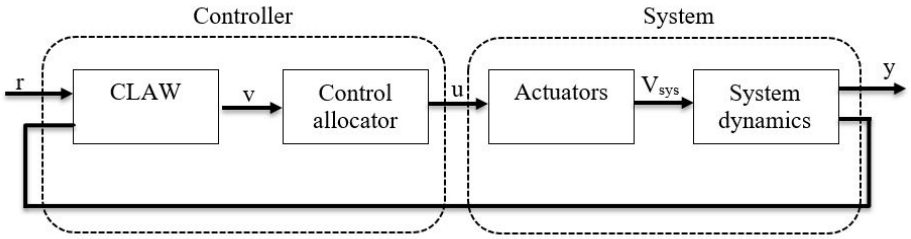


Figure 1.13: Control Allocation Block Diagram [Sadeghzadeh, (2015)].

When control allocation is used it may be further classified into optimal and non-optimal techniques. Optimal techniques consist of (but is not limited to) generalized inverse, daisy chaining, cascaded generalized inverse, and multi-pass inverse methods. Non-optimal techniques include facet searching, edge bisector, and optimization techniques [Drozeski, (2005)].

1.4.2.1 Challenges in Fault-Tolerant Control System Design

The following are some of the challenges encountered during fault-tolerant control system design:

- Incorporation of an appropriate fault detection and identification unit;
- Integrated FTC and FDI working within an uncertain environment should incorporate the constraints of the system and control variables, the cost function describing quality of control, and restrain the effects of disturbances and noises. These factors, severely increase the complexity of the problem stated while making the solutions more universal and reliable [Witczak & Pazera, (2016)];
- In observer-based FTC [Limnaios, (2014)]:
  - Distinguishing between the effects of continuous faults and mode transitions (may include discrete faults) on the system; and
  - Maintaining the stability of the overall control system in the presence of the two types of faults.
- Reduced requirement of online computation for the use of mass implementation;
- Ability to handle unknown inputs; and
- Detection of faults effectively and timeously [Mahmoud & Xia, (2013)].

1.4.2.2 Current Trend in Fault-Tolerant Control Strategies

Table 1.2 presents a brief overview of the current trend in fixed-wing UAV FTC.

Table 1.2: A Brief Overview of Fixed-wing UAV Fault-Tolerant Control

Fault Tolerant Control Schemes	**Adaptive Control**	Gain Scheduling	(Sadeghzadeh & Zhang, 2013) (Sadeghzadeh, 2015), (Milhim, Zhang, & Rabbath, 2012)		
		Auto-Tuning	(Gautam & Ha, 2013)		
		Fuzzy logic (FL) Controllers	(Hafez & Kamel, 2016)		
		Model Reference Adaptive Control (MRAC)	Artificial Intelligence	Adaptive PID Using FL	(Higgs, 2014)
				Fuzzy Model Reference Adaptive Control	(Jiang, et al., 2010)
			Conventional Techniques	MIT Rule	(Chamseddine, et al, 2011)
				Lyapanov	(Zhou, Chen, & Jiang, 2016), (Girish, et al., 2015)
				Backstepping	(Zheng, Gong, & Zhen, 2015)
		Model Predictive Control	(Khan, 2016)		
	Fault Detection, Diagnosis & Isolation	Direct Measurement	(Yoon, et al., 2015), (Bateman, Noura, & Ouladsine, 2011)		
		System Identification	(Hoffer, et al., 2015), (Hoffer, 2014), (Hoffer, et al., 2014), (Grymin, 2013)		
		Model Based Approaches	Kalman Filtering	(Chamseddine, Amoozgar, & Zhang, 2015), (Zhang, 2015), (Ma, 2011)	
			Fault Model Comparison	Neural Networks	(Chowdhary, et al., 2013)
				Hypothesis Testing & Statistical Methods	(Freeman & Balas, 2014)

	Reconfigurable Control	Sliding Mode Control	(Ismail, et al., 2014)		
		Multiple Model Switching	(Guo, Zhang, & Jiang, 2010)		
		Control Allocation	Optimal	Dynamic Control Allocation	(Peni, et al., 2014), (Qian, Jiang, & Xu, 2012), (Drozeski, 2005)
				Bisection Edge Searching	(Durham, et al., 2017)
			Non-Optimal	Daisy Chaining	(Higgs, 2014)
				Direct Allocation	(Johansen & Fossen, 2013)
				Generalized Inverse & Pseudo Weighted Inverse	(Girish, et al., 2015)
				Linear & Quadratic Programming	(Valavanis, Oh, & Piegl, 2008)
	Redundancy	Software	(Coopmans, Podhradsky, & Hoffer, 2015)		
		Hardware	Sensors	(Wang, et al., 2016), (Fuggetti, Ghetti, & Zanzi, 2015), (Ducard, et al., 2014)	
			Control Actuators	(Yu, Liu, & Zhang, 2016), (Ghazzawi, 2011)	
			Processor	(Hansen & Blanke, 2014)	

1.4.3 Identified Gaps in Literature

The following gaps have been identified in the literature:

- There has been a large amount of work done on PID control of UAVs, but there has been limited work done on the optimisation of a PID controller using an evolutionary algorithm for the purposes of FTC in a UAV;
- During the literature review no examples could be found on the use of radial basis function or backpropagation neural network applied for the FTC of fixed-wing UAVs; and
- Lastly, there is limited work done on the combination of different intelligent control schemes for the purposes of fault-tolerant control of fixed-wing UAV.

1.5 Research Scope, Strategy and Methodology

This research can be broken down into several tasks which may operate in a loop. These include literature review, system modelling, controller design, simulation and the generation of a report.

The project was originally defined as follows:

1. Development of simulation model: Here a 6 DOF flight dynamic model needs to be developed. The simulation of the proposed algorithm will be carried out in the MATLAB/Simulink environment.
2. Development of control system: The non-intelligent, RBFNN-based and BPNN-based fault-tolerant control systems will be developed and tested utilizing the simulations in MATLAB/Simulink.
3. A complete system simulation will be carried out in Matlab/Simulink.
4. The performance of the intelligent controller will be compared to a non-intelligent controller as well as an intelligent controller that has previously been implemented on the same system.
5. The results will be compiled into a dissertation.

Figure 1.14 displays the basic methodology to be followed when carrying out this research.

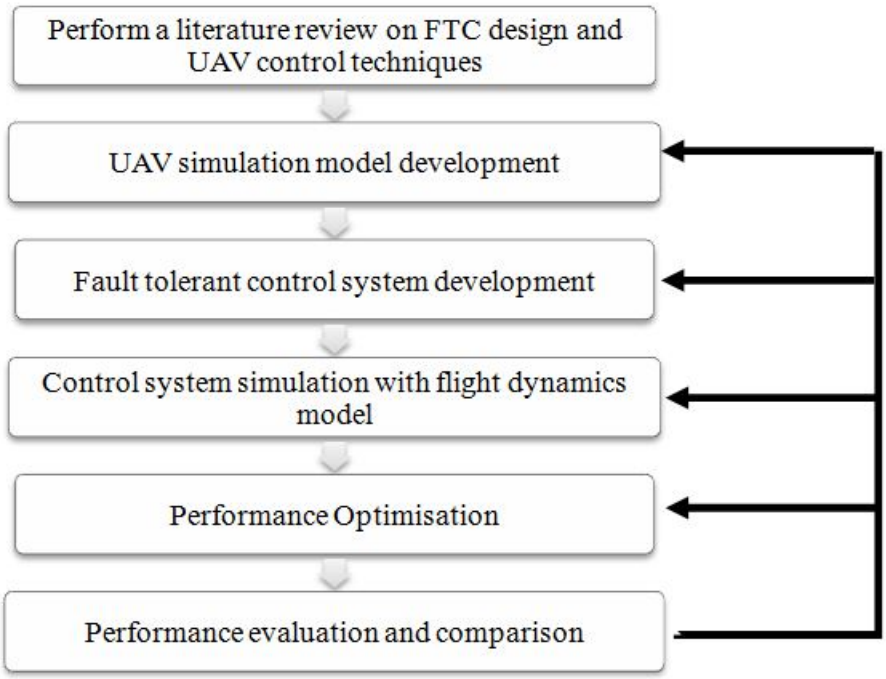


Figure 1.14: Work Flow Diagram

1.5.1 Proposed Controller Architecture

Figure 1.15 illustrates the proposed controller architecture.

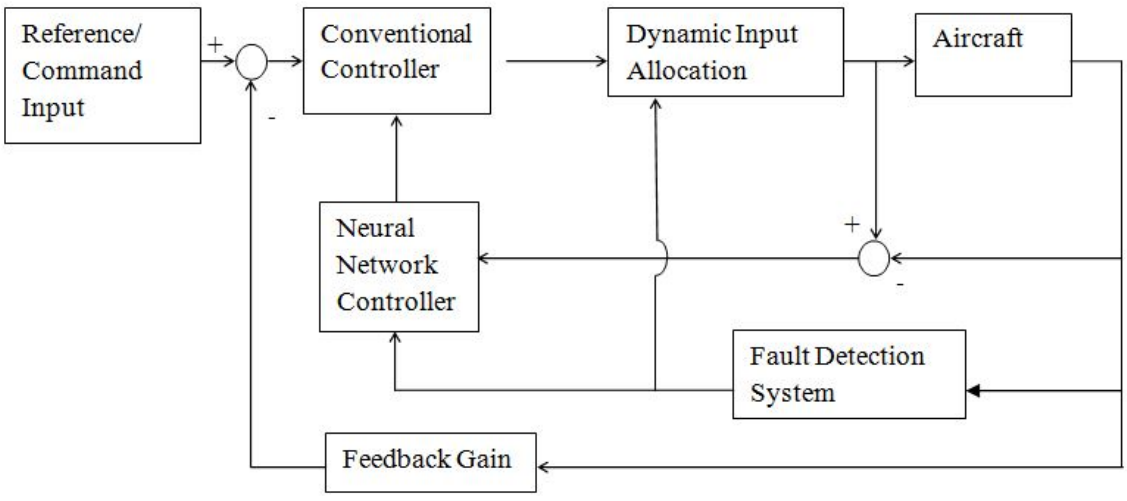


Figure 1.15: Proposed Controller Architecture

The control strategy proposed is aimed at combining several control architectures. The research began with applying a robust PID controller to the system. The PID controller was then optimised by means of an evolutionary algorithm to choose the optimal gains of the controller. A fault-tolerant control method such as reconfigurable control (in particular control allocation and re-allocation) was implemented on the system. Lastly, optimised radial basis function and backpropagation neural network was implemented along with the fault-tolerant control system.

1.6 Research Contributions

The contributions resulting from this research is as follows:

1. Neural network-based adaptive controllers for control of a fixed-wing UAV have been developed.
2. The ability of the control system to stabilise the UAV under varying actuator faults has been demonstrated in a simulated environment.
3. The control system's fault tolerance in the presence of microburst windshear has also been demonstrated in a simulated environment.
4. Three types of controllers have been studied in this work: a linear PID controller, a radial basis function neural network-based (RBFNN) controller, and a backpropagation neural network-based (BPNN) controller. These controllers were implemented with internal feedback loops and combined with control allocation to stabilise the UAV and enhance its performance in the presence of faults.

1.7 Organisation of Dissertation

The dissertation consists of six chapters. Figure 1.16 illustrates the structure of this dissertation. The first chapter, labelled introduction deals with a background description of unmanned aerial vehicles (UAVs), a review of control schemes applied to unmanned aerial vehicle design, and challenges associated with fault-tolerant control design. It also identifies gaps in literature based on controller design strategies that have been applied and presents the objectives of this research.

Chapter two presents the mathematical model components which form the basis of the simulation model. These components include the model of the UAV, the forces and moments

acting on the UAV, the propulsion dynamics, and environmental modelling which includes wind effects.

Chapter three presents PID controller architecture for the UAV. It also presents RBFNN-based and BPNN-based controller design as well as the optimisation methods (genetic algorithm and particle swam optimisation) implemented in this research.

In Chapter four, the designed controllers are evaluated by simulation with varying actuator fault cases with and without the presence of microburst windshear.

Chapter five presents an overview and discussion of the results of the simulation. Chapter six deals with the concluding remarks and further recommendations of the research.

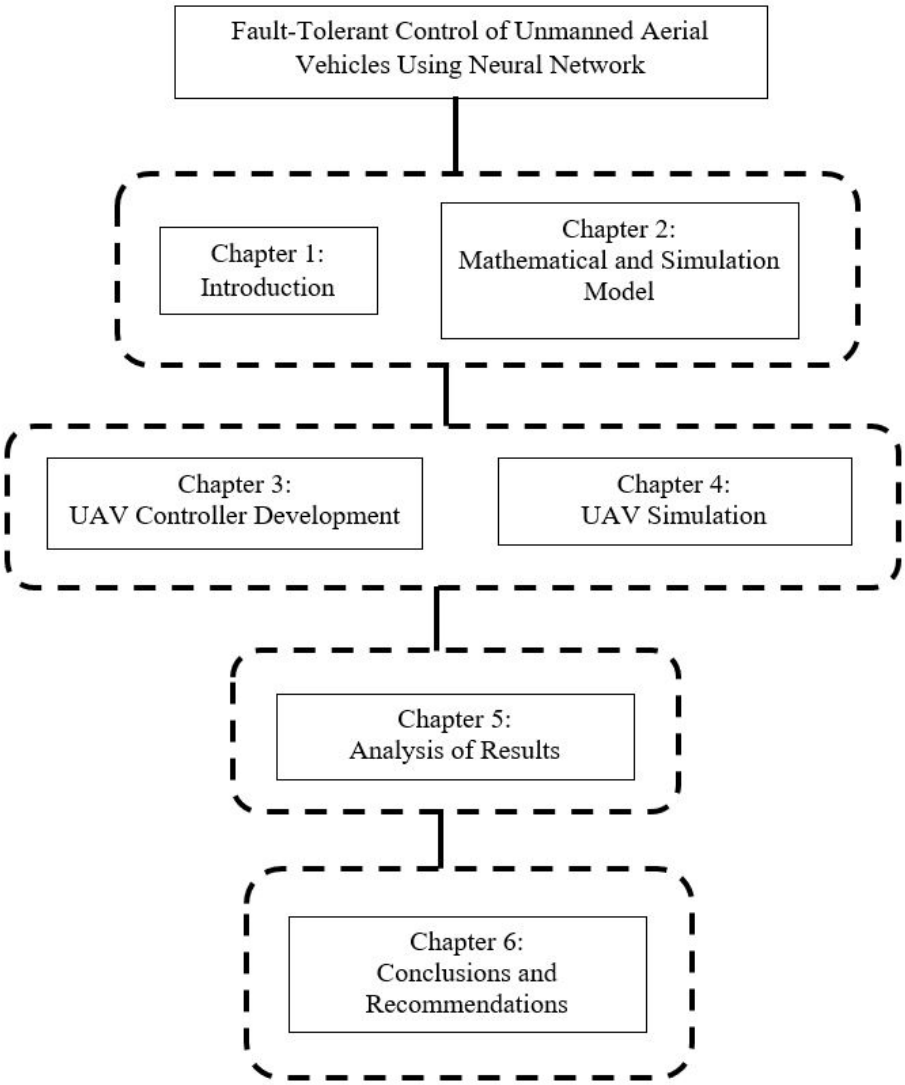


Figure 1.16: Structure of the Dissertation

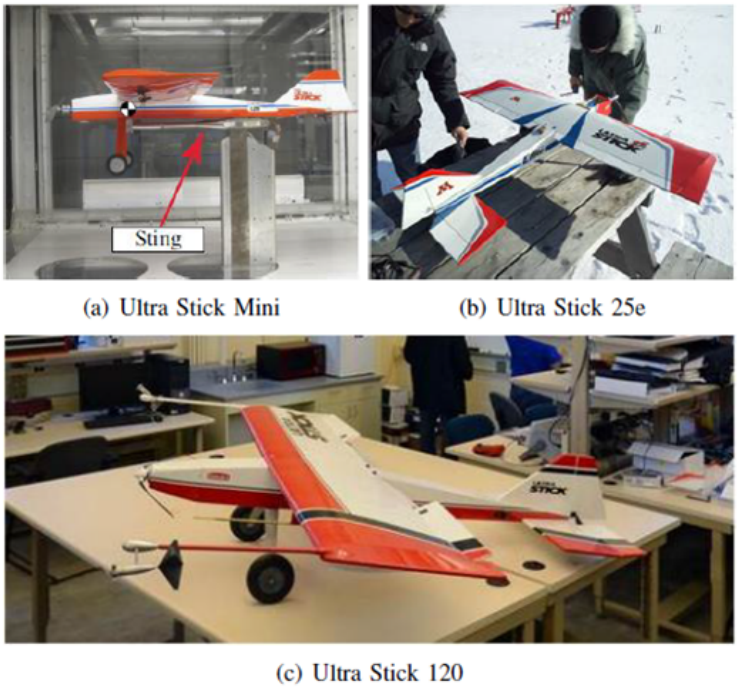
2 Mathematical and Simulation Model

2.1 Chapter Outline

In the control research field one typically includes the processes of plant identification, as well as physical system modelling. This is achieved by investigating the changes (due to plant dynamics) in the forces about some equilibrium point. The resultant set of equations are then linearised to enable a simpler way of obtaining the plant's response due to disturbances or changes in inputs. However, due to the complexity and nonlinearity of the UAV the small perturbation approach is not always sufficiently accurate. This results in the requirement of developing a model that takes nonlinearity into account to ensure the control algorithms developed are able to handle the faults within the system.

2.2 Ultra Stick Airframe

The UAV utilised in this research is a commercial-off-the-shelf (COTS) Radio-Controlled (RC) aeroplane model called the Ultra Stick, illustrated in Figure 2.1. The airframe was developed to be utilised for control research. In order for a simulation model to be useful, it should be able to predict the actual system response accurately. However, a mathematical model is simply an approximation of the actual physical system. Hence there is uncertainty in predicting the actual system response using the developed model. The airframe consists of a fuselage, with a tractor propeller configuration joined to a rectangular wing. The wing has a symmetrical aerofoil wing and both inboard flaps and ailerons. The tail of the UAV has elevators and a rudder.



(a) Ultra Stick Mini (b) Ultra Stick 25e

(c) Ultra Stick 120

Figure 2.1: Ultra Stick Aircraft [Paw, (2009)]

The advantage of using this airframe is in the wind tunnel and flight test data available as well as the redundancy in control surfaces. This leads to the ability to develop a high-fidelity flight dynamic model that takes non-linear effects into consideration. The airframe data used in this study was taken from the study done at the University of Minnesota. The important parameters and geometric specifications of the Ultra Stick are presented in Table 2.1 [Paw, (2009)], [Dorobantu, (2013)].

Table 2.1: Ultra Stick Parameters as obtained from literature and used in this study

Parameter	**Value**	**Symbol**
Mass (kg)	7.411	m
CG from firewall (m)	0.315	CG
Aero Ref from firewall (m)	0.320	A_{Ref}
Moment of inertia about x-axis ($kg.m^2$)	0.857	I_x
Moment of inertia about y-axis ($kg.m^2$)	1.010	I_y
Moment of inertia about z-axis ($kg.m^2$)	1.701	I_z
Product of inertia about x and y axes ($kg.m^2$)	0	I_{xy}
Product of inertia about x and z axes ($kg.m^2$)	-0.190	I_{xz}
Product of inertia about y and z axes ($kg.m^2$)	0	I_{yz}
Wing Chord (m)	0.434	c
Wing Span (m)	1.917	b
Wing Area (m^2)	0.769	S_{REF}

2.3 Flight Dynamic Simulation Methodology

A high-level overview of the simulation approach is depicted in Figure 2.2. The layout of this model is based on the layout of the nonlinear aircraft model block diagram from Aerospace Blockset in Matlab. The approach undertaken is to initialise the aircraft at a given state, determine all the forces and the accelerations about all three axes over small time intervals. This data is fed back into the system to update the aircraft states. The states' results are then stored and the process is repeated until a predetermined time period has lapsed.

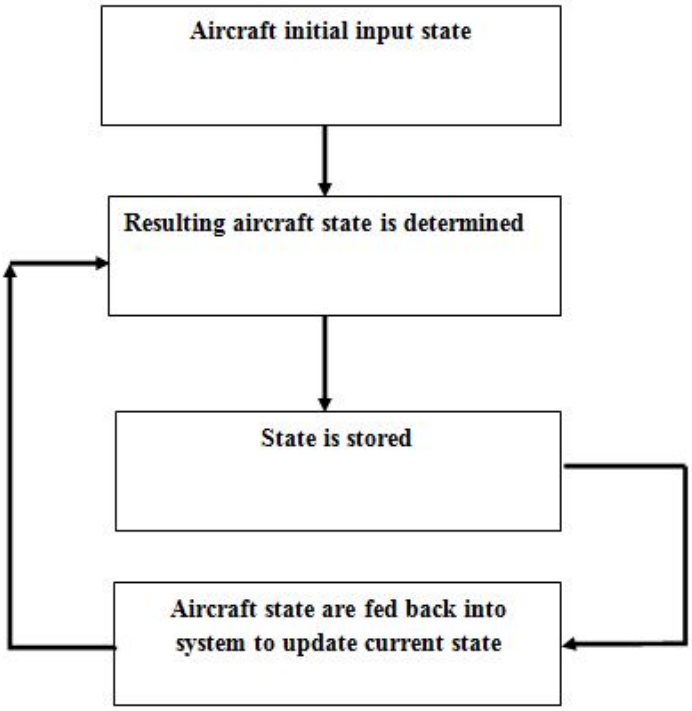


Figure 2.2: High-Level Simulation Methodology

2.3.1 Simulation Implementation

A simplified layout of the nonlinear aircraft model is illustrated is Figure 2.3. The main components of this simulation model which were created, based on the equations in the sections which follow, were the wind model (in particular the microburst windshear simulink model), the propulsion model, the control allocation model, the control system model, as well as the versions of the simulations which were used to investigate the effects of the microburst windshear.

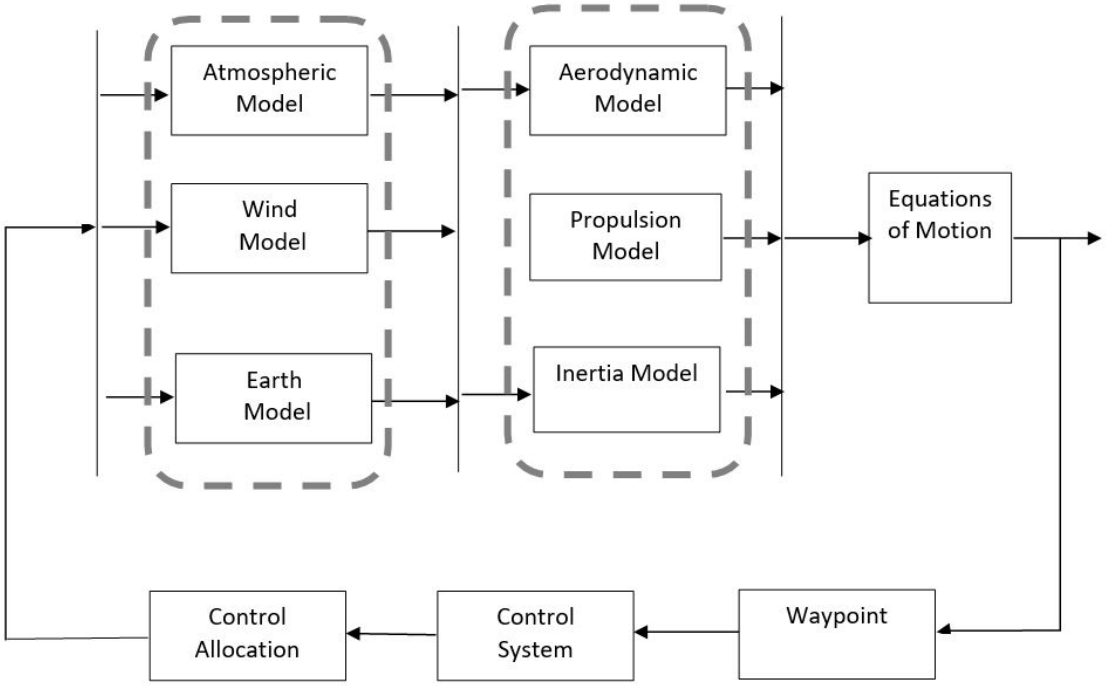


Figure 2.3: Simplified layout of nonlinear 6-DOF aircraft model

2.4 Coordinate Systems

There are three main coordinate systems which were used for this simulation.

- The Earth-fixed coordinate system, as illustrated in Figure 2.4;
- The body-fixed coordinate system, as illustrated in Figure 2.5; and
- The wind axis system.

Earth-fixed coordinate system:

The coordinate system assumes a flat Earth. This assumption was due to the modelling being done over relatively short time periods and low velocities. This study was not focused on detailed and accurate long distance navigation model, hence a flat Earth model was considered satisfactory. An additional assumption of the coordinate system being centred about an arbitrary location on the surface of the Earth was also used. The x-axis is positive in the North direction, y-axis positive in the East direction and the z-axis positive downwards. The orientation of the aircraft, relative to the Earth, will be described using this system. The system will also be used as a reference for navigational algorithms.

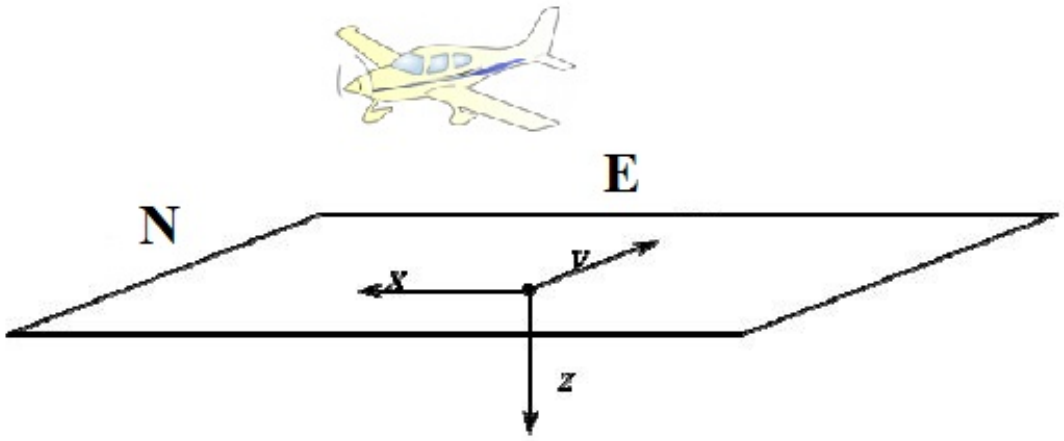


Figure 2.4: Conventions for earth-fixed coordinate system

Body-fixed coordinate system:

Another axis system that was used is the body-fixed system (see Figure 2.5). The aircraft's centre of gravity (CG) is the origin of this axis system. The x-axis is positive towards the nose of the aircraft, the y-axis is positive towards the starboard side of the aircraft, and the z-axis is positive towards the bottom of the aircraft. This is the primary axis system used to determine acceleration as well as the axis system the equations of motion are based on for the simulations presented in this dissertation. The body-fixed axis system is not the only primary frame of reference one could use as there are several possible choices, it is simply the one used in the simulations.

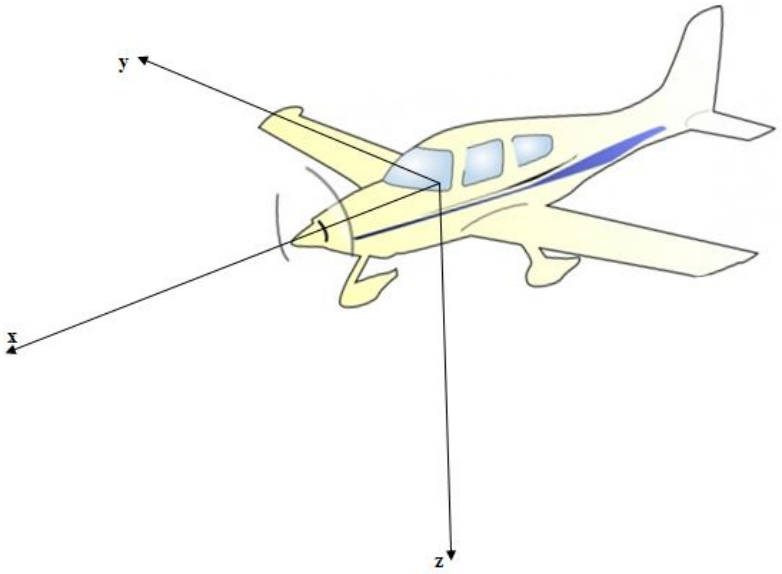


Figure 2.5: Convention for body-fixed coordinate system

Wind axis system:

The final axis system is the wind axis system which is used in the flight model. The is required as the wind tunnel data available is referenced to the wind axis. The wind axis system has a similar sign convention to that of the body-centred axis and is centred at the aircraft CG with the difference being that it is rotated away from the body axis by angle of attack as well as slideslip angle, resulting in a system aligned with the local flow direction.

The Euler angles are illustrated in Figure 2.6

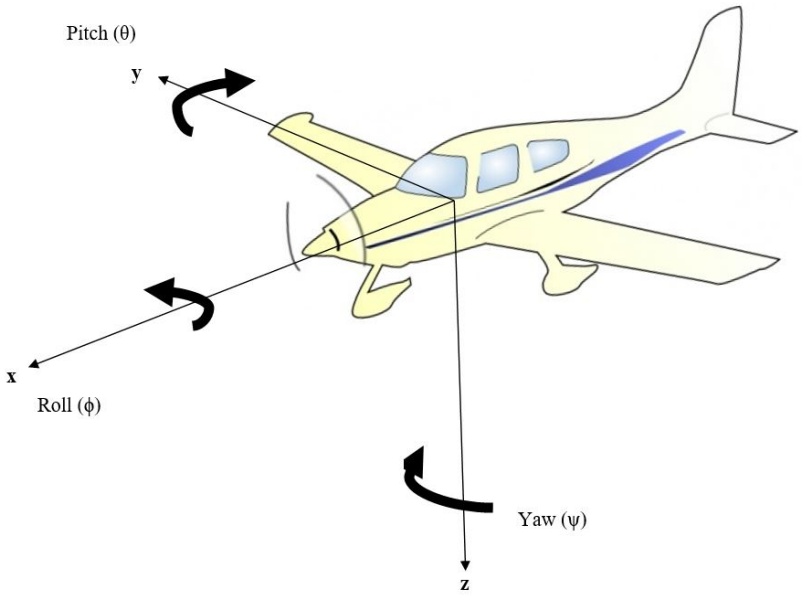


Figure 2.6: Euler angles

The use of these systems requires the transformation from one axis system to the other during flight simulation. These transformations are commonly used and formulated by rotations about each of the axes in turn. The order in which this occurs is imperative to the transformation. The transformations used are presented as Equations 2.1 and 2.2. For full derivation see [Cook, (2007)] and [Etkin & Reid, (1996)].

Transformation from body-fixed axis system to Earth-fixed axis system:

$$\begin{bmatrix} X_E \\ Y_E \\ Z_E \end{bmatrix} = \begin{bmatrix} \cos\theta\cos\psi & \sin\phi\sin\theta\cos\psi - \cos\phi\sin\psi & \cos\phi\sin\theta\cos\psi - \sin\phi\sin\psi \\ \cos\theta\sin\psi & \sin\phi\sin\theta\sin\psi + \cos\phi\cos\psi & \cos\phi\sin\theta\sin\psi - \sin\phi\cos\psi \\ -\sin\theta & \sin\theta\cos\theta & \cos\phi\cos\theta \end{bmatrix} \begin{bmatrix} X_b \\ Y_b \\ Z_b \end{bmatrix} \tag{2.1}$$

where ϕ, θ, and ψ are the roll, pitch and yaw angles; respectively. Transformation from wind-axis system to body-fixed system:

$$\begin{bmatrix} X_b \\ Y_b \\ Z_b \end{bmatrix} = \begin{bmatrix} \cos\alpha\cos\beta & -\cos\alpha\sin\beta & -\sin\alpha \\ \sin\beta & \cos\beta & 0 \\ \sin\alpha\cos\beta & -\sin\alpha\sin\beta & \cos\alpha \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \end{bmatrix} \tag{2.2}$$

where α and β are the angle of attack and sideslip respectively.

2.5 Derivation of the Equations of Motion

The dynamic equations of motion are derived by applying the Boltzmann-Hamel equations for mechanical systems with nonholonomic constraints [Pedro, (1992)]:

$$Q_\mu = \frac{d}{dt}\left(\frac{\partial T^*}{\partial \omega_\mu}\right) - \frac{\partial T^*}{\partial \pi_\mu} + \sum_{r=1}^{k}\sum_{\alpha=1}^{k} \gamma_{\alpha r}^{r} \frac{\partial T*}{\partial \omega_r}\omega_\alpha \tag{2.3}$$

where:
T^* is the kinetic energy of the system expressed in quasi-coordinate and quasi- velocities.
$\gamma_{\alpha r}^{r}$ are the three-index Boltzmann multipliers.
ω_μ are the quasi-velocities.
π_μ are the quasi-coordinates.
$\mu = 1, 2 \ldots k$ - degrees of freedom.

The system total kinetic energy T^*, based on Koenig's formula, is given by [Meirovitch, (2003)]:

$$T^* = \frac{1}{2}mV^2(s) + \frac{1}{2}\overline{\omega}(s)\widehat{J}_\omega(s)\overline{\omega}(s) \tag{2.4}$$

where:
m is the mass of the body;
$V(s)$ is the velocity of centre of mass with respect to inertial reference frame;
$\widehat{I}_\omega(s)$ is the tensor of inertia; and
$\omega(s)$ is the instantaneous angular velocity.

The kinetic energy of the aircraft in quasi-generalised velocity in spatial motion takes the following form:

$$\begin{aligned} T^* = {} & \frac{1}{2}\left[\left(U^2+V^2+W^2\right)m + P^2I_x + Q^2I_y + R^2I_z\right] - PQI_{xy} \\ & - QRI_{yz} - PRI_{xz} + S_x\left(VR - WQ\right) + S_y\left(WP - UR\right) + S_z\left(QU - VP\right) \end{aligned} \tag{2.5}$$

where U, V, and W are the aircraft velocity components along X, Y, and Z-body axes, respectively; P, Q, and R are the roll, pitch, and yaw rates respectively. I_x, I_y, and I_z are the

vehicle moments of inertia about x, y, and z axes respectively; I_{xy}, I_{xz}, and I_{yz} are the vehicle products of inertia. S_x, S_y and S_z are the static moments of inertia about x, y, and z-body axes.

Equation (2.5) is the general form of kinetic energy for any arbitrary 6-DOF rigid body expressed in any non-central reference frame or coordinate system $Oxyz$ rigidly connected with the body. Equation (2.3) can be written in scalar form as follows:

$$\frac{d}{dt}\left(\frac{\partial T^*}{\partial U}\right) + Q\frac{\partial T^*}{\partial W} - R\frac{\partial T}{\partial V} = X \tag{2.6}$$

$$\frac{d}{dt}\left(\frac{\partial T^*}{\partial V}\right) + R\frac{\partial T^*}{\partial U} - P\frac{\partial T^*}{\partial W} = Y \tag{2.7}$$

$$\frac{d}{dt}\left(\frac{\partial T^*}{\partial W}\right) + P\frac{\partial T^*}{\partial V} - Q\frac{\partial T^*}{\partial U} = Y \tag{2.8}$$

$$\frac{d}{dt}\left(\frac{\partial T^*}{\partial P}\right) + Q\frac{\partial T^*}{\partial R} - R\frac{\partial T^*}{\partial Q} + V\frac{\partial T^*}{\partial W} - W\frac{\partial T^*}{\partial V} = L \tag{2.9}$$

$$\frac{d}{dt}\left(\frac{\partial T^*}{\partial Q}\right) + R\frac{\partial T^*}{\partial P} - P\frac{\partial T^*}{\partial R} + W\frac{\partial T^*}{\partial U} - U\frac{\partial T^*}{\partial W} = M \tag{2.10}$$

$$\frac{d}{dt}\left(\frac{\partial T^*}{\partial R}\right) + P\frac{\partial T^*}{\partial Q} - Q\frac{\partial T^*}{\partial P} + U\frac{\partial T^*}{\partial V} - V\frac{\partial T^*}{\partial U} = N \tag{2.11}$$

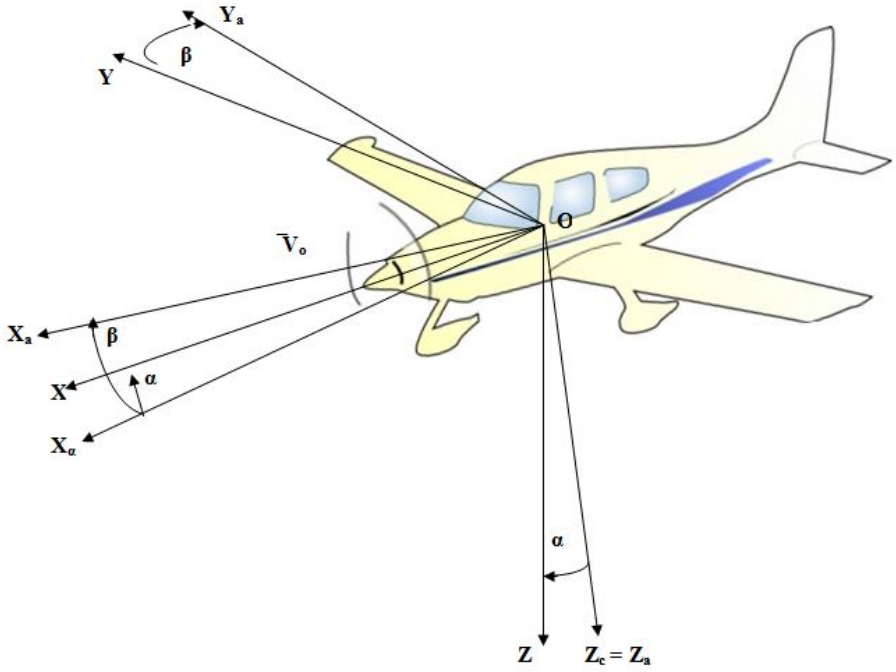


Figure 2.7: Body-fixed reference frames a) aerodynamic $Ox_ay_az_a$ b) arbitrary $Oxyz$

Vectors of external forces and moments of forces acting on the aircraft have the following components (see Figure 2.8):
Vector of external forces $\overline{F}$:

$$\overline{F} = X\bar{i} + Y\bar{j} + Z\bar{k} \tag{2.12}$$

where: X - Longitudinal force, Y - Side force, and Z - Lifting force
Vector of moments $\overline{M}_o$:

$$\overline{M}_o = L\bar{i} + M\bar{j} + N\bar{k} \tag{2.13}$$

The derivatives of kinetic energy with respect to components of linear velocity U, V, W and components of angular velocity P, Q, and R are:

$$\frac{\partial T^*}{\partial U} = mU - S_yR + S_zQ \tag{2.14}$$

$$\frac{\partial T^*}{\partial V} = mV + S_xR - S_zP \tag{2.15}$$

$$\frac{\partial T^*}{\partial W} = mW - S_xQ + S_yP \tag{2.16}$$

$$\frac{\partial T^*}{\partial P} = I_x P - I_{xy} Q - I_{xz} R + S_y W - S_z V \tag{2.17}$$

$$\frac{\partial T^*}{\partial Q} = I_y Q - I_{xy} P - I_{yz} R - S_x W + S_z U \tag{2.18}$$

$$\frac{\partial T^*}{\partial R} = I_z R - I_{yz} Q - I_{xz} P + S_x V - S_y U \tag{2.19}$$

Substituting Equations (2.14) to (2.19) into the system of Equations (2.6) to (2.11) and taking into consideration time derivatives, we obtain the following equations:

$$m\left(\dot{U} + QW - RV\right) - S_x\left(Q^2 + R^2\right) - S_y\left(\dot{R} - PQ\right) + S_z\left(\dot{Q} + PR\right) = X \tag{2.20}$$

$$m\left(\dot{V} + R - PW\right) - S_x\left(\dot{R} + QP\right) - S_y\left(P^2 + R^2\right) - S_z\left(\dot{P} - QR\right) = Y \tag{2.21}$$

$$m\left(\dot{W} + PV - QU\right) - S_x\left(\dot{Q} + PR\right) + S_y\left(\dot{P} + QR\right) - S_z\left(Q^2 + P^2\right) = Z \tag{2.22}$$

$$\begin{aligned} I_z\dot{P} - \left(I_y - I_z\right)QR - I_{xy}\left(\dot{Q} - PR\right) - I_{xz}\left(\dot{R} + PQ\right) - I_{yz}\left(Q^2 - R^2\right) + \\ S_x\left(\dot{W} + VP - UQ\right) + S_z\left(PW - UR - \dot{V}\right) = L \end{aligned} \tag{2.23}$$

$$\begin{aligned} I_y\dot{Q} - \left(I_z - I_x\right)RP - I_{xy}\left(\dot{P} + QR\right) - I_{yz}\left(\dot{R} - PQ\right) - I_{xz}\left(R^2 - P^2\right) - \\ S_x\left(\dot{W} + VP - UQ\right) + S_z\left(\dot{U} - VR + QW\right) = M \end{aligned} \tag{2.24}$$

$$\begin{aligned} I_z\dot{R} - \left(I_x - I_y\right)PQ - I_{yz}\left(\dot{Q} + PR\right) - I_{xz}\left(\dot{P} - RQ\right) - I_{xy}\left(P^2 - Q^2\right) + \\ S_x\left(\dot{V} - WP + RU\right) - S_y\left(\dot{U} - RV + WQ\right) = N \end{aligned} \tag{2.25}$$

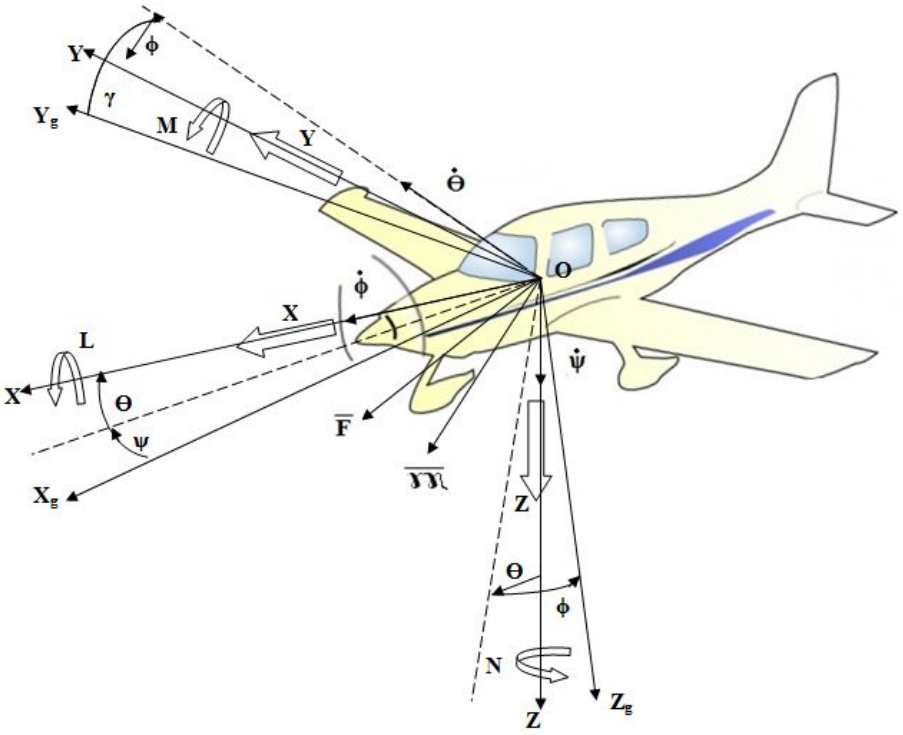


Figure 2.8: Forces and moments and their components in system $Oxyz$

For the purposes of this research the equations of motion will be about the aircraft's Centre of gravity. Geometric symmetry about the xz-plane is assumed, which will simplify the above equations to:

$$X = m\left(\dot{U} + QW - RV\right) \tag{2.26}$$

$$Y = m\left(\dot{V} + R - PW\right) \tag{2.27}$$

$$Z = m\left(\dot{W} + PV - QU\right) \tag{2.28}$$

$$L = I_z\dot{P} - (I_y - I_z)\,QR - I_{xz}\left(\dot{R} + PQ\right) \tag{2.29}$$

$$M = I_y\dot{Q} - (I_z - I_x)\,RP - I_{xz}\left(R^2 - P^2\right) \tag{2.30}$$

$$N = I_z\dot{R} - (I_x - I_y)\,PQ - I_{xz}\left(\dot{P} - RQ\right) \tag{2.31}$$

Vectors of external forces $\overline{F}$ with components X, Y, Z and moments of external forces $\overline{M}_o$ with components L, M, N acting on the aircraft come from thrust and gyroscopic moments from the engine, gravitational forces, aerodynamic forces, control forces as well as moments from the deflection of control surfaces i.e., aileron, elevator, flap and rudder.

Vector of external forces:

$$\overline{F} = \overline{F}_g + \overline{F}_a + \overline{F}_T + \overline{F}_{\delta i} \tag{2.32}$$

Vector of external moments:

$$\overline{M} = \overline{M}_g + \overline{M}_a + \overline{M}_T + \overline{M}_{\delta i} \tag{2.33}$$

where:

$\overline{F}_g, \overline{M}_g$ - Force and moment due to gravitational force;
$\overline{F}_a, \overline{M}_a$ - Force and moment due to aerodynamic forces;
$\overline{F}_T, \overline{M}_T$ - Force and moment due to thrust; and
$\overline{F}_{\delta i}, \overline{M}_{\delta i}$ - Force and moment due to control deflections: δ_r, δ_e, δ_f, δ_a - Rudder, elevator, flap and aileron deflections.

Equations (2.20) to (2.25) may be written in matrix form as:

$$\mathbf{A}\dot{x} + \mathbf{B}\mathbf{A}x = \mathbf{Q} \tag{2.34}$$

where:

$$x = [U,\ V,\ W, P,\ Q,\ R]^T \tag{2.35}$$

$$\mathbf{A} = \begin{bmatrix} m & 0 & 0 & 0 & 0 & 0 \\ 0 & m & 0 & 0 & 0 & 0 \\ 0 & 0 & m & 0 & 0 & 0 \\ 0 & 0 & 0 & I_x & 0 & -I_{xz} \\ 0 & 0 & 0 & 0 & I_y & 0 \\ 0 & 0 & 0 & -I_{xz} & 0 & I_z \end{bmatrix} \tag{2.36}$$

$$\mathbf{B} = \begin{bmatrix} 0 & -R & Q & 0 & 0 & 0 \\ R & 0 & -P & 0 & 0 & 0 \\ -Q & P & 0 & 0 & 0 & 0 \\ 0 & -W & V & 0 & -R & Q \\ W & 0 & -U & R & 0 & -P \\ -V & U & 0 & -Q & p & 0 \end{bmatrix} \tag{2.37}$$

$$\mathbf{Q} = [X,\ Y,\ Z,\ L,\ M,\ N]^T \tag{2.38}$$

2.6 Model of the Atmosphere

The atmospheric model implemented in the simulation is the International Standard Atmosphere (ISA) model. The primary function of this model is to account for the air density variation with altitude for the simulation. For this simulation, the atmosphere will only be modelled to the top of the Troposphere. The properties of the atmosphere used in this simulation are presented in Table 2.2 [ISO, (1975)].

Table 2.2: Atmospheric Properties

Parameter	**Value**	**Symbol**
ISA Temperature at S.L (K)	288.16	T_0
Environmental Lapse Rate (K/m)	-0.0065	L
ISA Pressure at Sea Level (Pa)	101325	P_0
Universal Gas Constant (J/mol)	8.31447	R
Molar Mass (Kg/mol)	0.0289644	M
Air Density at Sea Level (Kg/m^3)	1.225	ρ_0

Equation (2.39) is used to calculate the temperature at any given altitude:

$$T = T_0 + hL \tag{2.39}$$

where the height above sea level (h) is in metres.

The temperature equation is then utilised to determine air pressure as in Equation (2.40):

$$P = P_0 \left(1 + \frac{Lh}{T_0}\right)^{\frac{gM}{RL}} \tag{2.40}$$

The air density ratio is calculated using the following Equation (2.41):

$$\frac{\rho}{\rho_0} = \frac{PM}{RT} \tag{2.41}$$

The dynamic pressure of the air can be found using the air density ratio resulting in it affecting all aerodynamic force calculations.

2.6.1 Modelling of Wind Effects

The movement of the wind is described by a three-dimensional velocity vector of the wind field. The effects of this wind field on the dynamics of the aircraft are generated by the following:

- stationary wind; and
- components of the wind in the instantaneous position of the aircraft.

These gradients in principle are an interference acting on the aircraft, due to the coupling between the movement of the wind and the movement of the aircraft shows the additional combined process.

The aerodynamic forces and moments acting on the aircraft depend on the following:

- geometric shape of the aircraft;
- air density; and
- the relative motion between the flying object and surrounding mass of air i.e., from:

$$\overline{V}_a = \overline{V}_o - \overline{W} \tag{2.42}$$

$$\overline{\omega}_a = \overline{\omega} - \overline{\omega}_w \tag{2.43}$$

where:
$\overline{V}_a$ - linear relative velocity vector between the aircraft and surrounding air;
$\overline{V}_o$ - linear aircraft velocity vector;
$\overline{W}$ - linear wind velocity vector;
$\overline{\omega}_a$ - relative angular velocity between the aircraft and the air;
$\overline{\omega}$ - angular velocity vector of the aircraft; and
$\overline{\omega}_w$ - angular velocity vector of the wind.

In Figure 2.9 one observes that the velocity vectors are located in different reference systems and must therefore be transformed into references associated with the moving aircraft $Oxyz$. Equation (2.42) then takes the following form:

$$\begin{bmatrix} \cos\alpha\cos\beta \\ \sin\beta \\ \sin\alpha\cos\beta \end{bmatrix} \overline{V}_a = \begin{bmatrix} \cos\alpha_0\cos\beta_0 \\ \sin\beta_0 \\ \sin\alpha_0\cos\beta_0 \end{bmatrix} V_o - \begin{bmatrix} W_x \\ W_y \\ W_z \end{bmatrix}_b \tag{2.44}$$

The transformation of the wind velocity with the Earth during flight is of the form:

$$\left[\bar{W}\right]_b = L_{BE}\left[\bar{W}\right]_g \tag{2.45}$$

where:
L_{BE} - Matrix of transformation from the inertial reference system $O_1x_1y_1z_1$ to the aircraft body-fixed reference system $Oxyz$.

After substitution of the matrix L_{BE} Equation (2.45) becomes:

$$\begin{bmatrix} W_x \\ W_y \\ W_z \end{bmatrix}_b = \begin{bmatrix} \cos\theta\cos\psi & \sin\psi\cos\theta & -\sin\theta \\ \sin\phi\cos\psi\sin\theta - \sin\psi\cos\phi & \sin\phi\sin\theta\sin\psi + \cos\phi\cos\psi & \sin\phi\cos\theta \\ \cos\phi\cos\psi\sin\theta + \sin\psi\sin\phi & \cos\phi\sin\theta\sin\psi - \sin\phi\sin\psi & \cos\phi\cos\theta \end{bmatrix} \begin{bmatrix} W_x \\ W_y \\ W_z \end{bmatrix}_g \tag{2.46}$$

where:

$[W_x, W_y, W_z]_g^T$ - Wind velocity vector field in the inertial reference system in the Earth system.

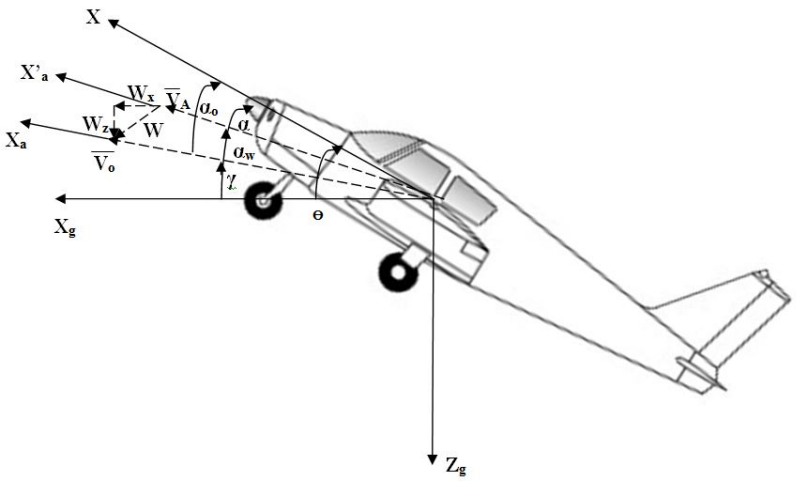


Figure 2.9: Aircraft velocity relative to wind V_A , aircraft velocity V_o and wind velocity W in the aircraft's plane of symmetry

We obtain the following geometric relations from Figure 2.9:

$$\alpha = \theta - \gamma - \alpha_w \tag{2.47}$$

where γ is the flight path angle and α_w is the angle of attack induced by the wind.

$$\sin \alpha_w = \frac{1}{V_A} \left(W_{xg} \sin \gamma + W_{zg} \cos \gamma \right) \tag{2.48}$$

Figure 2.10 results in the following:

$$\chi = \psi + \beta - \beta_w \tag{2.49}$$

where β_w is the sideslip angle induced by the wind and χ is the angle that the aircraft travels over the ground, relative to North.

$$\sin \beta_w = \frac{1}{V_A} \left(-W_{xg} \sin \chi + W_{yg} \cos \chi \right) \tag{2.50}$$

The change of wind speed experienced by the aircraft is defined as follows:

$$\frac{d\bar{W}}{dt} = \frac{\partial \bar{W}}{\partial t} + \frac{\partial \bar{W}}{\partial r} \frac{d\bar{r}}{dt} \tag{2.51}$$

The first term $\frac{\partial \bar{W}}{\partial t}$ is usually ignored because the wind velocity field is assumed to be approximately constant (stationary) as the aircraft velocity is much greater than the wind velocity. This is applicable to quasi-steady and steady wind terms [Etkin, (1959)]. Equation (2.51) in the Earth reference system has the form:

$$\begin{bmatrix} \dot{W}_x \\ \dot{W}_y \\ \dot{W}_z \end{bmatrix}_g = \begin{bmatrix} W_{xx} & W_{xy} & W_{xz} \\ W_{yx} & W_{yy} & W_{yz} \\ W_{zx} & W_{zy} & W_{zz} \end{bmatrix}_g \begin{bmatrix} \cos \gamma \cos \chi \\ \cos \gamma \sin \chi \\ -\sin \gamma \end{bmatrix} V_o \tag{2.52}$$

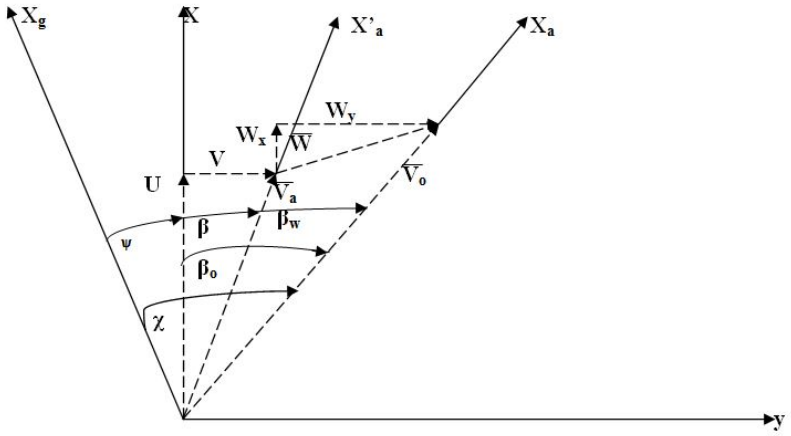


Figure 2.10: Velocities: Aircraft relative to wind $\bar{V}_A$, aircraft $\bar{V}_o$ and wind $\bar{W}$ on the horizontal plane

The differential equations describe the components of the wind velocity in the aircraft's instantaneous position. Elements of the matrix $W_{xx} = \frac{\partial W_x}{\partial x}$, $W_{xy} = \frac{\partial W_x}{\partial y}, \ldots$, etc. are the components of the local wind gradients. These are really noisy input variables acting on the aircraft. The method of modelling the effects of wind elements is incorrect by applying the "Taylor Hypothesis", of which the spatial distribution of wind velocity changes in them without considering the changing vector flight path. Vector of the wind speeds affect the acting aerodynamic forces and moments through the relationships in Equations (2.42) and (2.43) because the aerodynamic forces and moments depend directly on $\bar{V}_A$ and $\bar{\omega}_A$. Generally, very complicated non-linear functions involving density ρ, velocity $\bar{V}_A$, angle of attack α, sideslip angle β and other variables. Thus the aerodynamic force is determined by the relationship:

$$P_x^a = \frac{\rho V_A^2}{2} S\left(\bar{V}_A, \bar{\omega}_A, M_A, \alpha, \beta, \delta_e, i_t, \delta_f\right) \tag{2.53}$$

where i_t, and δ_f are the horizontal tailplane setting angle and flap deflection angle respectively.

The effect of wind described previously results from the impact of the local vector in the middle of the aircraft mass, considered as a point mass, only changes $\bar{V}_A$, α and β. This effect can be considered as an expression of a zero-order Taylor series of the influence of the wind [Brockhaus, (1987)]. The expression of the first row of that number of Taylor series expansion is the effect of the linear decomposition of the wind on the aircraft (Figure 2.11).

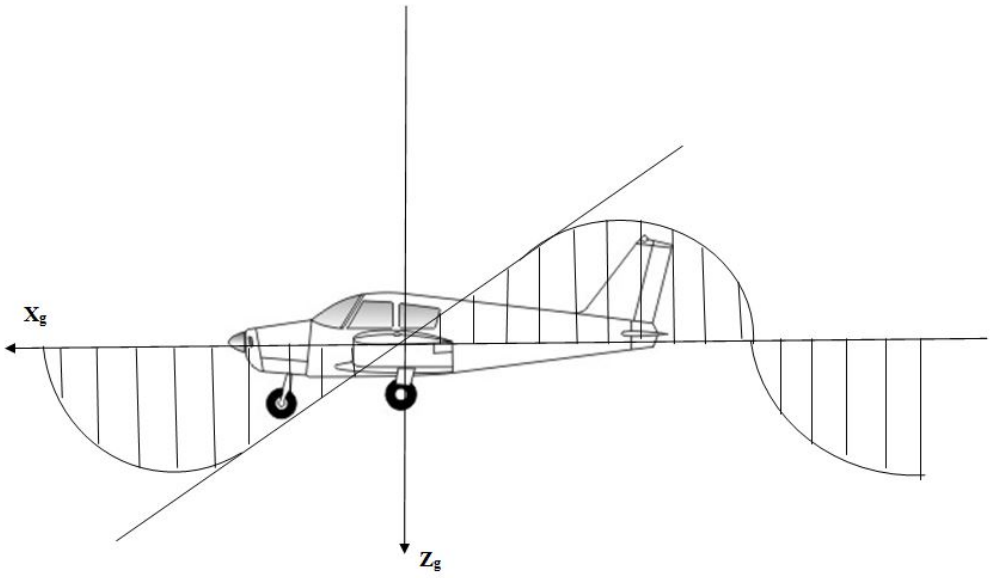


Figure 2.11: Sinusoidal wind field

The linear terms, which are determined by the local components of the wind gradient, can be transformed to the equations of motion plane as the effects of a rotating mass of air. This leads to a relatively simple model, because of the dependence on the derivative aerodynamic from the angular velocity of the aircraft being known.

The instantaneous wind angular velocity vector is given by:

$$\bar{\omega}_w = \begin{bmatrix} P_w \\ Q_w \\ R_w \end{bmatrix} = \frac{1}{2}\bar{V} \times \vec{W} = \frac{1}{2} L_{BE} \begin{bmatrix} i & j & k \\ \frac{\partial}{\partial x} & \frac{\partial}{\partial y} & \frac{\partial}{\partial z} \\ W_x & W_y & W_z \end{bmatrix} \tag{2.54}$$

Figure 2.12 presents the components of instantaneous angular wind velocity ω_w, depending on the wind gradient.

The diagonal elements of the wind gradient matrix, are not included in the equation, which results from the decomposition of the non-uniform air flow in the direction of the aircraft axis, as shown in Figure 2.13. Their effects are negligible and cannot be measured during flight, hence they are often omitted in the analysis [Brockhaus, (1987)], [Pedro, (2004)].

More accurate models should continue to develop the Taylor series expansion effects of the wind to higher-order terms. However, this requires a local derivative of the wind gradient, and greatly complicates the model. As a result of the effects of non-linear decomposition of the wind on the aerodynamic forces and moments not being known, the higher-order terms

may be neglected. The linear model is sufficient, provided the length of the wind's wave motion is 8-12 times bigger than the aircraft dimensions [Pedro, (1992)]. This condition is fulfilled for the approximation of the decomposition of the wind and quasi-stationary model aerodynamics. This means that the wind model must be limited to a frequency not greater than the frequency of brief periodic aircraft movement.

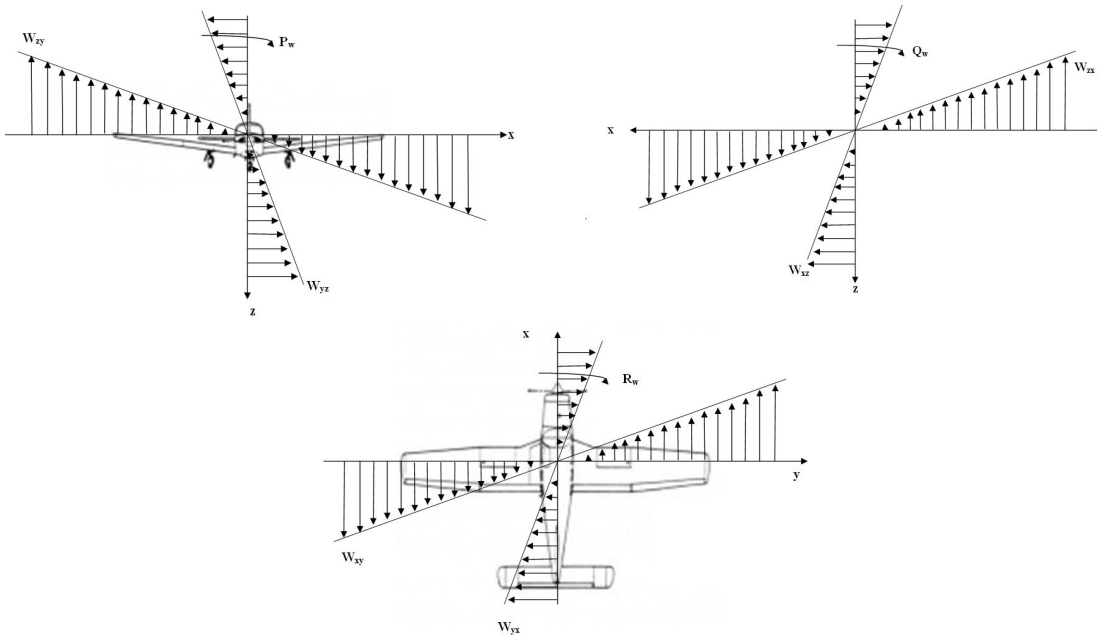


Figure 2.12: Angular and gradient wind velocity

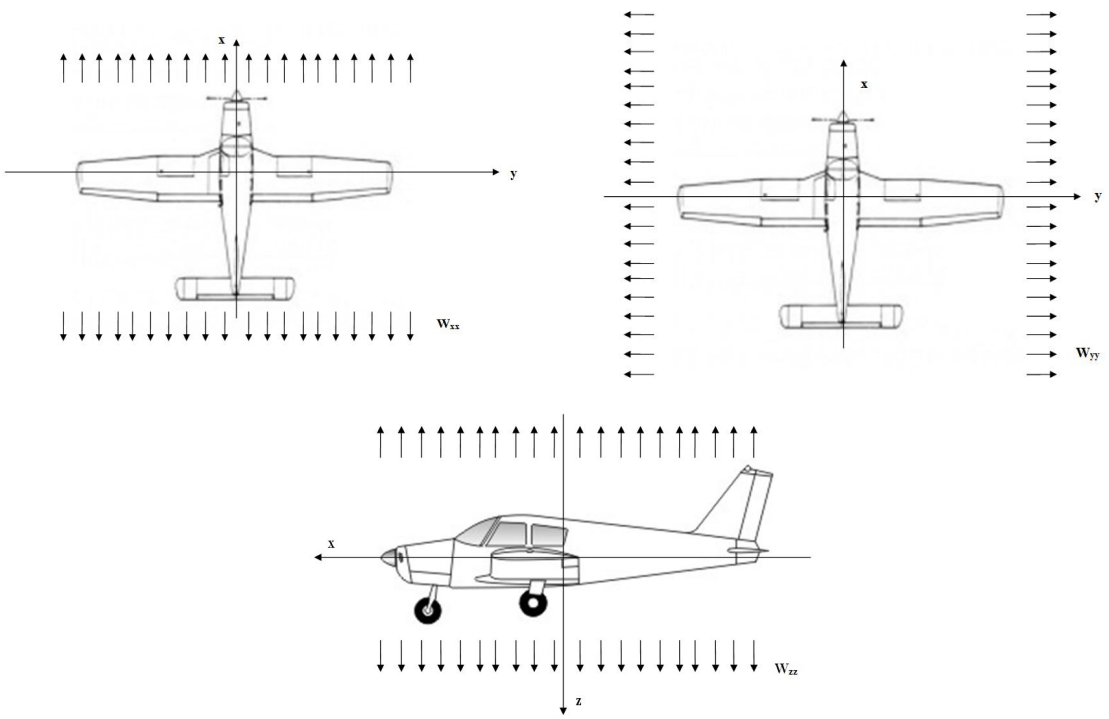


Figure 2.13: Wind gradients in the direction of the aircraft axes

Microburst Windshear Model

The modelling of microburst windshear begins with the *Vortex-Ring Principle*. The principle of the vortex-ring model is discussed in more detail in [Zhenxing, et al., (2009)]. It has been applied in different forms such as in [Wang, et al., (2016)] and [Bobbitt, (1991)]. The vertical component of the wind velocity must be equal zero on the ground. To ensure this holds true, two vortex rings denoted by Γ and $-\Gamma$, which are equal and opposite in size, are arranged symmetrically above and 'below' the ground. This is illustrated in Figure 2.14, where the ground coordinates $Oxyz$ are fixed at an arbitrary position on the ground as its origin, and the center of the primary vortex ring is fixed at $O_p(x_p, y_p, z_p)$ in the ground coordinates. The opposing imaginary vortex ring is arranged in much the same manner at the symmetrical point $O_I(x_p, y_p, -z_p)$. Any point $N(x_N, y_N, z_N)$ in the flow field is regarded as the reference point.

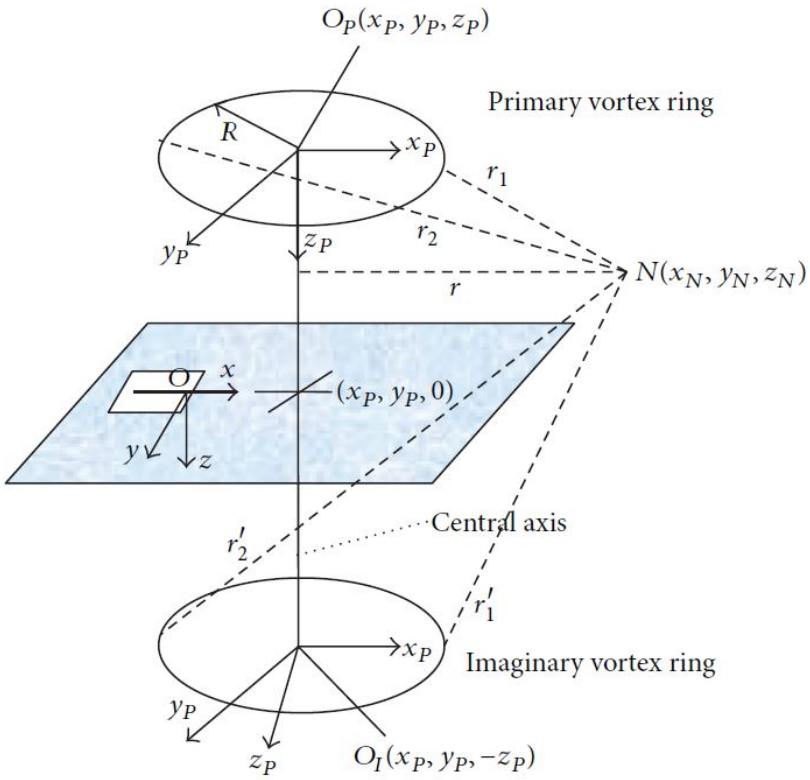


Figure 2.14: Vortex-ring model [Wang, et al.,(2016)]

If R is the radius of the vortex ring, the stream function then becomes [Wang, et al.,(2016)],[Bobbitt, (1991)]:

$$\Lambda_p = -\frac{\Gamma}{2\pi}(r_1 + r_2)(Z_1(k) - Q_1(k)) \tag{2.55}$$

where Γ is the strength of the vortex pair; r_1 and r_2 denote the minimum and maximum distances from the reference point to the vortex ring, respectively; and $Z_1(k)$ and $Q_1(k)$ are the elliptic integral functions, where the scaling factor $k = \|(r_2 - r_1)/(r_1 + r_2)\|$. $Z_1(k) - Q_1(k)$ may be replaced by the approximation

$$Z_1(k) - Q_1(k) = -\frac{3.153k^2}{1 + 3\sqrt{1 - k^2}} \tag{2.56}$$

r_1 and r_2 are calculated as follows:

$$r_1 = \sqrt{(z_N - z_p)^2 + (r - R)^2} \tag{2.57}$$

$$r_2 = \sqrt{(z_N - z_p)^2 + (r + R)^2} \tag{2.58}$$

where the radial distance to the central axis r is calculated as:

$$r = \sqrt{(x_N - x_p)^2 + (y_N - y_p)^2} \tag{2.59}$$

The stream function of the primary vortex ring is then:

$$\Lambda_p = -\frac{\Gamma}{2\pi}(r_1 + r_2)\frac{3.153k^2}{1 + 3\sqrt{1 - k^2}} \tag{2.60}$$

The imaginary vortex ring Λ_I is found in a similar manner. The stream function reference point is found by:

$$\Lambda = \Lambda_p + \Lambda_I \tag{2.61}$$

The induced velocity of this model is:

$$W_x = \frac{x - x_p}{r}\frac{1}{r}\frac{\partial \Lambda}{\partial z} \tag{2.62}$$

$$W_y = \frac{y - y_p}{r}\frac{1}{r}\frac{\partial \Lambda}{\partial z} \tag{2.63}$$

$$W_z = \frac{1}{r}\frac{\partial \Lambda}{\partial r} \tag{2.64}$$

There is a vortex core around the filament due to viscosity. The velocity gradually tends to zero in it. To solve this issue, a velocity damping factor is implemented [Bobbitt, (1991)]:

$$\tau = 1 - exp\left(\frac{r_1^2}{\epsilon d^2}\right) \tag{2.65}$$

where ϵ is the weight coefficient of influence and d is an empirical value related to the radius of the vortex core. The induced velocity is calculated using Equations (2.62) to (2.64) outside the vortex core. However within the core, a velocity damping factor must be applied. This causes the induced velocity to smoothly transition to zero in the vortex core.

The microburst wind field is built using the double-vortex-ring algorithm used in [Wang, et al., (2016)] and [Bobbit, (1991)]. This is done by using two vortex pairs with different parameters, including four vortex rings (such as the vortex ring model previously presented), set symmetrical about the ground coordinates. This is combined to form the total velocity components as:

$$W_x = \prod_{i=1}^{4} \tau_i \left(\sum_{i=1}^{4} W_{xi} \right) \tag{2.66}$$

$$W_y = \prod_{i=1}^{4} \tau_i \left(\sum_{i=1}^{4} W_{yi} \right) \tag{2.67}$$

$$W_z = \prod_{i=1}^{4} \tau_i \left(\sum_{i=1}^{4} W_{zi} \right) \tag{2.68}$$

where $\prod_{i=1}^{4} \tau_i$ is the total damping factor at the reference point.

The model of the microburst wind field was built using the following equations in the Matlab environment. The dimension are similar to that found in [Wang, et al., (2016)] and [Bobbit, (1991)] . It was found during simulation that the vortex strengths have to be decreased to account for the size and speed of the UAV being used for simulation. The microburst wind field used has a large ring radius of $2134m$, small ring radius of $396m$, large core radius of $610m$, small core radius of $100m$, large vortex strength of $4013.12m^2/s$, and small vortex strength of $531.45m^2/s$.

2.7 Modelling of Forces and Moments

2.7.1 Gravitational Forces and Moments

The components of the gravitational force $\bar{F}_g$ and moments $\bar{M}_g$ in the reference $Oxyz$ (see Figure 2.15) are as follows:

$$\bar{F}_g = mg\ L_{BE} = \begin{bmatrix} X_g \\ Y_g \\ Z_g \end{bmatrix} \tag{2.69}$$

where:
m - mass of the aircraft;
g - gravitational acceleration; and
L_{BE} - transformation matrix from the Earth reference system to the body-fixed reference system.

The components of the forces of gravity of the aircraft are as follows:

$$X^g = -mg\sin\theta \tag{2.70}$$
$$Y^g = mg\cos\theta\sin\phi \tag{2.71}$$
$$Z^g = mg\cos\theta\cos\phi \tag{2.72}$$

The components of the moments due to gravitational forces of the aircraft are as follows:

$$\bar{M}_g = \bar{r}_c \times \bar{F}_g = \begin{bmatrix} \bar{i} & \bar{j} & \bar{k} \\ x_c & y_c & z_c \\ X^g & Y^g & Z^g \end{bmatrix} = \begin{bmatrix} L_g \\ M_g \\ N_g \end{bmatrix} \tag{2.73}$$

where:

$$L^g = Z^g y_c - Y^g z_c \tag{2.74}$$
$$M^g = X^g z_c - Z^g x_c \tag{2.75}$$
$$N^g = Y^g x_c - X^g y_c \tag{2.76}$$

After substitution of Equation (2.72) into (2.76) gives:

$$L^g = mg\left(y_c\cos\theta\cos\phi - z_c\cos\theta\sin\phi\right) \tag{2.77}$$
$$M^g = -mg\left(z_c\sin\theta + x_c\cos\theta\cos\phi\right) \tag{2.78}$$
$$N^g = mg\left(x_c\cos\theta\sin\phi + y_c\sin\theta\right) \tag{2.79}$$

where:
$\bar{r}_c = [x_c, y_c, z_c]^T$ - vector of the center of gravity associated with the aircraft reference system.

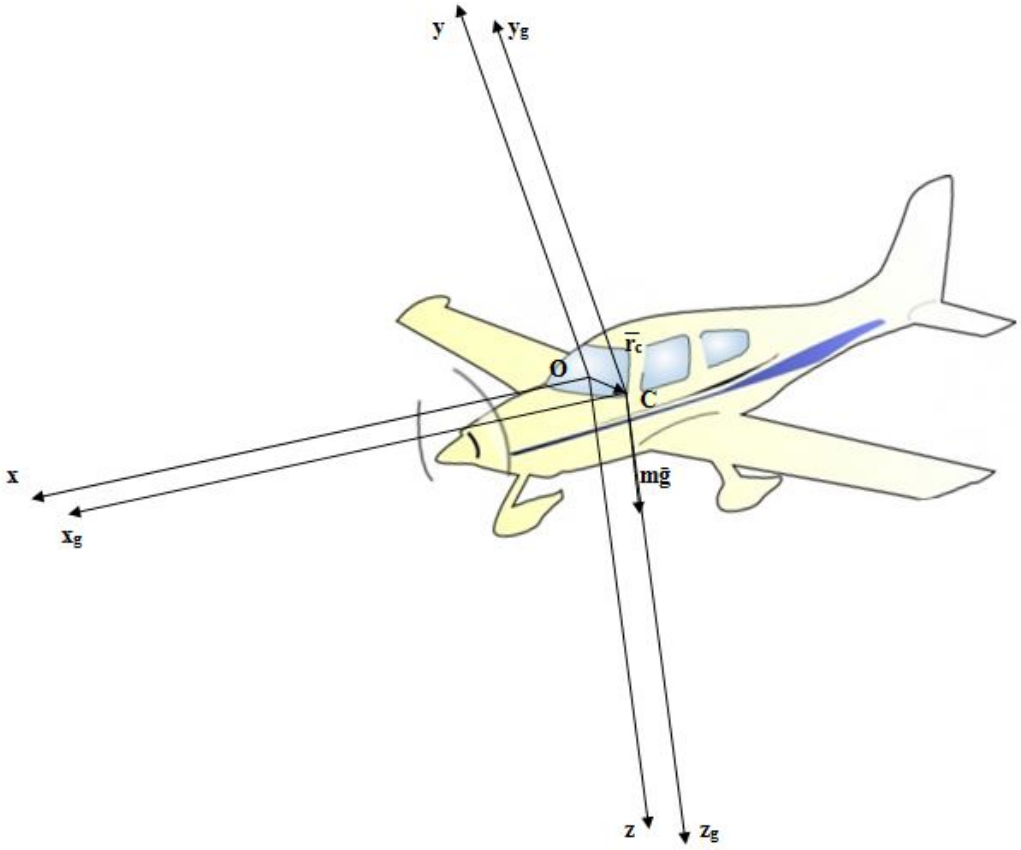


Figure 2.15: Illustrated system references for aircraft reference $Oxyz$ and weight reference $Cx_gy_gz_g$

2.7.2 Aerodynamic Forces and Moments

The dynamic model of an aircraft is commonly described by a six DOF equations, which are derived from the X, Y and Z forces and L, M and N moment equations. Figure 2.16 illustrates the forces and moments on the aircraft body axis.

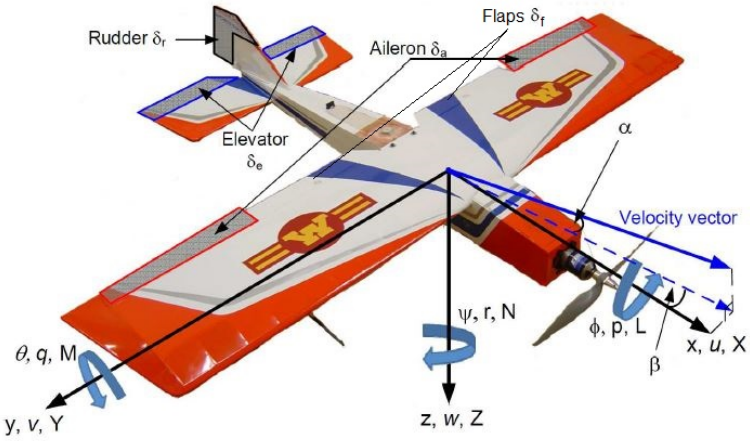


Figure 2.16: Aircraft body axis forces and moments [Paw, (2009)]

The aerodynamic force coefficients in the body axis are related to those in the wind axis by the following:

$$\begin{aligned} C_X &= C_L \sin\alpha - C_D \cos\alpha \cos\beta - C_y \sin\beta \cos\beta \\ C_Z &= -C_D \sin\alpha \cos\beta - C_L \cos\alpha - C_y \cos\beta \sin\alpha \end{aligned} \tag{2.80}$$

where α and β are angle of attack and sideslip angle of the aircraft in the wind axis. The forces in the body x, y and z axis are given by:

$$\begin{aligned} X &= \bar{q} S C_X \\ Y &= \bar{q} S C_y \\ Z &= \bar{q} S C_Z \end{aligned} \tag{2.81}$$

where $\bar{q}$ is the dynamic pressure. The lift (C_L), drag (C_D) and lateral force (C_y) coefficients are functions of the non-dimensional coefficients (where the small perturbation assumption is made and only linear terms are considered) are given by:

$$
\begin{aligned}
C_L &= C_{L0} + C_{L_\alpha}\alpha + C_{L_{\delta_e}}\delta_e + C_{L_{\delta_f}}\delta_f + \frac{\bar{c}}{2V_A}\left(C_{L_{\dot{\alpha}}}\dot{\alpha} + C_{L_q}Q_A\right) \\
C_D &= C_{D_0} + C_{D_{\delta_e}}\delta_e + C_{D_{\delta_r}}\delta_r + \frac{\left(C_L - C_{L_{min}}\right)^2}{\pi e AR} \\
C_y &= C_{y_0} + C_{y_\beta}\beta + C_{y_{\delta r}}\delta_r + C_{y_{\delta_a}}\delta_a + C_{y_{\delta_f}}\delta_f + b/(2V_A)\left(C_{y_P}P_A + C_{y_r}R_A\right)
\end{aligned}
\tag{2.82}
$$

The non-dimensional moment coefficients are given by:

$$
\begin{aligned}
C_l &= C_{l_0} + C_{l_\beta}\beta + C_{l_{\delta_a}}\delta_a + C_{l_{\delta_f}}\delta_f + C_{l_{\delta_r}}\delta_r + \frac{b}{2V_A}\left(C_{l_P}P_A + C_{l_r}R_A\right) \\
C_m &= C_{m_0} + C_{m_\alpha}\alpha + C_{m_{\delta e}}\delta_e + C_{m_{\delta f}}\delta_f + \frac{\bar{c}}{2V_A}\left(C_{m_{\dot{\alpha}}}\dot{\alpha} + C_{m_q}Q_A\right) \\
C_n &= C_{n_0} + C_{n_\beta}\beta + C_{n_{\delta a}}\delta_a + C_{n_{\delta_f}}\delta_f + C_{n_{\delta_r}}\delta_r + \frac{b}{2V_A}\left(C_{n_p}P_A + C_{n_r}R_A\right)
\end{aligned}
\tag{2.83}
$$

where $AR$is the wing aspect ratio; and e Oswalds efficiency factor. The moments about the body x, y and z axis are given by:

$$
\begin{aligned}
L &= \bar{q}Sc_l \\
M &= \bar{q}Sc_m \\
N &= \bar{q}Sc_n
\end{aligned}
\tag{2.84}
$$

2.7.3 Thrust Forces and Moments

The forces and moments due to thrust are:

$$X_T = T\cos\psi_T\cos\theta_T \tag{2.85}$$

$$Y_T = T\sin\psi_T \tag{2.86}$$

$$Z_T = T\cos\psi_T\sin_T \tag{2.87}$$

$$L_T = I_T\omega_P\left(Q\cos\psi_P\sin\theta_T + R\sin\phi_T\right) \tag{2.88}$$

$$M_T = I_P\omega_P\cos\psi_T\left(R\cos\theta_T + P\sin\theta_T\right) \tag{2.89}$$

$$N_T = I_P \omega_P (Q \cos\psi_T \cos\theta_T - P \sin\psi_T) \tag{2.90}$$

where I_T is the engine moment of inertia; ψ_T and θ_T are the engine thrust yaw and pitch angles respectively. The aircraft model includes correction terms for the gyroscopic effects of the spinning propeller.

2.8 Kinematic Equations

The kinematics of the aircraft rotation motion relating the body angular rates P, Q and R, Euler angles ϕ, θ and ψ and aerodynamics angles α, and β are given by:

$$\dot{\phi} = P + \tan\theta \, (Q \sin\phi + R \cos\phi) \tag{2.91}$$

$$\dot{\theta} = Q \cos\phi - R \sin\phi \tag{2.92}$$

$$\dot{\psi} = (Q \sin\phi + R \cos\phi) \sec\theta \tag{2.93}$$

where ψ is the yaw angle.

2.9 Navigational Equations

The navigation equations for the flat Earth model are:

$$\dot{x}_E = U\cos\theta\cos\psi + V(\sin\phi\sin\theta\cos\psi - \cos\phi\sin\psi) + W(\cos\phi\sin\theta\cos\psi + \sin\phi\sin\psi) + W_{x_g} \tag{2.94}$$

$$\dot{y}_E = U\cos\theta\sin\psi + V(\cos\phi\sin\theta\cos\psi - \cos\phi\cos\psi) + W(\cos\phi\sin\theta\sin\psi - \sin\phi\cos\psi) + W_{y_g} \tag{2.95}$$

$$\dot{h} = -\dot{z}_E = U\sin\theta - V\sin\phi\cos\theta - W\cos\phi\cos\theta - W_{z_g} \tag{2.96}$$

where x_E, y_E, and h are the aircraft position components in the Earth-fixed frame and W_{x_g}, W_{y_g}, and W_{z_g} are the components of the Earth-fixed wind velocity vector (W_g).

$$\dot{\times}_E = [\dot{x}_E,\ \dot{y}_E,\ \dot{x}_E]^T = \mathbf{L}_{BE}\mathbf{v}_{BE} = \mathbf{L}_{BE}\,[U,\ V,\ W]^T \tag{2.97}$$

2.10 Aerodynamic Coefficients Determination

Table 2.3 summarizes the aerodynamic coefficients required to model the UAV.

Lift Force	Drag Force	Side Force	Roll Moment	Pitch Moment	Yaw Moment
C_{L_0}	C_{D_0}	C_{y_β}	C_{l_β}	C_{m_0}	C_{n_β}
C_{L_α}	$C_{D_{\delta e}}$	$C_{y_{\delta r}}$	$C_{l_{\delta r}}$	C_{m_α}	$C_{n_{\delta r}}$
$C_{L_{\dot{\alpha}}}$	$C_{D_{\delta r}}$	C_{y_p}	C_{l_p}	$C_{m_{\delta e}}$	C_{n_p}
C_{L_q}		C_{y_r}	C_{l_r}	$C_{m_{\dot{\alpha}}}$	C_{n_r}
$C_{L_{min}}$			$C_{l_{\delta a}}$	C_{m_q}	

Table 2.3: Ultrastick UAV Aerodynamic Coefficients

These coefficients were obtained using wind tunnel testing performed by [Paw & Balas, (2008)]. Other method of obtaining coefficients exist such as flight test parameter identification or numerical computation [Abdulrahim & Lind, (2005)]. The determination of these coefficients was done by approximating the coefficients using the simulator parameters tuning approach with some of the initial parameters set to values obtained from previous work by [Owens, Cox, & Morelli, (2006)]. These values were then refined using flight test parameter identification by [Paw, (2009)]. The Ultra Stick aircraft was initially used as a low-cost flight test platform at the NASA Langley Research Center. Aerodynamic modeling includes extensive static wind tunnel tests, that have been complemented with dynamic wind tunnel tests. The aerodynamic model for the Ultra Stick is nonlinear and includes effects due to the basic airframe, control surfaces, thrust, and angular rates [Dorobantu, (2013)], [Owens, Cox, & Morelli, (2006)].

2.11 Propulsion Modelling

The propulsion system models the interaction between electric motor and propeller dynamics. The rotation speed of propeller, ω_p, is used to describe the dynamics of the propulsion system. Small UAVs (with propeller propulsion system) flight dynamics are sensitive to the propulsion system dynamics. Small UAV are more sensitive due to the large torque from the propulsion system coupling with the aircraft rigid body dynamics as the small UAV system is being propelled by a larger propeller when compared to the UAV's size. Therefore the moment generated by the propeller is added to the total moment of the aircraft in the simulation model. Applying the conservation of angular momentum, the propulsion dynamics is given by [Paw, (2009)]:

$$\left(I_{\text{motor}} + I_{\text{propeller}}\right)\dot{\omega}_p = T_{\text{motor}} - T_{\text{propeller}} \tag{2.98}$$

where:
I_{motor} - moment of inertia of rotating motor body (kgm^2);
$I_{\text{propeller}}$ - moment of inertia of propeller with spinner hub attachment (kgm^2);
T_{motor} - output torque at motor shaft (Nm); and
$T_{\text{propeller}}$ - torque generated by propeller (Nm).

The moment of inertia for rotating motor body, I_{motor}, is included because the motor used is an outrunner motor in which the major part of the motor mass is rotating. This has a significant contribution to the total moment of inertia for the propulsion system, $I_p (I_p = I_{\text{motor}} + I_{\text{propeller}})$.

2.11.1 Propeller

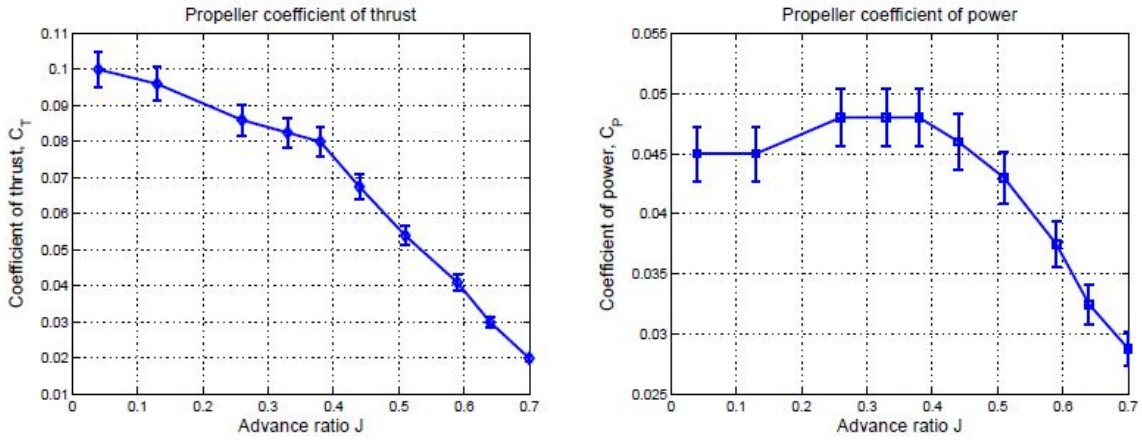
The thrust to propel the aircraft forward is generated by the rotating propeller using the torque generated at motor output shaft. The propeller rotation speed depends on both the input torque available and the speed of air flowing into the propeller disk. The propeller performance is mainly dependent upon the following parameters:

- advance ratio J:
$$J = \frac{\pi V_a}{\omega_p R} \tag{2.99}$$
- thrust coefficient C_T:
$$C_T = \frac{F_p \pi^2}{4\rho R^4 \omega_p^2} \tag{2.100}$$
- power coefficient C_p:
$$C_p = \frac{T_p \pi^3}{4\rho R^5 \omega_p^2} \tag{2.101}$$

where $R(m)$ is the diameter of the propeller, $F_p(N)$ is the propeller thrust, $T_p(Nm)$ is the propeller torque, and $\rho(kgm^3)$ is the air density.

The approximation made on the propeller performance data is based on experimental results published on an APC 12 x 8 propeller. The propulsion system thrust and torque generated were measured using a force-moment sensor in a wind tunnel with different propeller rotation speed and inflow airspeed. The data used in this research was obtained from [Paw, (2009)]. The coefficient of thrust and power obtained were plotted against advance ratio as shown in Figure 2.17. The data is used as lookup tables in the nonlinear simulation model. This

provides the propeller thrust (F_p) and moment (T_p) at different advance ratios condition during the simulation.



(a) Coefficient of thrust

(b) Coefficient of power

Figure 2.17: Propeller Performance [Paw, (2009)]

2.11.2 Motor

The propulsion motor used is E-flite Power 25 BL Outrunner Motor. The motor performance data is obtained from a commercial software, MotorCalc, since it is not available from the manufacturer. Figure 2.18 shows the plot for the relationship between throttle stick input (δ_T) and output power at the motor shaft (P_o).

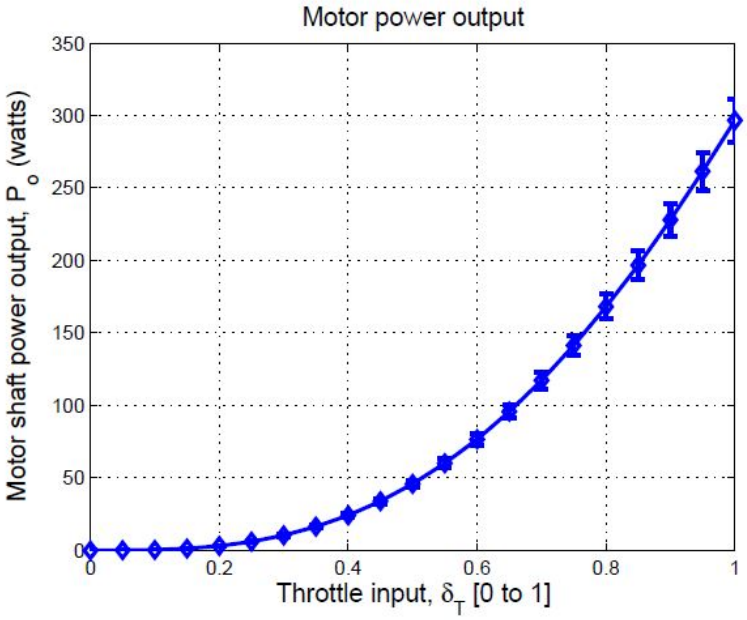


Figure 2.18: Motor Power Output [Paw, (2009)]

The motor shaft torque, $T_{motor}(Nm)$, generated from the motor shaft output power P_o is given by:

$$T_{motor} = \frac{P_o}{\omega_p} \tag{2.102}$$

2.11.3 Actuator Modelling

The UAV control surfaces are driven by Hitec S3108 micro servos using Pulse-Width Modulation (PWM) signal operating at 45 Hz. The servo rotation rate is 500 deg/s at no-load condition. The time delay for the servo actuator dynamics is approximated using a single period of PWM signal operating at 45 Hz frequency. This gives a time delay of 22 ms for the actuator delay. The actuators rotation angle saturation limits are set to the maximum mechanical deflection angles of the control surfaces.

2.12 Simulation Control Surface Convention

The sense of the control surface deflections as used in the simulation is depicted in Figures 2.19 to 2.21.

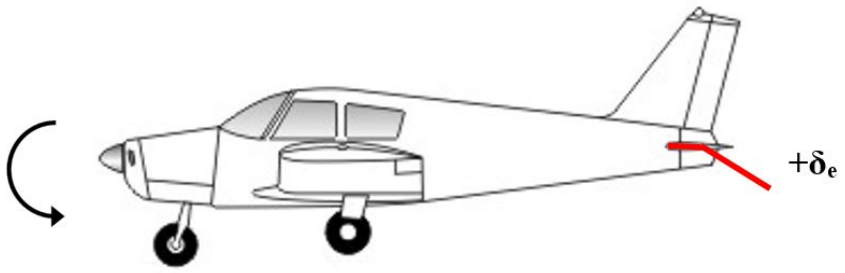


Figure 2.19: Elevator Deflection Convention

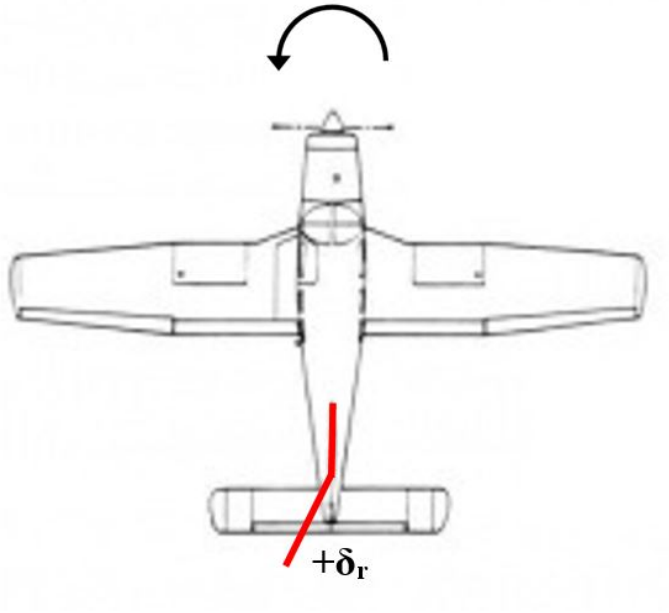


Figure 2.20: Rudder Deflection Convention

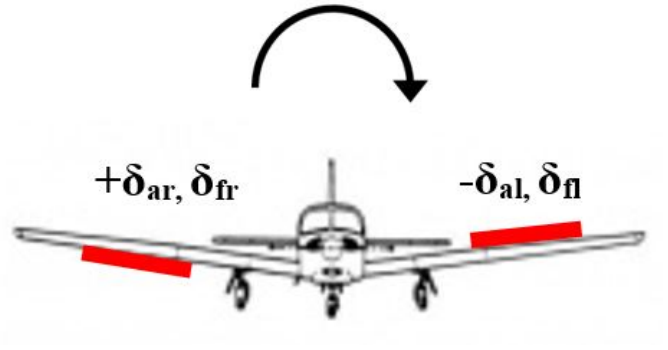


Figure 2.21: Aileron and Flap Deflection Convention

3 UAV Controller Development

3.1 Chapter Outline

This chapter provides an overview of the control structure, in addition to the detailed description of how each control loop works to control the aircraft.

3.2 Overall PID Control Structure

The basic principle of PID control is the regulation of a feedback signal, in particular of the error between the measured and the desired state. The PID controller gains adjust the control action according to the properties of the error signal $e(t)$. The starting point is a simple proportional gain K_P, which determines the control action $u_c(t)$:

$$u_c(t) = K_P e(t) \tag{3.1}$$

The limits of adopting only a proportional gain are the possible steady-state offset in response to a constant reference and the inability to reject a constant disturbance. Furthermore, an excessively large value of K_P might lead to instability, in particular when dealing with high-order systems. In order to eliminate the steady-state error the integral gain K_I is commonly adopted to support the proportional action. Its contribution is proportional through the gain K_I to the integral of the error from the initial time to the considered moment t.

$$u_c(t) = K_I \int_0^t e(t)dt \tag{3.2}$$

The integral contribution has the advantage of taking into account the past values of the error. For this reason, it is possible to have a control action different from zero when the error

is zero. The addition of the integral term to a simple proportional controller improves the tracking of the steady-state response. The drawback of increasing K_I is a slower response for equal overshoot, or a rise in overshoot for an unchanged response velocity. Finally, a derivative term is added to improve stability by increasing the damping of the response thanks to an anticipatory behaviour. The rate of change of the signal is used so that its contribution to the total control action is proportional through the gain K_D to the derivative of the error:

$$u_c(t) = K_D \frac{de(t)}{dt} \tag{3.3}$$

The sum of all the three contributions yields to the proportional-integral-derivative controller, a linear combination of the error, its integral and its derivative in time

$$u_c(t) = K_P e(t) + K_I \int_0^t e(t)dt + K_D \frac{de(t)}{dt} \tag{3.4}$$

In the longitudinal plane, the altitude h is controlled by elevator δ_e, while the linear speed u is controlled by throttle δ_{th}, (see Figure 3.1). Three PID controllers were designed in which there were two feedbacks on the altitude loop, pitch angle θ and altitude h, and one on the speed loop, linear speed u. The altitude is controlled by the elevator and the speed by the throttle. This implementation is not the only solution as other architectures have been implemented on the UAV. The state vector of the system in this plane is $x_{long} = [\theta,\ u,\ h,\ \alpha,\ q]^T$. In the lateral-directional plane the ailerons δ_a control the heading angle through two feedbacks of roll (ϕ) and heading angles (ψ) respectively. Two PID loops need to be configured. The lateral-directional response state vector of the system is $x_{lat} = [\phi,\ \psi,\ \beta,\ p,\ r]^T$.

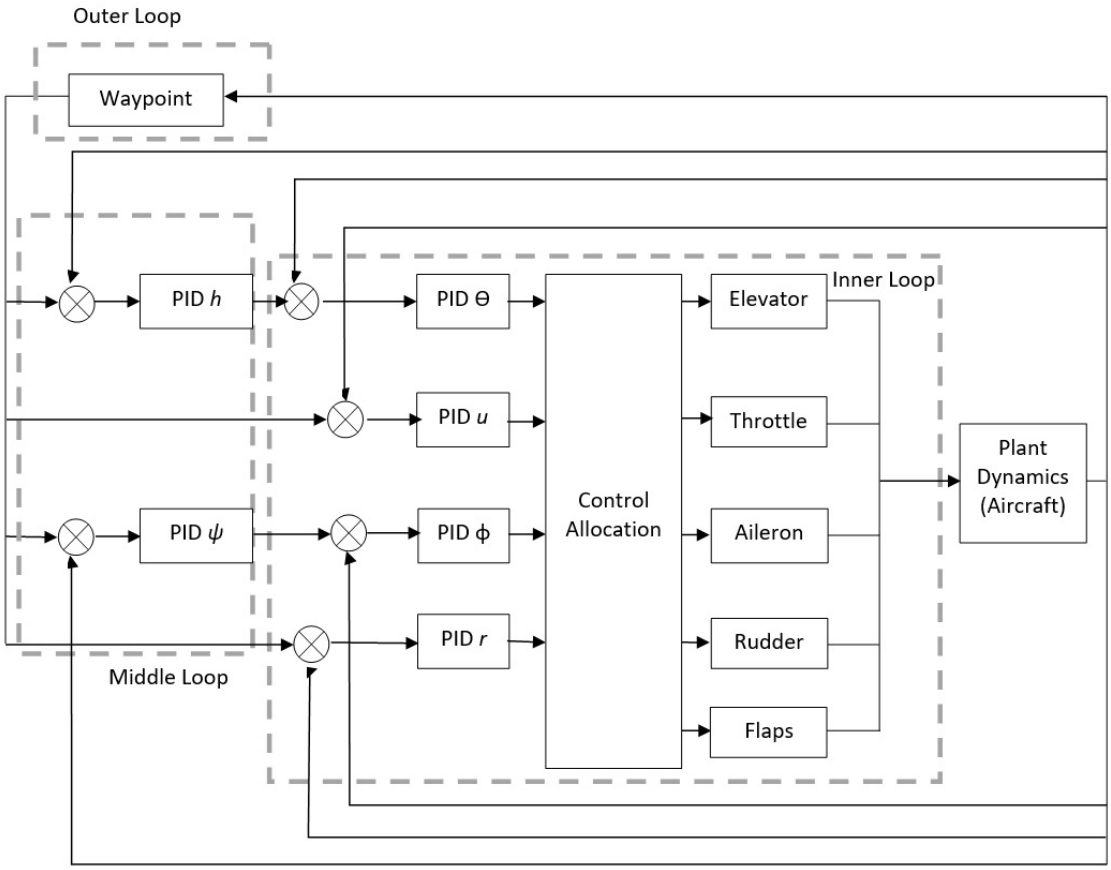


Figure 3.1: Block Diagram Showing Control Structure

The following equation is used to find the heading between two waypoints:

$$\Upsilon = \tan^{-1} \frac{\sin(\theta_{WP} - \theta_{\text{Current}}) \cos(\psi_{WP})}{\cos(\psi_{\text{Current}}) \sin(\psi_{WP}) - \sin(\psi_{\text{Current}}) \cos(\phi_{WP} - \phi_{\text{Current}})} \tag{3.5}$$

where ϕ_{WP}, θ_{WP}, and ψ_{WP} are the waypoint bank, pitch and yaw angles, and Υ is the set heading. The set of PID gains were determined by first tuning the inner-loop PID controllers, then tuning the middle-loop controllers and finally the outer-loop controllers in turn. A setpoint was chosen for each PID controller and the response to a disturbance observed. The gains were then adjusted until the response was deemed satisfactory.

The gains determined are presented in Table 3.1. These gains were obtained through trial and error.

PID Controller	Proportional Gain	Integral Gain	Derivative Gain
PID Pitch	0	0.0003	0
PID Altitude	0.001	0.001	0
PID Roll	0.0003	0	0
PID Yaw	-0.0001	0	0
PID Airspeed	0.	-0.001	-0.0001
PID Heading	1.2	0	0

Table 3.1: Controller PID Gains

3.2.1 Controller Optimization

3.2.1.1 Particle Swarm Optimisation

Particle swarm optimisation (PSO) was used to optimise the controller PID gains. PSO is inspired by social and cooperative behaviour displayed by various species to fulfil their needs in the search space. It presents a stochastic calculation based on the movement of swarms of particles biased towards the particle which has the most weight in the swarm. PSO has been utilised for aircraft control in works such as [Chollom, et al., (2016)] and [Mac, et al., (2016)]. In this application a swarm is made up of a set of particles, of which each particle is a possible solution to the optimisation problem. The algorithm is guided by personal experience (Pb) which is the best position found by the particle itself, global experience (Gb) which is the best position of the whole particles in the swarm and the present movement of the particles to decide their next locations in the search space. In addition, the experiences are accelerated by two factors c_1 and c_2 which are referred to as cognitive and social factors, respectively; and two random numbers (r_1 and r_2) generated between $[0, 1]$ which add the stochastic state to the algorithm [Mac, et al., (2016)]. The present movement is multiplied by an inertia weight factor w varying between $[w_{\min}; w_{\max}]$, according to [Hamidi, (2012)] it is possible to select a suitable w that balances between the Gb and Pb search space.

The initial population (swarm) of size N and search space D is denoted as $X = [X_1, X_2, \ldots, X_N]^T$. Each individual (particle) $X_i\,(i = 1, 2, \ldots, N)$ is given as $X_i = [X_{i1}, X_{i2}, \ldots, X_{iD}]$. Also, the initial velocity of the population is denoted as $V = [V_1, V_2, \ldots, V_N]^T$. Thus, the velocity of each particle $X_i\,(i = 1, 2, \ldots, N)$ is given as $V_i = [V_{i1}, V_{i2}, \ldots, V_{iD}]$. The index i varies from 1 to N whereas the index j varies from 1 to D. The details of the algorithm are presented below.

$$V_{i,j}^{k+1} = w \times V_{i,j}^{k} + c_1 \times r_1 \times \left(Pb_{i,j}^{k} - X_{i,j}^{k}\right) + c_2 \times r_2 \times \left(Gb_{j}^{k} - X_{i,j}^{k}\right) \tag{3.6}$$

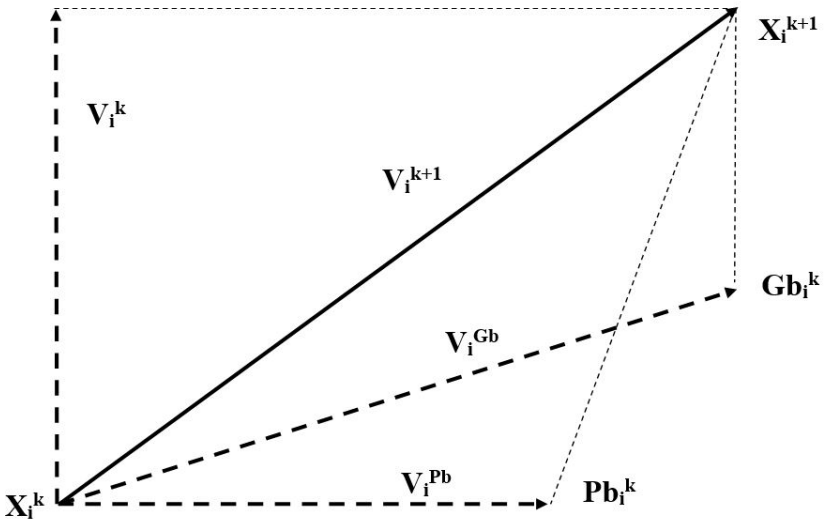


Figure 3.2: PSO search mechanism in multidimensional search space [Alam, et al, (2015)]

$$X_{i,j}^{k+1} = X_{i,j}^{k} + V_{i,j}^{k+1} \tag{3.7}$$

In Equation (3.6) $Pb_{i,j}^k$ represents personal best j^{th} component of ith individual, whereas Gb_j^k represents j^{th} component of the best individual of population upto iteration k. The particles will update their locations in the swarm towards the global optimum (or target position) based on these two factors. Figure 3.2 shows the search mechanism of PSO in multidimensional search space.

The different steps of PSO are as follows [Alam, et al, (2015)]:

1. Set parameter $w_{\min}, w_{\max}, c_1$ and c_2 of PSO
2. Initialize population of particles having positions X and velocities V
3. Set iteration $k = 1$
4. Calculate fitness of particles $F_i^k = f\left(X_i^k\right), \forall i$ and find the index of the best particle b
5. Select $Pb_i^k = X_i^k, \forall i$ and $Gb^k = X_b^k$
6. $w = w_{\max} - k \times (w_{\max} - w_{\min}) = k_{\max}$
7. Update velocity and position of particles
   $V_{i,j}^{k+1} = w \times V_{i,j}^k + c_1 \times \text{random number} \times \left(Pb_{i,j}^k - X_{i,j}^k\right) + c_2 \times \text{random number} \times \left(Gb_j^k - X_{i,j}^k\right); \forall j$ and $\forall i$
   $X_{i,j}^{k+1} = X_{i,j}^k + V_{i,j}^{k+1}; \forall j$ and $\forall i$

8. Evaluate fitness $F_i^{k+1} = f\left(X_{k+1i}\right); \forall i$ and find the index of the most optimal particle b_1

9. Update Pb of population $\forall i$
   If $F_i^{k+1} \leq F_i^k$ then $Pb_{k+1i} = X_i^{k+1}$ else $Pb_i^{k+1} = Pb_i^k$

10. Update Gb of population
    If $F_{b1}^{k+1} \leq F_b^k$ then $Gb^{k+1} = Pb_{b1}^{k+1}$ and set $b = b1$ else $Gb^{k+1} = Gb^k$

11. If $k < k_{\max}$ then $k = k + 1$ and go to step 6 else go to step 12

12. Print best solution as Gb^k

The most commonly used parameters of PSO algorithm are considered as follows:

- Inertial weight (w): 0.9 to 0.4
- Acceleration factors (c_1 and c_2): 2 to 2.05
- Population size (N): 10 to 100
- Maximum iteration ($k_{\max}$): 500 to 10000
- Initial velocity: 10 % of position

A detailed flowchart of the PSO algorithm considering the above steps is shown in Figure 3.3.

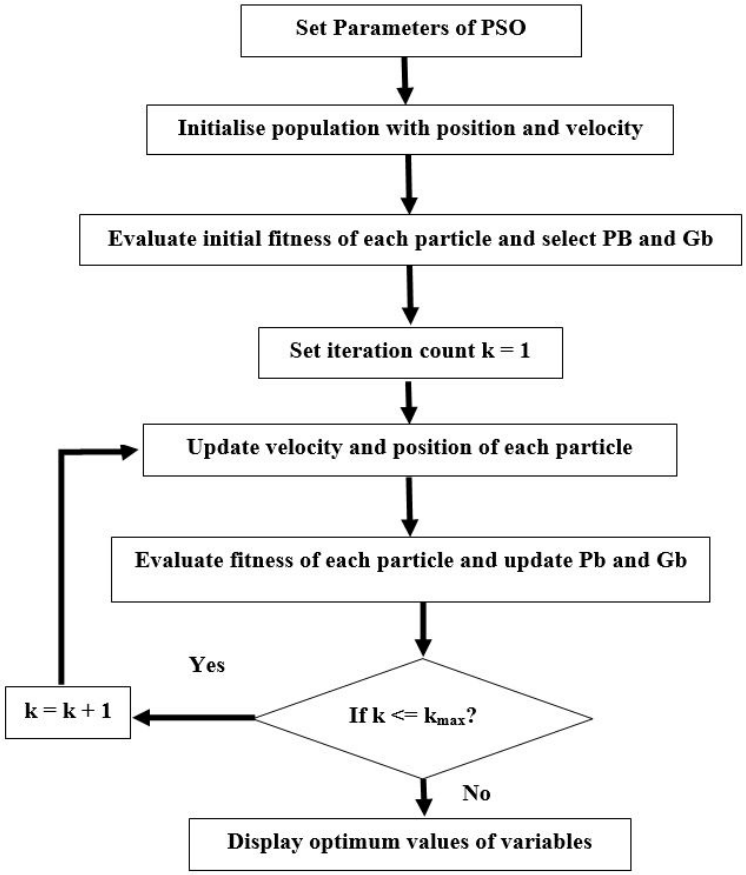


Figure 3.3: Flowchart of PSO

Automatic tuning of PID can be done based on minimising the performance index calculated by the integral absolute error (IAE):

$$IAE = \int_0^\infty |e(t)|\, dt \tag{3.8}$$

The control structure of the PID controllers optimised with PSO is illustrated in Figure 3.4.

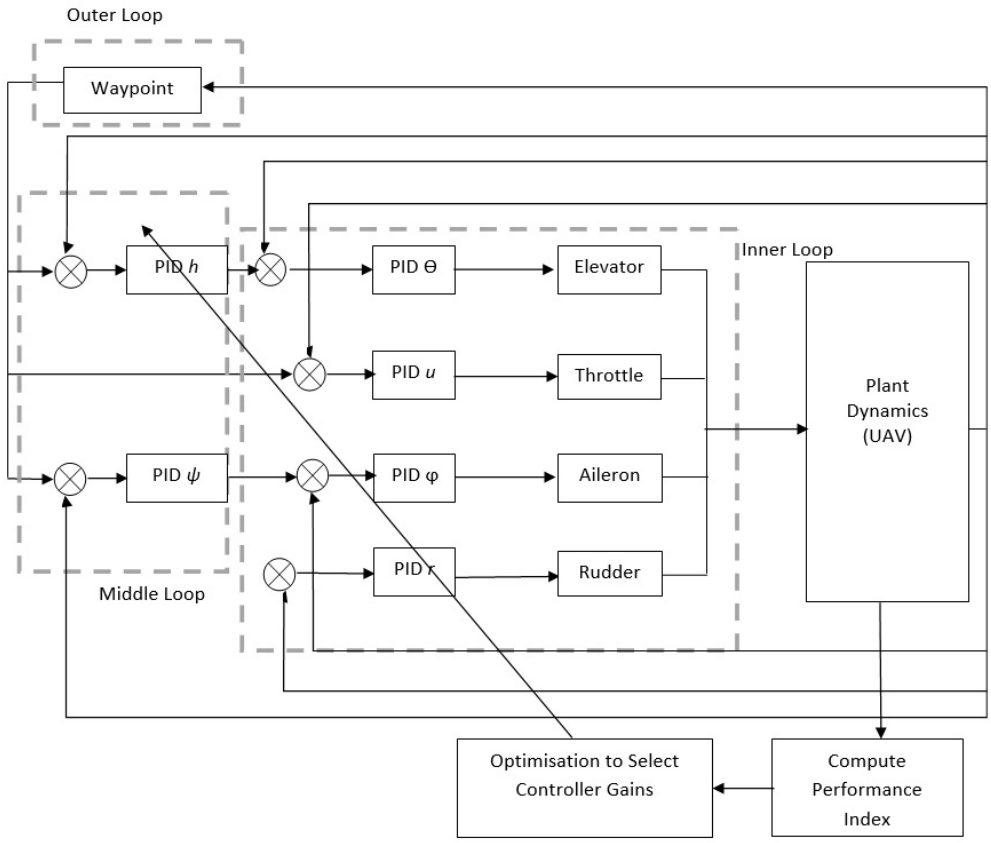


Figure 3.4: Architecture of Optimised PID Controller

The trend of the magnitude of the fitness function is illustrated in Figure 3.5.

PID Controller	Proportional Gain	Integral Gain	Derivative Gain
PID Pitch	-1.0	-0.08	0
PID Altitude	0.023	0.001	0
PID Roll	0.08	0	0
PID Yaw	-0.065	0	0
PID Airspeed	0.15	-0.0001	-0.0001
PID Heading	0.15	0	0

Table 3.2: Optimised Controller PID Gains

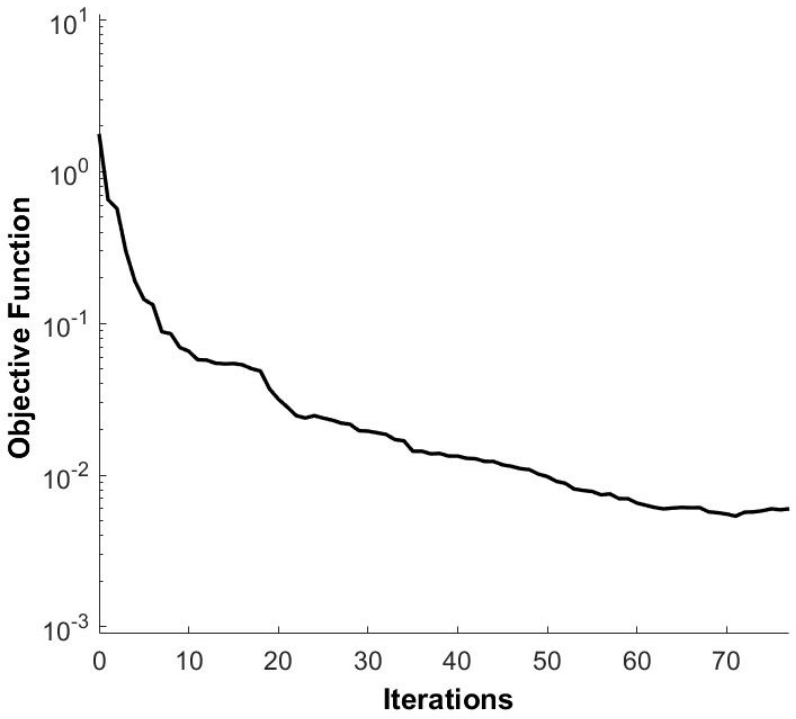


Figure 3.5: Fitness function

The optimised gains are presented in Table 3.1.

3.3 Flight Modes

The aircraft has a number of flight modes. These basic scenarios are as follows

- Steady wings-level flight.
- Steady turning flight.

- Steady wings-level climb.
- Climbing turn.

For steady-state flight:

$$\dot{p},\ \dot{q},\ \dot{r},\ \dot{u},\ \dot{v},\ \dot{w}\ \left(or\ \dot{V}_a,\ \dot{\beta},\ \dot{\alpha}\right) = 0 \tag{3.9}$$

$$U = \text{constant}$$

With the following additional constraints according to the flight condition:

1. Steady wings-level flight:
$$\phi,\ \dot{\phi},\ \dot{\theta},\ \dot{\psi},\ \dot{h} = 0 \tag{3.10}$$
2. Steady turning flight:
$$\dot{\phi},\ \dot{\theta} = 0,\ and\ \phi,\ \dot{\psi} \neq 0 \tag{3.11}$$
3. Steady wings-level climb:
$$\phi,\ \dot{\phi},\ \dot{\psi},\ \dot{\theta} = 0,\ and\ \dot{h} \neq 0 \tag{3.12}$$
4. Steady climbing turn:
$$\dot{\phi},\ \dot{\theta} = 0,\ and\ \dot{h},\ \dot{\psi} \neq 0 \tag{3.13}$$

3.4 Control Allocation

One of the problems that is envisaged when a failure occurs is the saturation of a control surface in an attempt to stabilise the aircraft or even maintain a trimmed position for extreme control failures. If this saturation occurs, the ideal behaviour will not be able to be maintained, creating a number of potential problems some of which have been considered previously. A method of control allocation is proposed which allows for the transferring of control authority to other control surfaces effective in the same plane in the event of control saturation. Control allocation has been implemented in various forms for the purposes of FTC. Some of these methods include adaptive control allocation [Tohidi, et al., (2016)], on-line sliding mode control allocation [Alwi & Edwards, (2008)]; direct allocation [Durham, et al., (2017)]; or nonlinear constraint optimisation [Zaki, et al., (2017)]. The chosen control allocation technique to be used is similar to that implemented by [Harkegard, (2002)]; and [Lombaerts, et al., (2011)].

3.4.1 Control Allocation Algorithm

There are different ways of redistributing the required control effort, some of these methods include distributing the control signal equally among all the actuators [Shin, et al, (2005)] or distributing it based on limits (position and rate) of the actuators [Davidson, et al., (2001)], [Higgs, (2014)]. The control allocator determines actuator commands $o \in R^m$, which compensate for the required control effort. This is done by means of a virtual control effort, $f \in R^n$ where $m > n$. In the UAV control case, o represents the desired deflections of the control surfaces, while o is indicative of the moments or angular accelerations to be enacted in pitch, roll, and yaw ($n = 3$) to achieve the required aircraft dynamics.

It is assumed that the actuator dynamics may be neglected (though this may not be true in general) as it has much quicker response time than the rest of the UAV dynamics, the resulting virtual control may be represented by Bo, where B is the control effectiveness matrix. The control vector o is limited by

$$\underline{o} \leq o \leq \overline{o} \tag{3.14}$$

At each sampling point the lower and upper limits ($\underline{o}$ and $\overline{o}$) are calculated using the position and rate limits of the actuators. The satisfactory performance of the determined control effort is measured by the 2-norm, $\|o\|^2 = o^T o$. The resulting sequential least squares formulation of the control allocation problem then becomes:

$$o_s = \arg\min_{o \in \kappa} \|W_o(o - o_p)\| \tag{3.15}$$

$$\kappa = \arg\min_{\underline{o} \leq o \leq \overline{o}} \|W_f(Bo - f)\| \tag{3.16}$$

where, o_p is the preferred control vector and W_o and W_f are nonsingular weighting matrices. Equations 3.15 and 3.16 should be seen as follows: Given κ, the set of achievable controls that minimize $Bo - f$ (weighted by W_f), determine the control that minimizes $o - o_p$ (weighted by W_o). This is enacted by considering the limited and equality limited least squares problem:

Algorithm 1 (Active set algorithm)

Consider the bounded and equality constrained least squares problem:

$$\min_o \|Ao - b\| \tag{3.17}$$

$$Bo = f \tag{3.18}$$

$$Co \geq O \tag{3.19}$$

where 3.19 is equivalent to 3.14 if we define $C = \begin{pmatrix} I \\ -I \end{pmatrix}$ and $O = \begin{pmatrix} \underline{o} \\ -\overline{o} \end{pmatrix}$. This problem is solved by the algorithm solving a sequence of equality limited problems. The working set W in each step is made up of some of the inequality constraints, which are taken as equality constraints. The remaining inequality constraints are not taken into account. The active set of the solution becomes the working set at its optimum.

In other words, the achievable starting point is o^0. A point is achievable if it satisfies 3.18 and 3.19. The active inequality constraints at o_0 are encompassed by the working set W.

An algorithm loops through $k = 0, 1, 2, \ldots, N-1$ (using a 'for' loop) and determines the ideal perturbation. The ideal perturbation (if presented with o^k) p is determined by considering the constraints in the working set as equality constraints along with disregarding the remaining inequality constraints. The following equations are then solved:

$$\min_p \left\| A(o^k + p) - b \right\| \tag{3.20}$$

$$Bp = 0 \tag{3.21}$$

$$p_i = 0, i \in W \tag{3.22}$$

The algorithm uses an outer 'if' statement depending on whether $o^k + p$ is achievable. If the 'if' statement is true the it sets $o^{k+1} = o^k + p$ and determines the Lagrange multipliers, $\begin{pmatrix} \mu \\ \lambda \end{pmatrix}$, where μ is affiliated with 3.18 and λ with the active constraints in 3.19. The inner 'if' statement then checks if all $\lambda \geq 0$. If this inner statement is true then the achievable solution to the bounded and equality constrained least squares problem is o^{k+1}. Otherwise if this inner statement is not true then the constraint affiliated with the most negative λ is discarded from the working set. If the outer 'if' statement is not true then the maximum step length is resolved by α such that $o^{k+1} = o^k + \alpha p$ is achievable. The primary bounding constraint is then included in the working set. This algorithm is summarized in Figure 3.6.

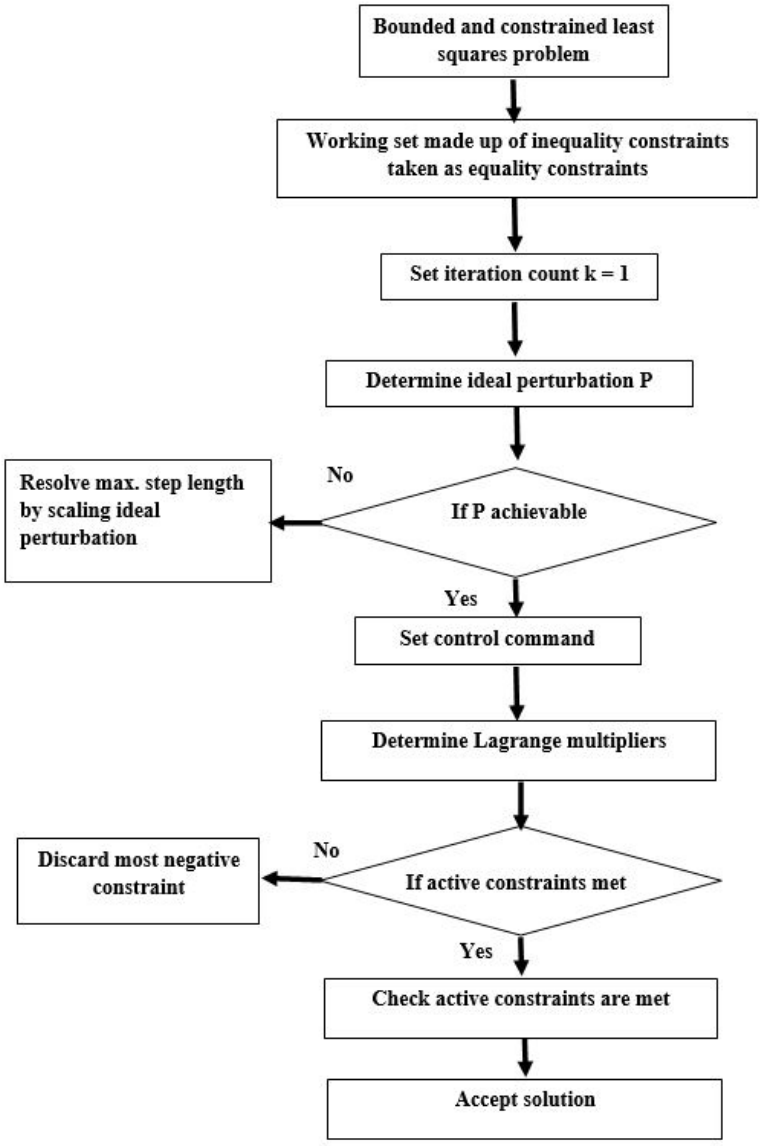


Figure 3.6: Flowchart of active set algorithm

Sequential Least Squares

There are two parts to resolving the sequential least squares problem 3.15 and 3.16. The first part involves (if possible) finding an achievable solution to $Bo = f$, which is used as the offset point in part two. An assumption is made that no three actuators produce coplanar controls, resulting in any three columns of B being linearly independent.

Part one begins with letting u^0 and W be the resulting solution and working set from the previous sampling instant. One solves $o_1 = \arg\min_o \|W_f(Bo - f)\|$ and $\underline{o} \leq o \leq \overline{o}$ using the previous algorithm with the following changes. When the number of free variables exceeds n,

the ideal perturbation p is not uniquely determined. In this case, the minimum perturbation is chosen. If $Bo_1 = f$ is true, move on to part two, else set $o_S = o_1$.

Part two begins with letting u^0 and W be the resulting solution and working set from step one. The equations $u_s = \arg\min_o \|W_o(o - o_p)\|$ and $Bo = f$ as well as the inequality $\underline{o} \leq o \leq \overline{o}$ are also solved using the previous algorithm.

If rate limits affecting $\underline{o}$ and $\overline{o}$ occur, o^0 and W may not be 'compatible' since the fixed components in o^0 are set to the limits specified by $\underline{o}$ and $\overline{o}$ from the previous sampling instant. When this occurs one updates the values of the fixed variables in o_0 to show the new limits. In the case where the control allocator is activated and there is no previous solution available, one sets $o^0 = (\underline{o} + \overline{o})/2$ and $W = \oslash$. The two parts are depicted in Figure 3.7.

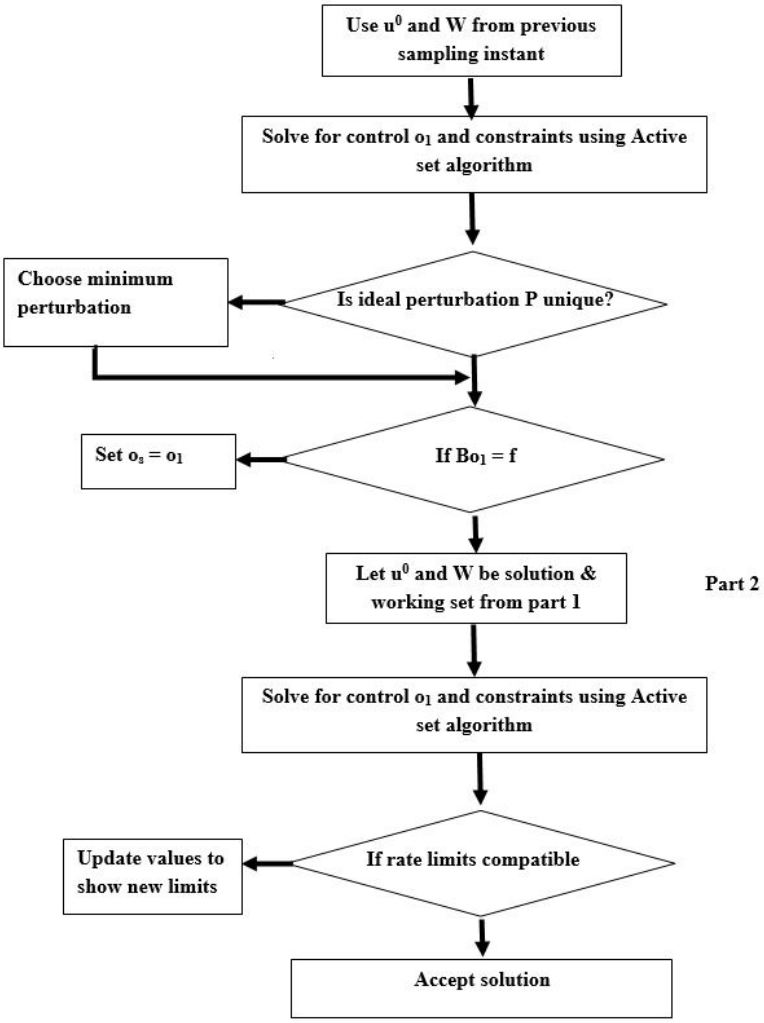


Figure 3.7: Flowchart of sequential least squares

3.4.2 Genetic Algorithm

This optimisation technique was applied to the weights of the control effectiveness matrix. The flowchart in Figure 3.8 provides an overview of a continuous GA. Each block will be discussed in detail. This method has been implemented in [Wilburn, et al., 2014)], [Vishal & Ohri, (2014)], [Saad, et al., (2012)].

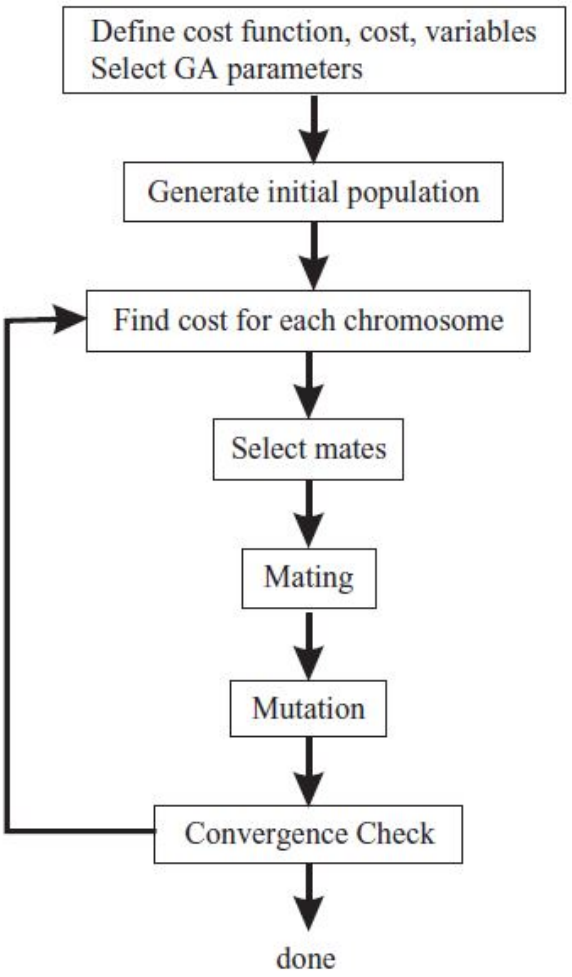


Figure 3.8: Flowchart of a continuous GA [Haupt & Haupt, (2004)]

The aim is to search for an optimal (minimum) solution to objective function (the combined integral square error of each controller) in terms of the system variables. Therefore one begins the process of fitting it to a GA by defining a chromosome as a set of variable values to be optimised. If the chromosome has N_{var} variables (an N-dimensional optimisation problem) given by $p_1, p_2, \ldots, p_{N\text{var}}$ then the chromosome is written as an array with $1 \times N_{\text{var}}$ elements so that $chromosome = [p_1, p_2, p_3, \ldots, p_{N\text{var}}]$.

3.4.2.1 Origin of the Species

To begin the GA, one creates the origin of the species consisting of N_{species} chromosomes. There are two ways to create these individuals. One method is to randomly generate values between some upper and lower limits for each gene or gain. The other method is using heuristically tuned set of individuals which offer some rudimentary operational ability. These individuals are inserted into the species. The subsequent species are generated as previously with a range closer to the initial guess. In this work the second method is used due to the sensitivity of the system to parameter variations. A matrix represents the species with each row in the matrix being a $1 \times N_{\text{var}}$ array (chromosome) of continuous values. Given the origin of the species of N_{species} chromosomes, the full matrix of $N_{\text{species}} \times N_{\text{var}}$ random values is generated by

$$\text{Species} = rand(N_{\text{species}}, N_{\text{var}}) \tag{3.23}$$

All variables are normalized to have values between 0 and 1, the range of a uniform random number generator. The values of a variable are unnormalised in the cost function. If the range of values is between p_{lo} and p_{hi}, then the unnormalised values are given by

$$p = (p_{\text{hi}} - p_{\text{lo}})p_{\text{norm}} + p_{\text{lo}} \tag{3.24}$$

where:

p_{hi} = highest number in the variable range,
p_{lo} = lowest number in the variable range, and
p_{norm} = normalized value of variable.

These chromosomes are then evaluated by the objective function and ranked in ascending order by their corresponding cost function values.

3.4.2.2 Population Selection

From the rankings the population is culled to make space for new offspring as of the N_{species} chromosomes, only the upper N_{keep} are retained for mating with the remaining ones being discarded. This purpose of population selection is to ensure that at each iteration of the algorithm the species consisting of chromosomes evolve over the generations to the optimal members as defined by the cost function.

3.4.2.3 Offspring Generation

Partnering

The N_{keep} number of optimal members form the mating pool. Parents are joined in some arbitrary manner with the two offspring possessing characteristics from each parent. In addition to this these parents proceed to be part of the following generation. The more similar the two parents, the higher the chance the offspring to carry characteristics of the parents.

Mating

This process starts with arbitrarily selecting a variable in the first pair of parents to be the crossover point:

$$\alpha = \{\text{random number} * N_{\text{var}}\} - \text{rounded up to the nearest integer.} \tag{3.25}$$

Say these two parents are:

$$\text{parent}_1 = [p_{f1}, p_{f2}, p_{f3}, \ldots, p_{f\alpha}, \ldots, p_{fNvar}] \tag{3.26}$$

$$\text{parent}_2 = [p_{m1}, p_{m2}, p_{m3}, \ldots, p_{m\alpha}, \ldots, p_{mNvar}] \tag{3.27}$$

where the f and m subscripts differentiate between the *female* and the *male* parent. The joining results in offspring:

$$p_{\text{new1}} = p_{f\alpha} - \beta\,[p_{f\alpha} - p_{m\alpha}] \tag{3.28}$$

$$p_{\text{new2}} = p_{m\alpha} - \beta\,[p_{m\alpha} - p_{f\alpha}] \tag{3.29}$$

where β is a parameter values between 0 and 1. The last step is to finish the crossover with the remaining chromosome.

$$\text{offspring}_1 = [p_{m1}, p_{m2}, \ldots, p_{new1}, \ldots, p_{dNvar}] \tag{3.30}$$

$$\text{offspring}_2 = [p_{d1}, p_{d2}, \ldots, p_{new2}, \ldots, p_{mNvar}] \tag{3.31}$$

If the first variable of the chromosomes is selected, then only the variables to the right of the selected variable are swapped. If the last variable of the chromosomes is selected, then only

the variables to the left of the selected variable are swapped. This method does not allow offspring variables outside the bounds set by the parent unless $\beta > 1$.

Mutations

Within the process of optimisation a problem may occur whereby the GA can converge too quickly into one region of the cost surface. If this area is in the region of the global minimum, that is advantageous. Albeit, an instance may happen where a function has many local minima. This issue must be solved to avoid ending up in a local rather than a global minimum. To avoid this problem of overly fast convergence, one must force the routine to investigate additional parts of the cost surface by arbitrarily introducing changes or mutations, in some of the variables.

The total number of variables that will be mutated in the population are multiplied by the mutation probability. The row and columns of the variables to be mutated are arbitrarily chosen then substituted by a random variable.

New Generation

The initial population for the following generation is ranked by cost. Offspring from the upper half of the parents take the place of the lower half of the chromosomes. The arbitrary variables chosen from the discarded ones are used for mutation.

3.4.2.4 Convergence

Once some stopping criteria (either maximum number of iteration or acceptable error) has been met the algorithm then finds the minimum cost.

Optimisation is based on minimising the performance index calculated by the integral time absolute error (ITAE):

$$ITAE = \int_0^\infty t\,|e(t)|\,dt \tag{3.32}$$

The general parameters for optimization are shown in Tables 3.3 and 3.4.

Table 3.3: Optimization Parameters

Parameter	Value
Matrix value limits	[-5, 5]
Population size	100
Max. iterations	100

The objective function value is calculated over the simulation time of 120 seconds. It is made up of the cumulative error of the system over the duration of the simulation.

Table 3.4: Objective Function Parameters

Control Effectiveness Matrix	Function Minimum Value
B $\delta_e = 0rad$	1800.2
B $\delta_e = 50\%$	1802.66
B $\delta_e = 100\%$	1806.7
B $\delta_a = 0rad$	1800
B $\delta_a = 50\%$	1802.5
B $\delta_a = 100\%$	1831.6
B $\delta_e \& \delta_a = 0rad$	1800.1
B $\delta_e \& \delta_a = 50\%$	1802.8
B $\delta_e \& \delta_a = 100\%$	1826.4

3.4.3 Control Allocation Algorithm Testing

The control allocation system was tested by running the simulation in the steady wings-level flight mode. This typically results in small deflections of elevator and aileron to achieve this trim position for a given throttle setting. Figures 3.9 (a) to (e) show the required control inputs to maintain this flight condition and shows that indeed some elevator and aileron deflections are required. The allocation algorithm can be tested by reducing the primary control surface maximum and minimum deflections to 0 *rad*. The autopilot would be forced to make use of the secondary and tertiary surfaces.

The parameters used in the control allocation algorithm are presented in Table 3.5.

Table 3.5: Control Allocation Parameters

Parameter	**Value**
Control Effectiveness Matrix B $\delta_e = 0rad$	$diag$[-0.15 -0.15 -0.15 -0.15]
Control Effectiveness Matrix B $\delta_e = 50\%$	$diag$[-0.75 -0.75 -0.75 -0.75]
Control Effectiveness Matrix B $\delta_e = 100\%$	$diag$[-0.05 -0.05 -0.05 -0.05]
Control Effectiveness Matrix B $\delta_a = 0rad$	$diag$[1.25 1.25 1.25 1.25]
Control Effectiveness Matrix B $\delta_a = 50\%$	$diag$[2.4 2.4 0.83 0.83]
Control Effectiveness Matrix B $\delta_a = 100\%$	$diag$[5 5 2 2]
Control Effectiveness Matrix B $\delta_e \& \delta_a = 0rad$	$diag$[-0.005 -0.005]
Control Effectiveness Matrix B $\delta_e \& \delta_a = 50\%$	$diag$[-2.323 -2.323]
Control Effectiveness Matrix B $\delta_e \& \delta_a = 100\%$	$diag$[3 1.1]
Elevator Actuator Position Limits $[rad]$	[-0.4363 0.4363]
Aileron Actuator Position Limits $[rad]$	[-0.4363 0.4363]
Velocity Position Limits	[0 1]
Altitude Position Limits $[rad]$	[-0.786 0.786]
Heading Position Limits $[rad]$	[-0.349 0.349]
Elevator & Aileron Actuator Rate Limits $[rad/s]$	2.618

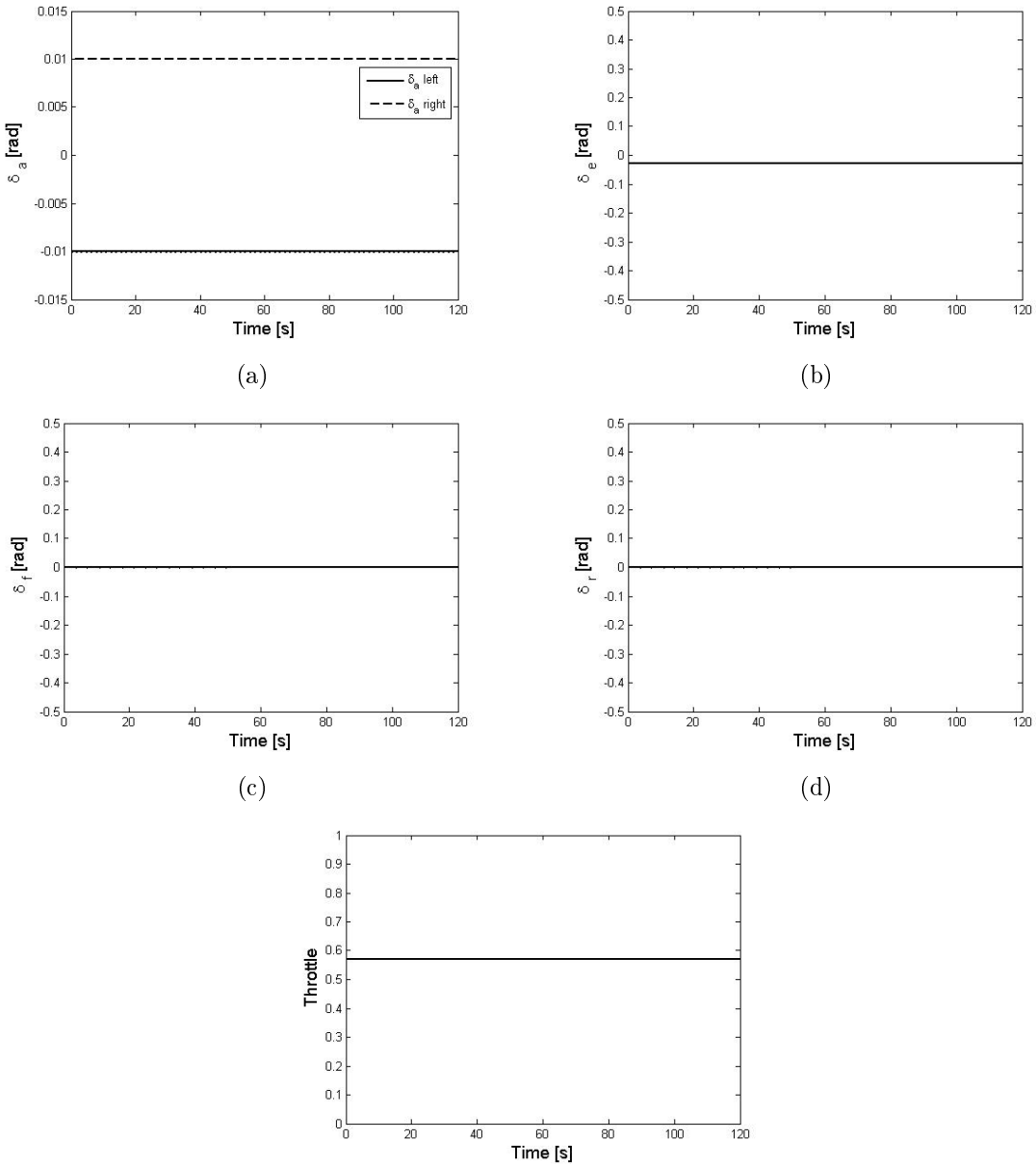


Figure 3.9: Simulated time history of aileron (a), elevator (b), flap (c), rudder (d), and throttle (e) deflections required to maintain level flight with default limitations on all control surfaces

The first control response to be checked was pitch. the maximum and minimum elevator deflections were set to 0 *rad*. Figure 3.10 (a) to (d) shows the required control deflections and that there are large flap and aileron deflections (Figure 3.10 (b) and (c)) that occur to generate a pitching moment to compensate for lack of elevator.

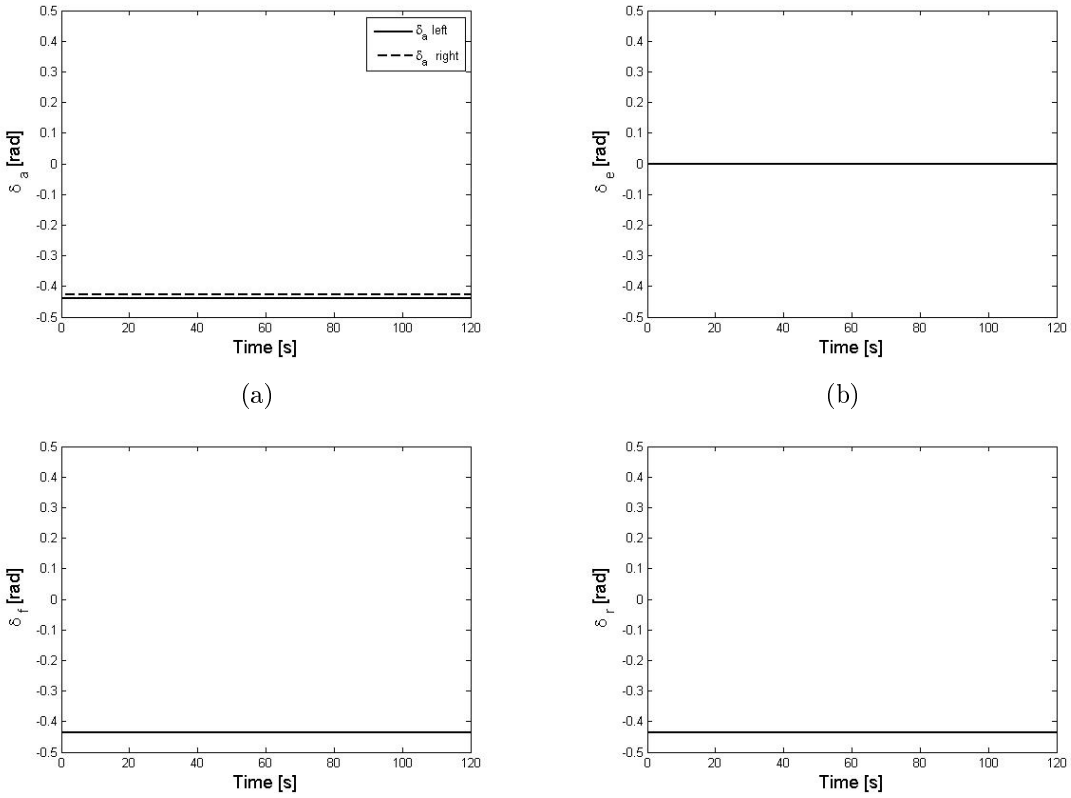


Figure 3.10: Simulated time history of aileron (a), elevator (b), flap (c), and rudder (d) deflections required to maintain stable flight with no elevator actuation

The simulation was re-run with maximum and minimum elevator deflections restored but now reducing the aileron deflections to 0 *rad*. Figure 3.11 (a) to (d) shows the required deflection without the ailerons, the flaps are used to achieve the desired roll effort.

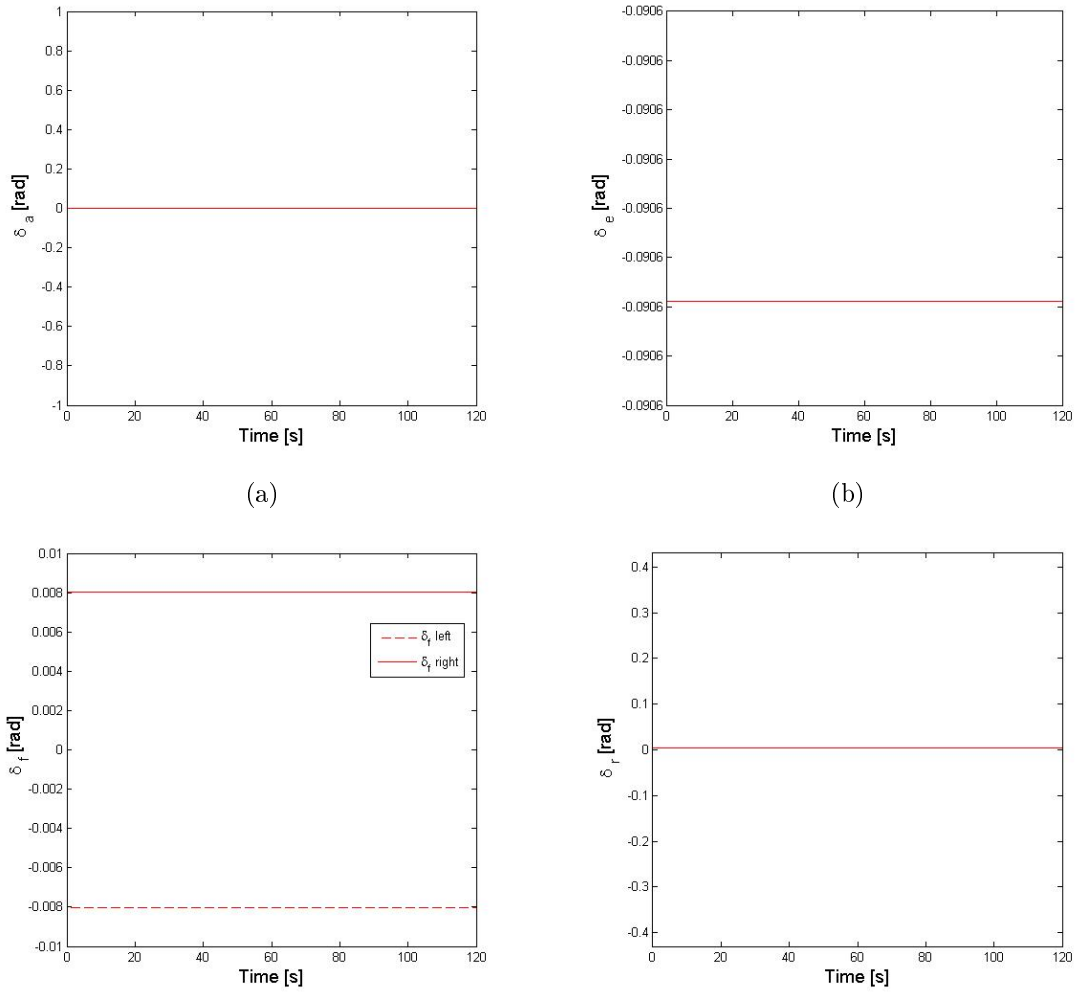


Figure 3.11: Simulated time history of aileron (a), elevator (b), flap (c), and rudder (d) deflections required to maintain stable flight with no aileron actuation

In order to further check control surface actuation, the simulation was run with aileron and flap maxima and minima set to 0 *rad*. This simulation result, shown in Figure 3.12 (a) to (e), shows that the elevators are now used to generate some of the required control effort.

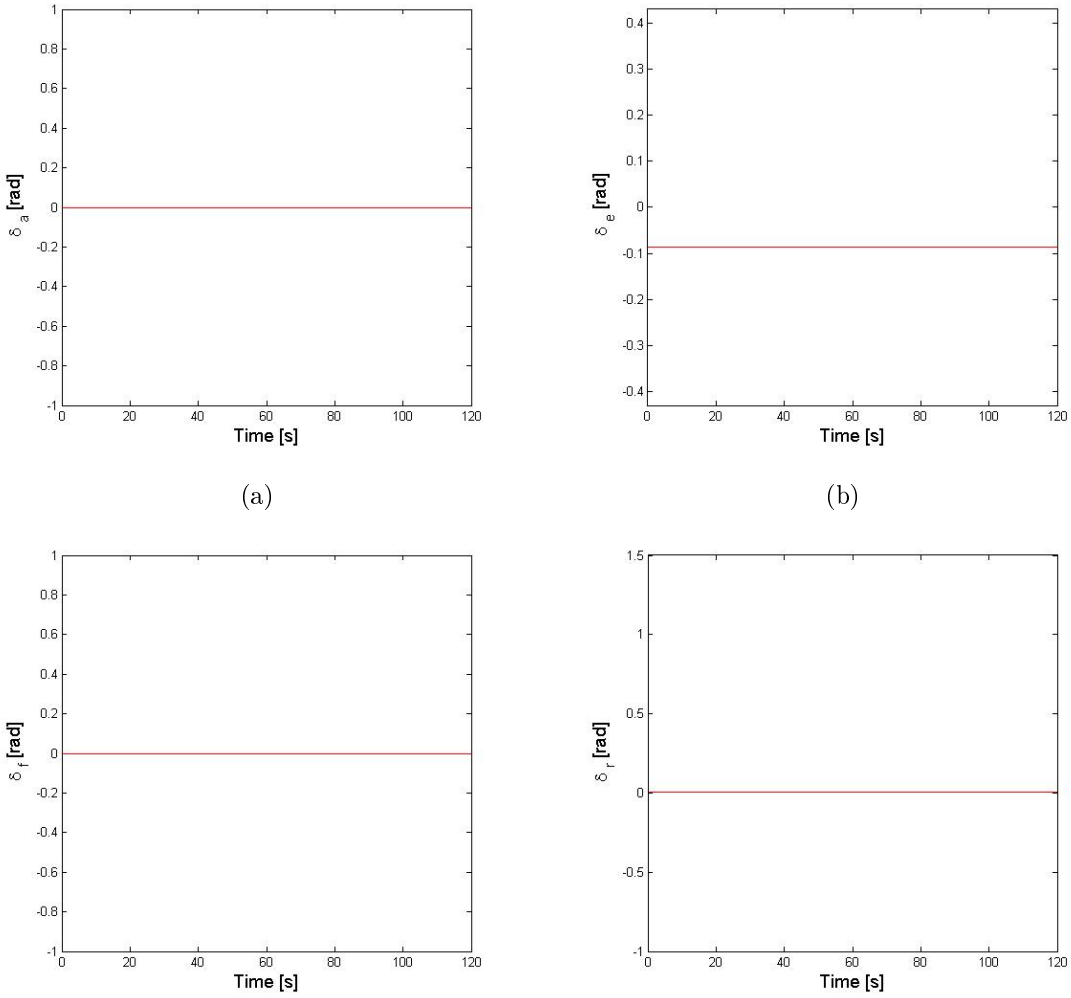


Figure 3.12: Simulated time history of aileron (a), elevator (b), flap (c), and rudder (d) deflections required to maintain stable flight with no aileron or flap actuation

In the above results it can be observed that the secondary and tertiary control actuators are working in the correct sense and when required and thus it can be concluded that the control allocation algorithm is functioning as intended.

3.5 Neural Network Controller Development

The work flow for the general neural network design process has seven primary steps:

1. Collect data
2. Create the network

3. Configure the network
4. Initialize the weights and biases
5. Train the network
6. Validate the network (post-training analysis)
7. Use the network

This process is represented in the following aspects.

3.5.1 System Identification

The first stage of neural network controller development is to train a neural network to represent the forward dynamics of the plant. The error between the plant output and the neural network output is used as the neural network training signal. The process is represented by Figure 3.13.

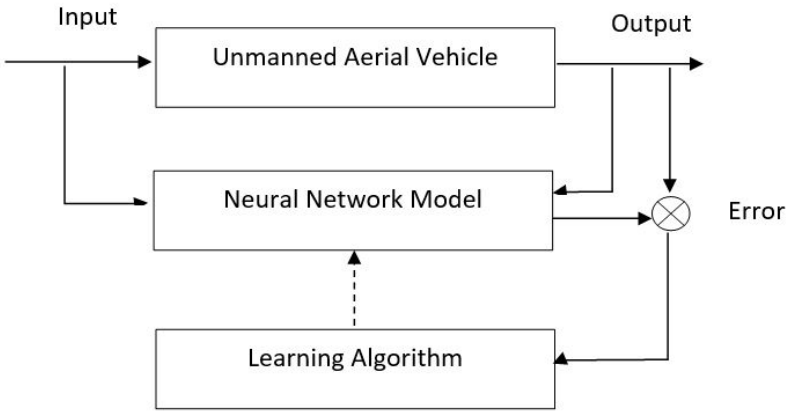


Figure 3.13: System Identification

The neural network plant model uses previous inputs and previous plant outputs to predict future values of the plant output. The structure of a neural network plant model is given in Figure 3.14.

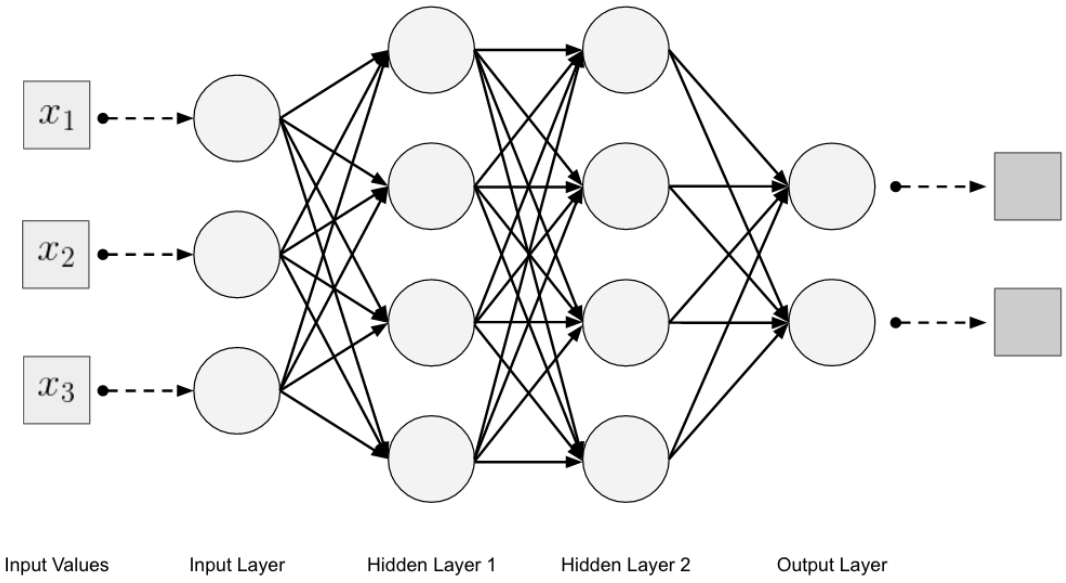


Figure 3.14: Neural Network Model [Gibson & Patterson, (2017)]

After the data has been collected, the next step in training a network is to create the network object. A multilayer neural network is created. The network object is configured by specifying the number of layers and the number of neurons in the hidden layer. The weights and biases of the network need to be initialised before the network is trained. When the network weights and biases are initialized, the network is ready for training. The process of training a neural network involves tuning the values of the weights and biases of the network to optimize network performance.

3.5.2 Radial Basis Function Neural Network

This section presents the structure of the radial basis function neural network-based (RBFNN) controller that consists of: a PID controller used for aircraft parameter control and a RBFNN used for regulating PID parameters: k_p , k_i and k_d . A locally-tuned overlapping receptive field is found in parts of the cerebral cortex, in the visual cortex, and in other parts of the brain. The radial basis function neural network model is based on characteristics of these biological systems. RBFNNs are local approximators where each basis function can only locally affect the network output, therefore, the construction of the RBFNNs and the resulting approximation are very dependent on the compact set they work on. The classical approach to the network implementation is to fix the number of hidden neurons *a priori* along with its centres and widths, based on some properties of the input data, and then estimate the weights using algorithm such as scaled conjugate gradient (SCG), backpropagation (BP), least mean squares (LMS), and [Sundararajan, et al, (2002)]. An RBFNN is a feedforward

artificial neural network consisting of three layers that uses radial basis functions as activation functions. The output of the network is a linear combination of radial basis functions of the inputs and neuron parameters. An important factor in system control and identification is the number of basis functions required for RBF networks to approximate a given function [Ge, et al., (2002)].

Figure 3.15 illustrates a RBFNN. There, the inputs are $x_i, i = 1, 2, \ldots, n$, and the output is $y = F_{\text{process}}(x)$. The processing by the whole RBFNN is represented by F_{process}. Let $x = [x_1, x_2, \ldots, x_n]$. The input to the ith receptive field unit or radial basis function is x, with output $R_i(x)$. The receptive field unit has weight b_i. An assumption of nR receptive field units is made. Hence, from Figure 3.15 [Passino, (2005)],

$$y = F_{\text{process}}(x, \Lambda) = \sum_{i=1}^{nR} b_i R_i(x) \tag{3.33}$$

is the output of the RBFNN, and Λ consists of the b_i parameters as well as the parameters of the receptive field units.

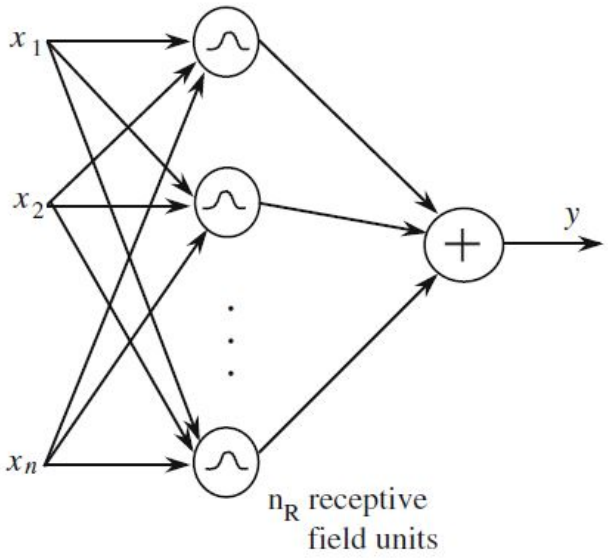


Figure 3.15: Radial basis function neural network model [Passino, (2005)]

The "receptive field unit" $R_i(x)$ is:

$$R_i(x) = exp\left(-\frac{|x - c^i|^2}{(\sigma)^2}\right) \tag{3.34}$$

where $c^i = \left[c_1^i, c_2^i, \ldots, c_n^i\right]^T$, σ^i is a scalar, and if z is a vector then $|z| = \sqrt{z^T z}$. For the case where $n = 1, c^1 = \left[c_1^1\right] = [2]$, and $\sigma^1 = 0.1$, $R_1(x)$ is shown in Equation 3.35. As x moves away from c_1^1, $R_1(x)$ decreases with the rate of decrease dictated by the size of σ^1 (a smaller value of σ^1 results in a steeper slope on the function, and so its value will decrease quicker as x moves away from c_1^1).

If one makes σ^i also depend on the input dimension (which makes sense if the input dimensions are scaled differently). One would have $\sigma^i = [\sigma_1^i, \sigma_2^i, \ldots, \sigma_n^i]$ and

$$R_i(x) = exp\left(-\sum_{j=1}^{n} \frac{\left(x - c^i\right)^2}{(\sigma_j^i)^2}\right) \tag{3.35}$$

where σ_j^i is the spread for the jth input for the ith receptive field unit.

Controller Input Choice and Control System Structure

The error e is used as one input to the radial basis function neural network, and the derivative of that error the other input. Hence, the inputs to the radial basis function neural network will be

$$e = \psi_r - \psi \tag{3.36}$$

and

$$e = \dot{\psi}_r - \dot{\psi} \tag{3.37}$$

The backward difference approximation to the derivative is denoted by $c(kT)$,

$$\dot{e} \approx \frac{e(kT) - e(kT - T)}{T} = c(kT) \tag{3.38}$$

where $T = 0.02sec$ (this corresponds to the sampling period of the plant during simulation). and k is an index for the time step (this is an Euler approximation of the derivative and T is the sampling period). As is standard in discrete-time systems, for convenience one will often use k rather than kT as the argument for the signals. With this, one can denote the radial basis function neural network by

$$\delta(k) = F_{\text{process}}(e(k), c(k)) \tag{3.39}$$

The parameter vector argument is omitted for convenience.

A radial basis function neural network of Equation (3.33) with $n = 2$ inputs, and $nR = 121$ so we will have to pick 121 strengths $b_i, i = 1, 2, \ldots, 121$. For the $R_i(e(k), c(k))$ we use Equation (3.34) and create a uniform grid for the c^i centers, $i = 1, 2, \ldots, 121$. To pick the grid points, assume that $e(k)$ lies in the range

$$e(k) \in \left[-\frac{\pi}{2}, \frac{\pi}{2}\right] \tag{3.40}$$

The angular rate (in rad/s) of movement limited by the actuators is

$$\dot{e}(k) \in [-0.01, 0.01] \tag{3.41}$$

A uniform grid with its four outer corners at $(-\frac{\pi}{2}, -0.01), (-\frac{\pi}{2}, 0.01), (\frac{\pi}{2}, 0.01)$, and $(\frac{\pi}{2}, 0.01)$ is created with $nR = 121$ centers uniformly placed at the grid points (i.e., with 11 points along each input dimension). For the receptive field units one uses spreads σ_j^i (i.e., so that the size of the spread depends on which input dimension is used) with

$$\sigma_1^i = 0.7\frac{\pi}{\sqrt{nR}} \tag{3.42}$$

and

$$\sigma_2^i = 0.7\frac{\pi}{\sqrt{nR}} \tag{3.43}$$

for $i = 1, 2, \ldots, 121$. For σ_1^i, the σ_1^1 factor makes the spread size depend on the number of grid points along the e input dimension (similarly for σ_2^i), and the 0.7 factor was chosen to get a smooth interpolation between adjacent receptive field units.

The input-output mapping of the radial basis function neural network is shaped by the choice of the scaling parameters b_i. Suppose that one views the parameters as being loaded in a matrix

$$\begin{bmatrix} b_1 & b_{12} & b_{23} & b_{34} & b_{45} & b_{56} & b_{67} & b_{78} & b_{89} & b_{100} \\ b_{111} & & & & & & & & & \\ b_2 & & & & & \cdots & & & & \\ b_{112} & & & & & & & & & \\ \vdots & & & & & \cdots & & & & \\ \vdots & & & & & & & & & \\ b_{11} & b_{22} & b_{33} & b_{44} & b_{55} & b_{66} & b_{77} & b_{88} & b_{99} & b_{110} \\ b_{121} & & & & & & & & & \end{bmatrix} \tag{3.44}$$

Adequate tuning of the PID parameters is paramount and the designed RBF network is able to tune k_p, k_i and k_d accurately in different situations by using the above matrix.

Figures 3.16 to 3.19 illustrate the centres of the RBFNN-based controllers. The RBF parameters figures for the pitch and bank controllers are similar as the controller's output, input and rate limits are similar.

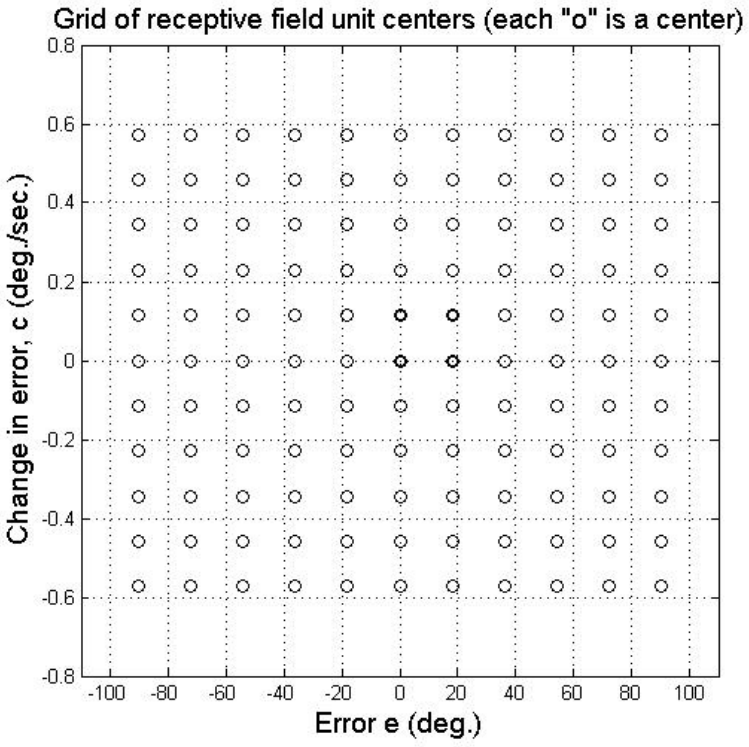


Figure 3.16: RBF centres for pitch and roll controllers

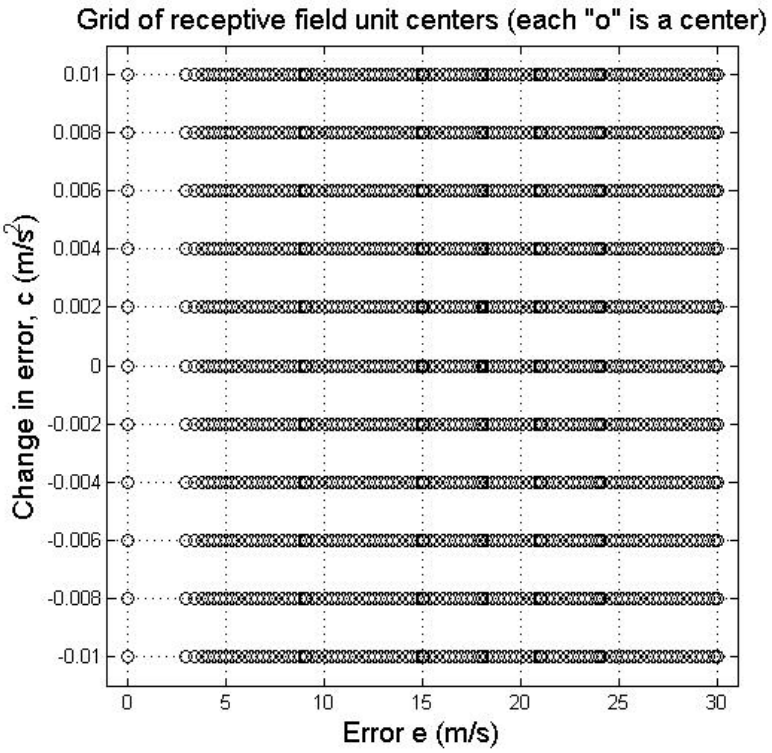


Figure 3.17: RBF centres for velocity controller

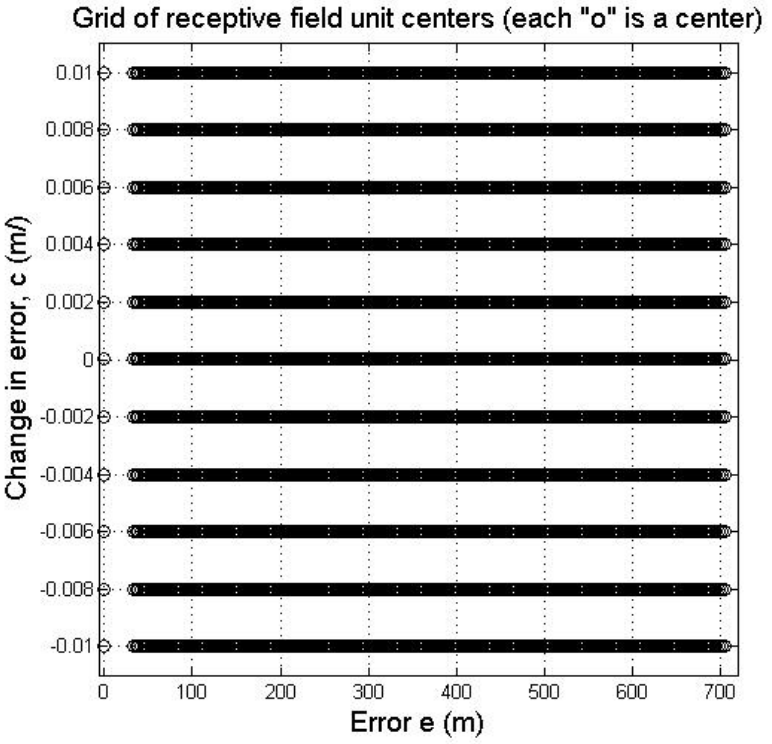


Figure 3.18: RBF centres for altitude controller

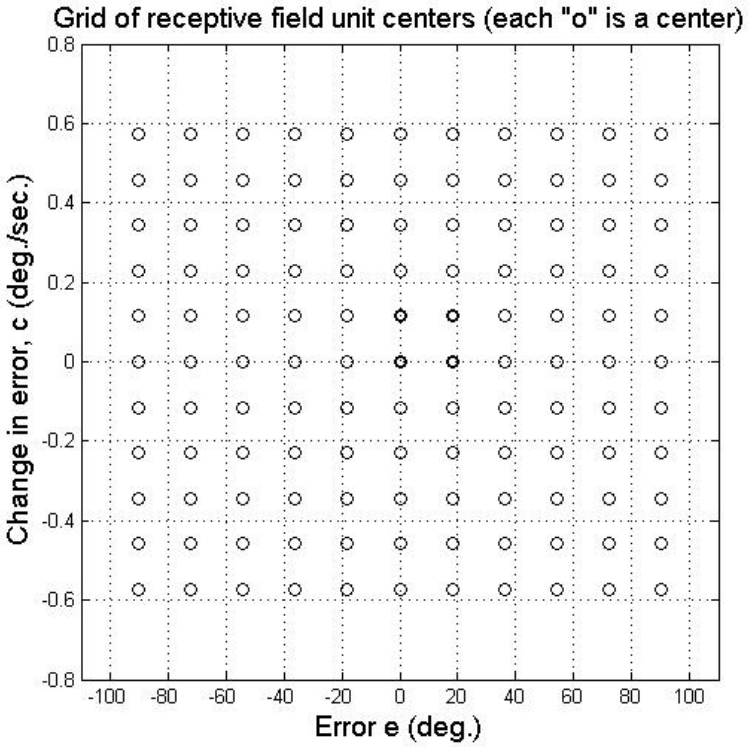


Figure 3.19: RBF centres for heaing controller

Figures 3.20 to 3.23 illustrate the mapping of the RBFNN controllers.

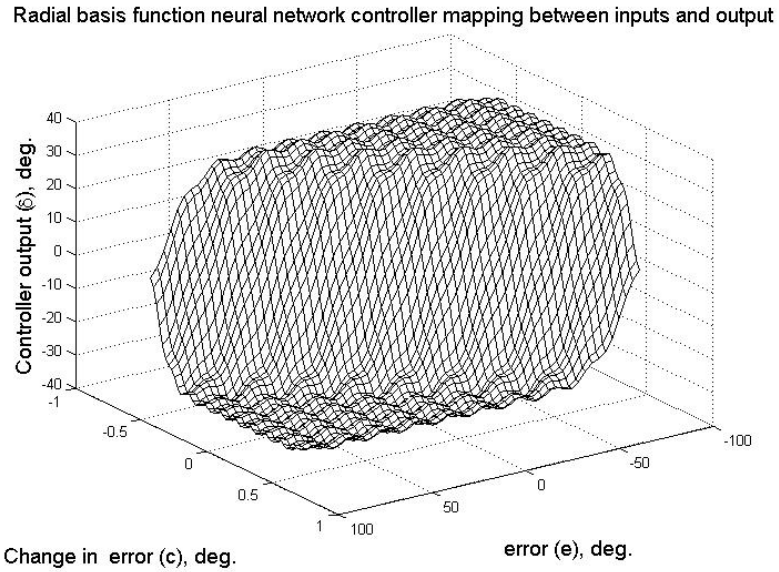


Figure 3.20: RBF mapping for pitch and roll controllers

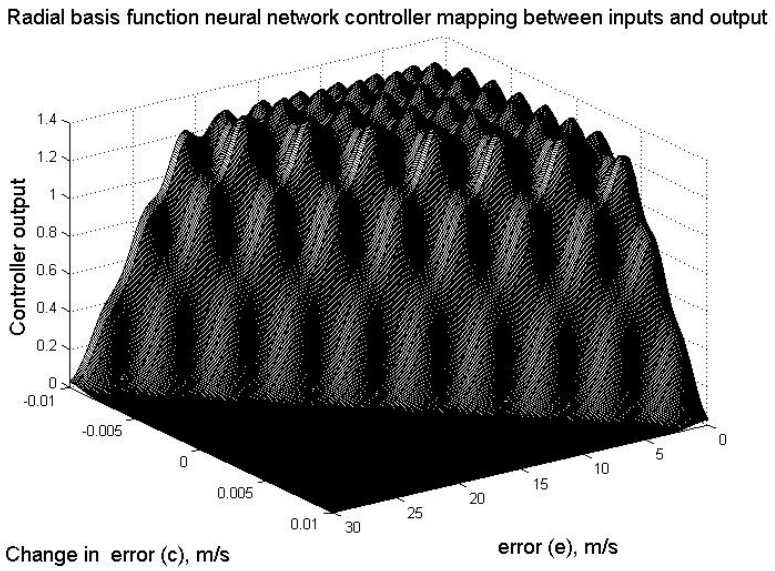


Figure 3.21: RBF mapping for velocity controller

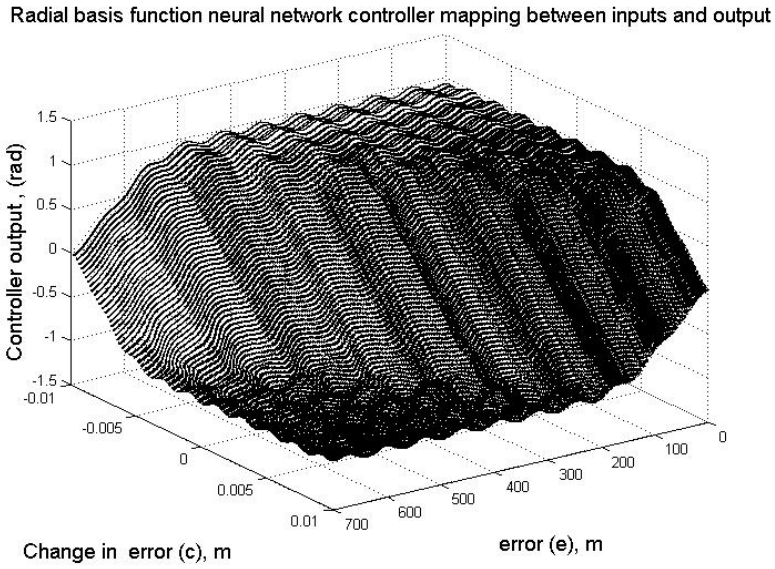


Figure 3.22: RBF mapping for altitude controller

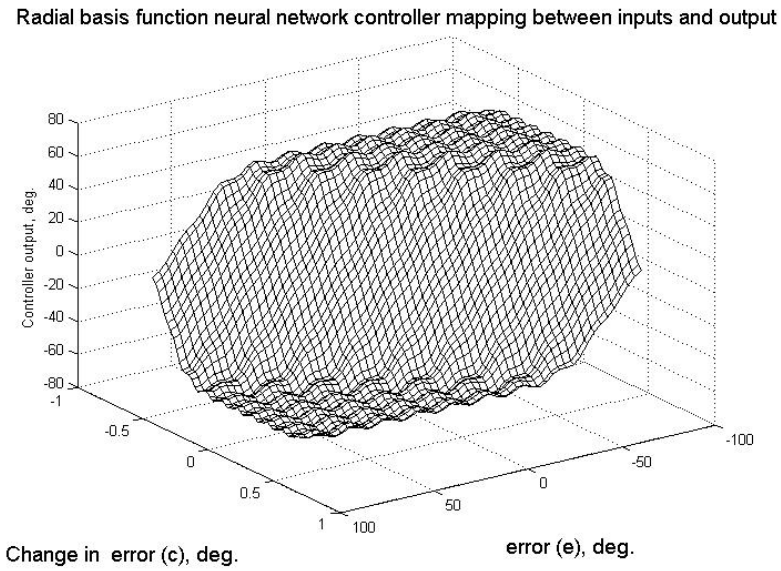


Figure 3.23: RBF mapping for heading controller

Figures 3.24 to 3.27 illustrate one of the central receptive field units of the RBFNN controllers.

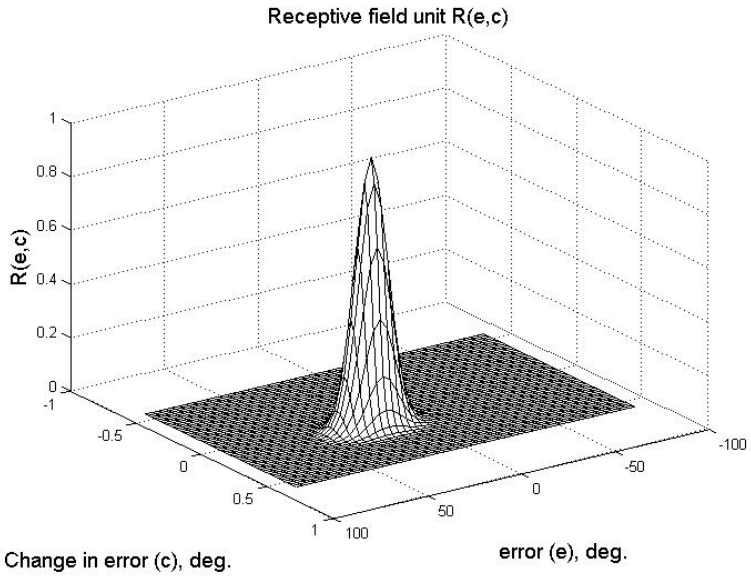


Figure 3.24: RBF central receptive field unit for pitch and roll controllers

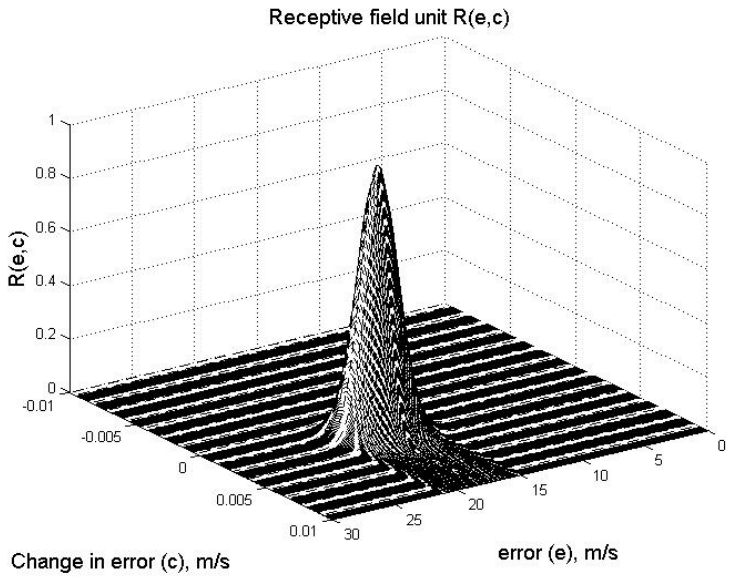


Figure 3.25: RBF central receptive field unit for velocity controller

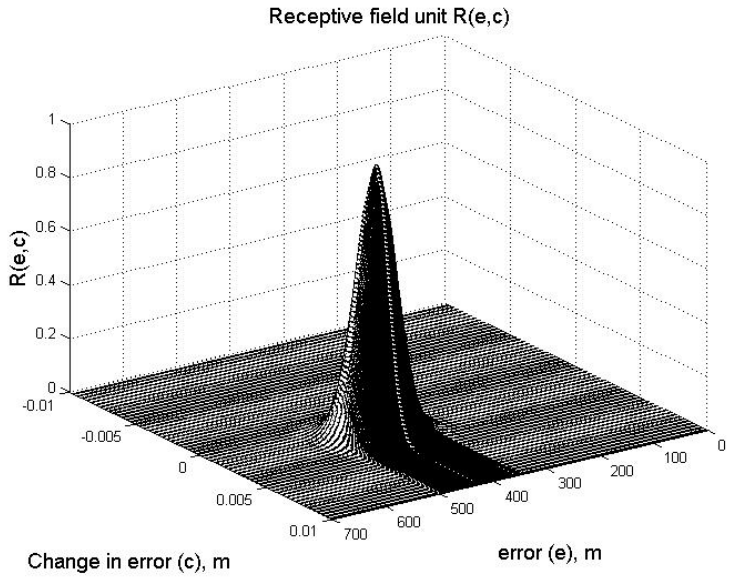


Figure 3.26: RBF central receptive field unit for altitude controller

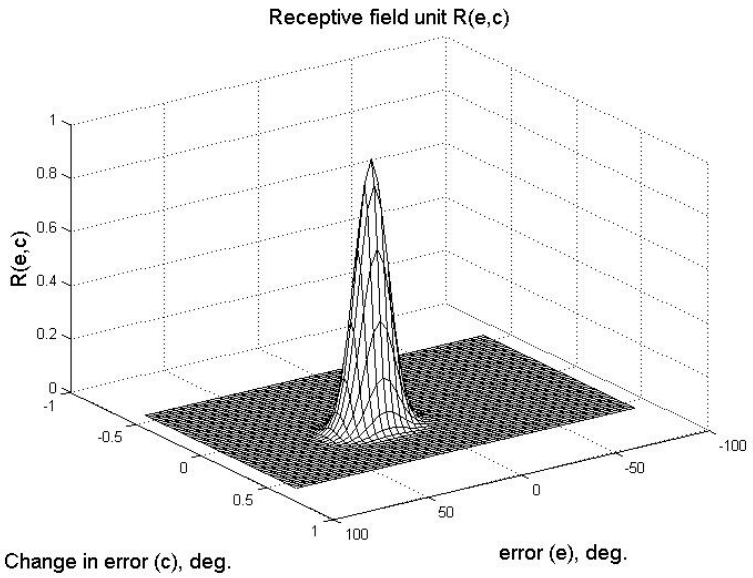


Figure 3.27: RBF central receptive field unit for heading controller

3.5.3 Backpropagation Neural Network (BPNN)

In order to achieve an optimal system performance in terms of speed of response, system stability, disturbance rejection and small overshoot, a neural network proportional-integral-derivative (PID) controller with back-propagation-based weight updating algorithm is proposed. The structure and algorithm of the BPNN-PID control is introduced in this section. The performance of the BPNN-PID controller is tested by computer simulation using Matlab.

3.5.3.1 Structure of the BPNN-PID controller

Figure 3.29 presents the structure of the proposed intelligent PID controller based on BPNN learning algorithm. It contains two parts: 1) a classical PID controller and 2) BPNN. The PID controller is used to control the UAV. The control performance depends on the setting of PID control parameters k_p, k_i and k_d which can be auto-tuned by the BPNN. The BPNN uses an on-line training algorithm based on a scaled conjugate gradient approach to update network weights and ensures that the designed neural network is able to calculate the desired PID control parameters for the PID controller. Therefore, in this control approach, by combining the classical PID control and the intelligent BPNN the targeted system output can be tracked with a guaranteed stability [Yang, (2014)]. A BPNN's structure is shown in Figure 3.28. The output of each neuron is a combination of the numbers of neurons of the previous level multiplied by its corresponding weights. The input values are converted into output signals with the calculations of activation functions [Che, et al., (2011)]. Figure 3.29 illustrates a

BPNN-based PID control scheme. Figure 3.30 illustrates a BPNN-based PID control scheme combined with control allocation.

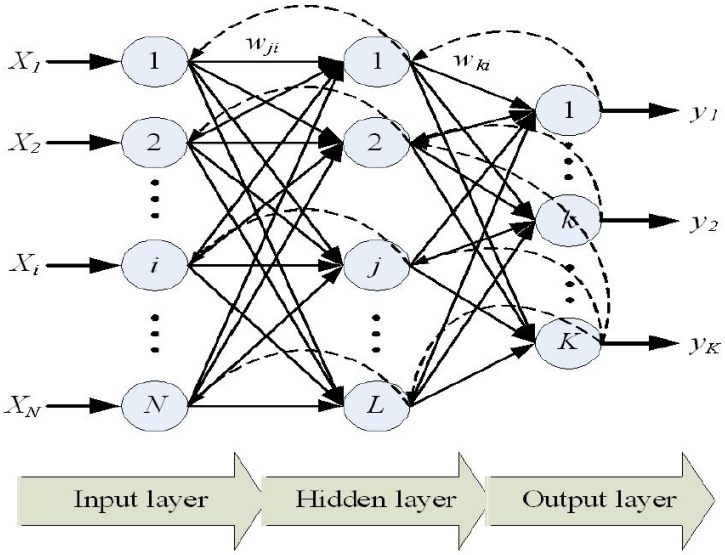


Figure 3.28: BPNN Structure [Che, et al., (2011)]

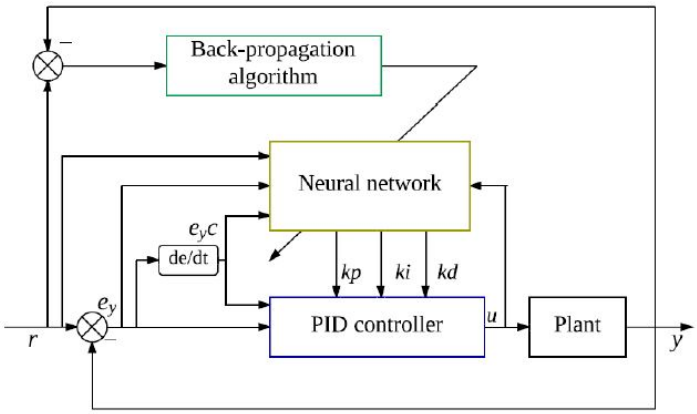


Figure 3.29: BPNN-based PID control scheme [Yang, (2014)]

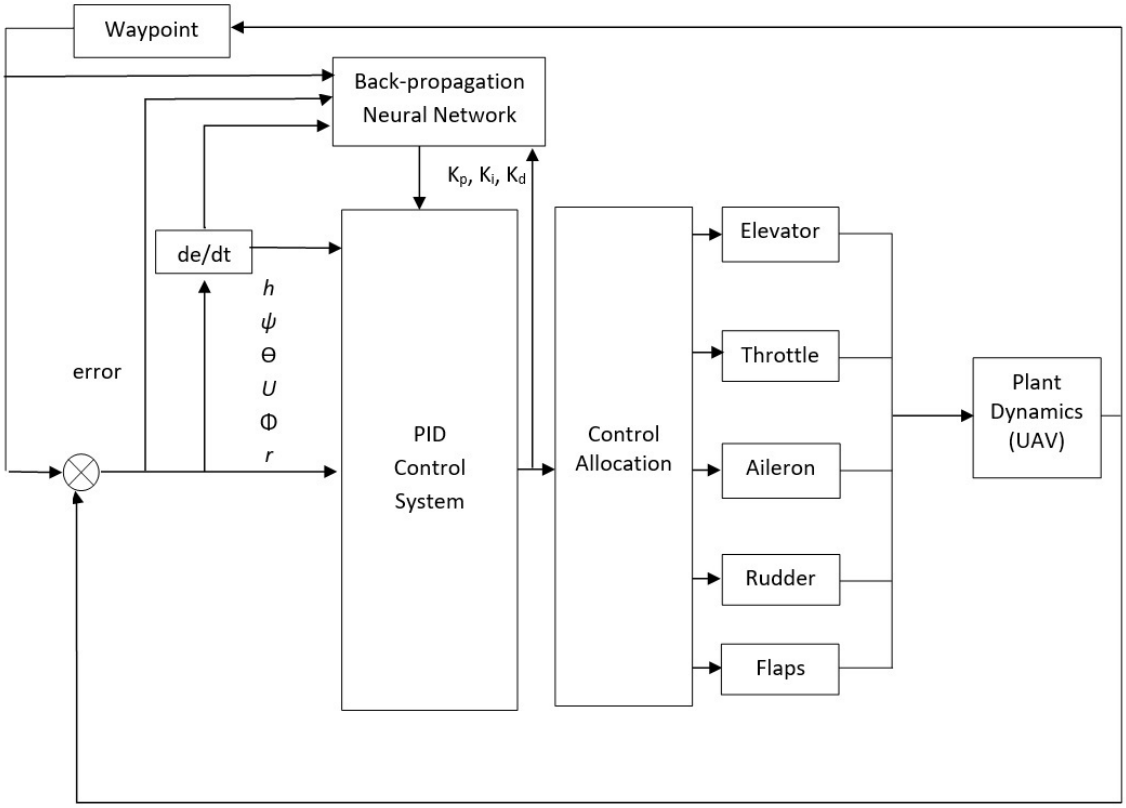


Figure 3.30: BPNN-based PID control scheme with control allocation

3.5.4 Network Training

Training data is generated by applying a series of random step inputs to the Simulink plant model. The training of the model then occurs using a scaled conjugate gradient algorithm (SCG).

3.5.4.1 Scaled Conjugate Gradient Algorithm

A variation of a SCG used by [Moller, (1993)], [Danilin, et al., (2014)], [Kapela, et al., (2017)] , which avoids the line-search per learning iteration by using a Levenberg-Marquardt approach in order to scale the step size. Most of the optimisation methods used to minimise functions are based on the same strategy. The minimisation is a local iterative process in which an approximation to the function in a neighbourhood of the current point in weight space is minimised. The approximation is often given by a first or second-order Taylor expansion of the function. The idea of the strategy is illustrated in the pseudo-algorithm presented below, which minimises the error function $E(w)$.

1. Choose weight vector w_1 and set $k = 1$.
2. Determine a search direction p_k and a step size α_k so that $E(w_k + \alpha_k p_k) < E(w_k)$.
3. Update vector: $W_{k+1} = w_k + \alpha_k p_k$.
4. If $E'(w_k) \neq 0$ then set $k = k + 1$ and go to 2 else return w_{k+1} as the desired minimum,

Determining the next current point in this iterative process involves two independent steps. First a search direction has to be determined, i.e., in what direction in weight space do we want to go in the search for a new current point. Once the search direction has been found we have to decide how far to go in the specified search direction, i.e., a step size has to be determined. The search direction and the step size are chosen by using information from the second-order approximation $E(w + y) = E(w) + E'(w)T^y + \frac{1}{2}y^T E''(w)y$.

Quadratic functions have some nice properties that general functions do not necessarily have. Denote the quadratic approximation to E in a neighbourhood of a point w by $E_{qw}(y)$, so that $E_{qw}(y)$ is given by

$$E_{qw}(y) = E(w) + E'(w)T^y + \frac{1}{2}y^T E''(w)y \tag{3.45}$$

In order to determine minima to $E_{qw}(y)$ the critical points for $E_{qw}(y)$ must be found, i.e., the points where

$$E'_{qw}(y) = E''(w)y + E'(w) = 0 \tag{3.46}$$

The critical points are the solution to the linear system defined by Equation (3.46). If a conjugate system is available the solution can be simplified considerably. Let $p_1, \ldots, p_N$ be a conjugate system. Because $p_1, \ldots, p_N$ form a basis for $\Re^N$, the step from a starting point y_1 to a critical point y_* can be expressed as a linear combination of $p_1, \ldots, p_N$

$$y_* - y_1 = \sum_{i-1}^{N} \alpha_i p_1, \alpha_i \in \Re \tag{3.47}$$

Multiplying Equation (3.47) with $p_j^T E''(w)$ and substituting $E'(w)$ for $E''(w)y_*$ and then rearranging the subject of the formula gives

$$\alpha_j = \frac{p_j^T(E'(w)E''(w)y_1)}{p_j^T E''(w)p_j} = \frac{p_j^T E'_{qw}(y_1)}{p_j^T E''(w)p_j} \tag{3.48}$$

The critical point y_* can be determined in N iterative steps using Equations (3.47) and (3.48). Unfortunately y_* is not necessarily a minimum, but can be a saddle point or a maximum. Only if the Hessian matrix $E''(w)$ is positive definite then $E_{qw}(y)$ has a unique global minimum. This can be realized by:

$$E_{qw}(y) = E_{qw} + (y_*) + \frac{1}{2}(yy_*)^T E''(w)(yy_*) \tag{3.49}$$

It follows from Equation (3.49) that if y_* is a minimum then $\frac{1}{2}(yy_*)^T E''(w)(yy_*) > 0$ for every y, hence $E''(w)$ has to be positive definite. The Hessian $E''(w)$ will in the following if not told otherwise be assumed to be positive definite. The intermediate points $y_{k+1} = y_k + \alpha_k p_k$ given by the iterative determination of y_* are in fact minima for $E_{qw}(y)$ restricted to every k-plane π_k: $y = y_1 + \alpha_1 p_1 + \ldots + \alpha_k p_k$.

The formulated conjugate gradient algorithm consists of: 1) Select an initial weight vector y_1 and a conjugate system $p_1, \ldots, p_N$. 2) Find successive minima for E_{qw} on the planes $p_1, \ldots, p_N$, where π_k, $1 \leq k \leq N$, is given by $y = y_1 + \alpha_1 p_1 + \ldots + \alpha_k p_k$, $\alpha_i \in \Re$. The algorithm assures that the global minimum for a quadratic function is detected in at most N iterations. 3) If all the eigenvalues to the Hessian $E''(w)$ falls into multiple groups with values of the same size, then there is great possibility that the algorithm terminates in much less than N iterations.

It is not necessary to know the conjugate weight vectors $p_1, \ldots, p_N$ in advance, they can be determined recursively. Initially p_1 is set to the steepest descent vector $E'_{qw}(y_1)$. p_{k+1} is then determined recursively as a linear combination of the current steepest descent vector $E'_{qw}(y_{k+1})$ and the previous direction p_k. More precisely is p_{k+1} chosen as the orthogonal projection of $E'_{qw}(y_{k+1})$ on the $(N\ k)$-plane π_{N-k} conjugate to π_k.

For each iteration the above described algorithm is applied to the quadratic approximation E_{qw} of the global error function E in the current point w in weight space. Because the error function $E(w)$ is non-quadratic the algorithm will not necessarily converge in N steps. If the algorithm has not converged after N steps, the algorithm is restarted, i.e., initializing p_{k+1} to the current steepest descent direction r_{k+1}.

The first version of the SCG algorithm is

1. Choose initial weight vector w_1.
   Set $p_1 = r_1 = E'(w_1), k = 1$

2. Calculate second order information:

$$s_k = E''(w_k)p_k,$$
$$\delta_k = p_k^T s_k.$$

3. Calculate step size:

$$\mu_k = p_k^T r_k,$$
$$\alpha_k = \frac{\mu_k}{\delta_k}.$$

4. Update weight vector:

$$w_{k+1} = w_k + \alpha_k p_k,$$
$$r_{k+1} = E'(w_{k+1}).$$

5. If k mod $N = 0$ then restart algorithm: $p_{k+1} = r_{k+1}$
else create new conjugate direction:

$$\beta_k = \frac{|r_{k+1}|^2 r_{k+1} r_k}{\mu_k},$$
$$p_{k+1} = r_{k+1} + \beta_k p_k.$$

6. If the steepest descent direction $rk \neq 0$ then set $k = k + 1$ and go to 2 else terminate and return w_{k+1} as the desired minimum.

When the algorithm shows poor development the formula forces the algorithm to restart because of the following relation

$$r_{k+1} = r_k \Rightarrow \beta_k = 0 \Rightarrow p_{k+1} = r_{k+1}$$

For each iteration the Hessian matrix $E''(wk)$ has to be calculated and stored. Unfortunately it is not desirable to calculate the Hessian matrix explicitly, because of the calculation complexity and memory usage involved. The solution to the problem is to estimate the $E''(w_k)p_k$ with

$$s_k = E''(w_k)p_k = \frac{E'(w_k + \sigma_k p_k)E'(w_k)}{\sigma_k}, 0 < \sigma_k << 1 \tag{3.50}$$

This algorithm is modified by using a Levenberg-Marquardt algorithm to improve its poor performance when the current point is very far from the desired minimum. The idea is to introduce a scalar λ_k, which is supposed to regulate the indefiniteness of $E''(wk)$. This is done by setting

$$s_k = \frac{E'(w_k + \sigma_k p_k)E'(w-k)}{\sigma_k} + \lambda_k p_k \tag{3.51}$$

and for each iteration adjusting λ_k looking at the sign of δ_k, which directly reveals if $E''(w_k)$ is not positive definite. If $\delta_k \leq 0$ then λ_k should be increased and s_k is estimated again. Call the new s_k for $\bar{s}_k$ and the raised λ_k for $\bar{\lambda}_k$. Then $\bar{s}_k$ is

$$\bar{s}_k = s_k + (\bar{\lambda}_k \lambda_k) p_k \tag{3.52}$$

The Levenberg-Marquardt algorithm has to raise λ_k with a constant factor, whenever the Hessian is not positive definite, because the algorithm contains no information about how much λ_k must be raised. This is however possible when using this approach in the above algorithm, as the values of δ_k indirectly reveal how much λ_k should be raised. Assume in a given iteration that $\delta_k \leq 0$. It is possible to determine how much λ_k should be raised in order to get $\delta_k > 0$.

$$\bar{\delta}_k = p_k^T \bar{s}_k = p_k^T (s_k + (\bar{\lambda}_k \lambda_k) p_k) = \delta_k + (\bar{\lambda}_k \lambda_k) |p_k|^2 > 0 \Rightarrow \bar{\lambda}_k > \lambda_k - \frac{\delta_k}{|p_k|^2} \tag{3.53}$$

Equation (3.53) implies that if λ_k is raised with more than $-\frac{\delta_k}{|p_k|^2}$ then $\bar{\delta}_k > 0$. $\bar{\lambda}_k$ then becomes

$$\bar{\lambda}_k = 2\left(\lambda_k - \frac{\delta_k}{|p_k|^2}\right) \tag{3.54}$$

which leads to:

$$\bar{\delta}_k = -\delta_k + \lambda_k |p_k|^2 > 0 \tag{3.55}$$

The step size is given by:

$$\alpha_k = \frac{\mu_k}{p_k^T s_k - \lambda_k |p_k|^2} \tag{3.56}$$

The values of λ_k directly scale the step size in the way, that the bigger λ_k, the smaller the step size, which agrees well with our intuition of the function of λ_k. Because λ_k scales the Hessian matrix in an artificial way, the quadratic approximation E_{qw} on which the algorithm relies may in some points not be a very good approximation to $E(w)$. In order to get a good approximation, a mechanism to raise a lower λ_k even when the Hessian is positive definite is needed. Define

$$\Delta_k = \frac{2\delta_k \left[E(w_k)E(w_k + \alpha_k p_k)\right]}{\mu_k^2} \tag{3.57}$$

Δ_k is a measure of how well $Eqw(\alpha_k p_k)$ approximates $E(w_k + \alpha_k p_k)$ in the sense, that the closer Δ_k is to 1, the better is the approximation. λ_k is raised and lowered following the formula

$$\text{if}\Delta_k > 0.75 \text{ then } \lambda_k = \frac{1}{2}\lambda_k \tag{3.58}$$

$$\text{if}\Delta_k < 0.25 \text{ then } \lambda_k = 4\lambda_k \tag{3.59}$$

The final SCG algorithm is as follows:

1. Choose weight vector w_1 and scalars $\sigma > 0$, $\lambda_1 > 0$ and $\bar{\lambda}_1 = 0$.
   Set $p_1 = r_1 = E'(w_1)$, $k = 1$ and success = true.

2. If success = true then calculate second-order information:

$$\sigma_k = \frac{\sigma}{p_k} \tag{3.60}$$

$$s_k = \frac{E'(w_k + \sigma_k p_k)E'(w_k)}{\sigma_k} \tag{3.61}$$

$$\delta_k = p_k^T s_k \tag{3.62}$$

3. Scale s_k:

$$s_k = s_k + (\lambda_k \bar{\lambda}_k)p_k, \tag{3.63}$$

$$\delta_k = \delta_k + (\lambda_k \bar{\lambda}_k)\left|p_k\right|^2 \tag{3.64}$$

4. If $\delta_k \leq 0$ then make the Hessian matrix positive definite:

$$s_k = s_k + (\lambda_k 2\frac{\delta_k}{\left|p_k\right|^2})p_k, \tag{3.65}$$

$$\bar{\lambda}_k = 2(\lambda_k \frac{\delta_k}{\left|p_k\right|^2}), \tag{3.66}$$

$$\delta_k = -\delta_k + \lambda_k \left|p_k\right|^2, \tag{3.67}$$

$$\lambda_k = \bar{\lambda}_k \tag{3.68}$$

5. Calculate step size:

$$\mu_k = p_k^T r_k, \tag{3.69}$$

$$\alpha_k = \frac{\mu_k}{\delta_k} \tag{3.70}$$

6. Calculate the comparison parameter:

$$\Delta_k = \frac{2\delta_k \left[E(w_k)E(w_k + \alpha_k p_k)\right]}{\mu_k^2} \tag{3.71}$$

7. if $\Delta_k \geq 0$ then a successful reduction in error can be made:

$$w_{k+1} = w_k + \alpha_k p_k, \tag{3.72}$$

$$r_{k+1} = E'(w_{k+1}), \tag{3.73}$$

$$\bar{\lambda}_k = 0, \tag{3.74}$$

success = true.

- if $k \; mod \; N = 0$ then restart algorithm: $p_{k+1} = r_{k+1}$ else create new conjugate direction:

$$\beta_k = \frac{|r_{k+1}|^2 r_{k+1} r_k}{\mu_k}, \tag{3.75}$$

$$p_{k+1} = r_{k+1} + \beta_k p_k. \tag{3.76}$$

- If $\Delta_k \geq 0.75$ then reduce the scale parameter: $\lambda_k = \frac{1}{2}\lambda_k$.

else a reduction in error is not possible: $\bar{\lambda}_k = \lambda_k$, success = false.

8. If $\Delta_k < 0.25$ then increase the scale parameter: $\lambda_k = 4\lambda_k$

9. If the steepest descent direction $r_k \neq 0$ then set $k = k + 1$ and go to 2 else terminate and return w_{k+1} as the desired minimum.

4 UAV Simulation

4.1 Chapter Outline

This chapter begins with the verification of the flight dynamic model through the use of a linearised static flight model as well as through the use of well-defined and understood performance indicators. This is only a verification that the simulation code is running correctly and only serves to confirm that results obtained are a realistic reflection of the real UAV. After this verification, a report of the failures modelled and the testing procedure followed will be outlined. This chapter's primary concern is to test the adaptation and allocation algorithms under a variety of flight conditions and with a variety of aircraft failures to determine if these algorithms are tolerant to certain control actuator failures. The results for the inner loop simulations and middle loop simulations of the UAV are presented.

The PID controller alone was found to be unable to handle any faults without the control allocation algorithm so all the PID results presented are a combination of the PID and control allocation algorithm.

4.2 Fault Scenarios

The failures modelled stem from the objectives of this research described in Chapter 1.3. These failures are considered representative of typical UAS failures that one may expect in the operation of a research UAS. Severe structural damage was not considered but, with some modification to the flight dynamic code, this could be achieved and, as such, will be included in the recommendation for future work. For this research, the failures modelled pertain primarily to actuator or control surface failures. The failures tested are as follows:

- 0 *rad* deflection of one or more control surfaces
- A 50% of full deflection failure

- A hardover failure, where servo moves to either extreme position

Specifically, the controls failed are the elevator and right aileron. The left aileron did not fail in these tests as it shows similar results to the failure of the right control surfaces and provides no additional insight. Different combinations of left and right control surface failures could be modelled, but have not been simulated. This will be recommended for future work.

4.3 Procedures Followed

The procedures followed involved in testing the algorithm begins with testing the inner-loop controls in either steady level flight or steady turning flight mode. Once satisfied with the results, the middle loops are then tested. Lastly, the outer loops are tested in the climbing turn flight mode. Initially, the control allocation algorithm was disabled to gain an understanding of the behaviour of the adaptation algorithm.

4.3.1 Inner Loop Testing

The flight mode chosen for this simulation was the steady-level and turning flight mode as it provided a means to determine the suitability of the neural network and allocation algorithms during two common flight manoeuvres. While many combinations or manoeuvres could be performed, the manoeuvre chosen was to maintain level wing flight, and maintain a constant bank angle. This could be viewed at a system's level as the bare minimum for the aircraft to return to a "Home" location and loiter over the home location should failure occur. These manoeuvres also provide insight into the suitability of the proposed algorithms to achieve the desired goal and to determine any shortcomings or enhancements required. For inner loop tests the setpoint is unchanging and thus the tests will show the ability of the neural network to track a fixed setpoint value.

The high level procedure to test the inner loops was to establish the throttle and pitch settings to maintain level and constant banked flight. Using these settings, two simulations per failure were run. One with the classical controller autopilot and one with the neural network algorithm enabled. The allocation algorithm was only used if it is found that it is required and for the more severe control failures. A detailed procedure to test the inner loop is outlined:

1. Find the correct control input settings to maintain altitude for the manoeuvre being tested.

Test Number	Control Surface Failed	Direction	Failure
Control	None	N/A	None
1	Elevator	N/A	0 rad
2	Elevator	Pitch up	50% hardover
3	Elevator	Pitch up	100% hardover
4	Right aileron	N/A	0 rad
5	Right aileron	Roll right	50% hardover
6	Right aileron	Roll right	100% hardover
7	Elevator & right aileron	N/A	0 rad
8	Elevator & right aileron	Pitch up and roll right	50% hardover
9	Elevator & right aileron	Pitch up and roll right	100% hardover

Table 4.1: Inner loop simulation test cases

2. Set the control inputs.

3. Determine the failure to be modelled from the test matrix, Table 4.1 and set up the aircraft failure algorithm to apply the desired failure. Run the simulation for a period of 120 seconds with neural network and allocation algorithm disabled.

4. Rerun the simulation with the neural network enabled, then proceed to the next test point in the test matrix.

4.3.2 Middle loop testing

The flight mode used to test the middle loop is one in which the aircraft climbs to an altitude of $600m$ from $300m$ above mean sea level (MSL) in the first 60 seconds and then maintains a constant airspeed and altitude for the remaining 60 seconds. The reason for this is that most aircraft accidents happen during takeoff and landing. This allows one to test if the UAV can recover just after the takeoff phase during climb. One will also be able to determine the interaction between the coupled controllers and the ability of the controller to track a changing setpoint. A similar procedure to that outlined in Section 4.3.1 was performed, but with the test cases presented in Table 4.2.

Test Number	Control Surface Failed	Direction	Failure
Control	None	N/A	None
1	Elevator	N/A	0 rad
2	Elevator	Pitch up	50% hardover
3	Elevator	Pitch up	100% hardover
4	Right aileron	N/A	0 rad
5	Right aileron	Roll right	50% hardover
6	Right aileron	Roll right	100% hardover

Table 4.2: Middle loop simulation test cases

4.4 Results

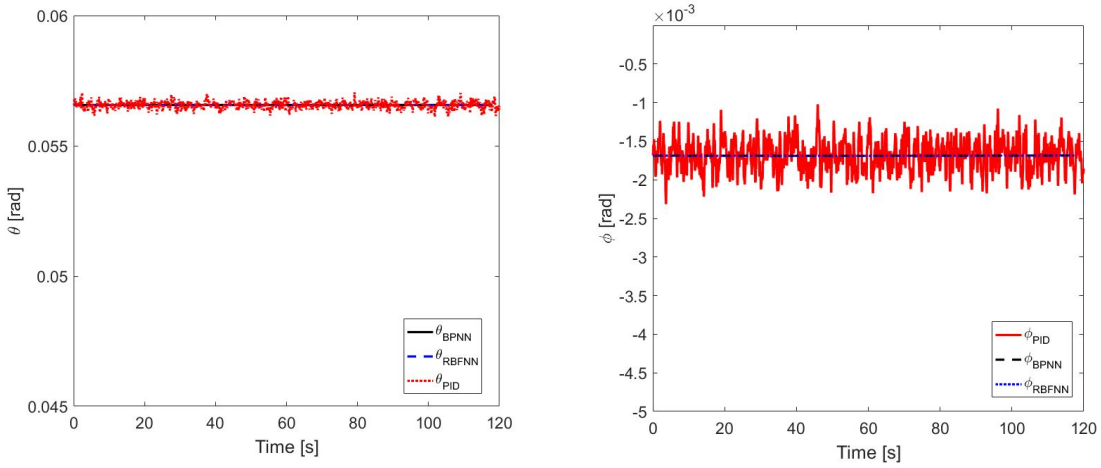
4.4.1 Inner control loop testing

The inner loop of the autopilot was tested first. The autopilot was simulated in the steady wings-level flight mode as described in Section 3.3 and the trim position determined. The pitch angle to maintain level flight was found to be 0.057 rad (3.27 deg) and a throttle setting of 55% was found to maintain a speed of approximately 17 m/s. These settings were then applied to the simulation and the test matrix followed.

To display the results of all aircraft state data would be considerable; hence for each test in this section the primary point of interest will be pitch and roll angles as these are the controlled parameters of the inner flight loop. The control deflection data, the PID gain coefficient data as well as the controller effort will also be supplied. For the PID-controlled cases the gains used are those in Table 3.2.

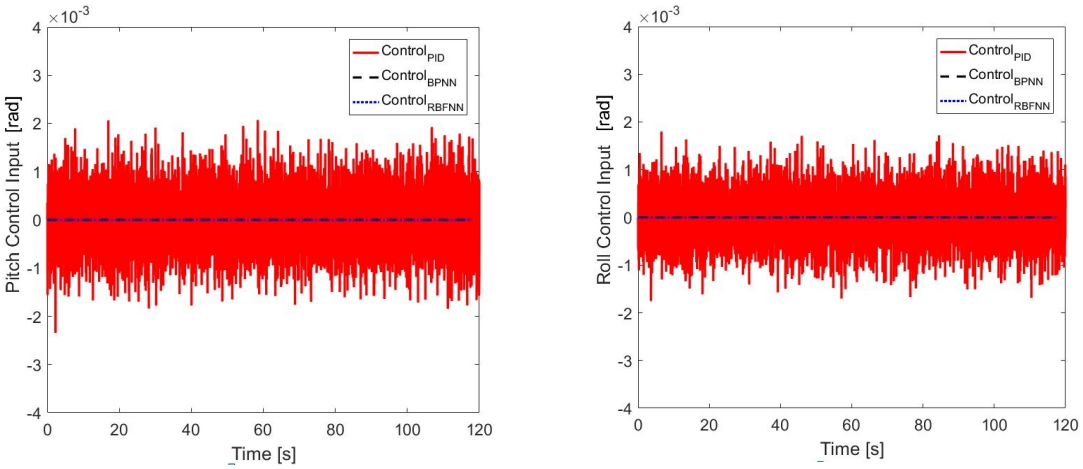
4.4.1.1 Control - No Failures

The simulation was run without inducing any failures, both with the PID and neural network controllers for commanded level flight. Figure 4.1, which shows the time history of the pitch and bank angle, shows that for both cases a continuous pitch and bank angle of 0.057 rad (3.27 deg) and -0.0014 rad (-0.08 deg), respectively. Though the desired bank angle is 0 rad this slight deviation is negligible as it is unlikely to be noticed in flight. It can be observed that with the neural network controllers some of the noise from the sensors is canceled out which indicates that the controller is working. Figure 4.2 illustrates the controller effort required and shows that there is a small variation in control effort required for both cases.



(a) (b)

Figure 4.1: Simulated time history of inner loop pitch and bank angles during level flight without control surface failures



(a) (b)

Figure 4.2: Simulated time history of the PID-controlled pitch and roll inputs required for level flight without failures: a), and b)

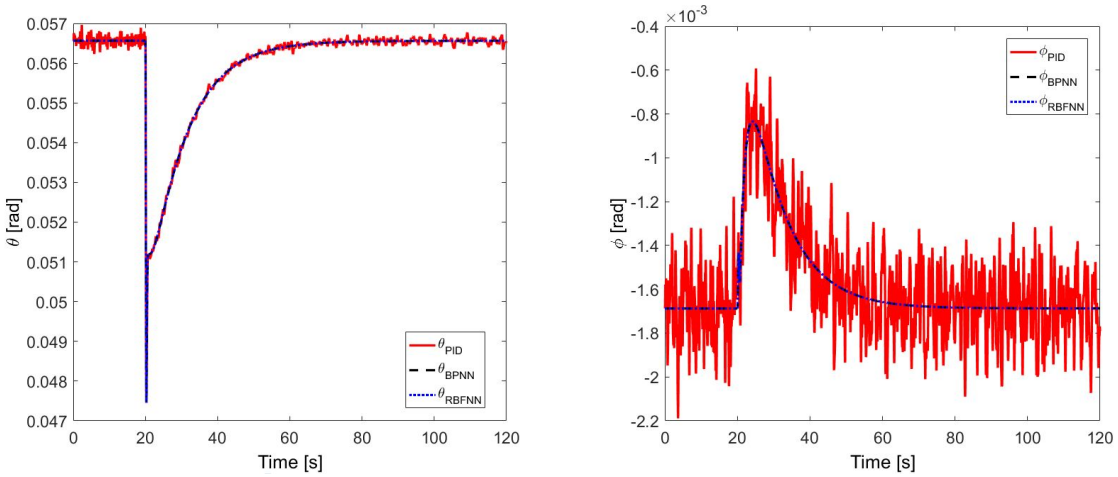
The performance of these controllers were deemed to be satisfactory for the purposes of this research.

4.4.1.2 Elevator Failure - *0 rad* Deflection

The simulation was run with 0 *rad* elevator deflection failure. That is, both the left and right elevators move from their current position, at the time of failure to their neutral position. For this flight position and aircraft configuration, a slight down elevator trim is required. Hence,

as the elevator returns to its neutral position, a nose-up disturbance is created. Figure 4.3 clearly shows the point at which the elevator failed and shows that the failure causes a nose up pitching moment coupled with a rolling moment to the right which is observed by the diversion of the roll and pitch angles.

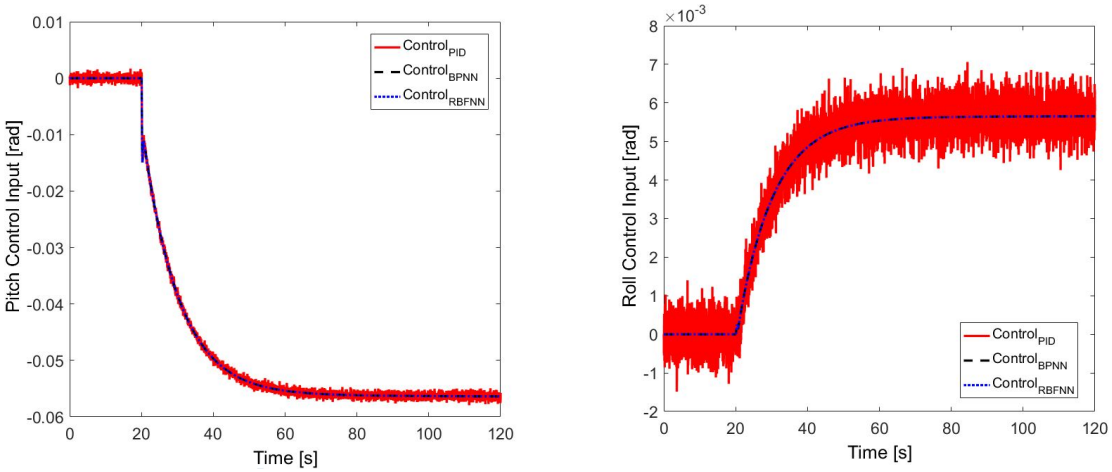
Figure 4.4 (a) shows that, the pitch change is of the order of 0.01 *rad* (0.57 deg). The pitch change is then followed by a rapid increase in pitch angle as the control system attempts to correct the failure. A smooth return to the desired setpoint then occurs. With the NN-based controllers enabled there is no large difference in control performance noted at this point. The control inputs shown in Figure 4.4 show a negative pitch control input after the failure to aid in the reduction of the effects of the failure.



(a) (b)

Figure 4.3: Simulated time history of pitch and bank angles (0 *rad* elevator failure) during level flight

Figure 4.4 (b) shows the roll disturbance created as the elevator is moved to the neutral position. The effect was small and no noticeable difference can be observed between the PID and NN-based cases. Figure 4.4 shows a negative pitch control input after the failure for both the PID and NN-based cases.



(a) (b)

Figure 4.4: Simulated time history of pitch (a) and roll (b) control inputs (0 *rad* elevator failure) during level flight

Figure 4.5 shows the time history of the PID gains during 0 *rad* elevator failure. Figure 4.5(a) shows that for the BPNN-based pitch there is a slight change in the derivative gain while the proportional and integral gains remain constant. For the RBFNN-based case (Figure 4.5 (b)) the gains remain constant. In Figure 4.5 (c) and (d) show a slight change in the gains for the BPNN-based case while the RBFNN-based gains remain constant.

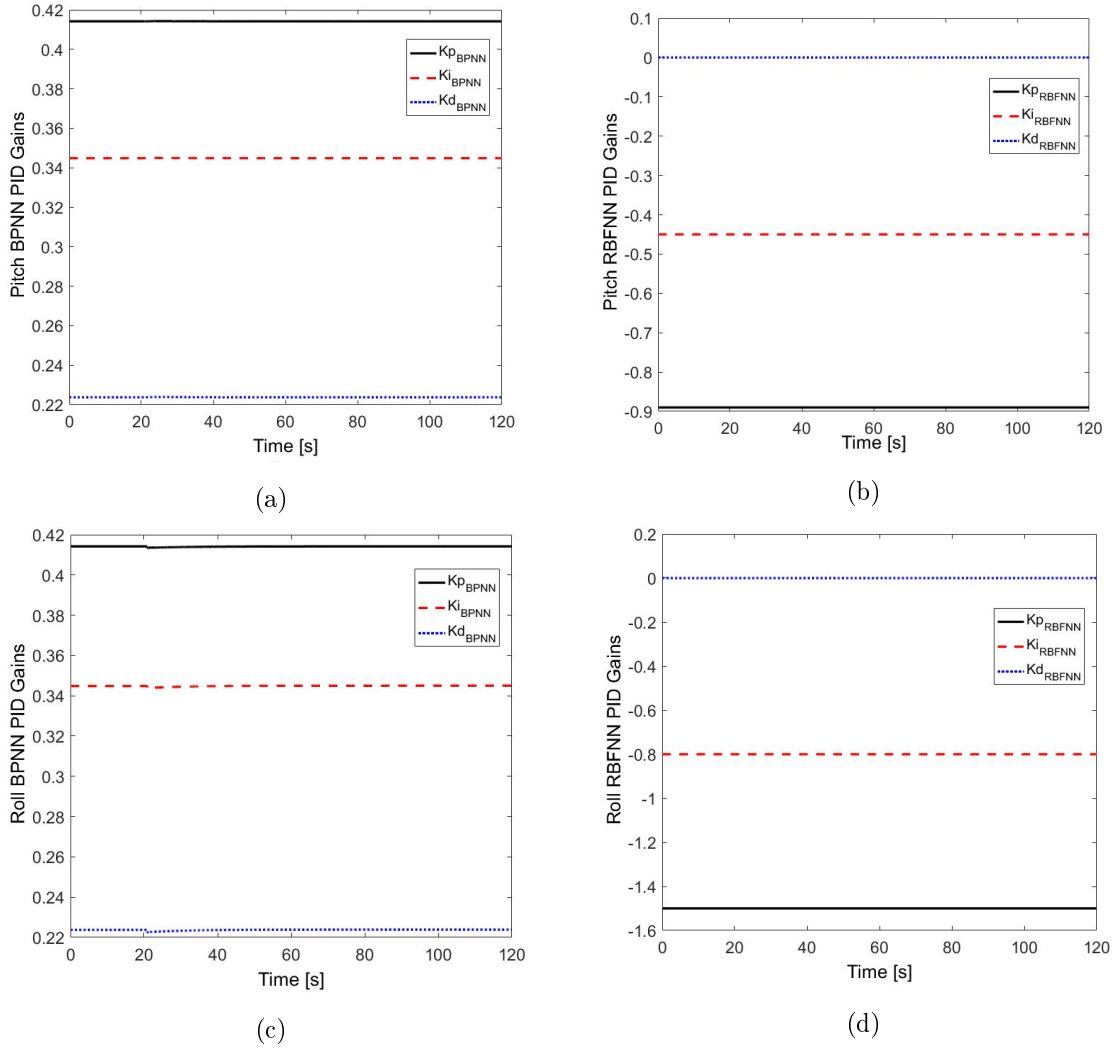
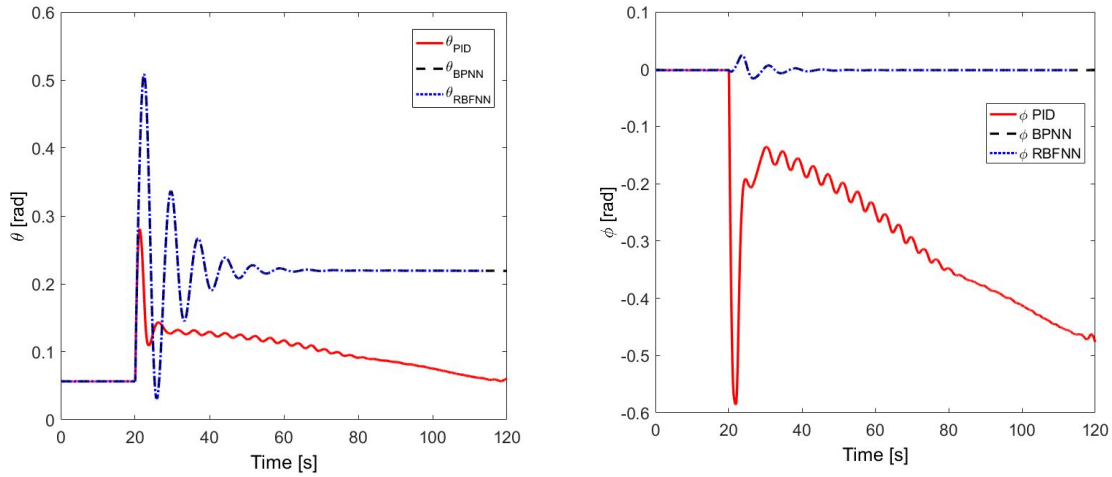


Figure 4.5: Simulated time history of pitch (BPNN-based (a), and RBFNN-based (b)), and roll (BPNN-based (c), and RBFNN-based (d)) gains during during level flight with 0 *rad* elevator failure

4.4.1.3 Elevator Failure - 50% Hardover

The simulation was run with a more severe failure of 50% hardover elevator failure. Figure 4.6 shows the result of this simulation in the form of a time history plot of pitch and roll angles. It can be observed in Figure 4.6 (a) that the elevator failure caused a large change in the nose up direction. In the case of the PID-controlled system, the pitch error reached a maximum of 0.22 *rad* (12.61 deg) and 0.45 *rad* (25.78 deg) for the NN-based cases after which oscillations occurred before stability was restored. The NN-based cases stabilise at a larger pitch angle while the oscillations in the PID-controlled case take longer to dampen and the angle moves to was the desired setpoint. Figure 4.7 shows that the pitch control input was increased rapidly by both the PID and NN-based controllers to accommodate the increased error with

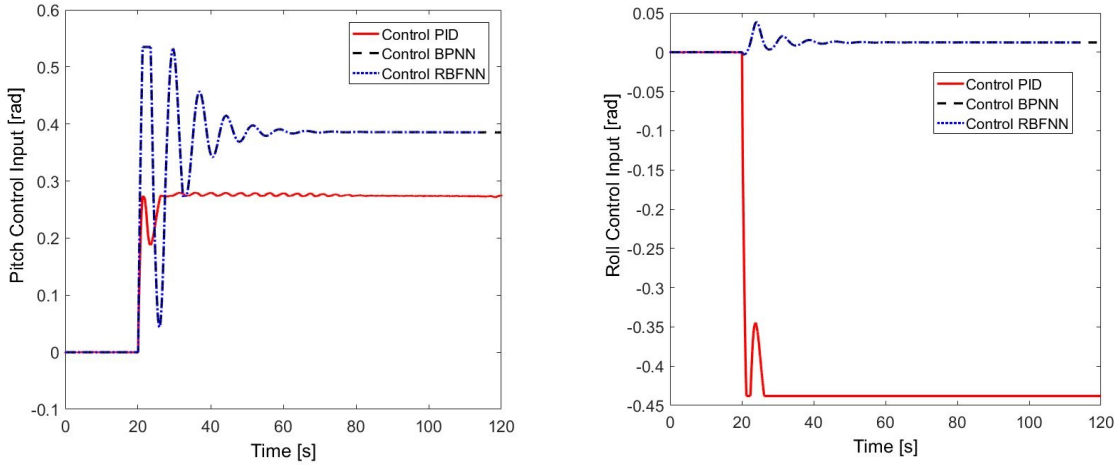
the NN-based one experiencing a larger increase. For the roll angle the NN-based cases have a far smaller as well as positive control input when compared to the PID-based case. The BPNN and RBFNN-based cases' responses overlap.



(a) (b)

Figure 4.6: Simulated time history of pitch (a) and bank (b) angles (50% hardover) elevator failure) during level flight

Figure 4.6 shows that the autopilot was unable to tolerate the roll perturbation in the PID-based case but in the NN-based cases, performance was greatly improved with the amplitude and number of oscillations being far less as well as the roll angle returning to the desired setpoint. From Figure 4.7 it can be observed that PID-based requires less control effort for pitch control while the NN-based case requires less control effort for roll control.



(a) (b)

Figure 4.7: Simulated time history of pitch (a) and roll (b) control inputs (50% hardover) elevator failure during level flight

Figure 4.8 shows the time history of the PID-based controllers' gains during 50% hardover elevator failure. For the pitch gains (Figure 4.8 (a) and (b)) in the BPNN-based case one can see a small disturbance at the time of failure which gradually dampens out while the RBFNN-based controllers' gains remain constant. In contrast to this the roll gains (Figure 4.8 (c) and (d)) show a large disturbance followed by small amplitude oscillations in the BPNN-based case and only a gradual increase the integral gain for the RBFNN-based case.

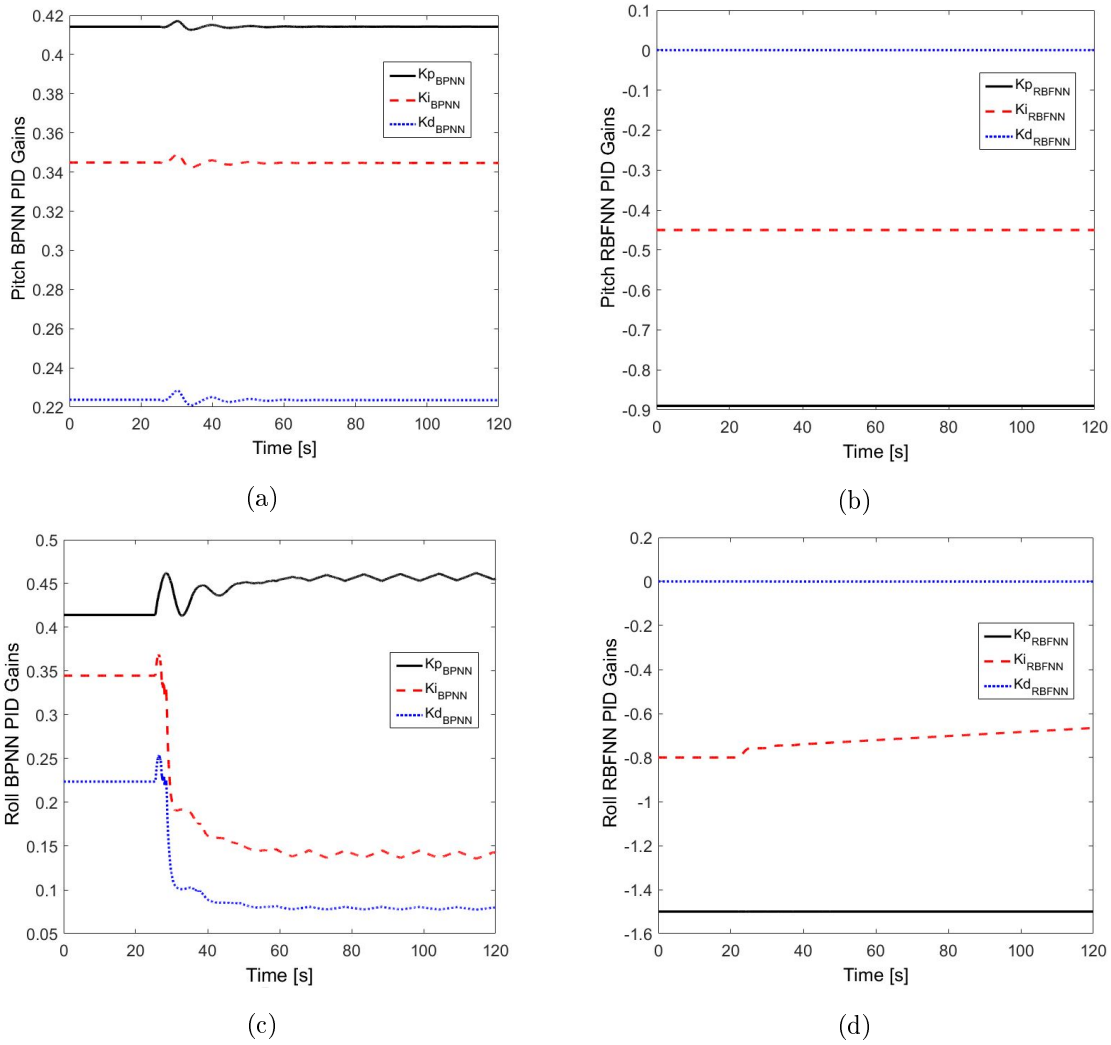


Figure 4.8: Simulated time history of pitch (BPNN-based (a), and RBFNN-based (b)), and roll (BPNN-based (c), and RBFNN-based (d)) gains during level flight with 50% hardover elevator failure

Figure 4.9 (a) to (f) show the time history of the control deflections during the simulation. Figure 4.9 (c) shows that both right and left elevators move to an angle of $-0.17\ rad$ ($-9.74\,\text{deg}$) (50% of $0.34\ rad$ ($19.48\,\text{deg}$)) and remain at that position for the duration of the simulation. A new equilibrium is then reached which is within the control limits of the actuators. Figure 4.9 shows a change in aileron and flap deflection as the required control input is reallocated for

the PID-based case. However, the NN-based cases do not utilise the flaps to return to stability and require smaller control deflections for the ailerons.

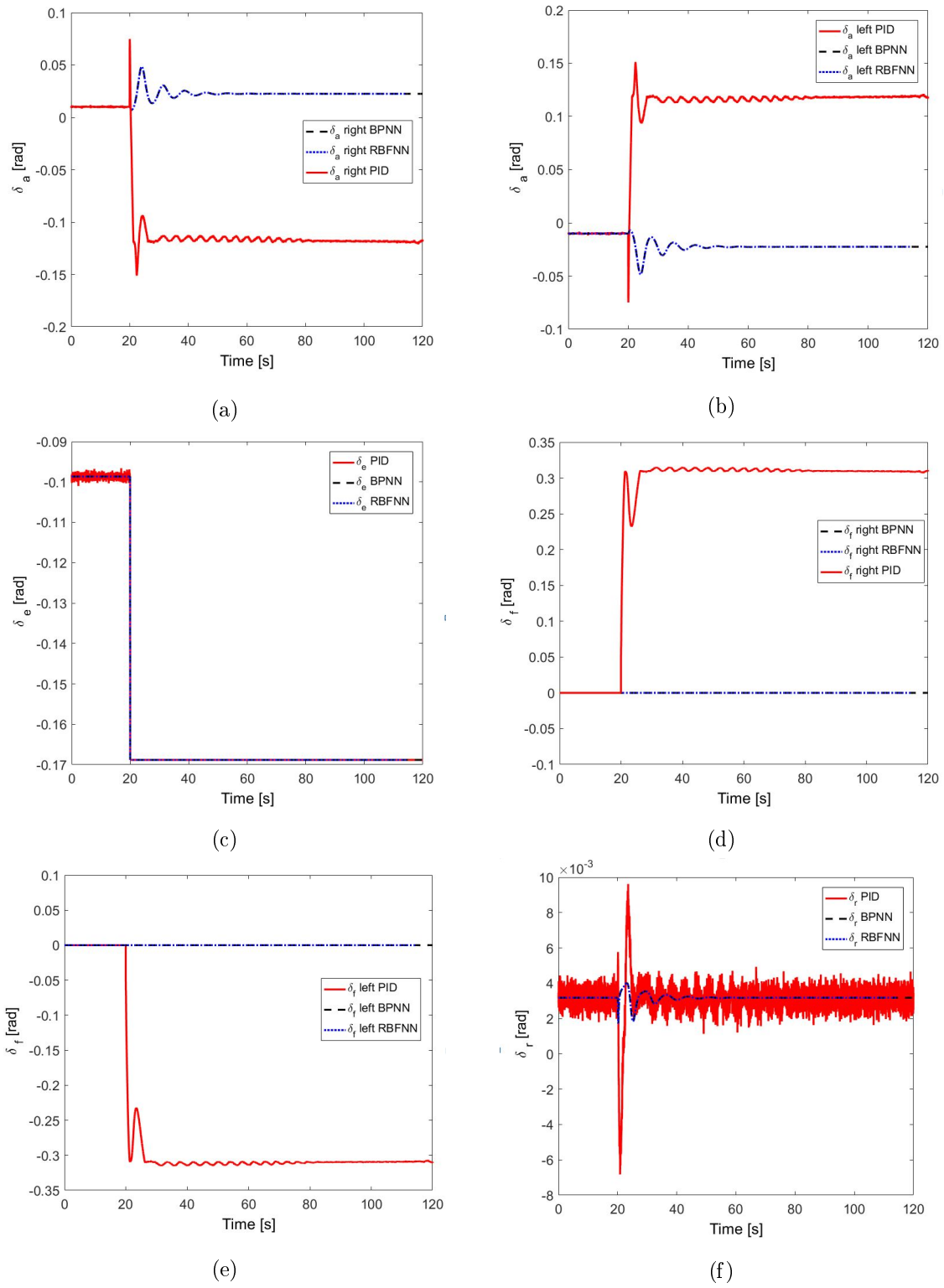


Figure 4.9: Simulated time history of right aileron (a), left aileron (b), elevator (c), right flap (d), left flap (e), and rudder (f) deflections (50% hardover) elevator failure during level flight

4.4.1.4 Elevator Failure - 100% Hardover

The simulation was run for a 100% hardover elevator failure. Figure 4.10 shows the time history of pitch and roll angles for this simulation. It shows that there was significant overshoot on the return to setpoint value. Again a number of oscillations occurred about the setpoint value (as observed in the 50% elevator hardover failure case). Figure 4.10 (a) shows that there is a rapid increase in pitch control input as the controller attempts to prevent the increase in pitch error. The NN-based cases reached the upper limit for the pitch controller. The angle of attack response plot is shown in Figure 4.10 (c), the plot shows that while the angle of attack of the PID-controlled case remains within reasonable limits during the simulation the NN-based cases go well past stall conditions.

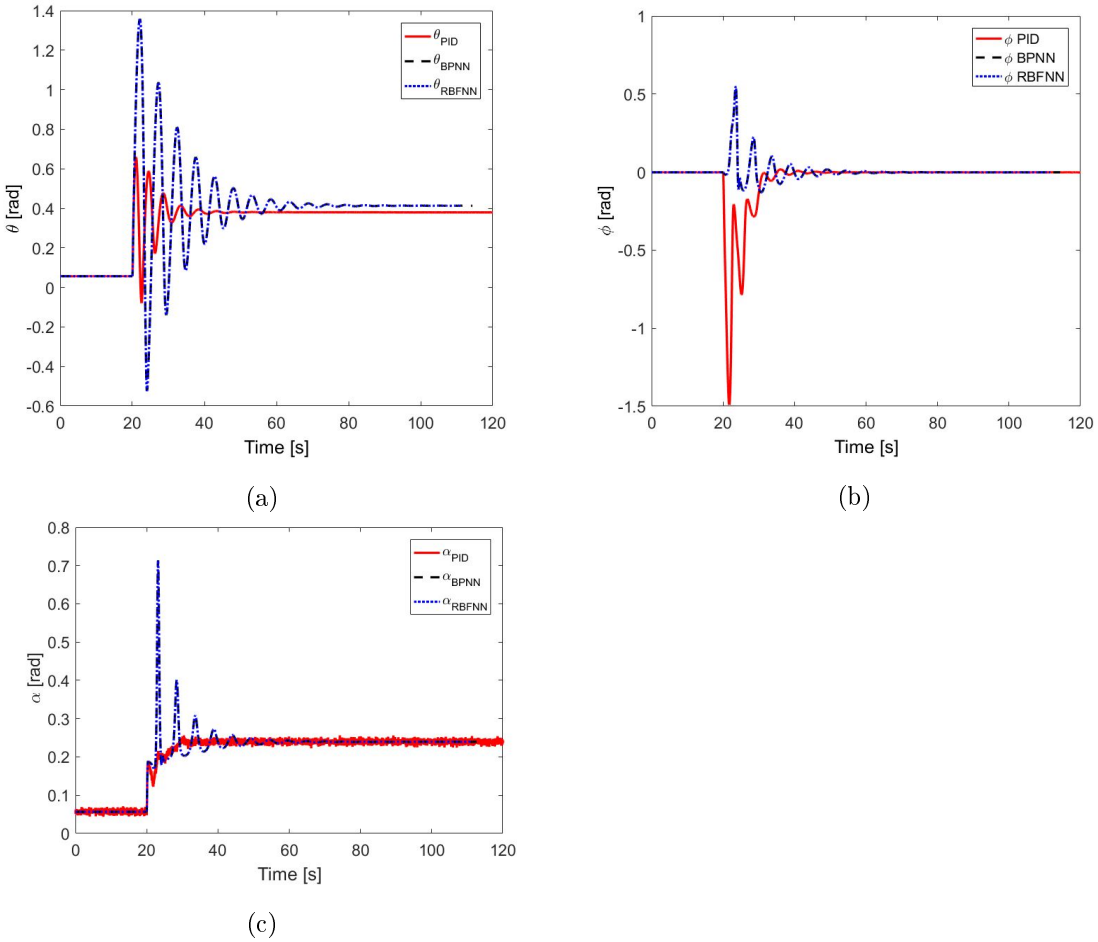
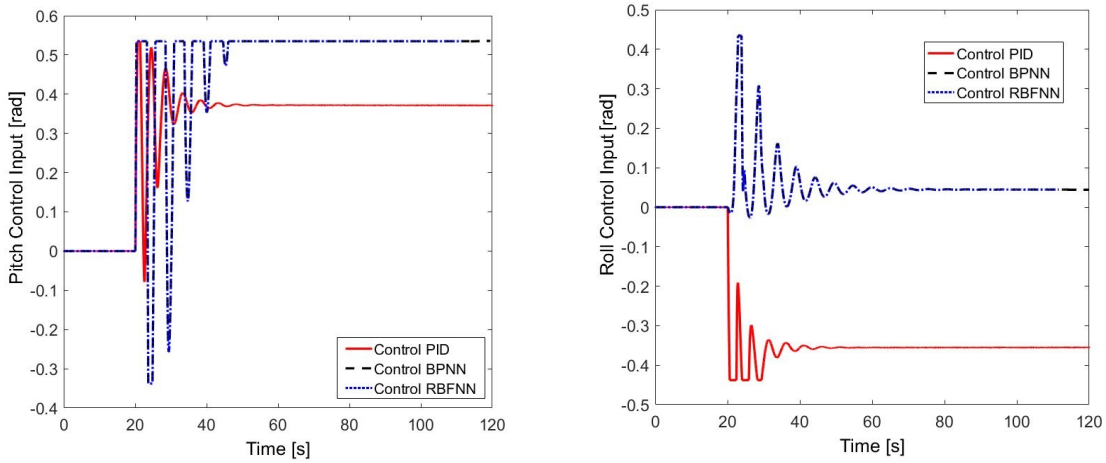


Figure 4.10: Simulated time history of pitch and bank angles and angle of attack (100% hardover elevator failure) during level flight

After the failure of the elevator, there is a corresponding roll right in the NN-based cases and roll left in the PID case. The roll control (Figure 4.10 (a)) for the NN-based cases settles at

a value closer to the original control input while the PID-based case settles at a value that is far more negative.



(a) (b)

Figure 4.11: Simulated time history of pitch (a) and roll (b) control inputs (100% hardover elevator failure) during level flight

There was a tradeoff in the NN-based cases between pitch and roll performance in that if the NN-based controller forced the inclusion of flap deflection, then the resulting responses would be similar to that of the PID-based case while its natural response improved the bank angle response. Figure 4.12 shows the time history of the PID-based controllers' gains during the 100% hardover elevator failure.

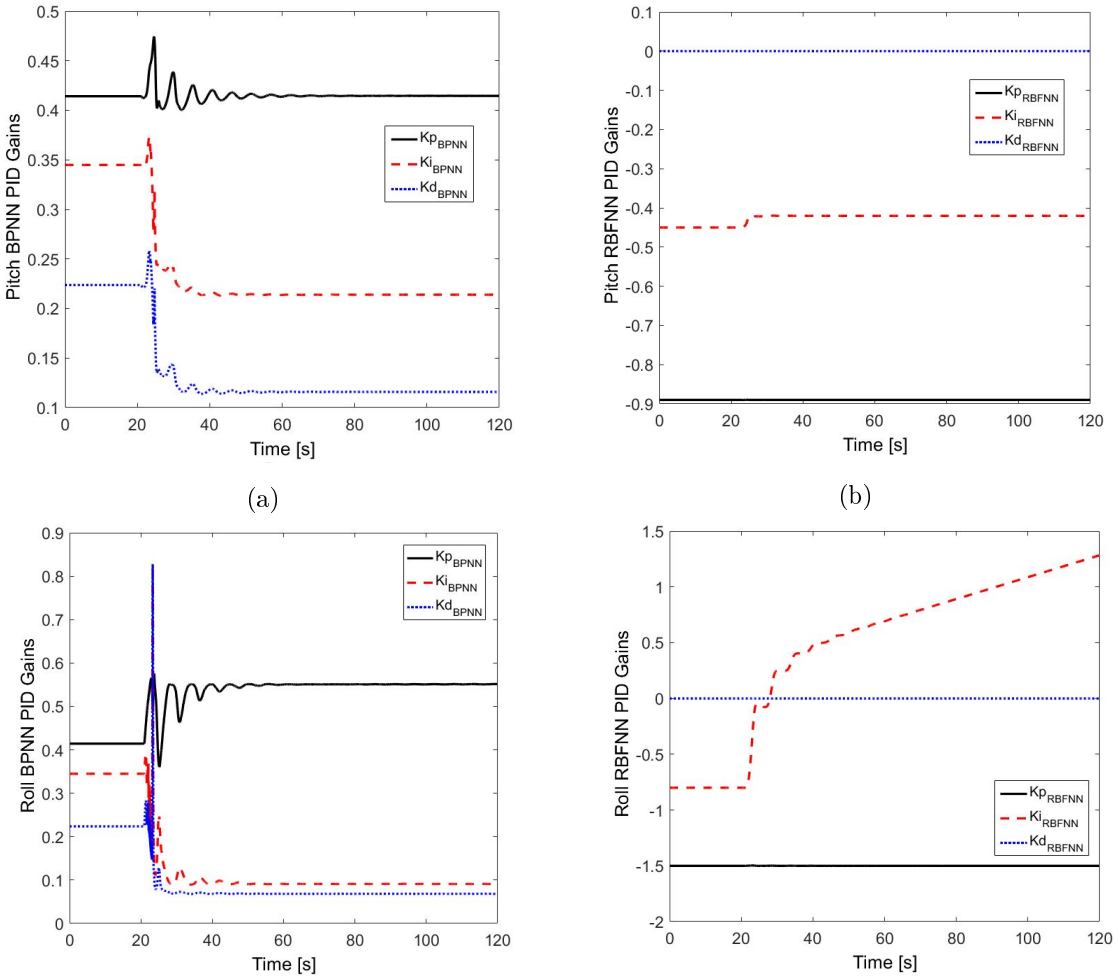


Figure 4.12: Simulated time history of pitch (BPNN-based (a), and RBFNN-based (b)), and roll (BPNN-based (c), and RBFNN-based (d)) gains during level flight with 100% hardover elevator failure

The oscillatory behaviour noted in Figure 4.10 can be explained by examining Figure 4.13, which shows that the aileron, flap and rudder deflections experience oscillatory behaviour. The variation in the magnitudes of the control deflections are also high.

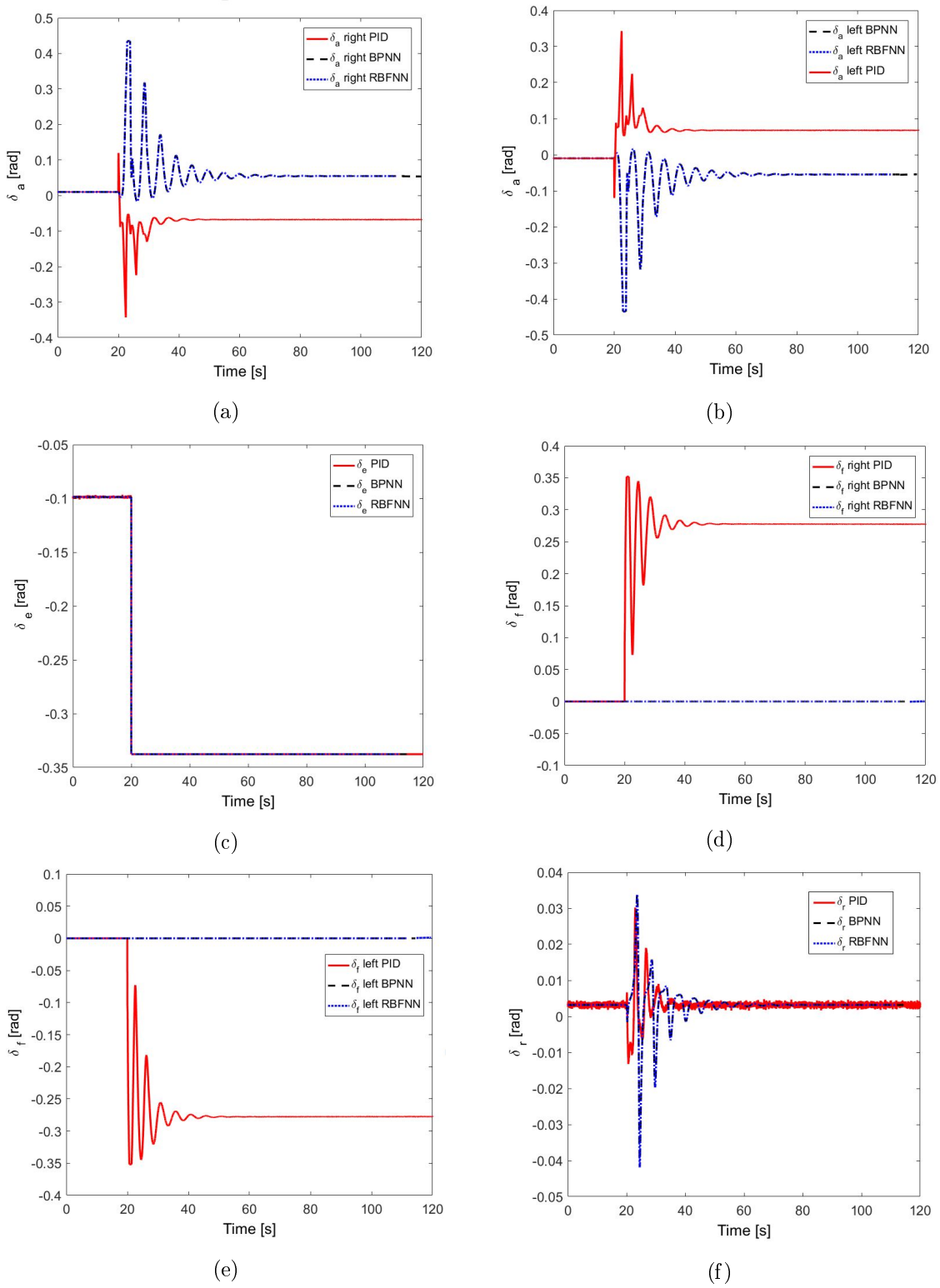
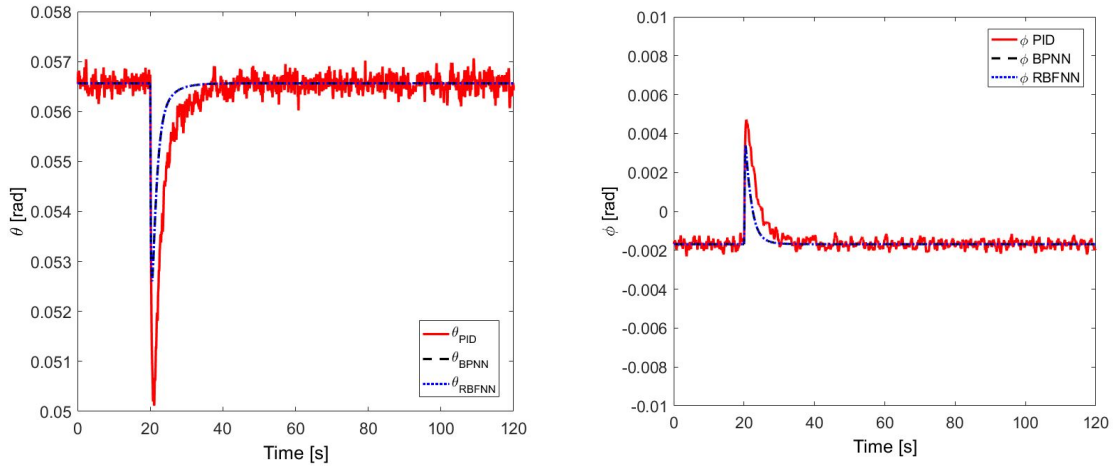


Figure 4.13: Simulated time history of right aileron (a), left aileron (b), elevator (c), right flap (d), left flap (e), and rudder (f) deflections (100% hardover) elevator failure during level flight

4.4.1.5 Right Aileron Failure - *0 rad* Deflection

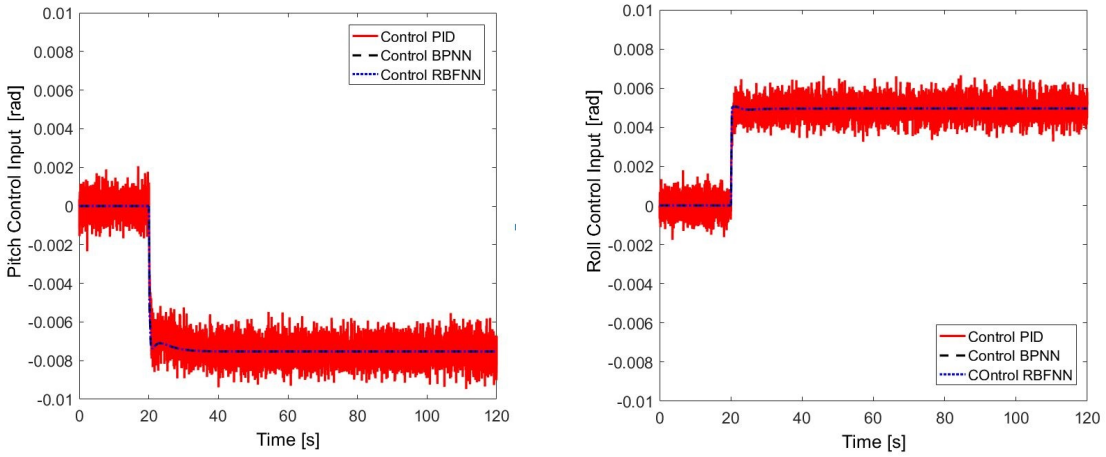
The simulation was run with a 0 *rad* aileron deflection failure at 20 seconds after the start of the simulation. Figure 4.14 (a) shows that there was an improved performance in pitch control with the initial jump being slightly less, a similar improvement was noted in the roll plot (Figure 4.14 (b)). The effect on roll was less pronounced and only a small bank error was induced. This is to be expected for level flight with only the rolling moment being generated by the asymmetries of the aircraft. Figure 4.15 does show a slight difference in control performance in roll with more rapid response being generated by the NN-based cases.



(a) (b)

Figure 4.14: Simulated time history of pitch (a) and bank (b) angles (0 *rad* right aileron deflection failure) during level flight

Figure 4.15 (a) and (b) show a small change in the control input for both pitch and roll controllers.



(a) (b)

Figure 4.15: Simulated time history of pitch (a) and roll (b) control inputs (0 *rad* right aileron deflection failure) during level flight

Figure 4.16 shows the time history of the PID-based controllers' gains during 0 *rad* right aileron failure. Figure 4.16 (a) and (c) show a very slight change in the gains for the BPNN-based case. In Figure 4.16 (b) and (d) the RBFNN-based gains remain constant.

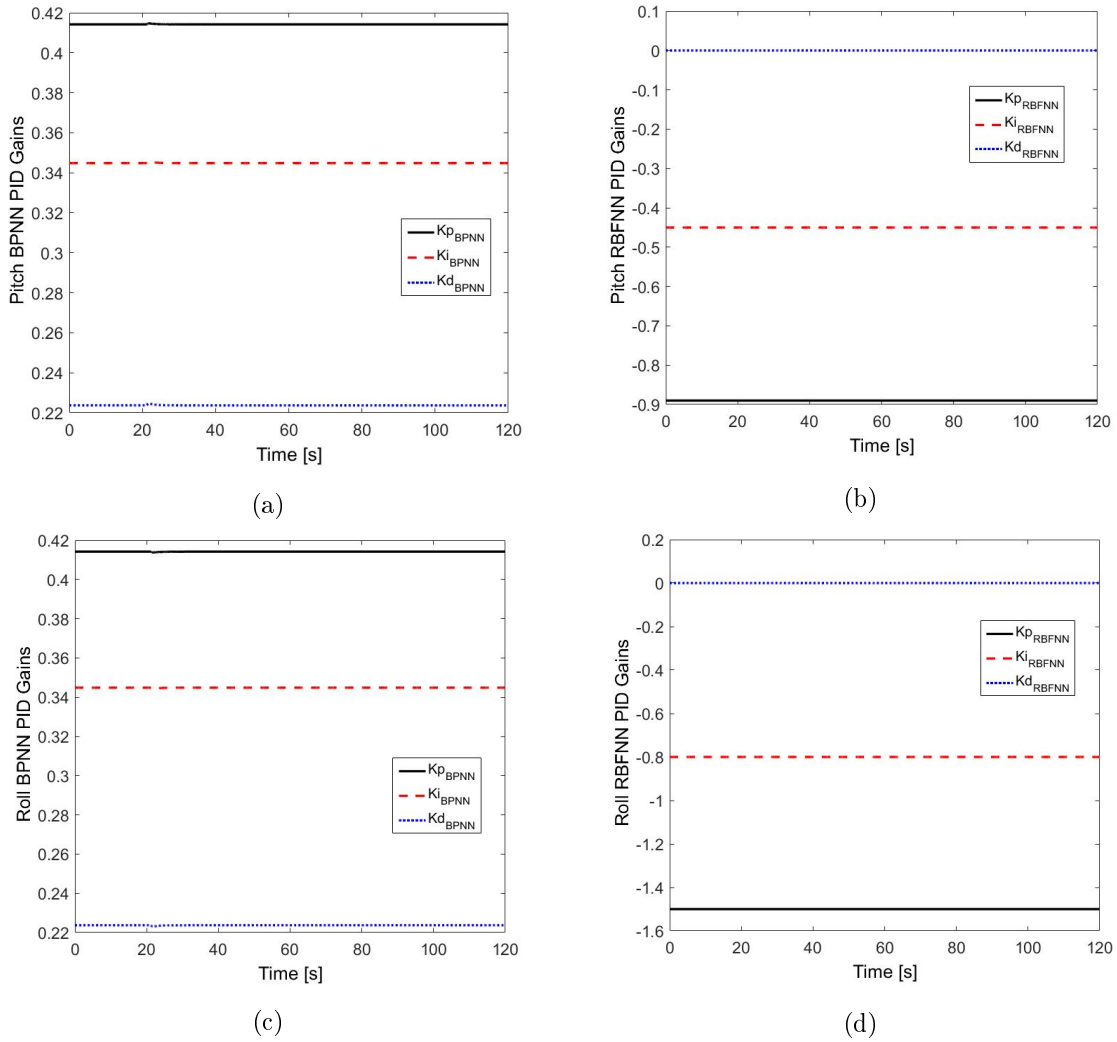
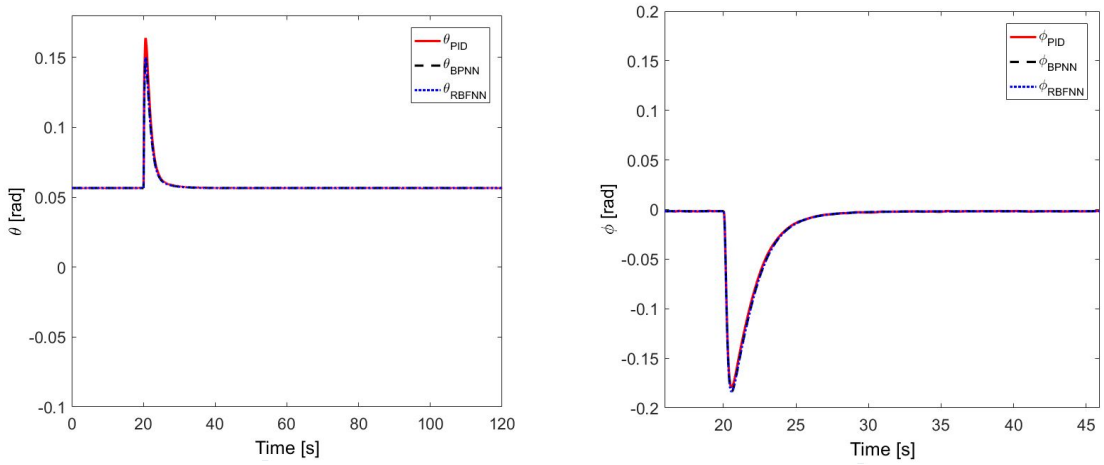


Figure 4.16: Simulated time history of pitch (BPNN-based (a), and RBFNN-based (b)), and roll (BPNN-based (c), and RBFNN-based (d)) gains with 0 *rad* right aileron failure during level flight

4.4.1.6 Right Aileron Failure - 50% Hardover

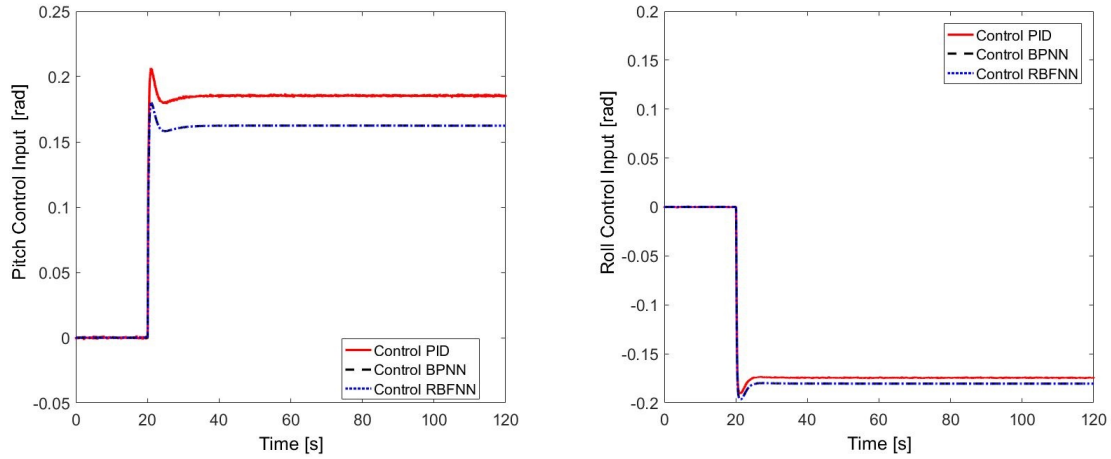
A 50% hardover failure of the right aileron was modelled, the results of which are depicted in Figure 4.17. It shows that there is a rapid decrease in bank angle followed by an equally rapid return to level flight as the controller compensates for this error. The maximum magnitude of the bank angle is found to be 0.18 *rad* (10.31 deg). Upon examination of Figure 4.18, it can be observed that there is a rapid change in the control input to mitigate the change in bank angle. The NN-based cases inputs are slightly more negative control input.



(a) (b)

Figure 4.17: Simulated time history of pitch (a) and bank angles (b) (50% right aileron hardover failure) during level flight

The pitch angle (see Figure 4.17 (a)) experienced a sharp increase before returning to the desired setpoint. The PID-based case required a larger control input to mitigate the failure.



(a) (b)

Figure 4.18: Simulated time history of pitch (a) and roll (b) control inputs (50% right aileron hardover failure) in level flight

Figure 4.19 shows the time history of the PID-based controllers' gains during 50% right aileron hardover failure. The roll gains (see Figure 4.19 (c) and (d)) show a small increase in the BPNN-based case while the RBFNN show a barely discernible change in the proportional and integral gains. For the pitch control gains (see Figure 4.19 (c) and (d)) show a dip in the BPNN-based case while as previously the RBFNN-based controllers' gains remain constant.

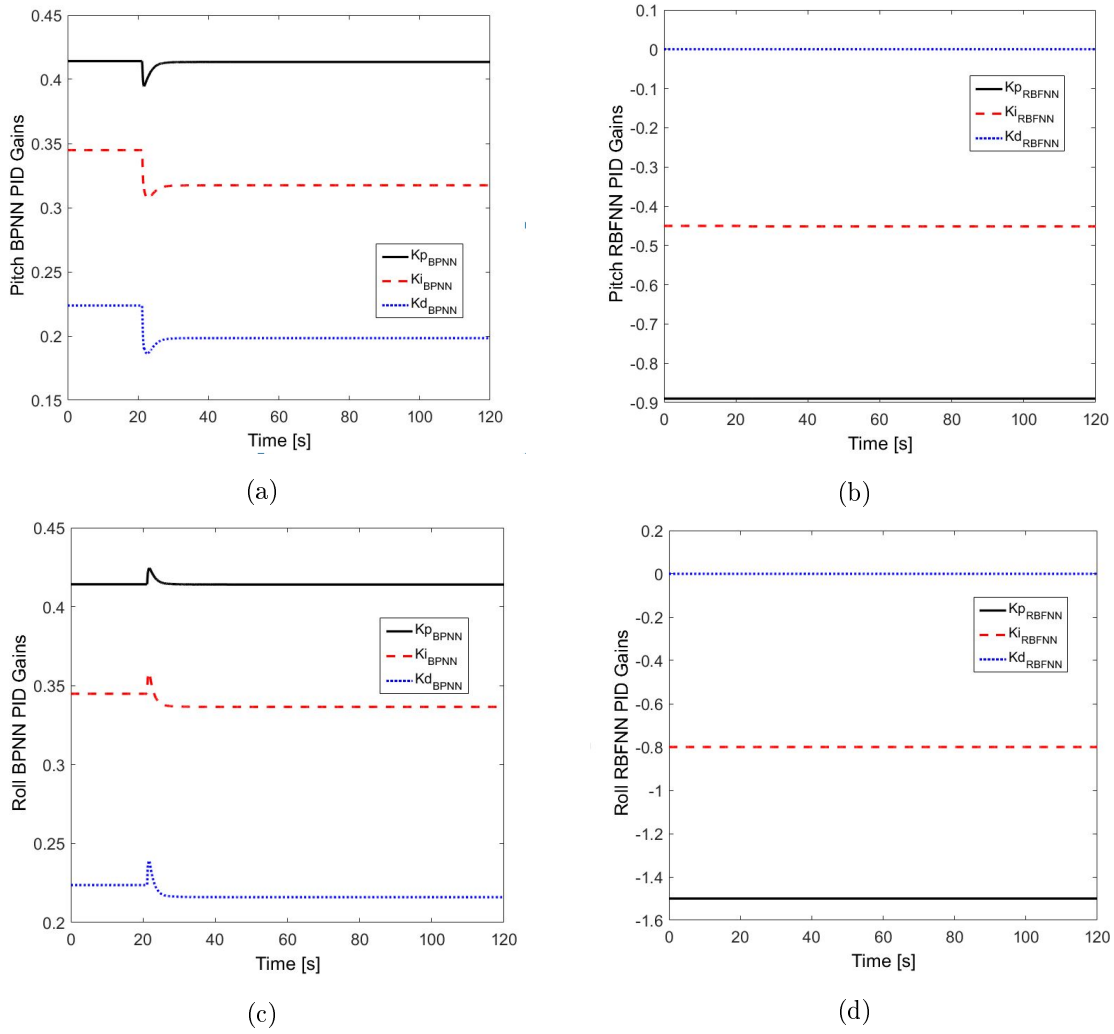


Figure 4.19: Simulated time history of pitch (BPNN-based (a), and RBFNN-based (b)), and roll (BPNN-based (c), and RBFNN-based (d)) gains with 50% right aileron hardover failure during level flight

Figure 4.20 (a) shows that the right aileron moves to 0.22 *rad* (12.61 deg) and remains in that position until the end of the simulation while the left aileron moves to compensate the induced rolling moment and achieves an angle slightly lower than the right aileron. One also notices a change in elevator and flap deflection to compensate for the failure (Figure 4.20).

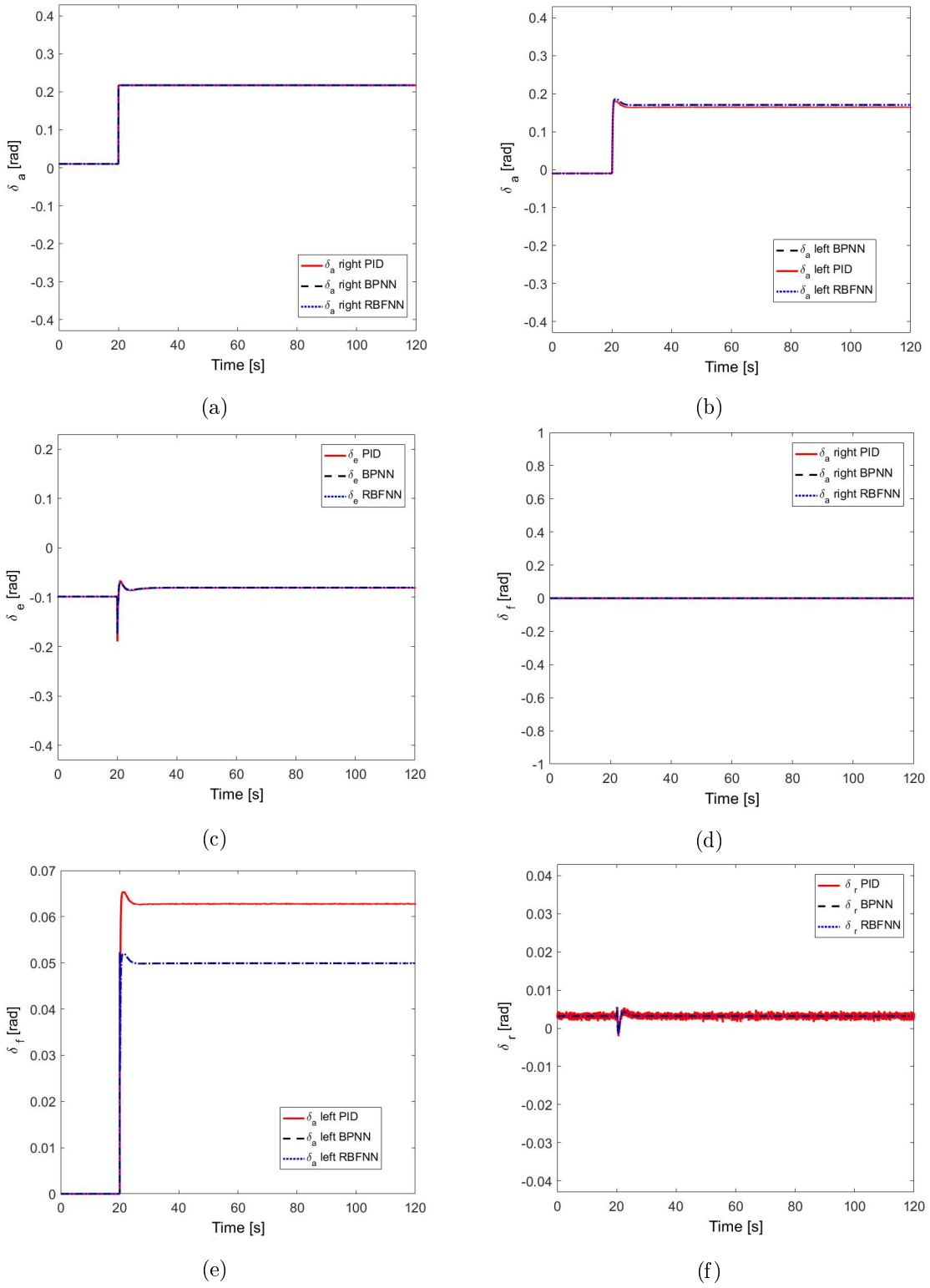
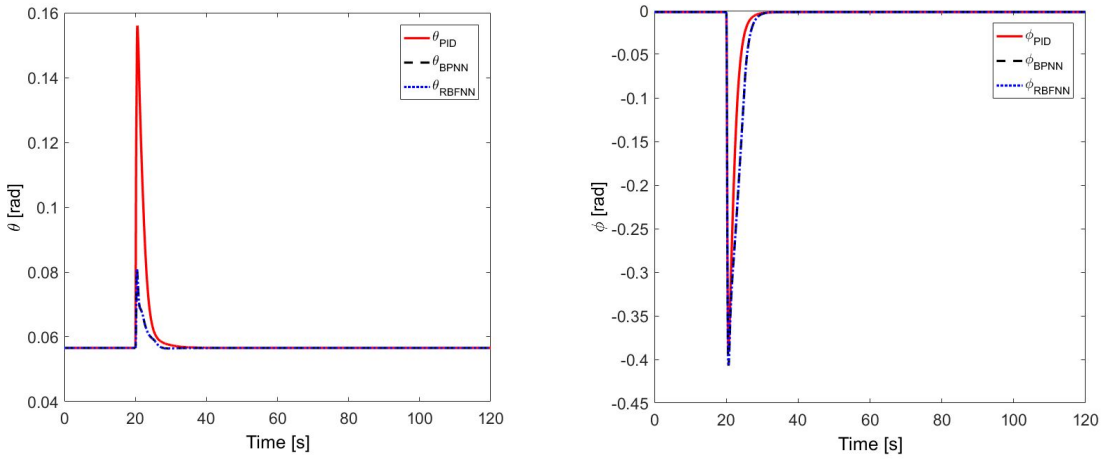


Figure 4.20: Simulated time history of right aileron (a), left aileron (b), elevator (c), right flap (d), left flap (e), and rudder (f) deflections (50% hardover right aileron failure) during level flight

4.4.1.7 Right Aileron Failure - 100% Hardover Failure

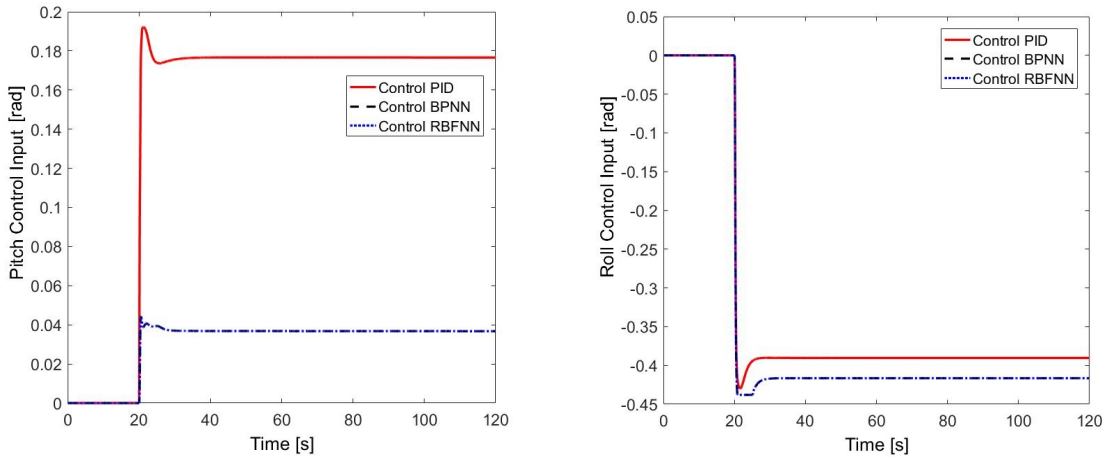
The simulation was run for a 100% hardover failure of the right aileron. The failure causes a strong rolling moment to the left with a small nose-up pitching moment due to change in the effective wing camber. Figure 4.21 shows the time history of the pitch and roll angles. It shows the superior performance of the NN-based in its ability to decrease the sharp increase in the pitch angle at the time of failure. The roll angle performance is similar for both the PID-based and NN-based cases (there is a sharp decrease followed by a rapid increase back to the desired setpoint). The changes in pitch angle were evident in Figure 4.21 and were relatively minor when compared to the roll change.



(a) (b)

Figure 4.21: Simulated time history of pitch and bank angles (100% right aileron hardover failure) during level flight

Figure 4.22 (a) shows that the control input for pitch experienced a step like increase with the PID-based case magnitude being greater. A similar response was found in the decrease of the roll control input (see Figure 4.18 (b)) with the PID-based control input case having a slight less decrease.



(a) (b)

Figure 4.22: Simulated time history of pitch (a) and roll (b) control inputs (100% right aileron hardover failure) during level flight

Figure 4.23 shows the time history of the PID-based controllers' gains during 100% right aileron hardover failure. It shows a drop in the BPNN-based controllers' pitch gains and only a drop in the integral RBFNN-based controllers' pitch gains. The roll gains for the BPNN-based controllers' show a slight change while the RBFNN-based controllers' gains remain constant.

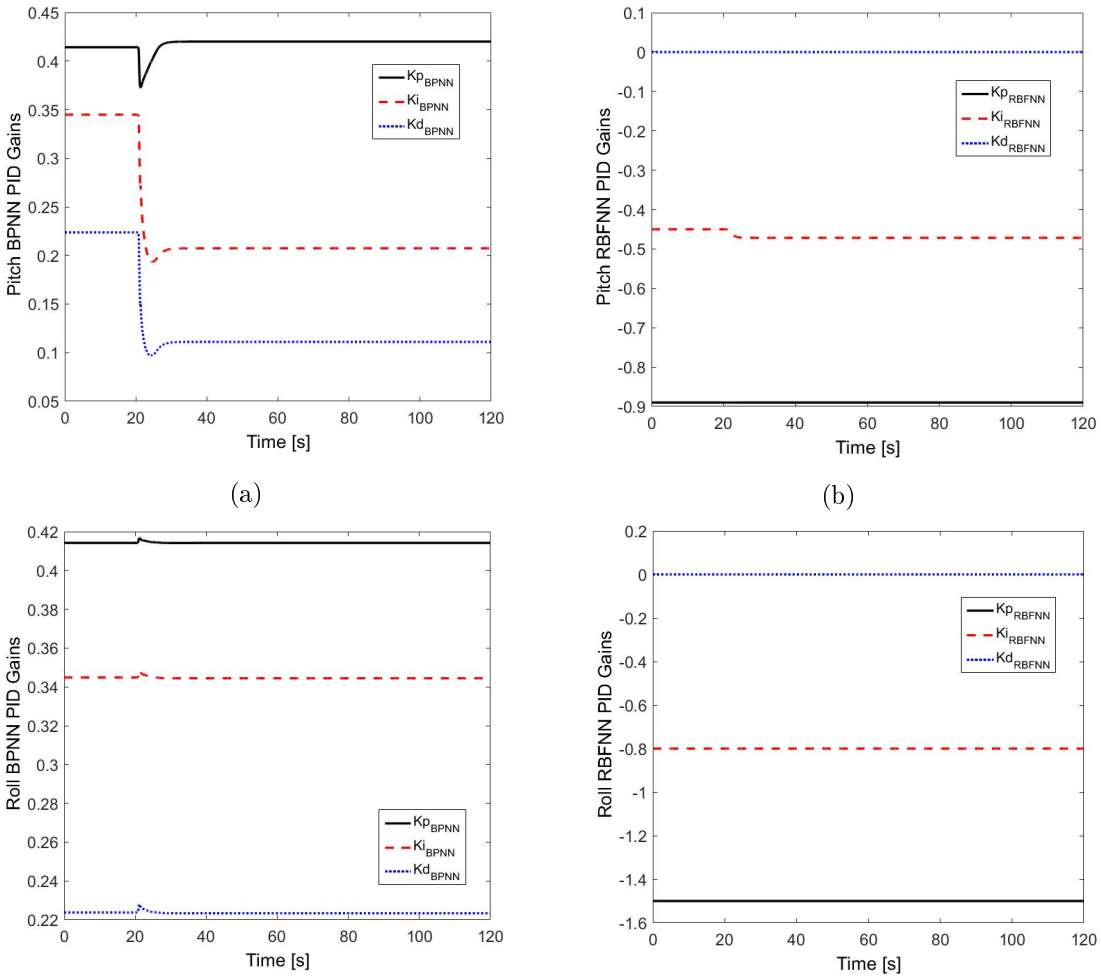


Figure 4.23: Simulated time history of pitch (BPNN-based (a), and RBFNN-based (b)), and roll (BPNN-based (c), and RBFNN-based (d)) gains during level flight with 100% right aileron hardover failure

Examining Figure 4.24 (b) shows the NN-based left ailerons achieve a slightly higher deflection than the PID-based case to compensate for the right aileron. Only the PID-based case uses flap deflection (see Figure 4.23 (d) and (e)). The NN-based controllers are handling the fault without the use of control allocation. The PID-based case initially experiences a sharp decrease in elevator deflection at the time of failure while the NN-based cases do not (see Figure 4.23 (c)). The controllers were constrained not use additional right flap deflection to test their limits with regards to right aileron failure.

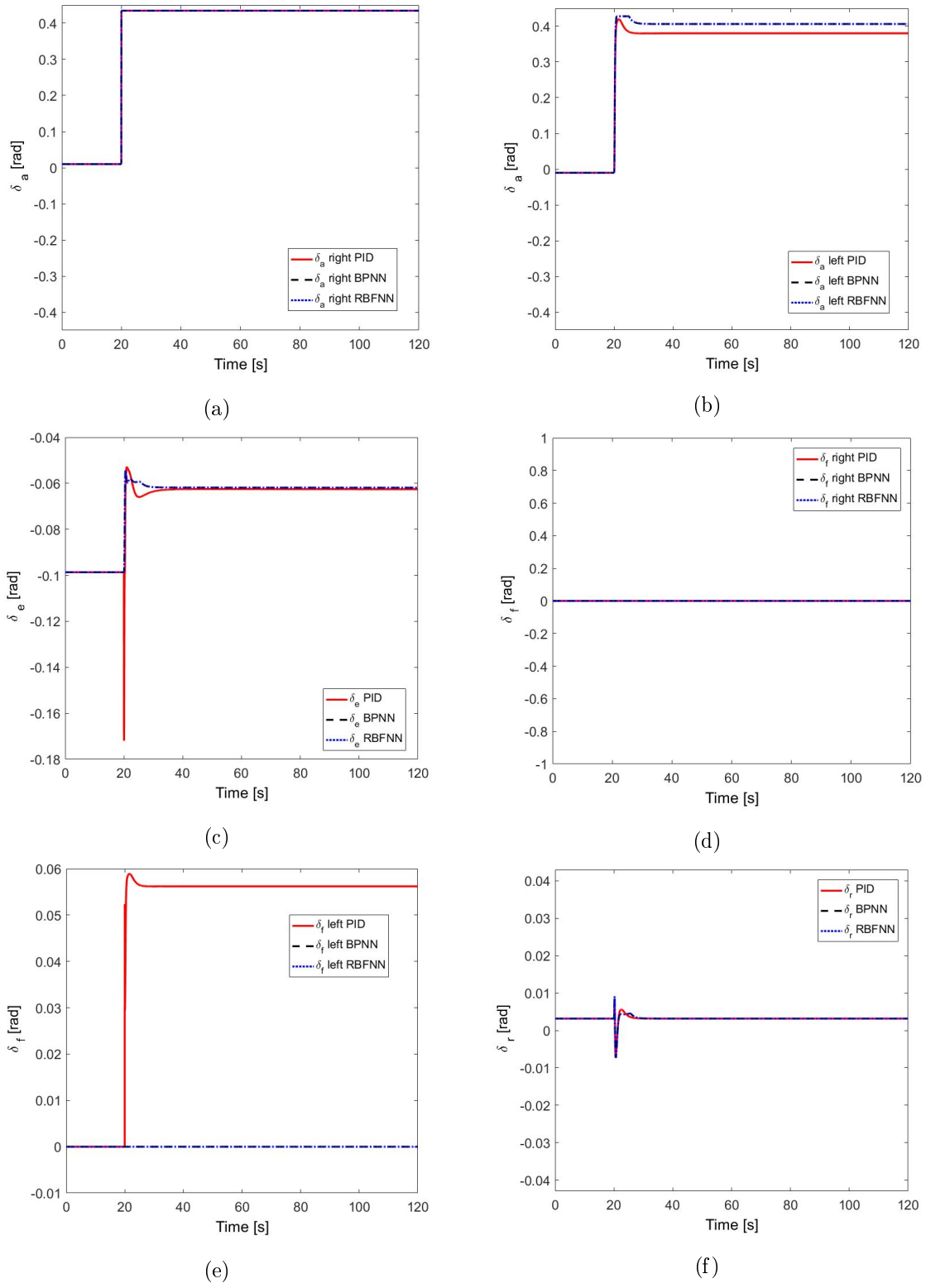
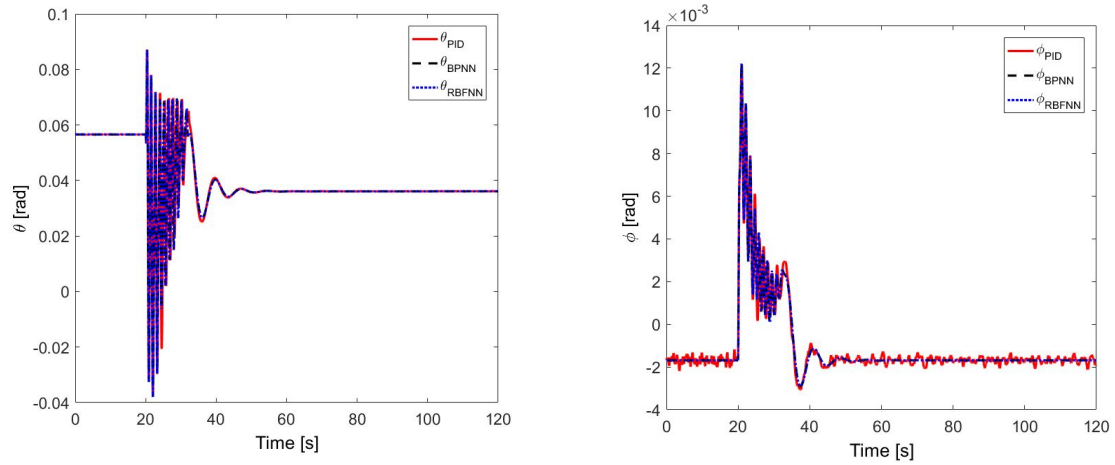


Figure 4.24: Simulated time history of right aileron (a), left aileron (b), elevator (c), right flap (d), left flap (e), and rudder (f) deflections (100% hardover right aileron failure) during level flight

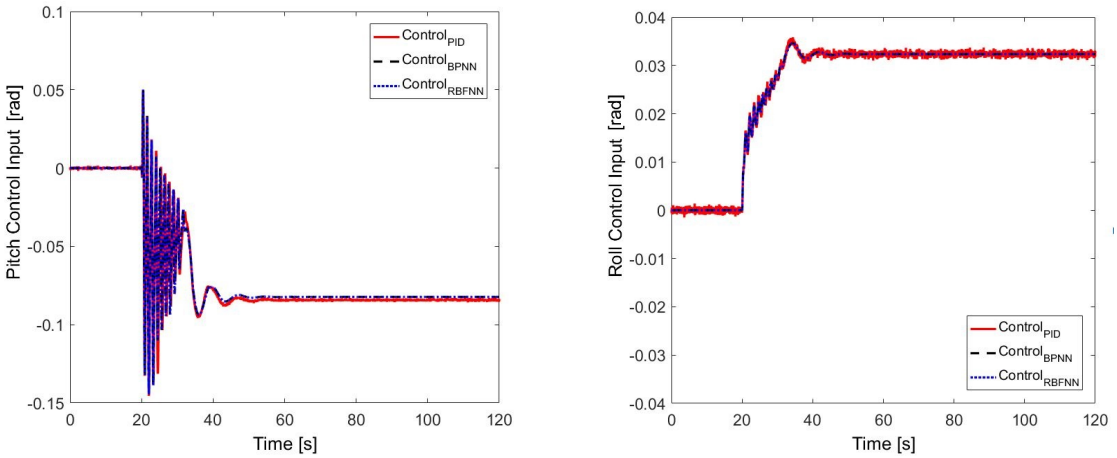
4.4.1.8 Elevator and Aileron Combined Failure - *0 rad* deflection

The 0 *rad* failure was again induced but in this case for both right aileron and elevators. The result of this simulation was a combination of the two previous cases: there was a change in both roll and pitch as shown by in Figure 4.25. Similar to the cases above, the aircraft was able to return to the desired setpoint in both the PID and NN-based cases after some initial oscillations. There was an increase in the roll control input to return to a stable level as shown in Figure 4.26, while the pitch control decreased.



(a) (b)

Figure 4.25: Simulated time history of pitch (a) and bank (b) angles (0 *rad* elevator and right aileron deflection failure) during level flight



(a) (b)

Figure 4.26: Simulated time history of pitch (a) and roll (b) control inputs (0 *rad* elevator and right aileron deflection failure) during level flight

Figure 4.28 shows the time history of the PID-based controllers' gains during 0 *rad* elevator and right aileron hardover deflection failure. There is a slight change in the BPNN-based controllers' pitch gains while the RBFNN-based controllers' pitch gains remain the same. Some slight initial oscillation can be observed in the BPNN-based controllers' roll gains.

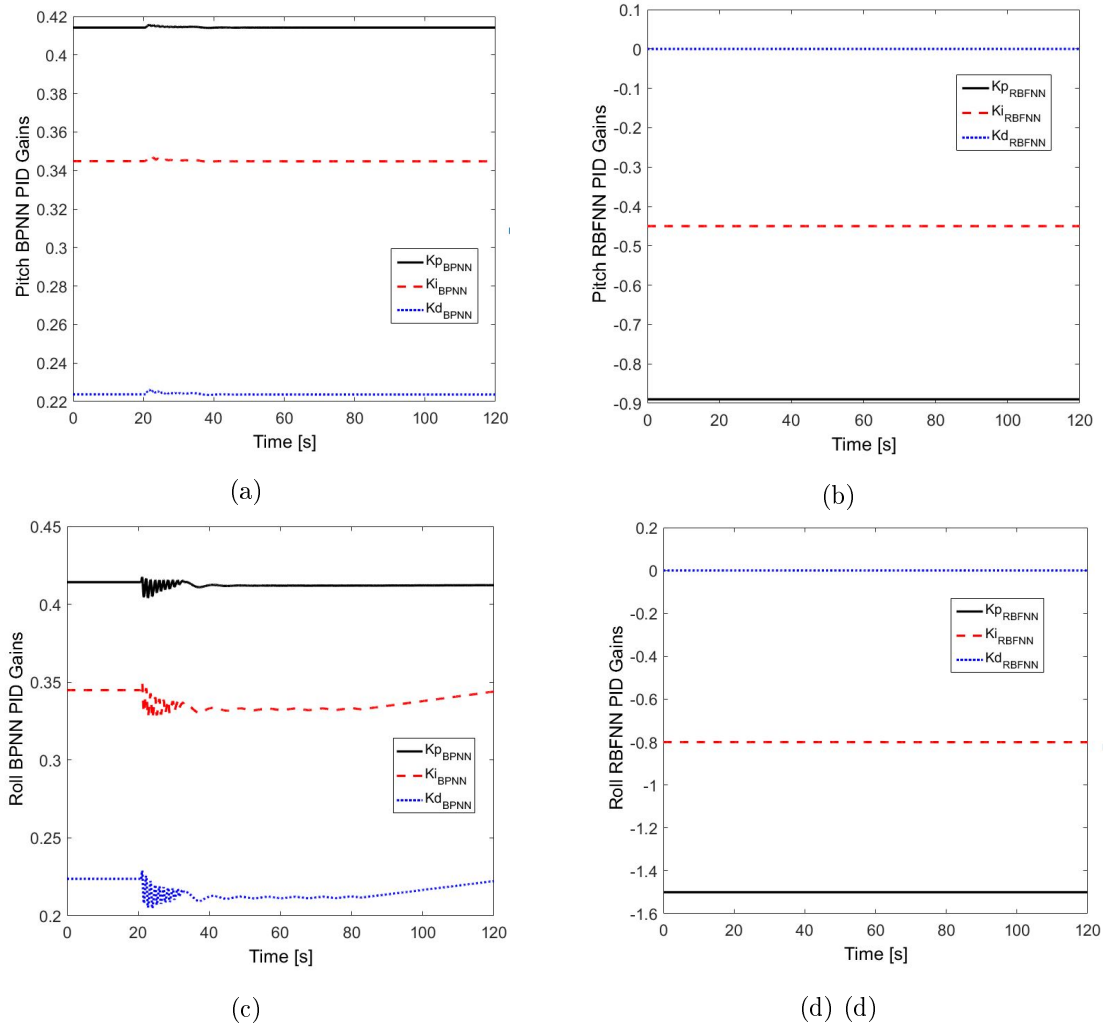


Figure 4.27: Simulated time history of pitch (BPNN-based (a), and RBFNN-based (b)), and roll (BPNN-based (c), and RBFNN-based (d)) gains during level flight with 0 *rad* elevator and right aileron failure

Figure 4.28 shows the control surface deflections during the simulation.

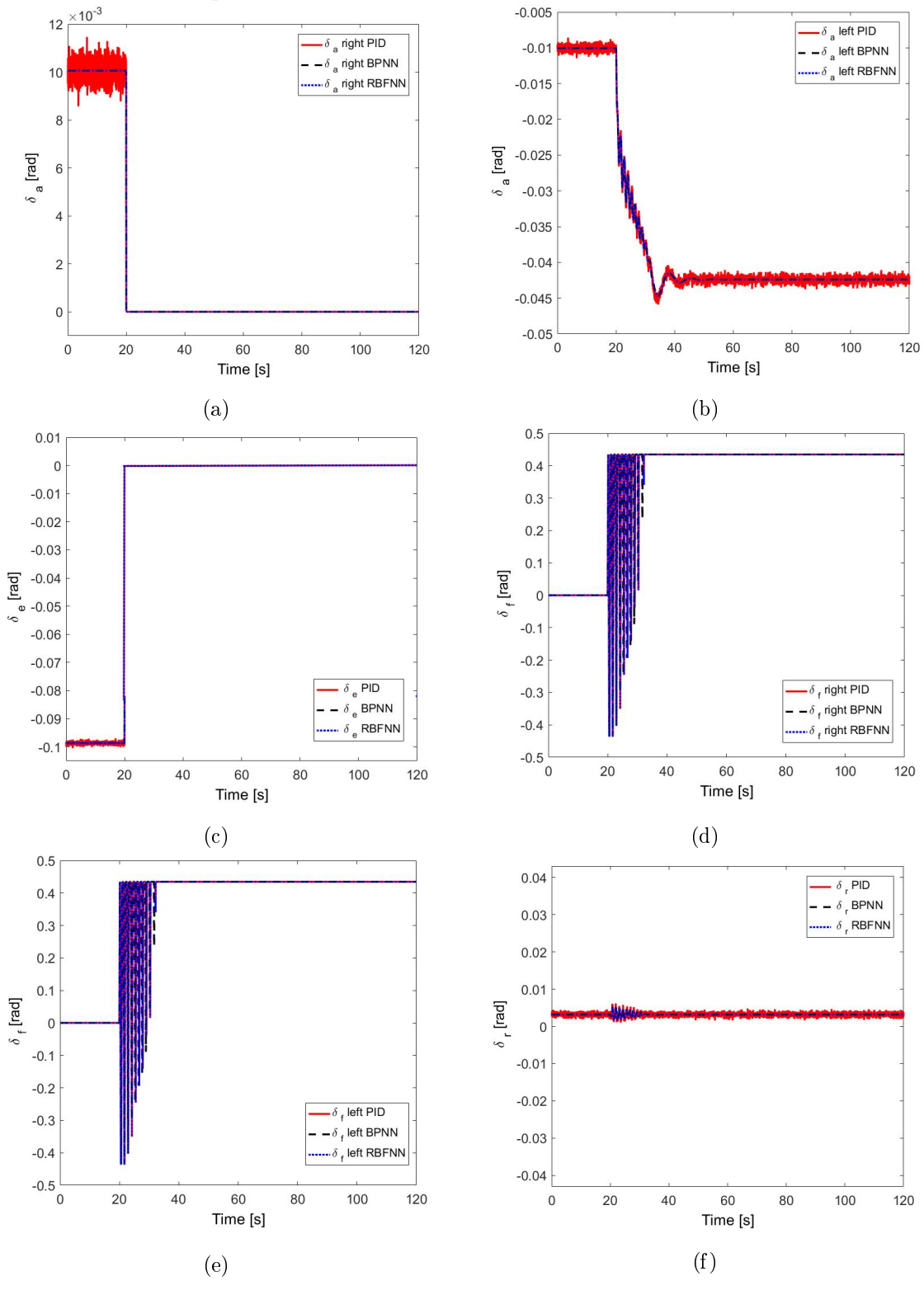
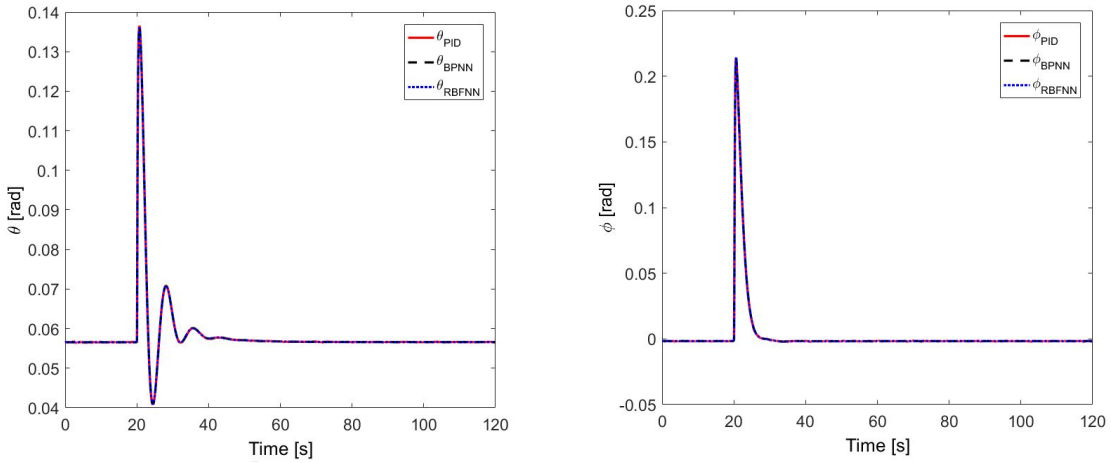


Figure 4.28: Simulated time history of right aileron (a), left aileron (b), elevator (c), right flap (d), left flap (e), and rudder (f) deflections (0 *rad* elevator and right aileron deflection failure) during level flight

4.4.1.9 Elevator and aileron combined failure - 50% Hardover

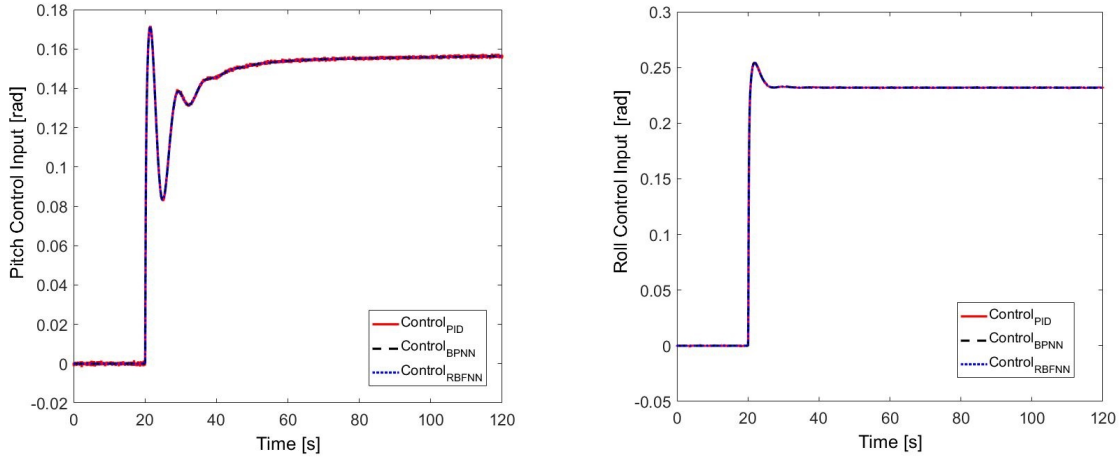
The simulation was run, but with a 50% hardover failure of both elevators and right aileron. It should be noted that the aileron deflection direction is not the same as for the aileron-only case. Figure 4.29 shows the time history of the pitch and roll angles for the level flight case and shows a large pitch and roll disturbance. The pitch disturbance was found to be similar in the PID-based and NN-based cases. There are also oscillations that occurred before reaching the setpoint value. The roll angle (Figure 4.29 (b)) shows an initial disturbance then returns to the setpoint.



(a) (b)

Figure 4.29: Simulated time history of pitch and bank angles (50% elevator and right aileron hardover failure) during level flight

Figures 4.30 (a) and (b) show the changes in control input for the pitch and roll loops and show that the input rose sharply with slight oscillation for the pitch to a higher value. This was then stable for the rest of the simulation.



(a) (b)

Figure 4.30: Simulated time history of pitch (a) and roll (b) control inputs (50% elevator and right aileron hardover failure) during level flight

Figure 4.31 shows the time history of the PID-based controllers' gains during 50% elevator and right aileron hardover failure. There is a jump in both the pitch and roll gains (see Figure 4.31 (a) and (c)) for the BPNN-based controllers' case before they settled. The RBFNN-based controllers' gains were hardly affected.

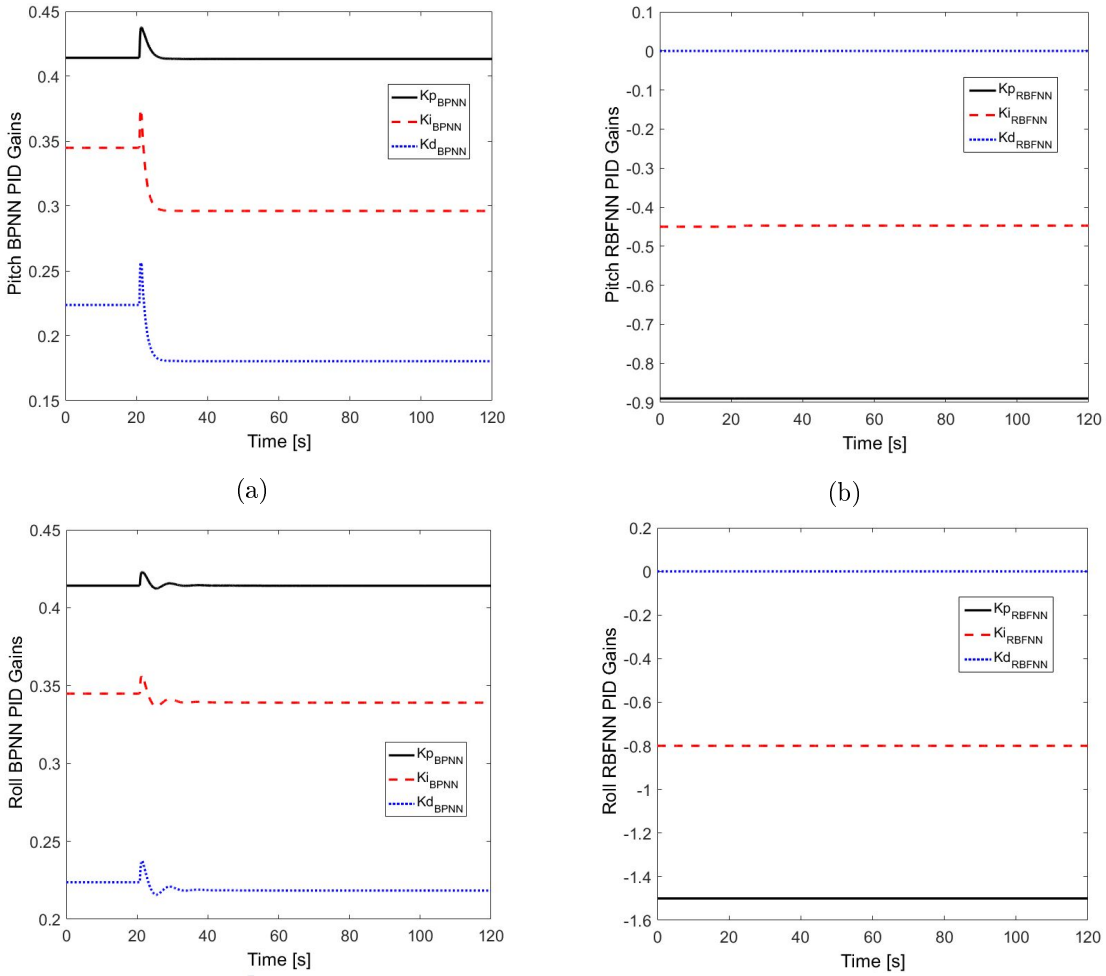


Figure 4.31: Simulated time history of pitch (BPNN-based (a), and RBFNN-based (b)), and roll (BPNN-based (c), and RBFNN-based (d)) gains during level flight with 50% elevator and right aileron hardover failure

Figures 4.32 (a) to (f) show the control deflections during the simulation and show the point at which the elevators and aileron failed.

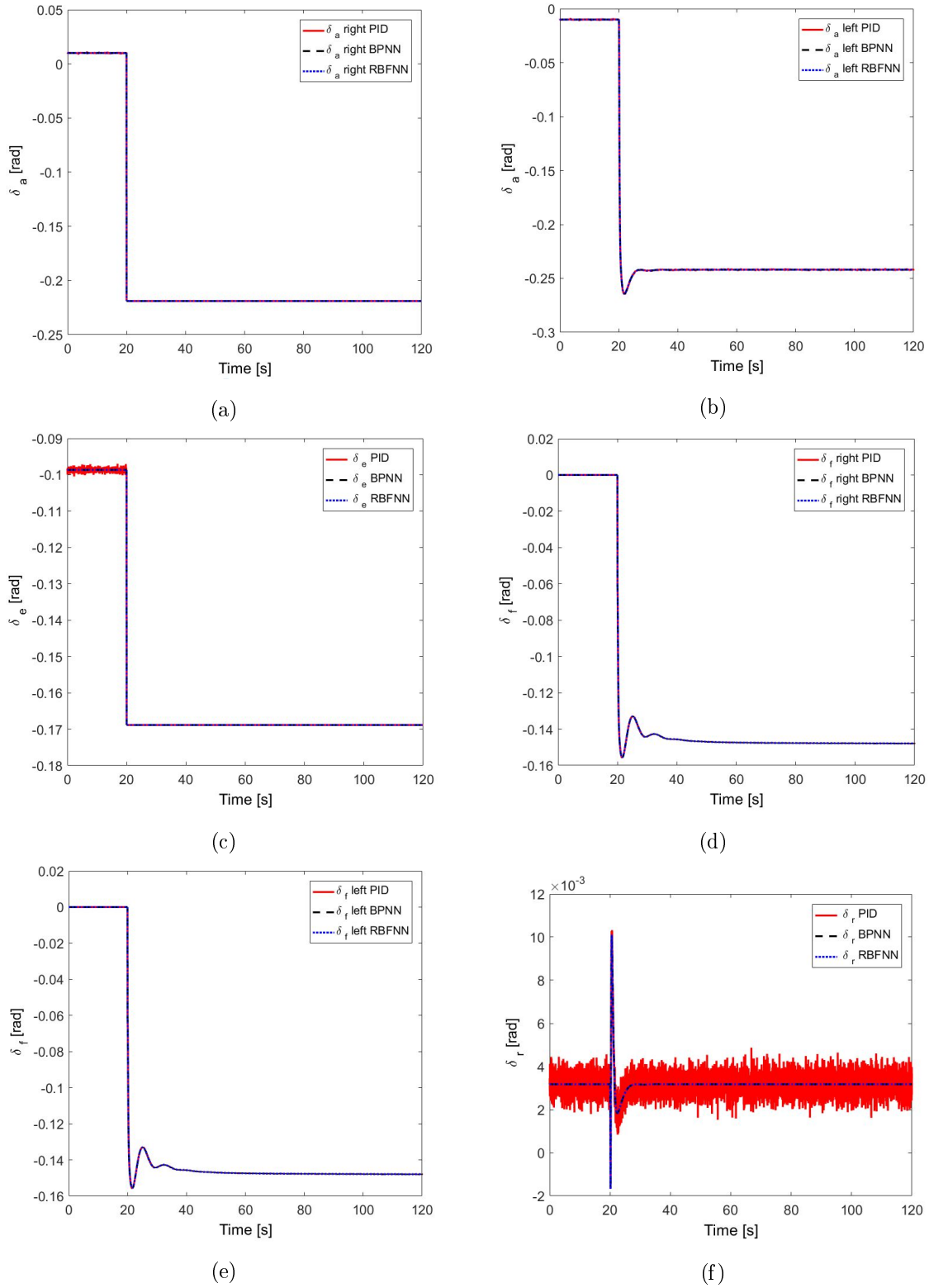


Figure 4.32: Simulated time history of right aileron (a), left aileron (b), elevator (c), right flap (d), left flap (e), and rudder (f) deflections (50% elevator and right aileron hardover failure) during level flight

4.4.1.10 Elevator and Aileron Combined Failure - 100% Hardover

The simulation was run on this occasion inducing a 100% control deflection failure for both the elevator and right aileron. For the roll, the controller was able to handle the failure but with the pitch angle it was noted that there was an inability to re-establish the desired setpoint as shown in Figure 4.33. There is an oscillatory behaviour in both the pitch and bank angles before they settle to a new equilibrium value. The angle of attack response plot is shown in Figure 4.33 (c), the plot shows that the angles of attack remain within reasonable limits throughout the simulation though they reached the limits at the maximum peak.

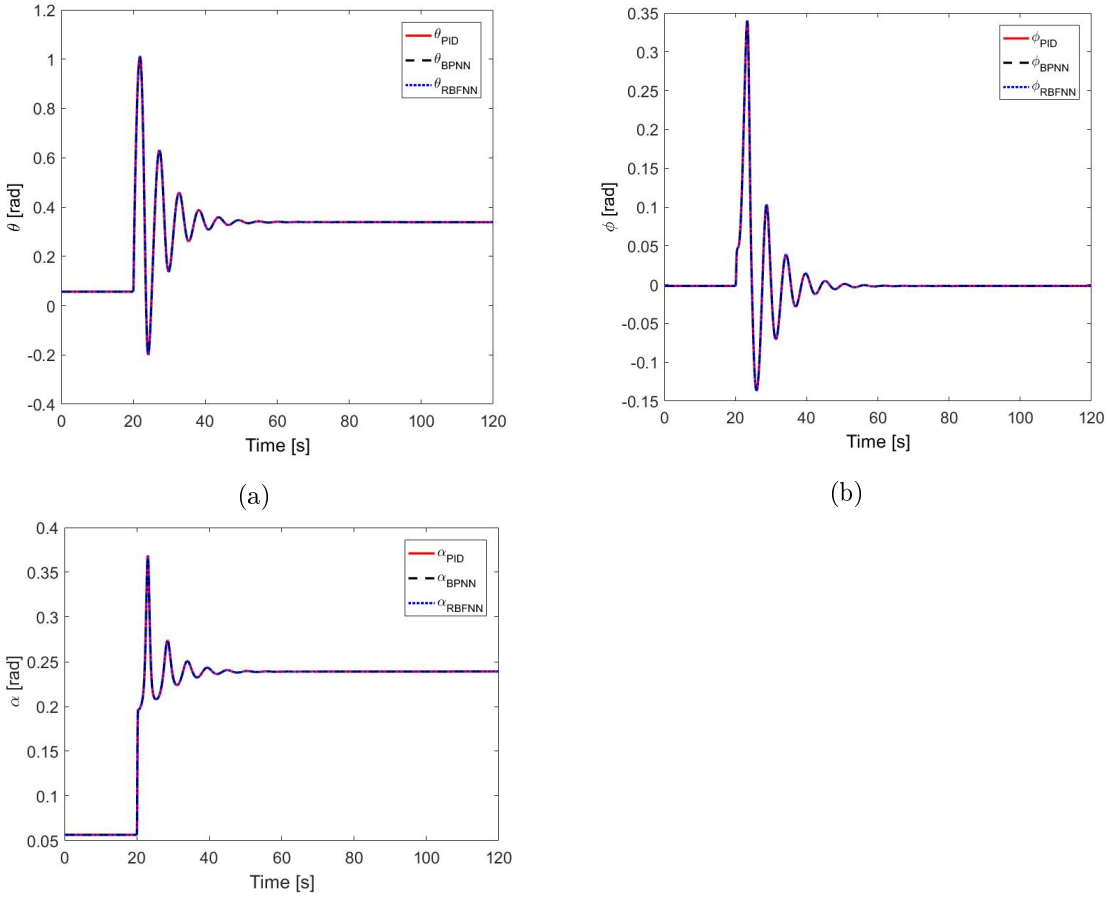
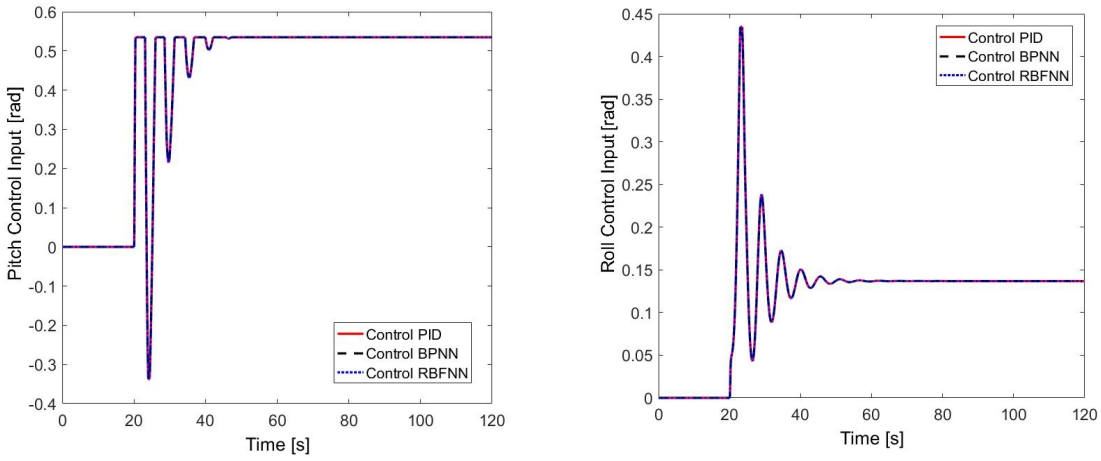


Figure 4.33: Simulated time history of pitch (a) and bank (b) angles and angle of attack (c) (100% elevator and right aileron hardover failure) during level flight

Figure 4.34 shows the rapid increase in control input for the pitch and roll controllers. This is in response to the insufficient pitch authority which is observed by the saturation of the pitch control input.



(a) (b)

Figure 4.34: Simulated time history of pitch (a) and roll (b) control inputs (100% elevator and right aileron hardover failure) during level flight

Figure 4.35 shows the time history of the PID-based controllers' gains during 100% elevator and right aileron hardover failure. The pitch BPNN-based controllers' gains show a slight oscillatory response after the initial increase at failure while the RBFNN-based controllers' pitch integral control shows a slight increase at failure. The BPNN-based controllers' roll proportional gain shows an initial large increase then it shows oscillatory behaviour which slowly dampens out, while the integral and derivative gains show similar behaviour with an initial large decrease. The RBFNN-based controllers' roll gains only show an initial jump then increase in integral action.

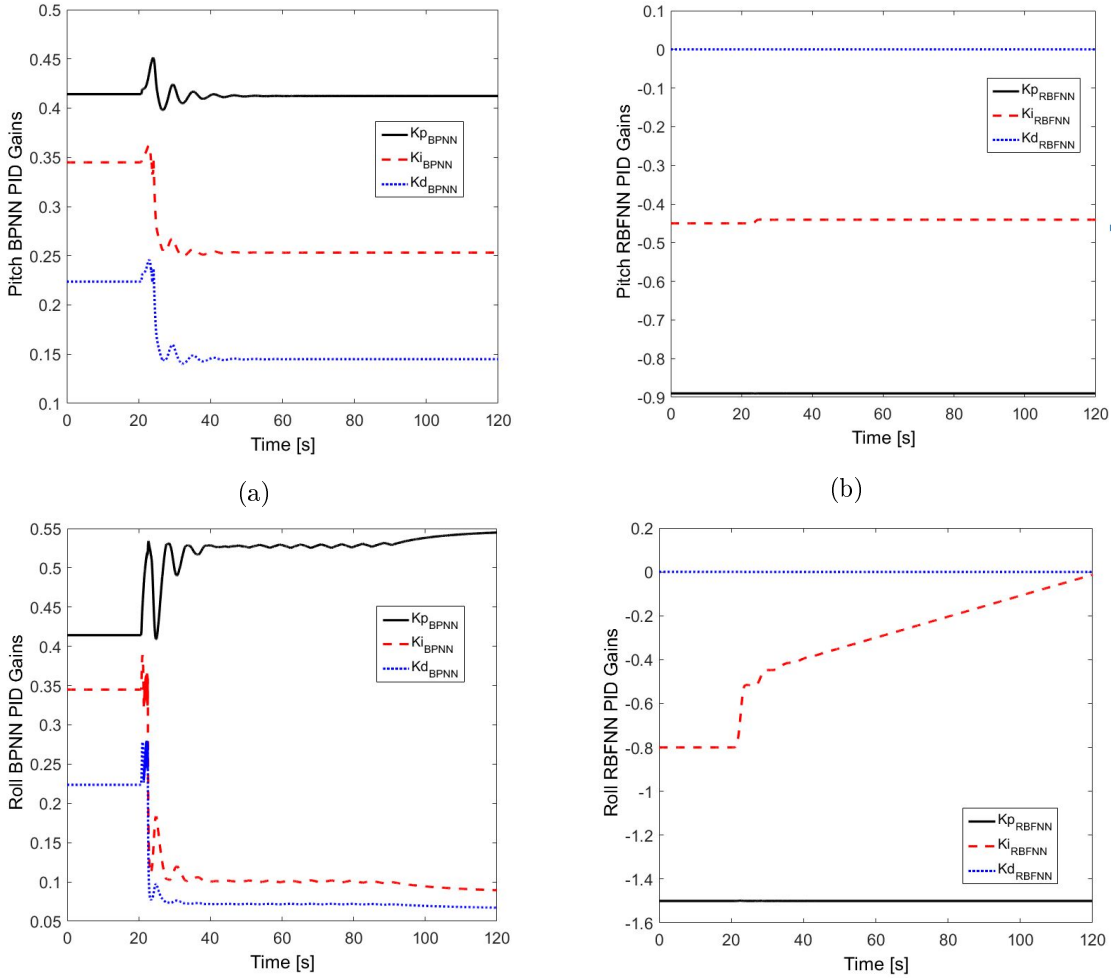


Figure 4.35: Simulated time history of pitch (BPNN-based (a), and RBFNN-based (b)), and roll (BPNN-based (c), and RBFNN-based (d)) gains during level flight with 100% elevator and right aileron hardover failure

Figure 4.36 (b) shows that there is a number of left aileron oscillations until some stable deflection is reached. The flap deflections immediately move to their maximum positions at the occurrence of failure. There is also some initial small oscillation induced in the rudder which dampens out.

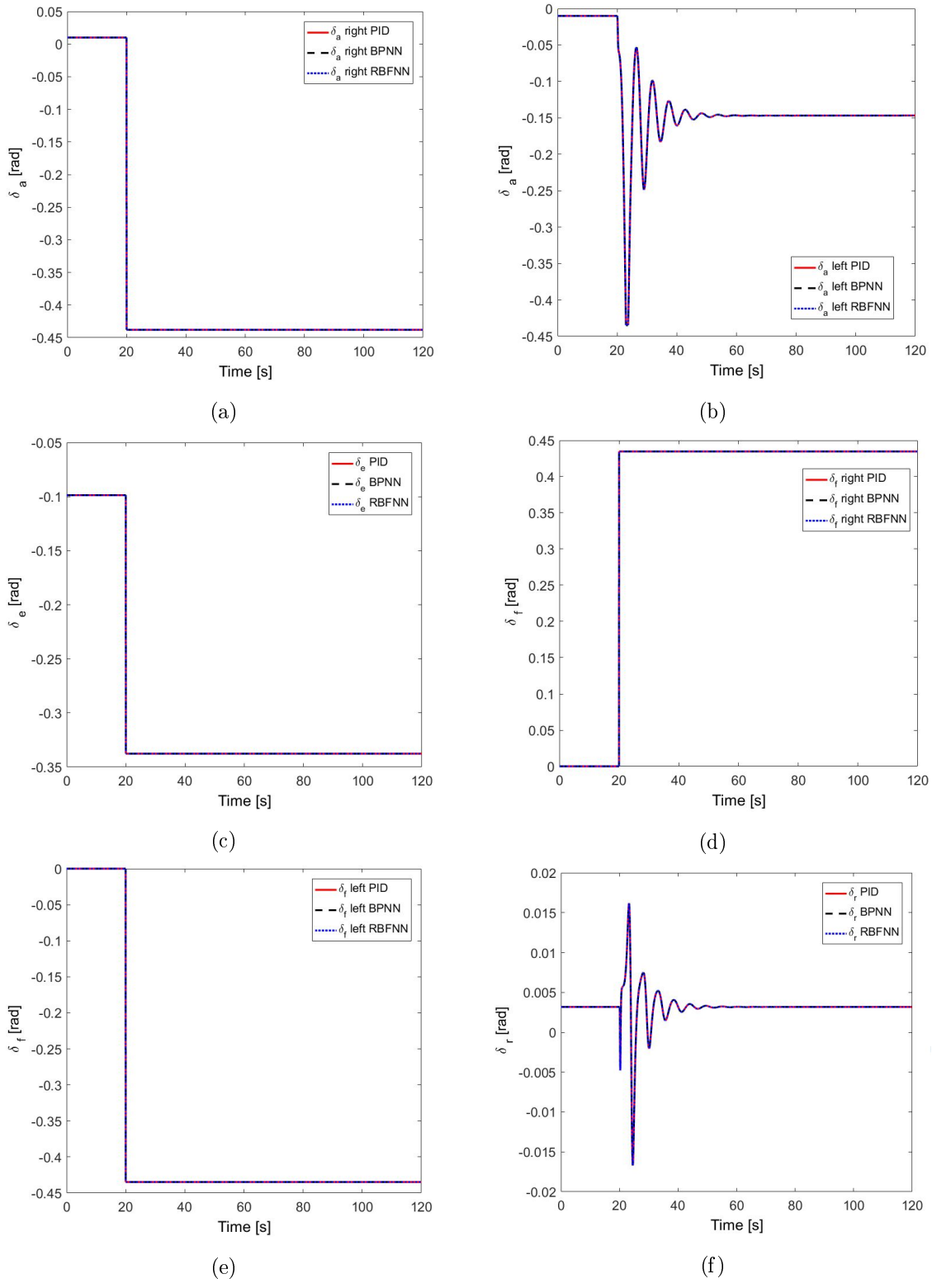


Figure 4.36: Simulated time history of right aileron (a), left aileron (b), elevator (c), right flap (d), left flap (e), and rudder (f) deflections (100% elevator and right aileron hardover failure) during level flight

4.4.2 Middle Control Loop Simulation

The middle control loop was tested in combination of first climb then the level flight mode. This flight mode controls airspeed through throttle input and altitude and heading through use of pitch and roll command respectively. For these tests, the airspeed was initially set to a value of $17m/s$ and altitude $300m$ above sea level. This altitude and airspeed was chosen as it was anticipated that this would be the primary flight condition should experimental testing be conducted.

4.4.2.1 Control - No Failure

A control simulation was run to determine the response of the aircraft under these conditions, with no failures. The simulation was run for both the PID-based and NN-based controllers' cases. The responses between the cases showed no noticeable differences. Figure 4.38 shows the pitch, bank and heading angles' responses in the no fault condition, one can see an initial overshoot each time the setpoint changes (at 0 and 60 seconds) before it moves to the setpoint value for the heading and bank responses while the pitch angle shows a gradual move to the setpoint.

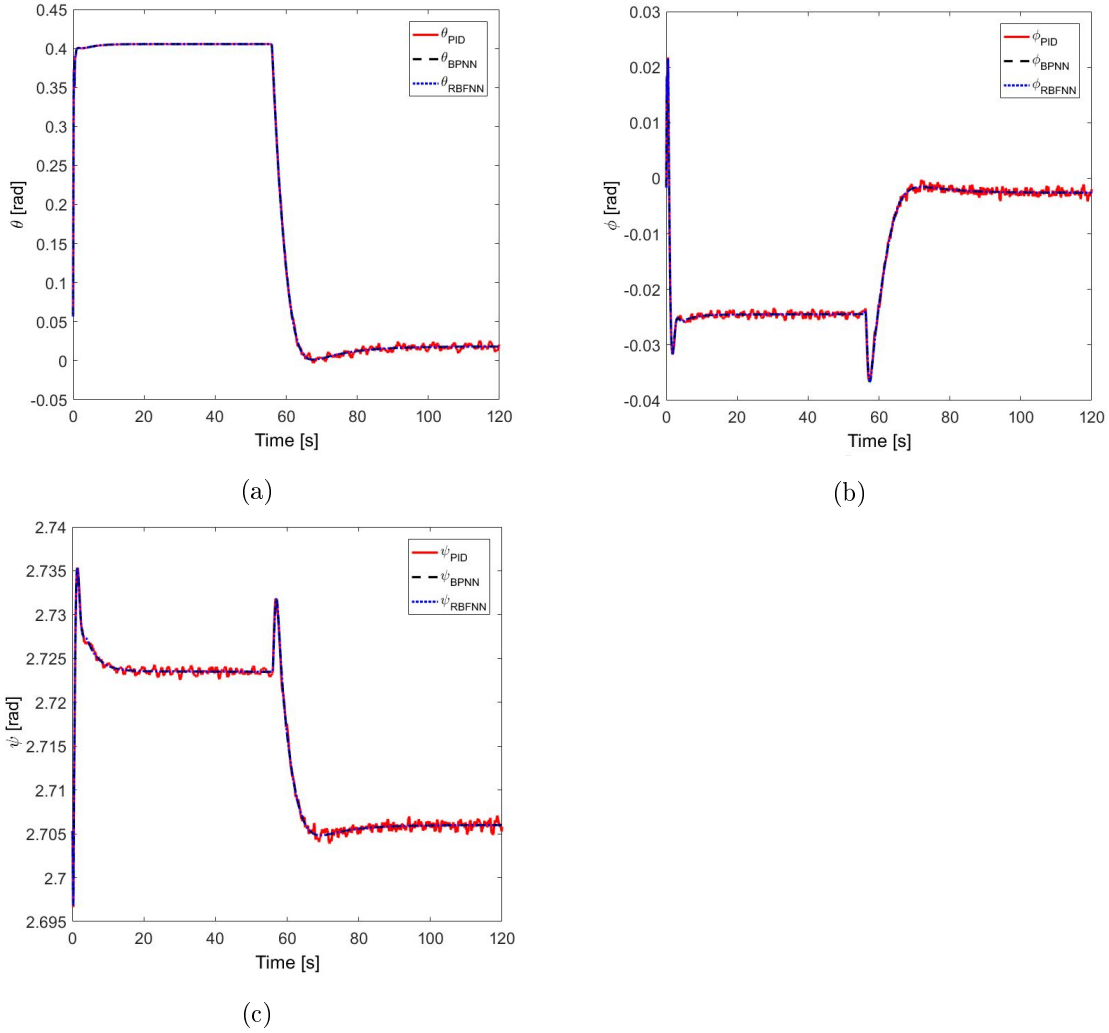
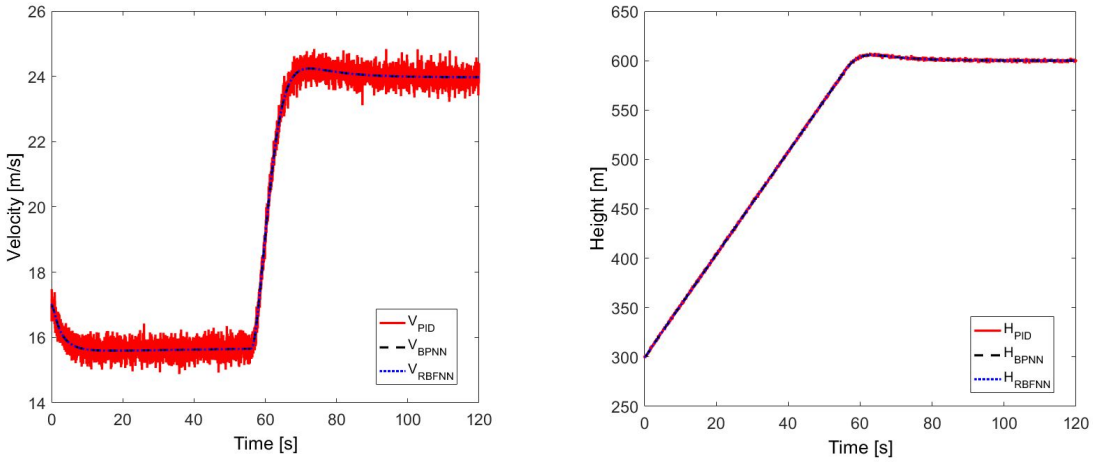


Figure 4.37: Simulated time history of middle loop pitch (a), bank (b), and heading (c) angles (no failures)

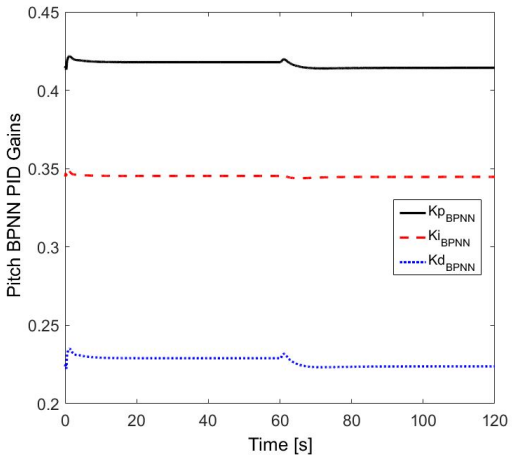
Figure 4.38 (a) shows that the airspeed gradually moves towards the setpoint value at the start then when the setpoint changes at 60 seconds it gradually moves to the new setpoint with a slight overshoot. Figure 4.38 (b) shows the UAV maintaining the desired climb rate of $5m/s$ until it is required to maintain level altitude at 60 seconds.



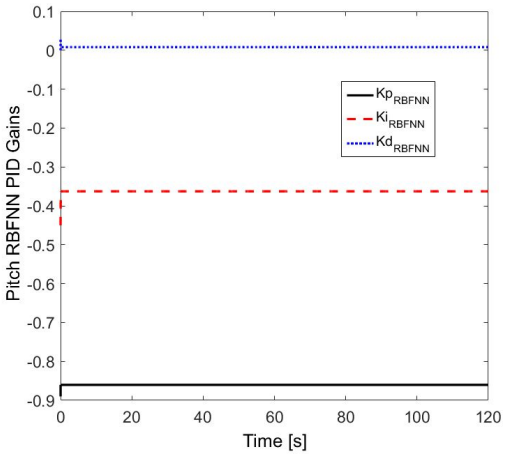
(a) (b)

Figure 4.38: Simulated time history of the airspeed (a), and altitude (b) (no failures)

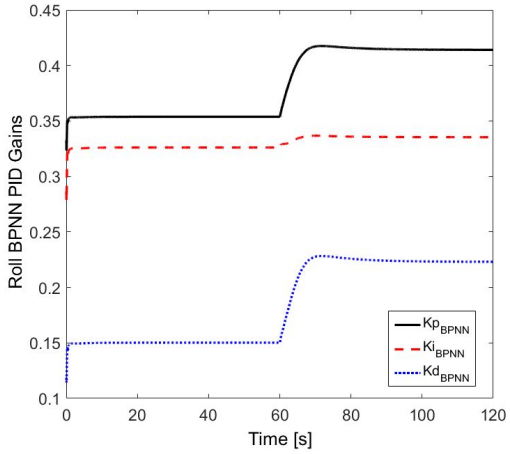
Figure 4.39 shows the variation in PID-based controllers' gains for all control loops. The pitch gains (Figure 4.39 (a) and (b)) shows that the BPNN-based controllers' gains have a slight change at the time when the setpoint changes while the RBFNN-based controllers' gains remain constant as before. The roll gains (Figure 4.39 (c) and (d)) show a similar pattern is found in the RBFNN-based controllers' gains while the BPNN-based controllers gains increase then level out when the setpoint changes, the proportional and derivative gains experience a much larger increase. The velocity gains (Figure 4.39 (e) and (f)) for the BPNN-based case are very close to zero while for the RBFNN-based case the integral gain initially rapidly decreases then increases at the next setpoint change. The altitude gains on the other hand (Figure 4.39 (g) and (h)) remain constant for all cases. Lastly, the heading gains ((Figure 4.39 (i) and (j)) show a response similar to that of the pitch gains with the BPNN-based gains having an initial gradual decrease before the setpoint change at 60 seconds.


Pitch BPNN PID Gains
Time [s]
Kp_BPNN
Ki_BPNN
Kd_BPNN

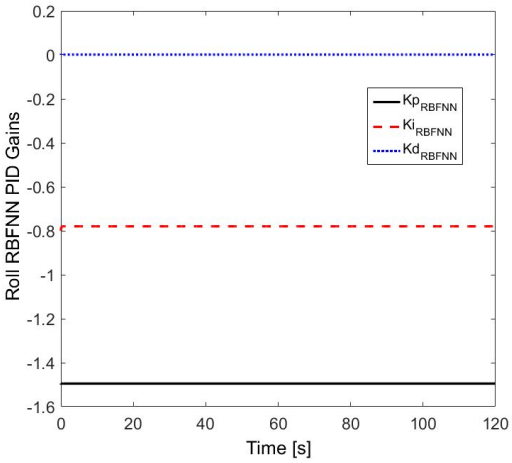
(a)


Pitch RBFNN PID Gains
Time [s]
Kp_RBFNN
Ki_RBFNN
Kd_RBFNN

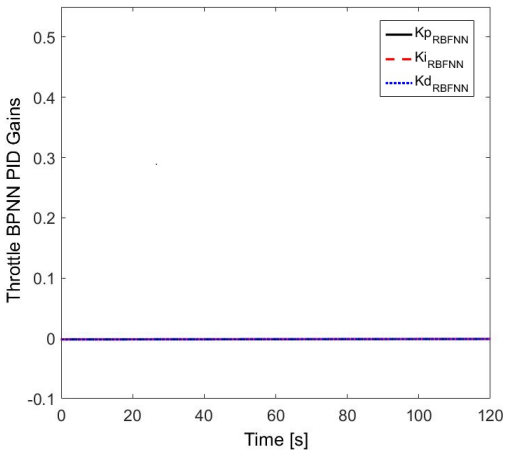
(b)


Roll BPNN PID Gains
Time [s]
Kp_BPNN
Ki_BPNN
Kd_BPNN

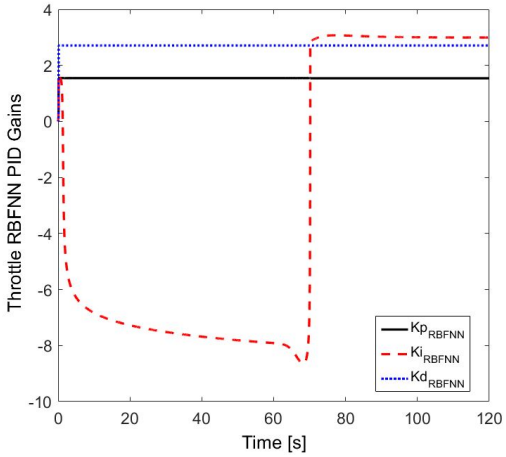
(c)


Roll RBFNN PID Gains
Time [s]
Kp_RBFNN
Ki_RBFNN
Kd_RBFNN

(d)


Throttle BPNN PID Gains
Time [s]
Kp_RBFNN
Ki_RBFNN
Kd_RBFNN

(e)


Throttle RBFNN PID Gains
Time [s]
Kp_RBFNN
Ki_RBFNN
Kd_RBFNN

(f)

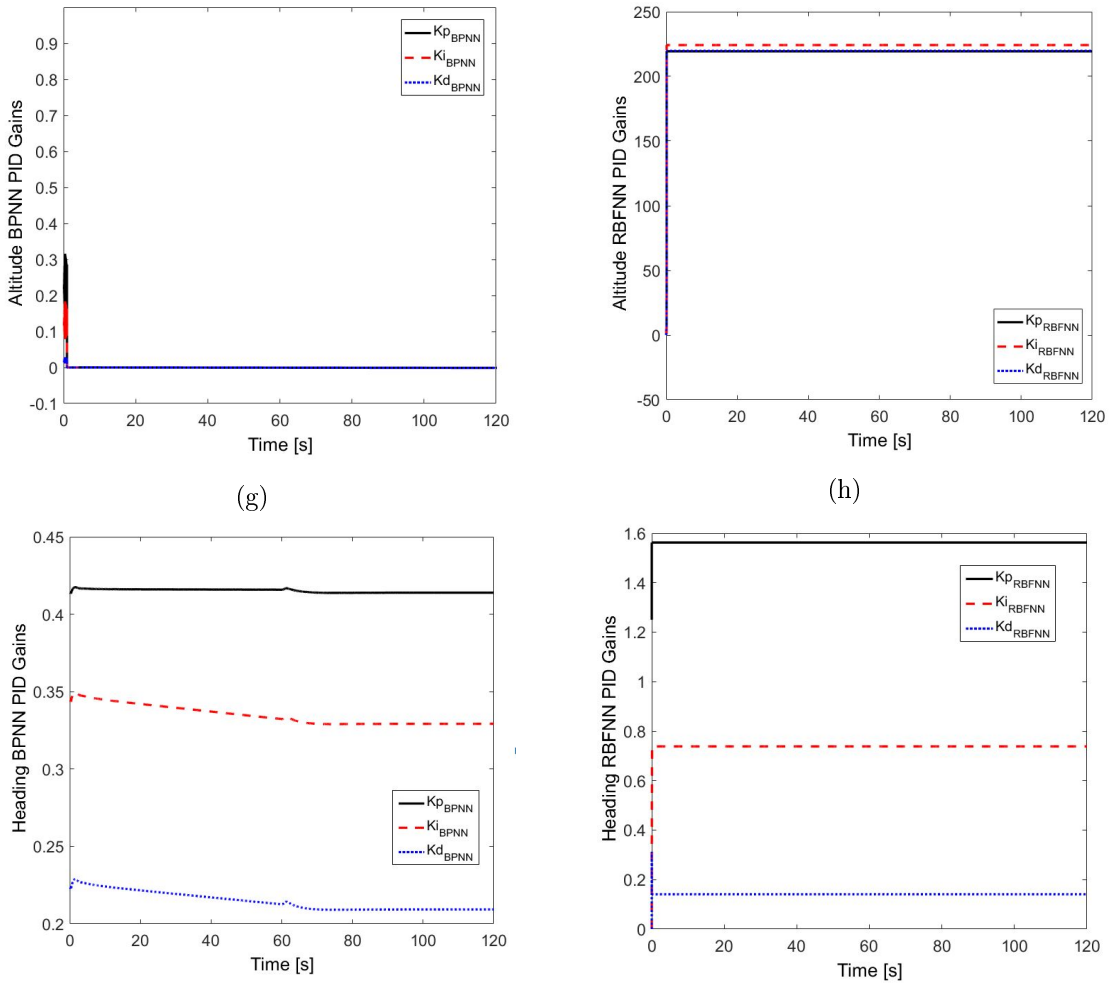


Figure 4.39: Simulated time history of the pitch (BPNN-based (a), and RBFNN-based (b)), and roll (BPNN-based (c), and RBFNN-based (d)), velocity (BPNN-based (e), and RBFNN-based (f)), altitude (BPNN-based (g), and RBFNN-based (h)), and heading (BPNN-based (i), and RBFNN-based (j)) PID gains with no failure

4.4.2.2 Elevator Failure - *0 rad* Deflection

The simulation was run inducing 0 *rad* elevator deflection failure. That is, the elevator moves from its current position, at the time of failure to its neutral position. To maintain consistency with the inner loop simulations, the failure was once again induced at 20 seconds. Figure 4.40 shows the time history of the pitch, roll and heading angles. The pitch angle (Figure 4.40 (a)) experiences a drop when the failure occurs; there is a very slight change from the no-fault case. A similar response is observed in the heading and roll angles.

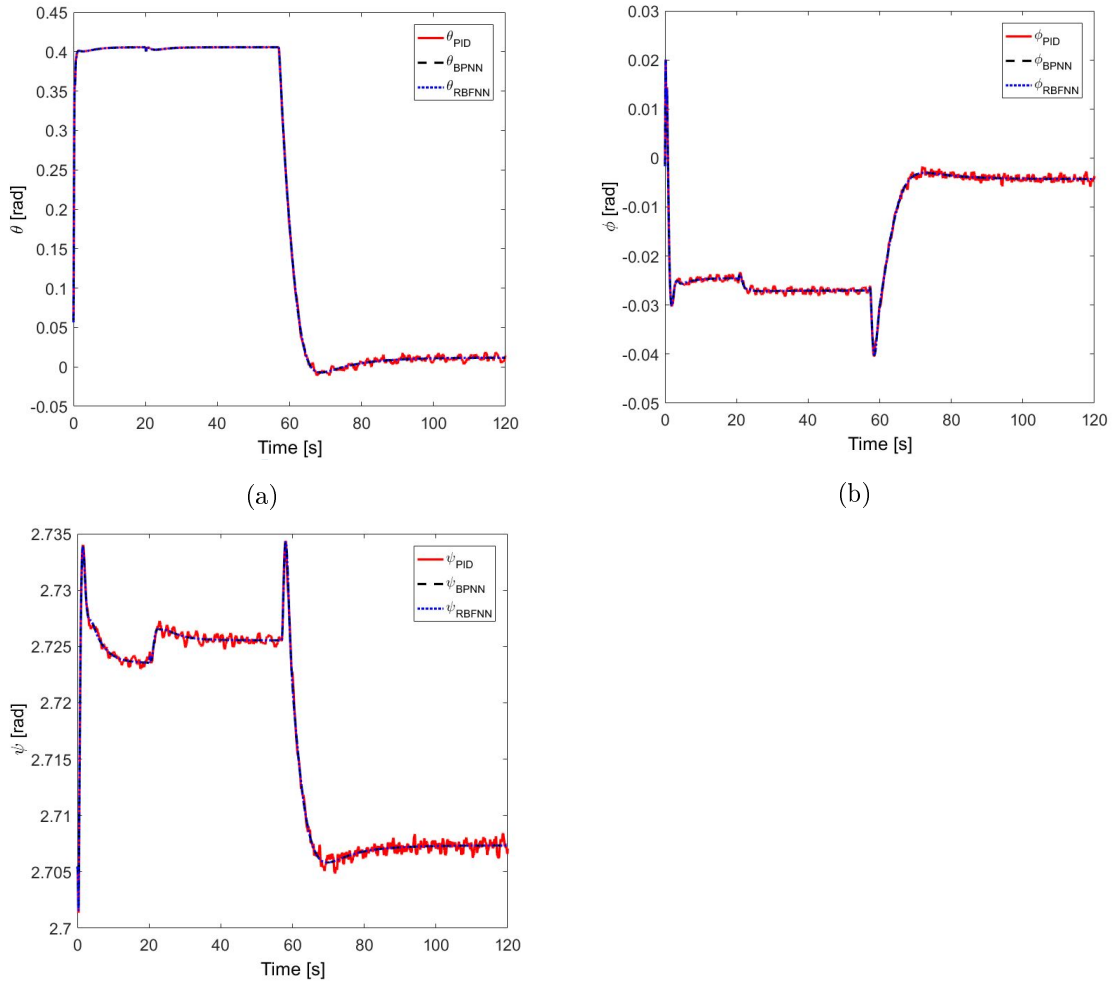
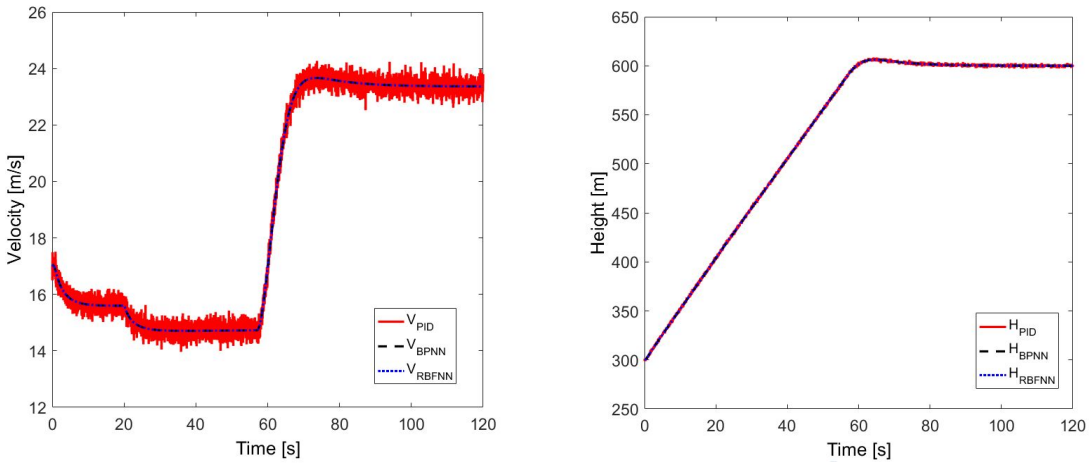


Figure 4.40: Simulated time history of middle loop pitch (a), bank (b) and heading (c) angles (0 *rad* deflection)

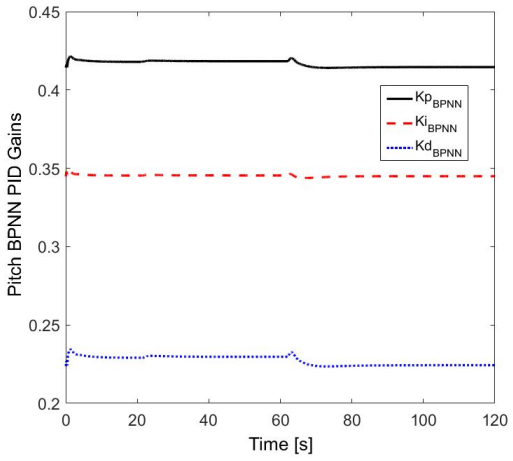
Figure 4.41 shows the time history of the airspeed. A similar result was obtained for the altitude plot of the aircraft, shown in Figure 4.41 (b) as there is no discernible difference in altitude due to failure. There is however a slight drop in the velocity when error occurs.



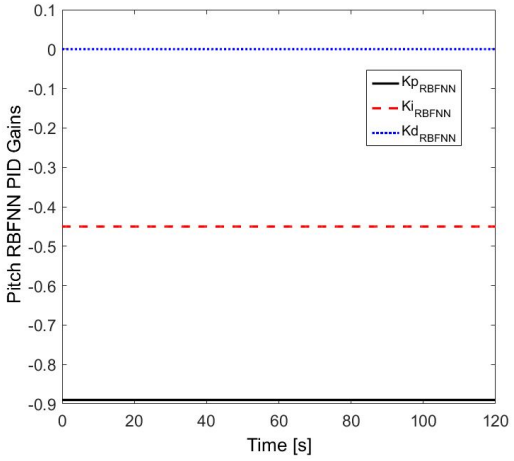
(a) (b)

Figure 4.41: Simulated time history of the airspeed (a), and altitude (b) (0 *rad* deflection)

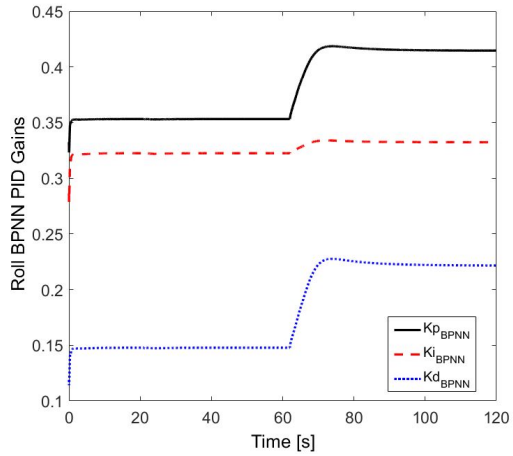
Figure 4.42 shows the variation in PID gains' for all control loops. The gains are similar to that of the no-fault cases with a hardly visible disturbance in some of the BPNN-based controllers' gains.


Pitch BPNN PID Gains
Time [s]
Kp_BPNN
Ki_BPNN
Kd_BPNN

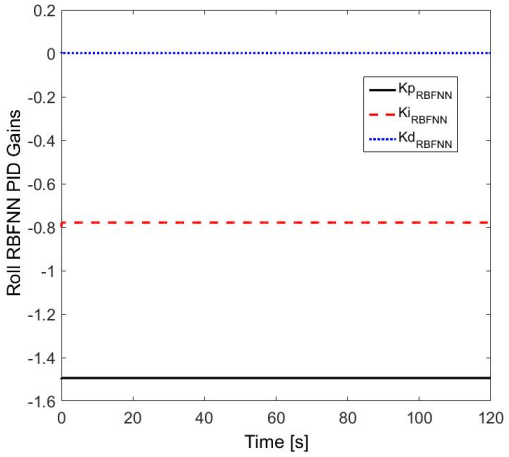
(a)


Pitch RBFNN PID Gains
Time [s]
Kp_RBFNN
Ki_RBFNN
Kd_RBFNN

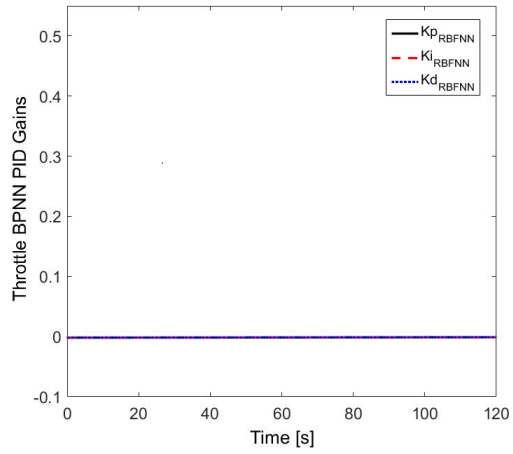
(b)


Roll BPNN PID Gains
Time [s]
Kp_BPNN
Ki_BPNN
Kd_BPNN

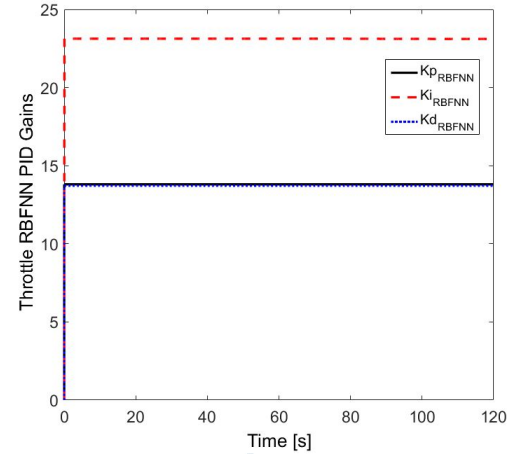
(c)


Roll RBFNN PID Gains
Time [s]
Kp_RBFNN
Ki_RBFNN
Kd_RBFNN

(d)


Throttle BPNN PID Gains
Time [s]
Kp_RBFNN
Ki_RBFNN
Kd_RBFNN

(e)


Throttle RBFNN PID Gains
Time [s]
Kp_RBFNN
Ki_RBFNN
Kd_RBFNN

(f)

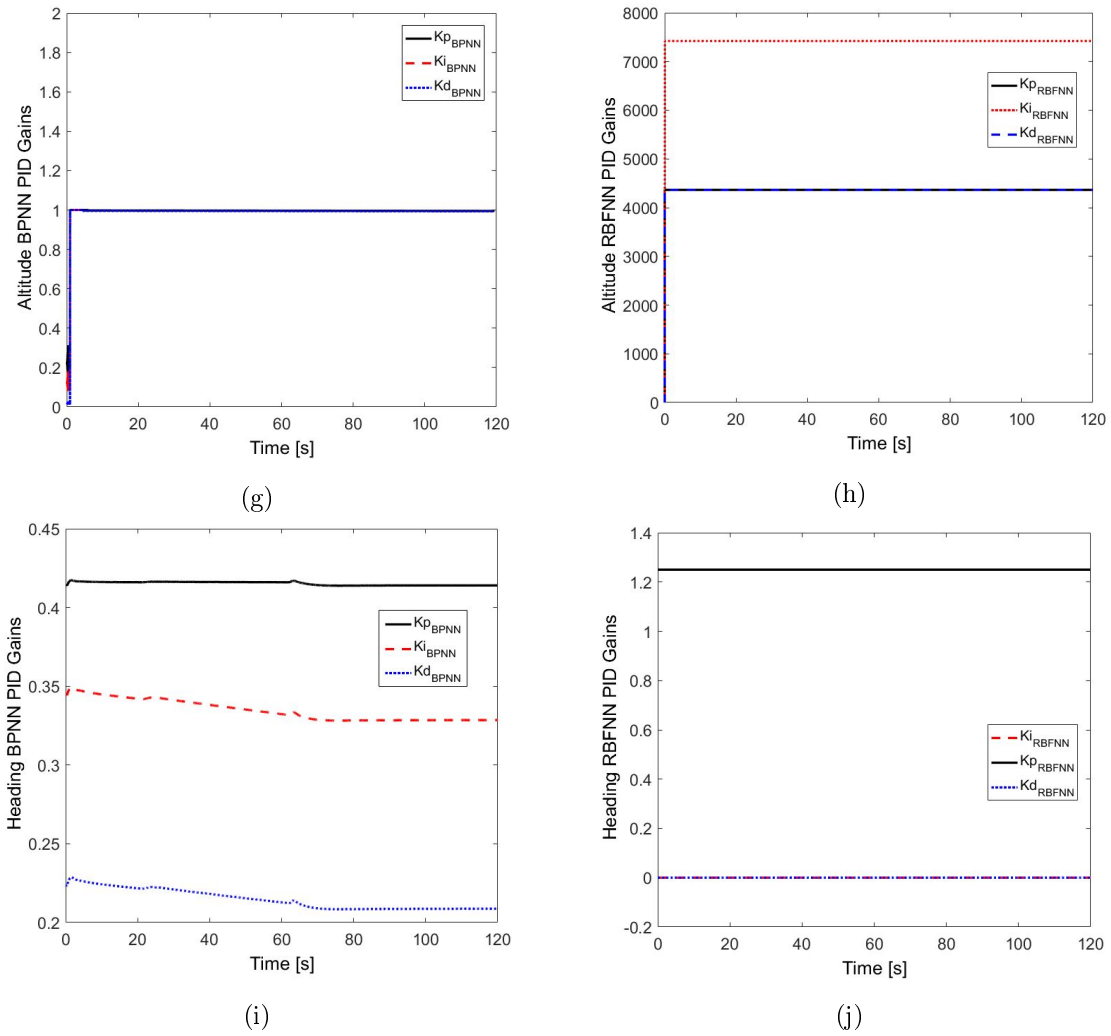


Figure 4.42: Simulated time history of the pitch (BPNN-based (a), and RBFNN-based (b)), and roll (BPNN-based (c), and RBFNN-based (d)), velocity (BPNN-based (e), and RBFNN-based (f)), altitude (BPNN-based (g), and RBFNN-based (h)), and heading (BPNN-based (i), and RBFNN-based (j)) PID gains with 0 *rad* deflection failure

4.4.2.3 Elevator Failure - 50% Hardover

The simulation was run with a more severe failure of 50% hardover elevator failure. Figure 4.43 shows the time history of the pitch, roll and heading angles. There is a disturbance when the failure occurs, but the response quickly returns to the desired setpoint.

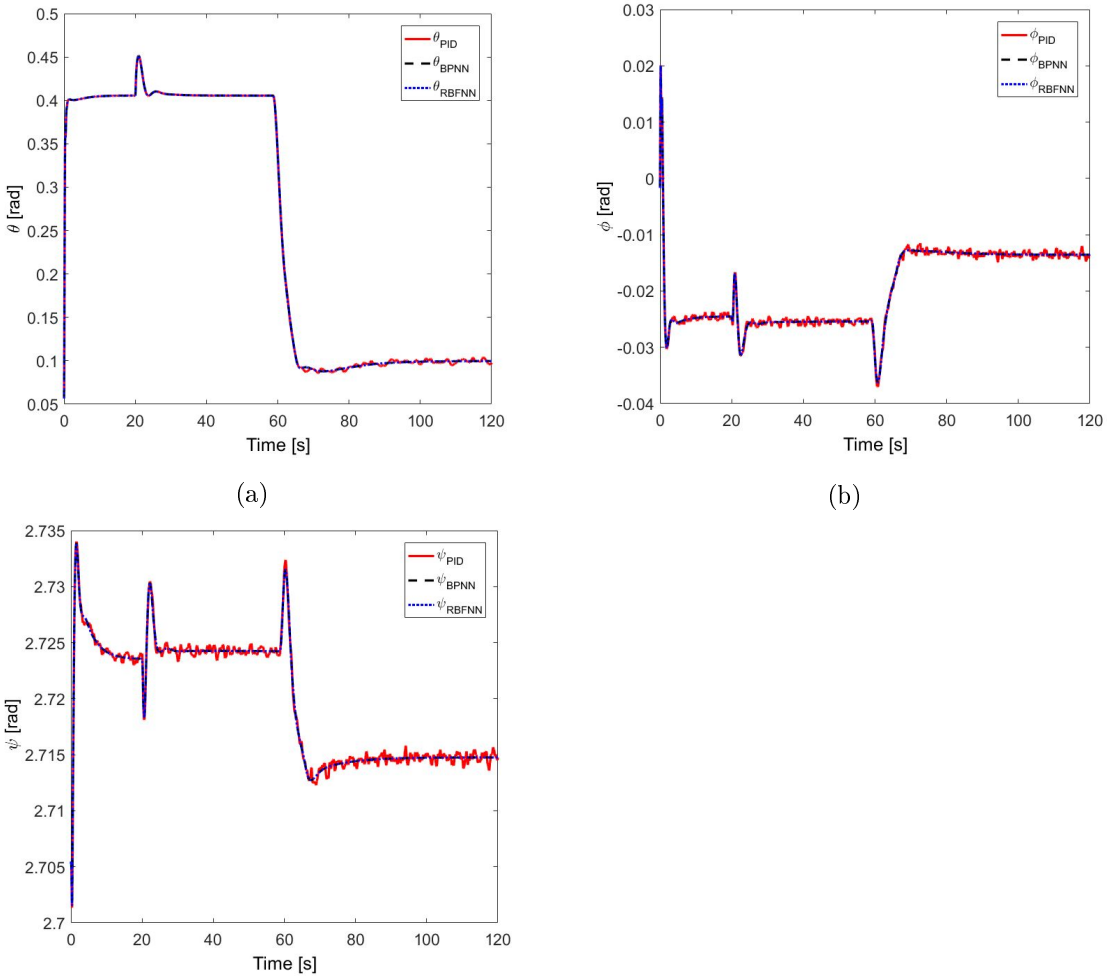
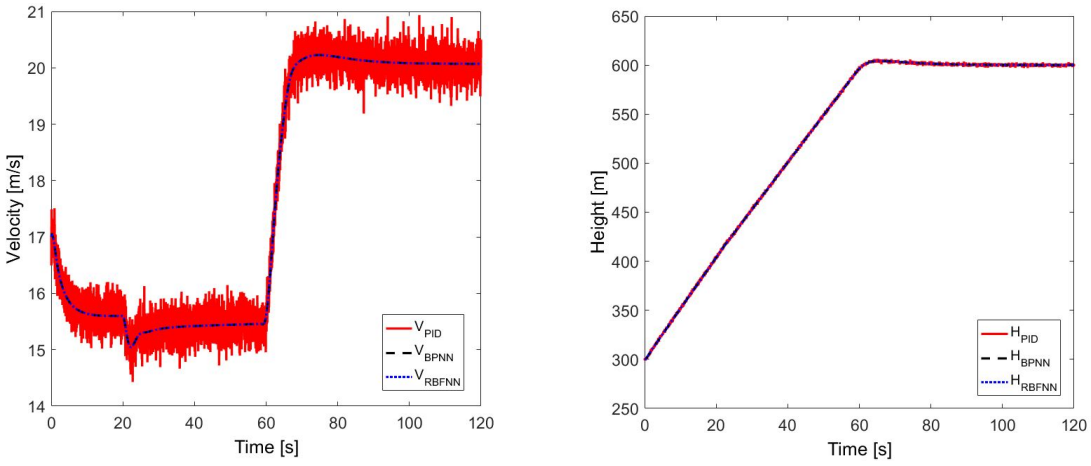


Figure 4.43: Simulated time history of middle loop pitch (a), bank (b) and heading (c) angles (50% hardover elevator failure)

Figure 4.44 shows the time history of the airspeed and altitude. Figure 4.44 (a) shows a slight drop in the airspeed when failure occurs which quickly returns back to the setpoint. Figure 4.44 (b) shows no change in the altitude response.



(a) (b)

Figure 4.44: Simulated time history of the airspeed (a), and altitude (b) (50% hardover elevator failure)

Figure 4.45 shows the variation in PID controllers' gains for all control loops. The pitch gains (Figure 4.45 (a) and (b)) are similar to those of the case with no fault with the difference being the slight jump which occurs in the BPNN-based gains when failure occurs. The roll gains (Figure 4.45 (c) and (d)) show no change for the RBFNN-based case while the BPNN-based gains show a slight jump when failure occurs and an increase when the setpoint is changed. The velocity and altitude gains (Figure 4.45 (e) to (h)) remain constant during the simulation. The heading gains (Figure 4.45 (i) to (j)) follows the same trends as the no-fault case with a slight jump when failure occurs like the pitch gains.

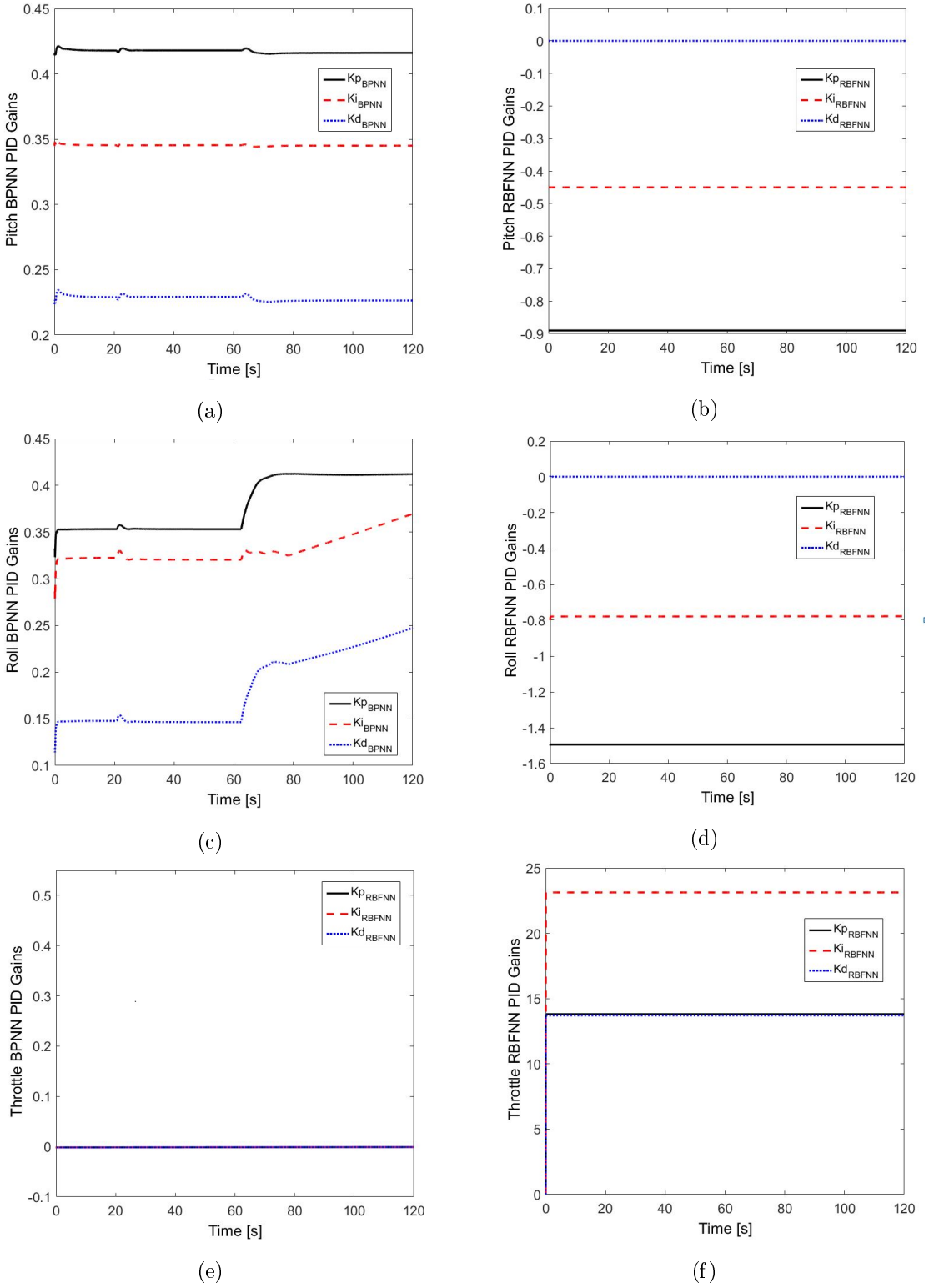
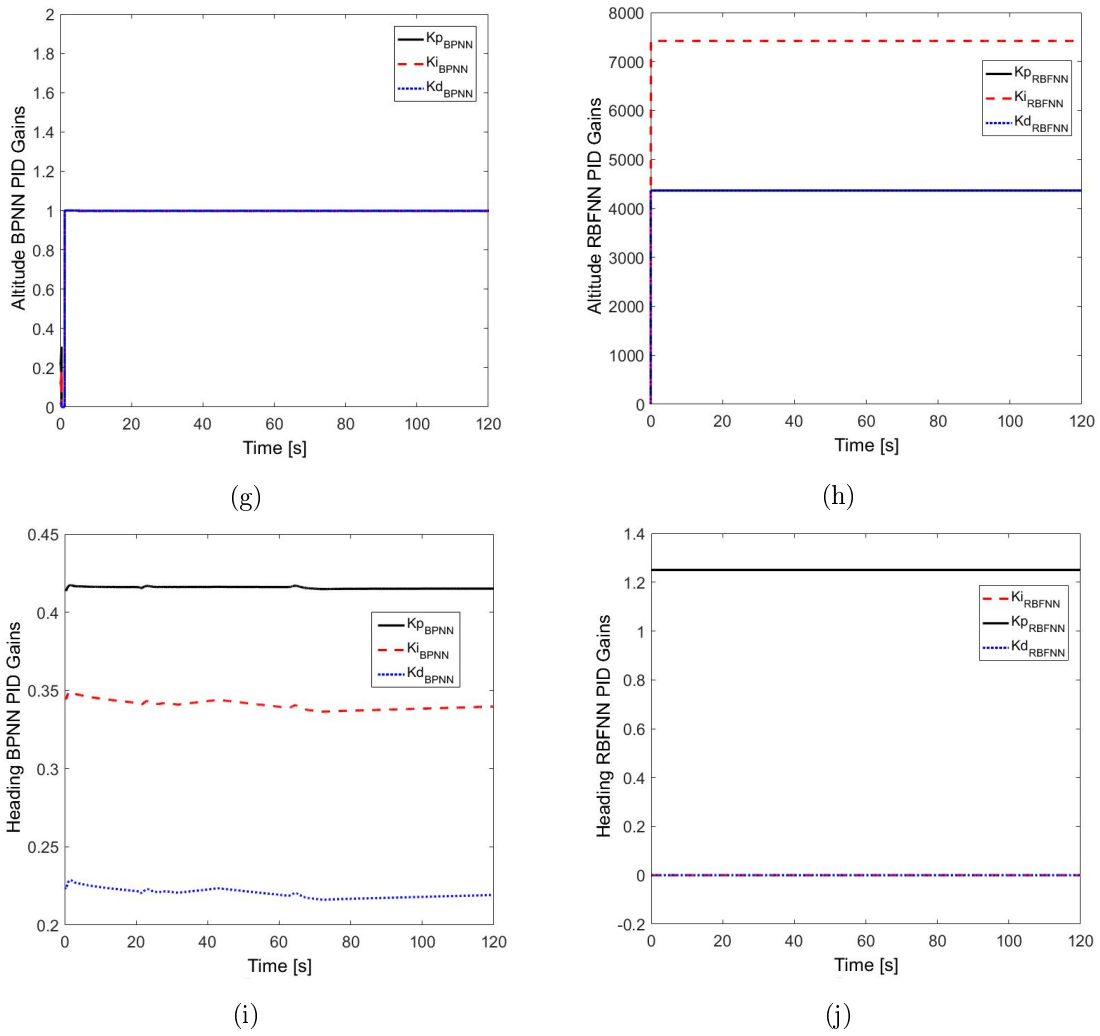


Figure 4.46 (a) to (f) show the time history of the control deflections during the simulation. Figure 4.46 shows a drop in aileron and flap deflection as the required control input is reallocated when failure occurs, this drops further when the setpoint is changed.



(g) (h)

(i) (j)

Figure 4.45: Simulated time history of the pitch (BPNN-based (a), and RBFNN-based (b)), and roll (BPNN-based (c), and RBFNN-based (d)), velocity (BPNN-based (e), and RBFNN-based (f)), altitude (BPNN-based (g), and RBFNN-based (h)), and heading (BPNN-based (i), and RBFNN-based (j)) PID gains with 50% hardover elevator failure

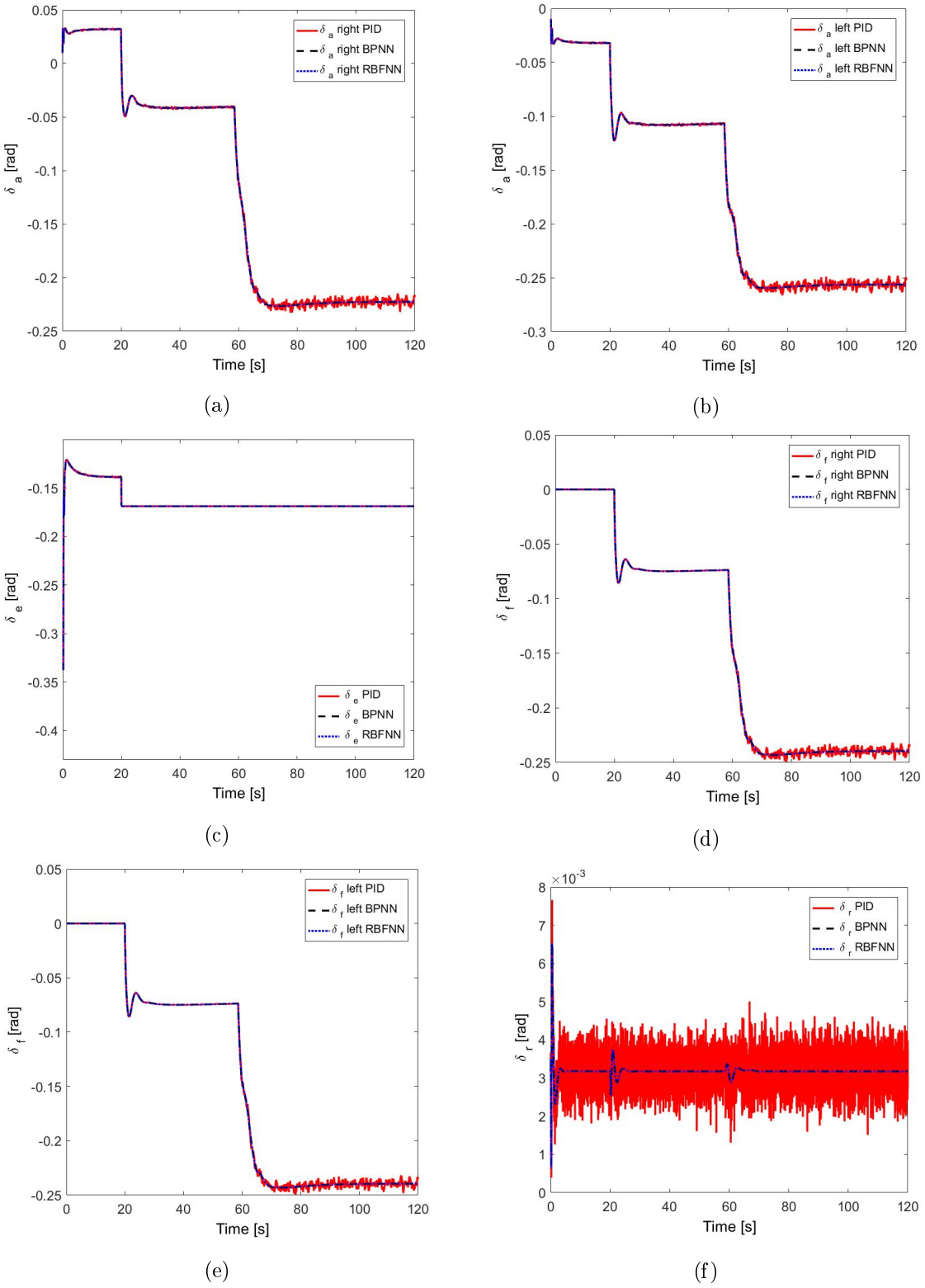


Figure 4.46: Simulated time history of right aileron (a), left aileron (b), elevator (c), right flap (d), left flap (e), and rudder (f) deflections (50% hardover) elevator failure

4.4.2.4 Elevator Failure - 100% Hardover

The simulation was run with a more severe failure of 100% hardover elevator failure. The response of all three cases is the same. Figure 4.47 shows the time history of the pitch, roll and heading angles. After some initial oscillations the angles become stable though they do not return to the desired setpoint. The pitch and heading angles settle at a slightly higher value while the bank angle at a slightly lower value. When the setpoint is required to change the responses remains the same and the change required visible.

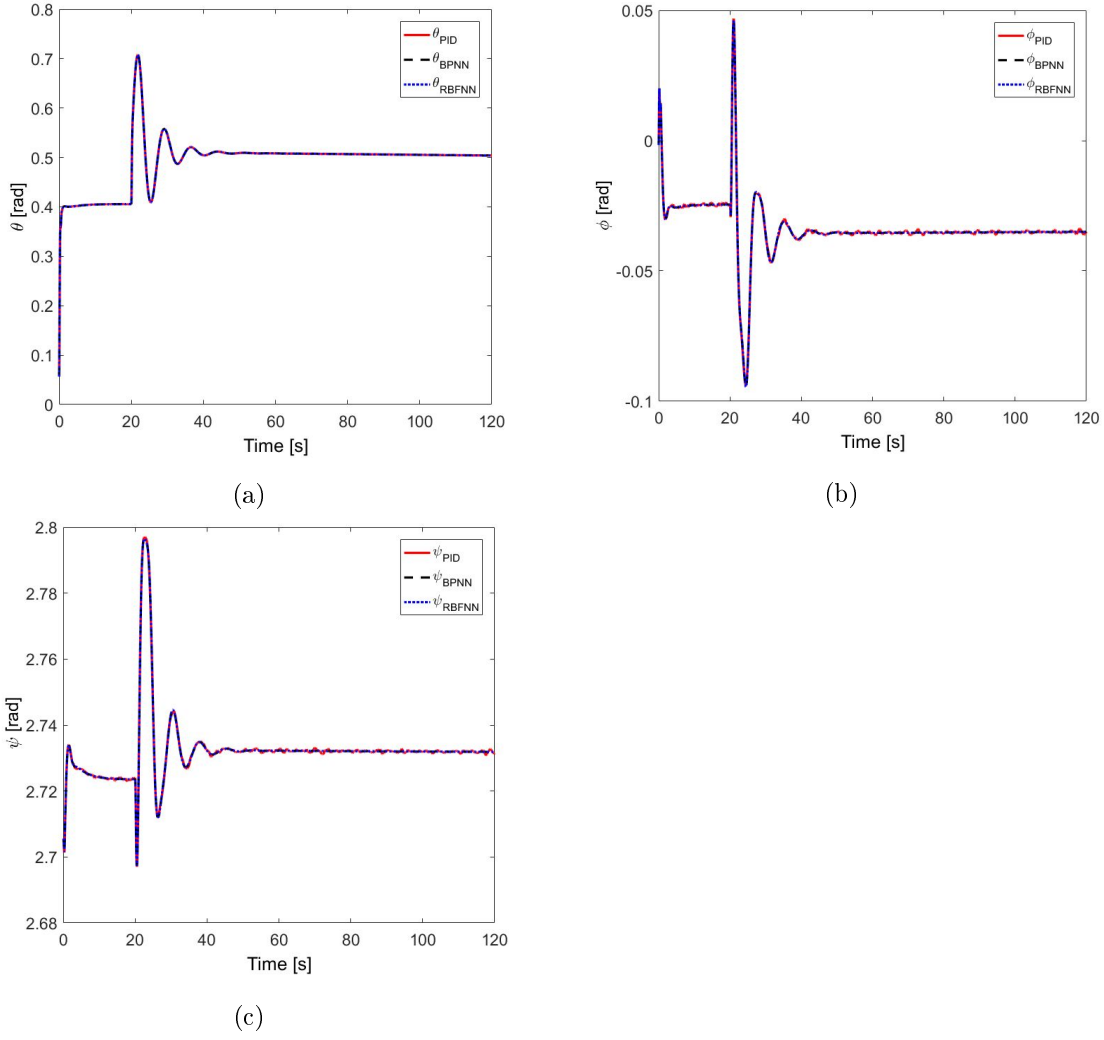
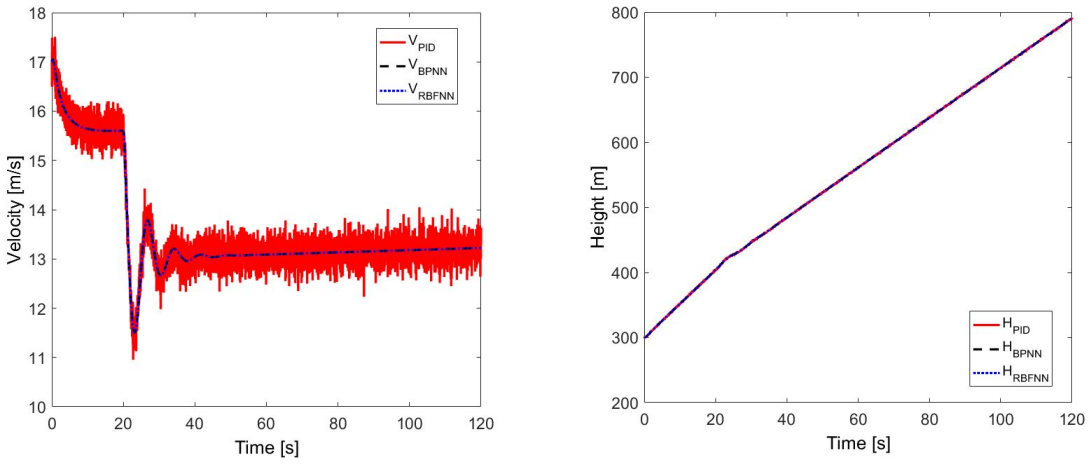


Figure 4.47: Simulated time history of middle loop pitch (a), bank (b), and heading (c) angles (100% hardover elevator failure)

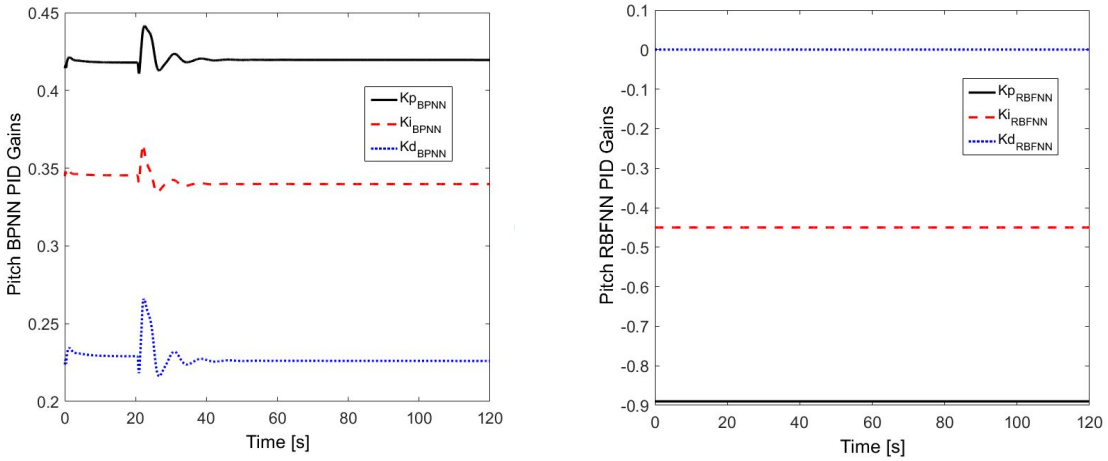
Figure 4.48 shows the time history of the airspeed and altitude. The speed decreases and one can see from the altitude plot that when the setpoint changes the UAV is unable to maintain constant altitude.



(a) (b)

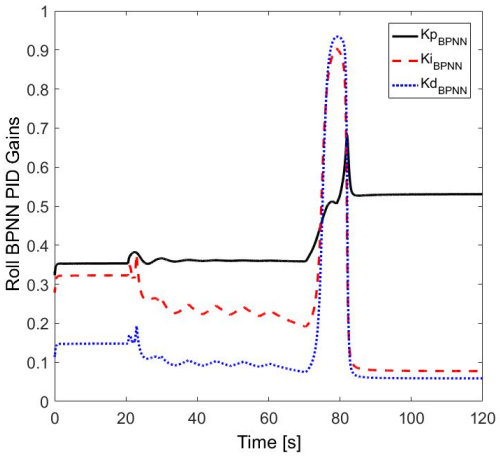
Figure 4.48: Simulated time history of the airspeed (a), and altitude (b) (100% hardover elevator failure)

Figure 4.49 shows the variation in PID controllers' gains for all control loops. From the roll gains one can see that the controllers attempt to put in more control effort by increasing the gains but from (Figure 4.50) one can see that both the flap and aileron deflections have already reached their maximum so no further control can be applied.

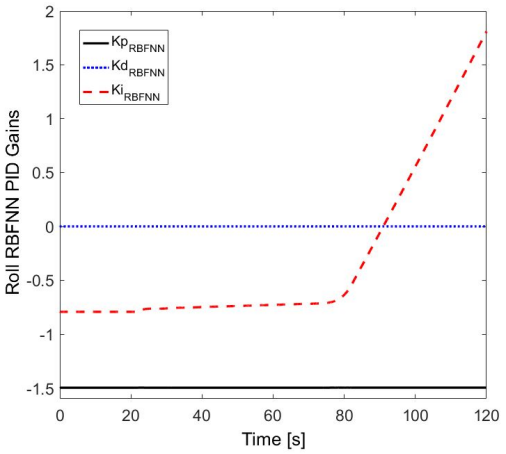


(a)

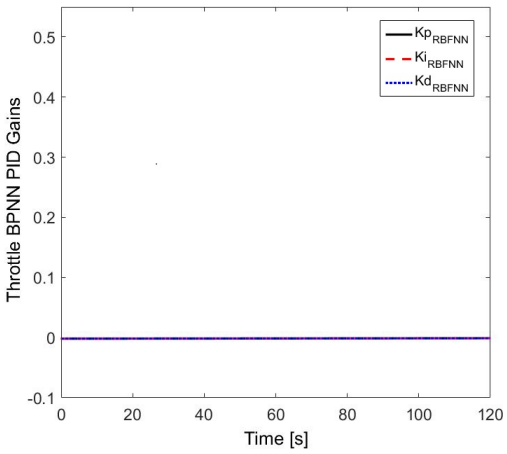
(b)



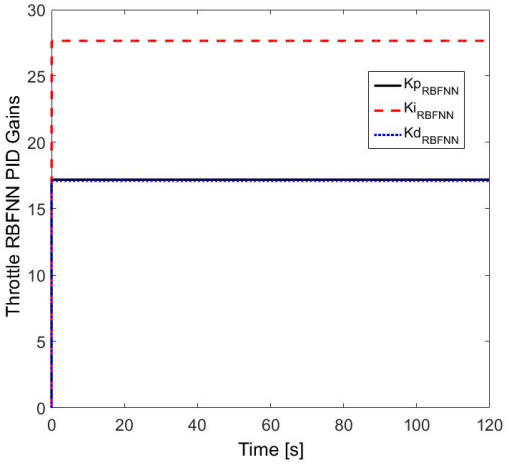
(c)



(d)



(e)



(f)

Figure 4.50 (a) to (f) shows the time history of the control deflections during the simulation.

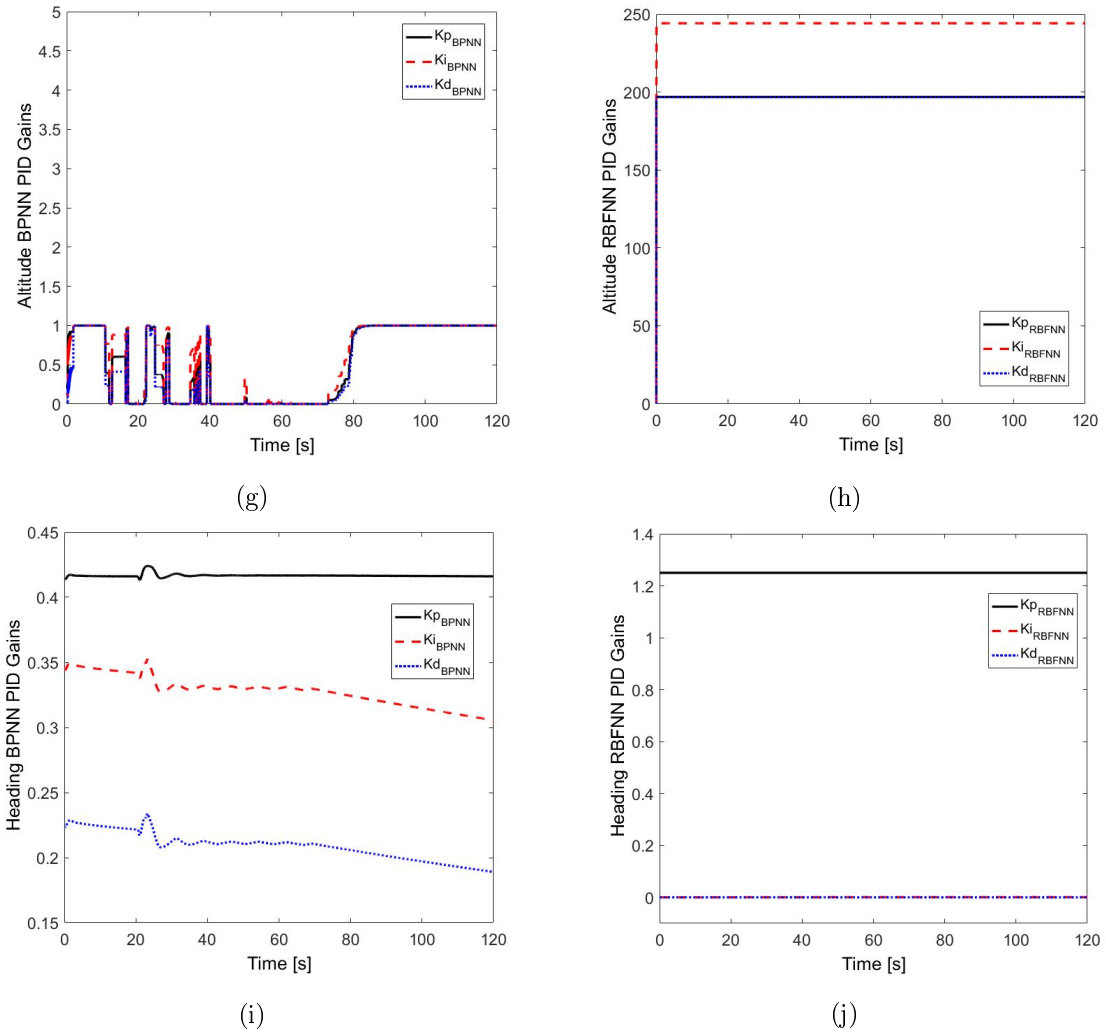


Figure 4.49: Simulated time history of the pitch (BPNN-based (a), and RBFNN-based (b)), and roll (BPNN-based (c), and RBFNN-based (d)), velocity (BPNN-based (e), and RBFNN-based (f)), altitude (BPNN-based (g), and RBFNN-based (h)), and heading (BPNN-based (i), and RBFNN-based (j)) PID gains with 100% hardover elevator failure

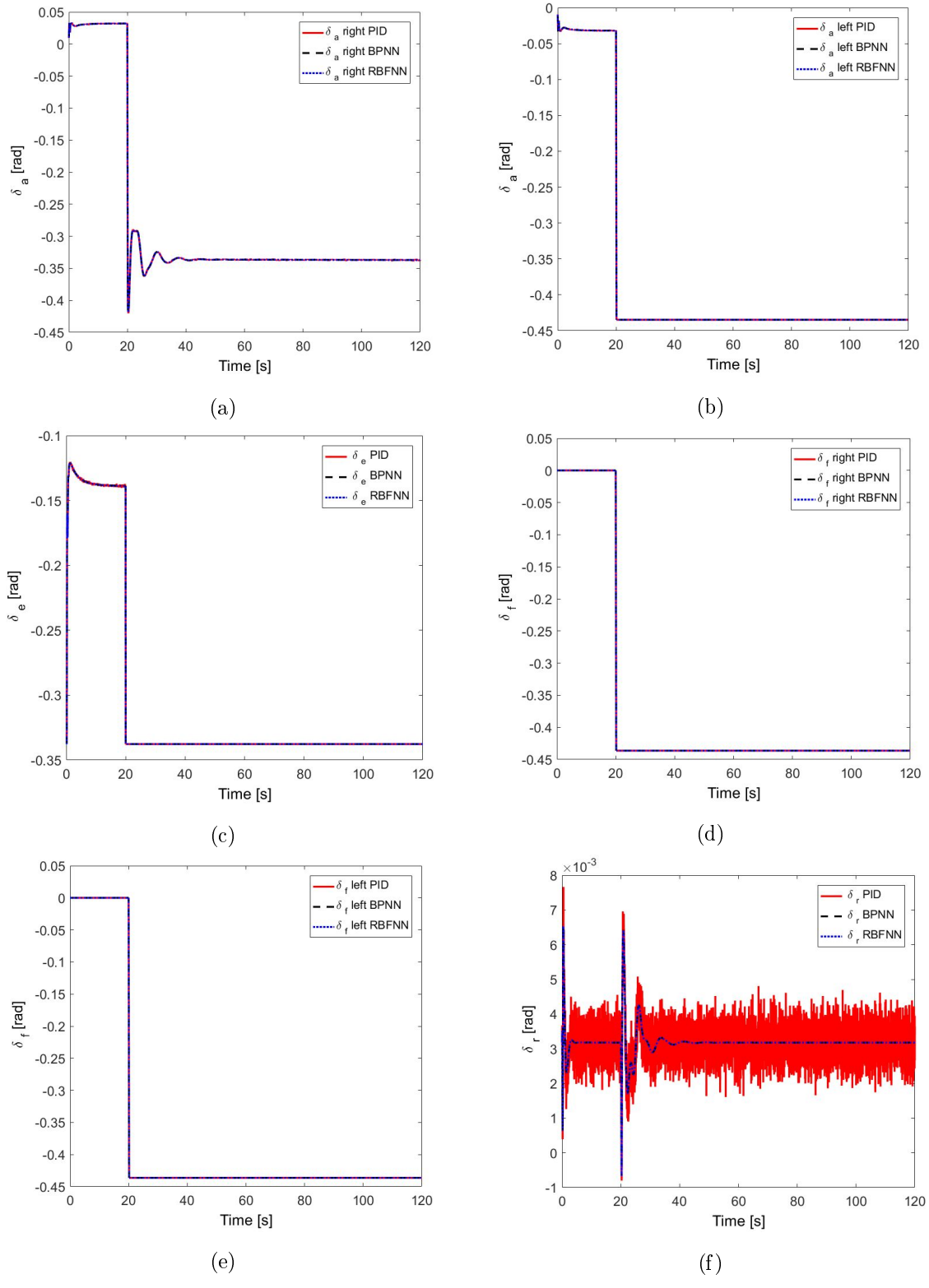


Figure 4.50: Simulated time history of right aileron (a), left aileron (b), elevator (c), right flap (d), left flap (e), and rudder (f) deflections (100% hardover) elevator failure

4.4.2.5 Right Aileron Failure - *0 rad* Deflection

The simulation was run with a 0 *rad* aileron deflection failure at 20 seconds after the start of the simulation. The angular response are shown in Figure 4.51. There is a slight change in the pitch angle at failure. The bank and heading angles experience a slightly sharp jump when failure occurs.

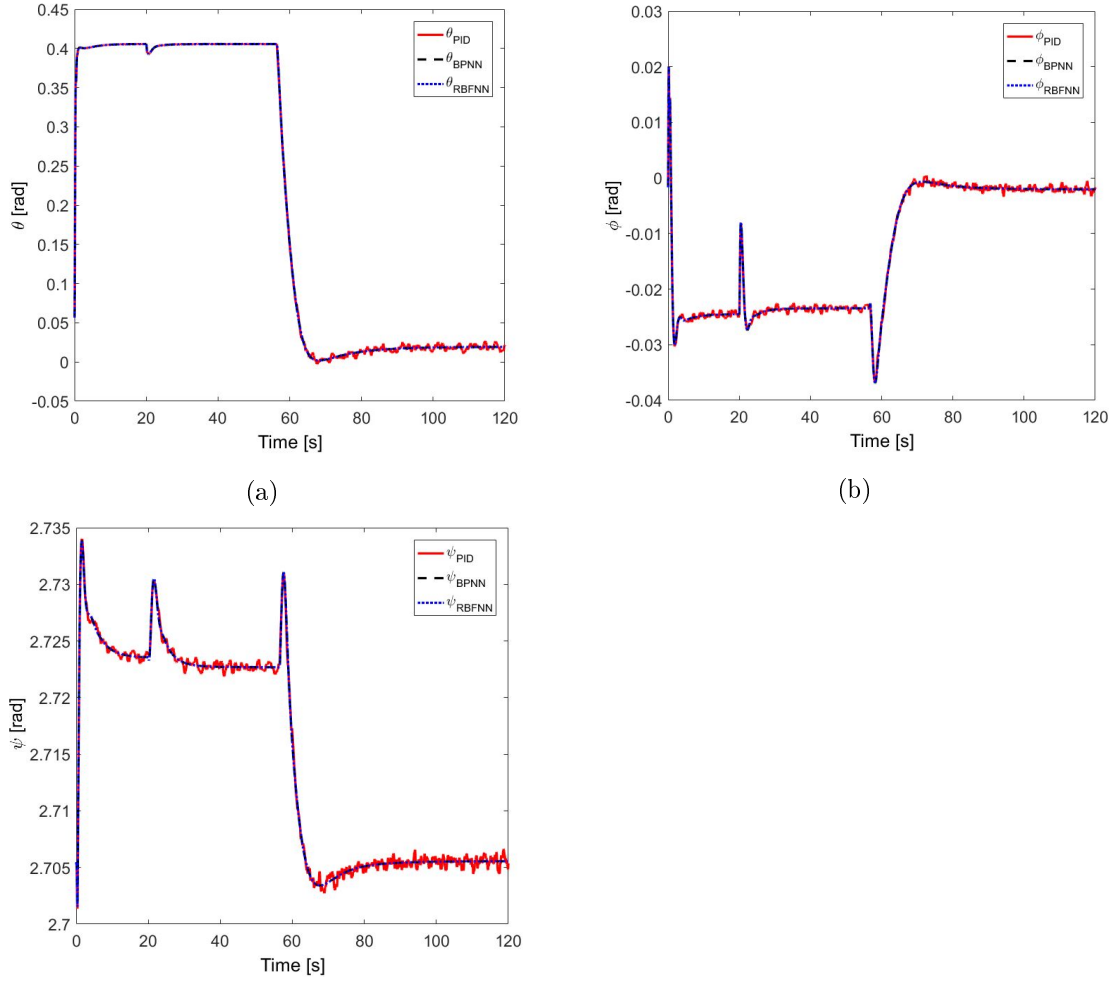
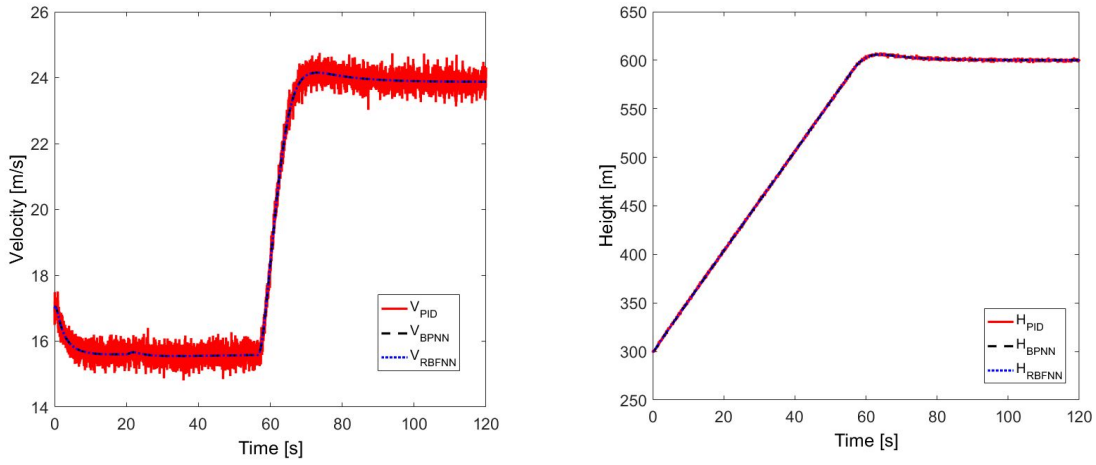


Figure 4.51: Simulated time history of middle loop pitch (a), bank (b) and heading (c) angles (0 *rad* right aileron deflection failure)

Figure 4.52 shows the time history of the airspeed and altitude. The is no noticeable difference in the airspeed and altitude responses when compared to the no-fault plots.



(a) (b)

Figure 4.52: Simulated time history of the airspeed (a), and altitude (b) (0 *rad* right aileron deflection failure)

Figure 4.53 shows the variation in PID controllers' gains for all control loops. There is a slight jump in the BPNN-based controllers' roll and heading gains when failure occurs while the RBFNN-based controllers' gains again remain constant. The roll gains show a response similar to that of the no-fault case. The velocity gains remain constant. The altitude gains remain constant though there is a spike in the BPNN-based controllers' gains at the start of the simulation.

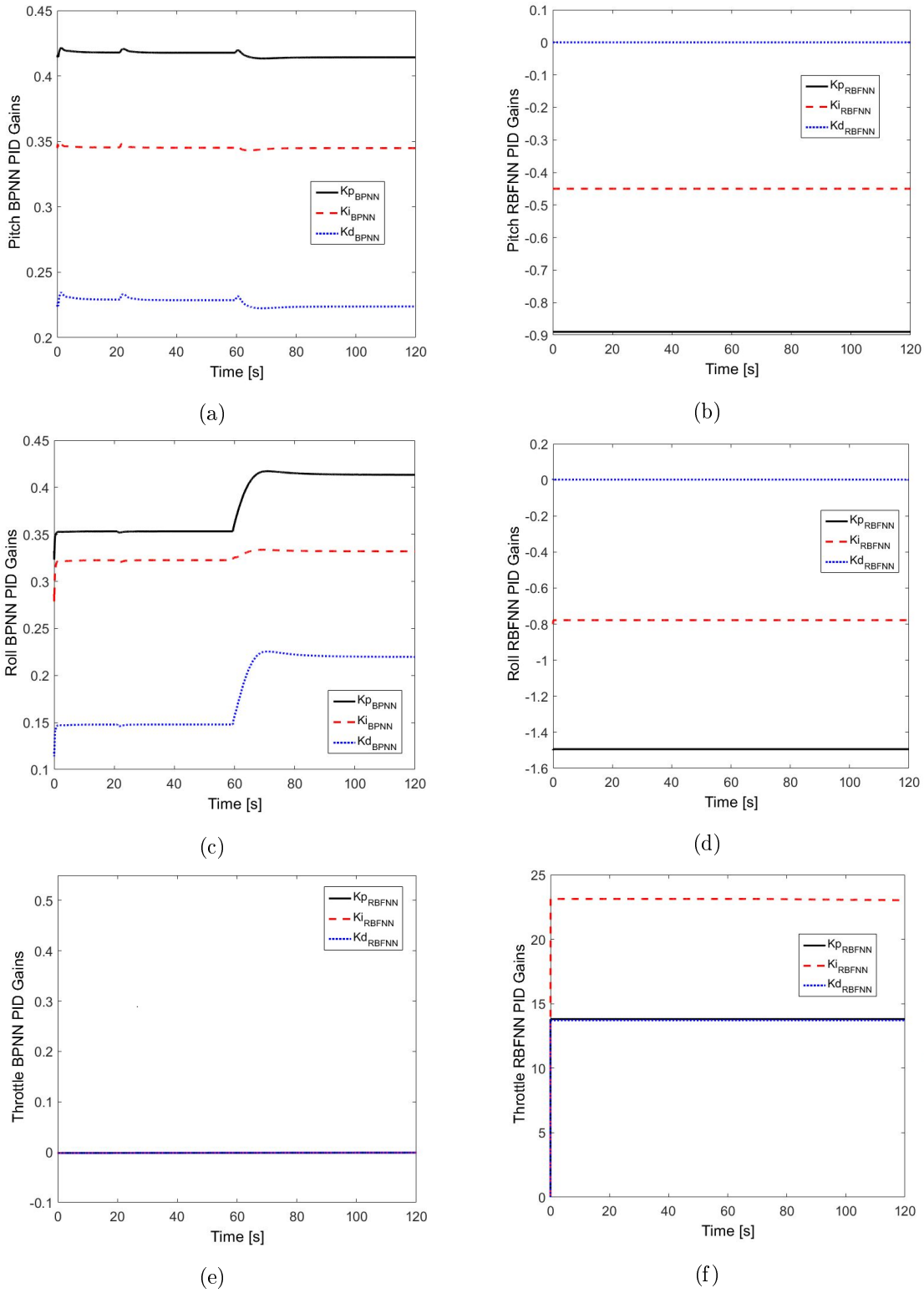


Figure 4.54 (a) to (f) shows the time history of the control deflections during the simulation. This shows the control reallocation at failure and effect of the change in setpoint.

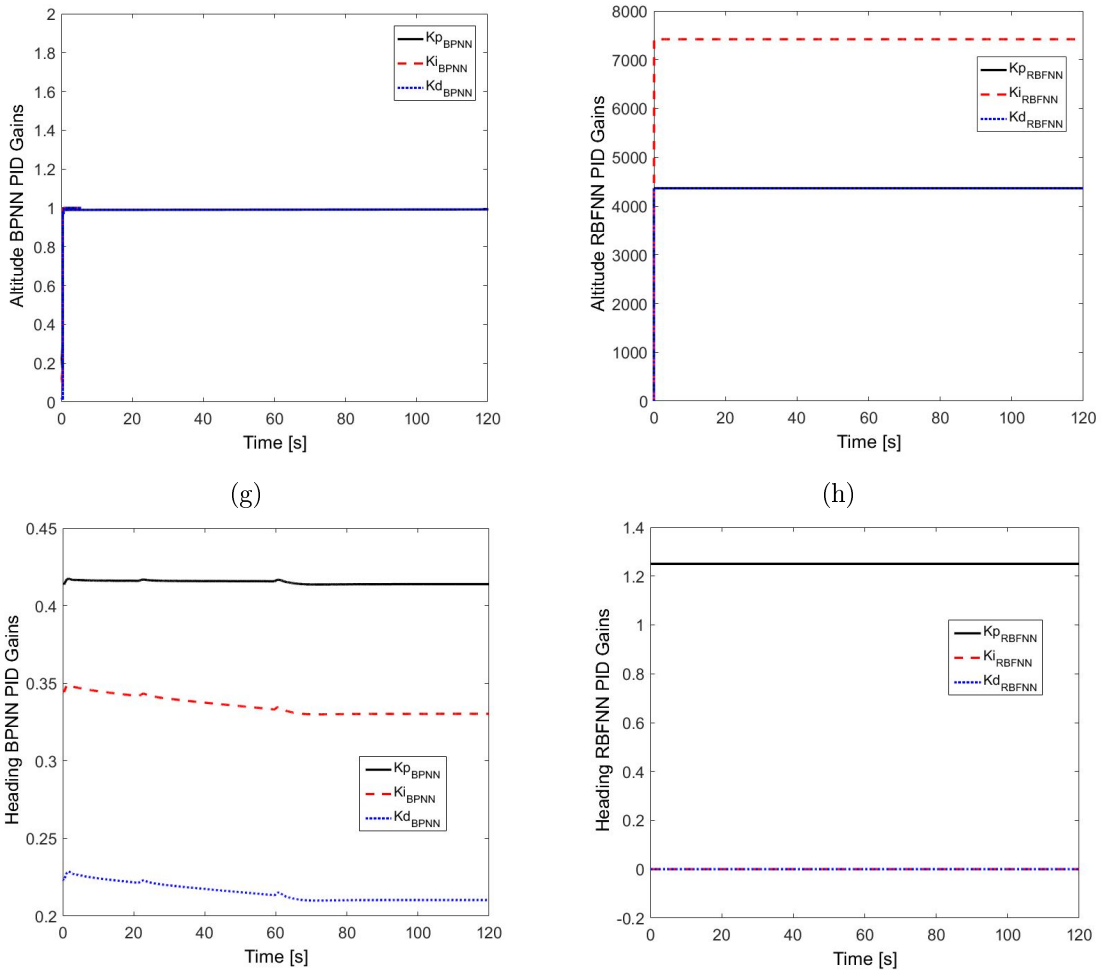


Figure 4.53: Simulated time history of the pitch (BPNN-based (a), and RBFNN-based (b)), and roll (BPNN-based (c), and RBFNN-based (d)), velocity (BPNN-based (e), and RBFNN-based (f)), altitude (BPNN-based (g), and RBFNN-based (h)), and heading (BPNN-based (i), and RBFNN-based (j)) PID gains with 0 *rad* right aileron deflection failure

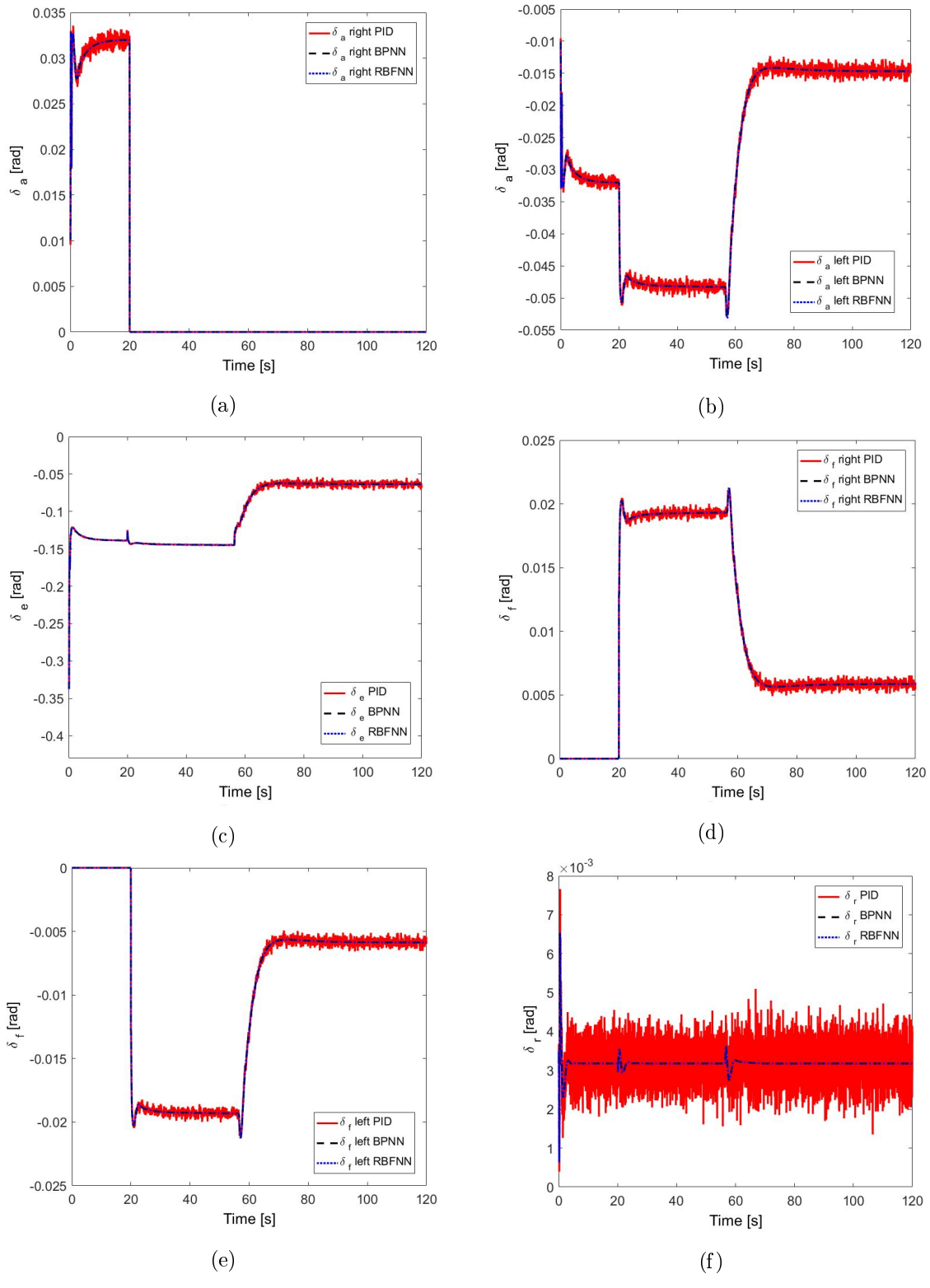


Figure 4.54: Simulated time history of right aileron (a), left aileron (b), elevator (c), right flap (d), left flap (e), and rudder (f) deflections (0 *rad* right aileron deflection failure)

4.4.2.6 Right Aileron Failure - 50% Hardover

A 50% hardover failure of the right aileron was modelled, the results of which can be found in Figure 4.55. It should be noted that the sense of the aileron failure has been reversed from that of the inner-loop right aileron-only failure case. There is a disturbance in the heading and pitch angle (Figure 4.55 (c) and (a)) while a few oscillations are observed in the bank angle (Figure 4.55 (b)).

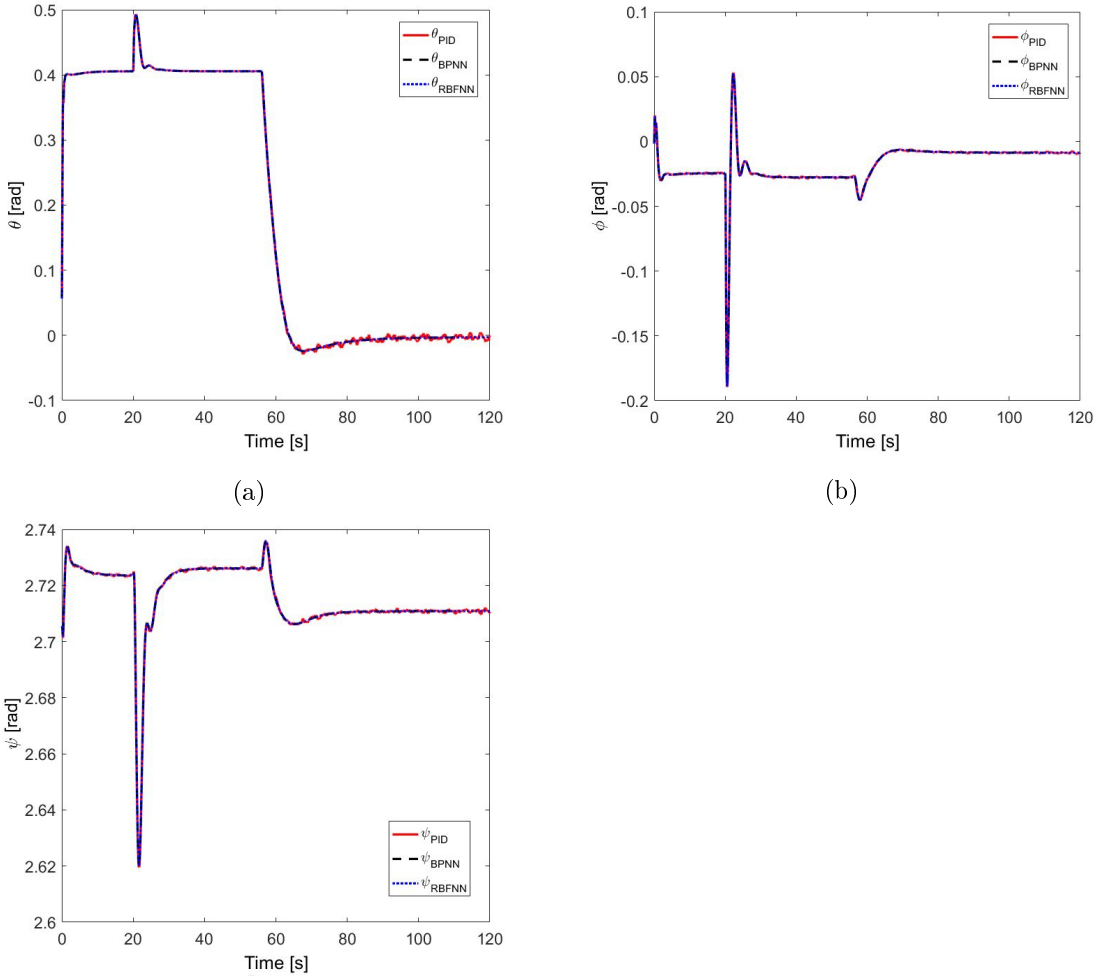
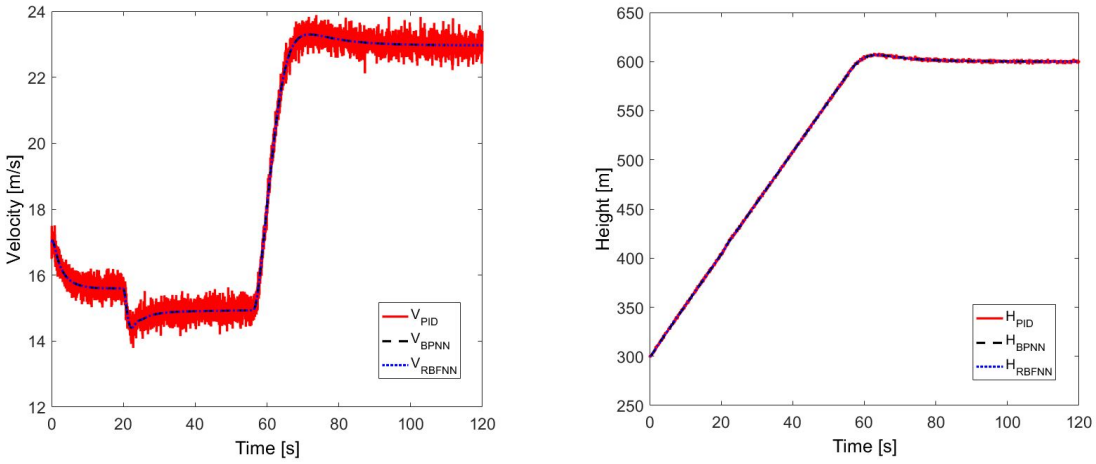


Figure 4.55: Simulated time history of middle loop pitch (a), bank (b), and heading (c) angles (50% hardover right aileron failure)

Figure 4.56 shows the time history of the airspeed and altitude. The airspeed and altitude plots are similar to that of the 50% elevator hardover case.



(a) (b)

Figure 4.56: Simulated time history of the airspeed (a), and altitude (b) (50% hardover right aileron failure)

Figure 4.57 shows the variation in PID controllers' gains for all control loops. There is a drop followed by a rapid increase in the BPNN-based controller pitch angle gains when failure occurs, while the roll gains experience a jump. The altitude gains display erratic behaviour and settle down after the setpoint is changed. All the RBFNN-based controllers' gains remain constant throughout the simulation.

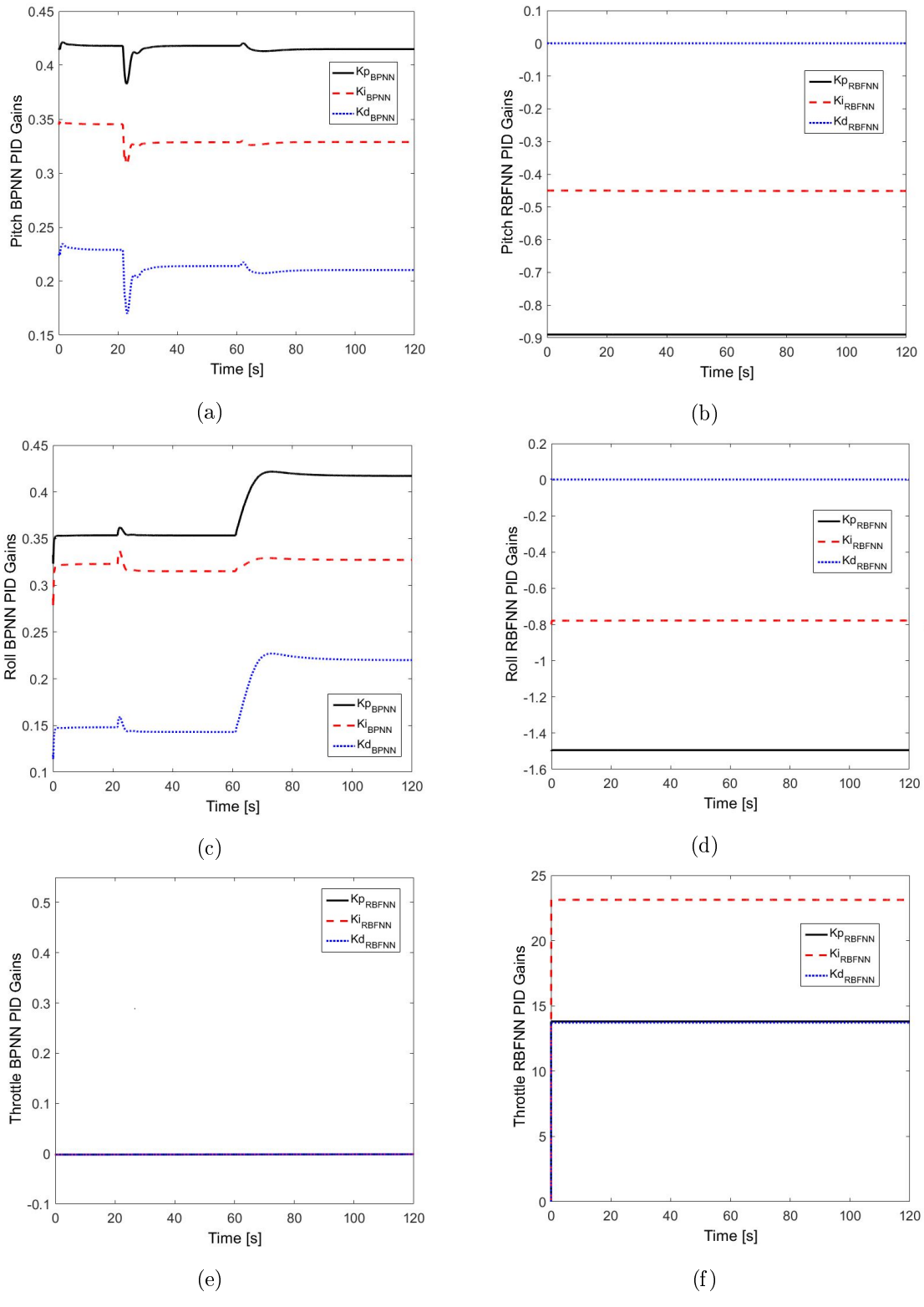


Figure 4.58 (a) to (f) shows the time history of the control deflections during the simulation.

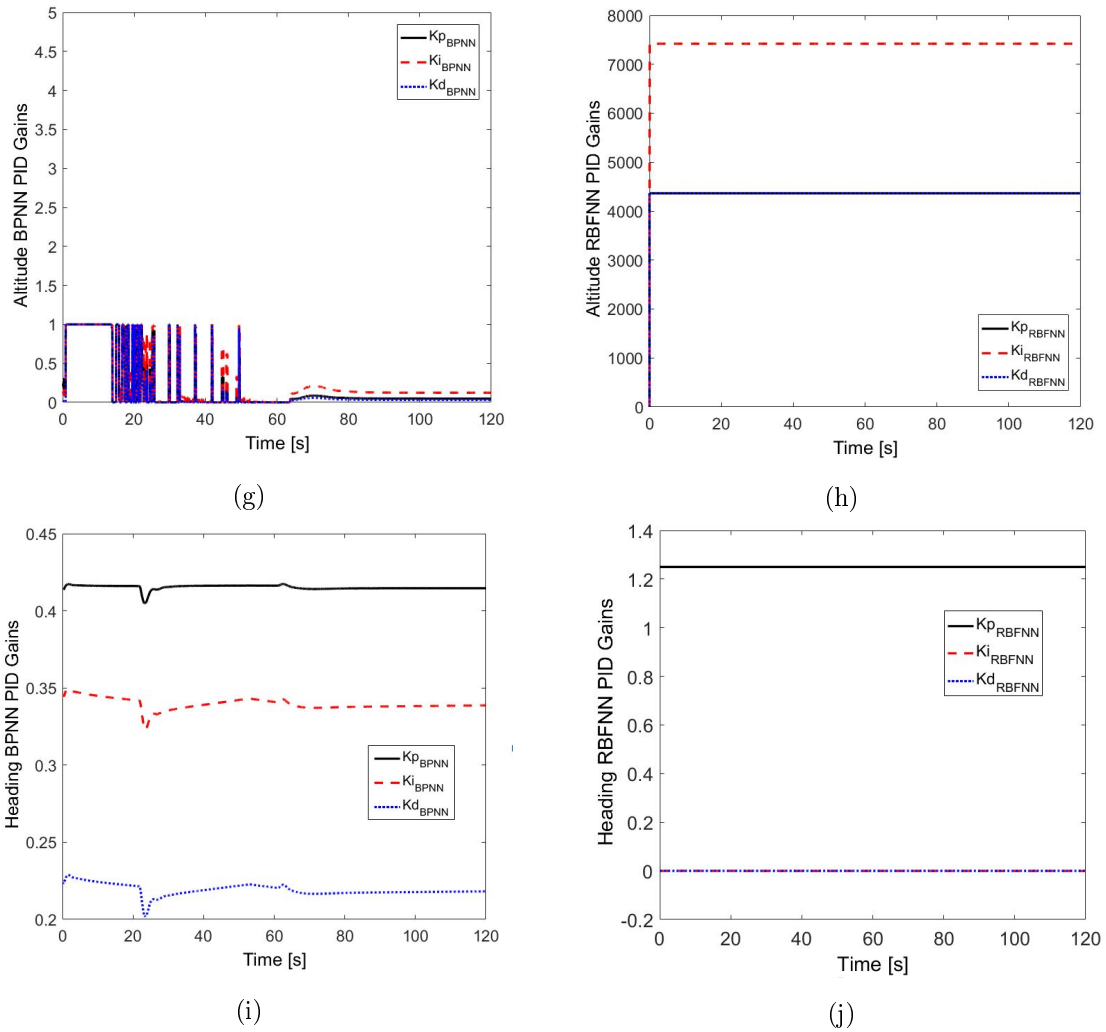


Figure 4.57: Simulated time history of the pitch (BPNN-based (a), and RBFNN-based (b)), and roll (BPNN-based (c), and RBFNN-based (d)), velocity (BPNN-based (e), and RBFNN-based (f)), altitude (BPNN-based (g), and RBFNN-based (h)), and heading (BPNN-based (i), and RBFNN-based (j)) PID gains with 50% hardover right aileron failure

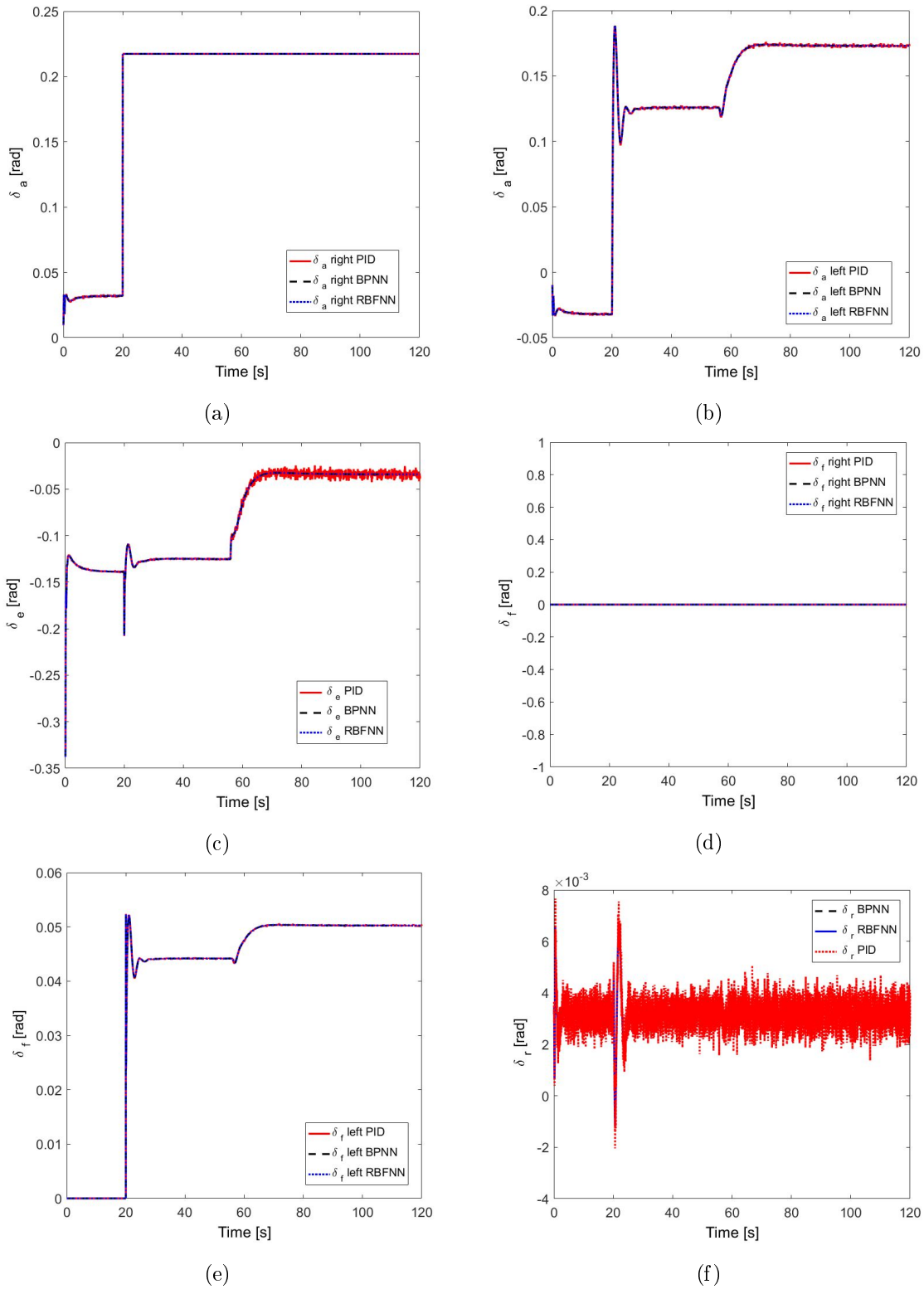


Figure 4.58: Simulated time history of right aileron (a), left aileron (b), elevator (c), right flap (d), left flap (e), and rudder (f) deflections (50% hardover right aileron failure)

4.4.2.7 Right Aileron Failure - 100% Hardover

A 100% hardover failure of the right aileron was modelled, the results of which can be found in Figure 4.59. The pitch angle shows an oscillation at failure with the NN-based magnitude being far less. For the bank angles a few oscillations are also observed with the PID-based amplitudes being slightly less. There is a dip in the heading angles with the PID-based magnitude again being slightly less.

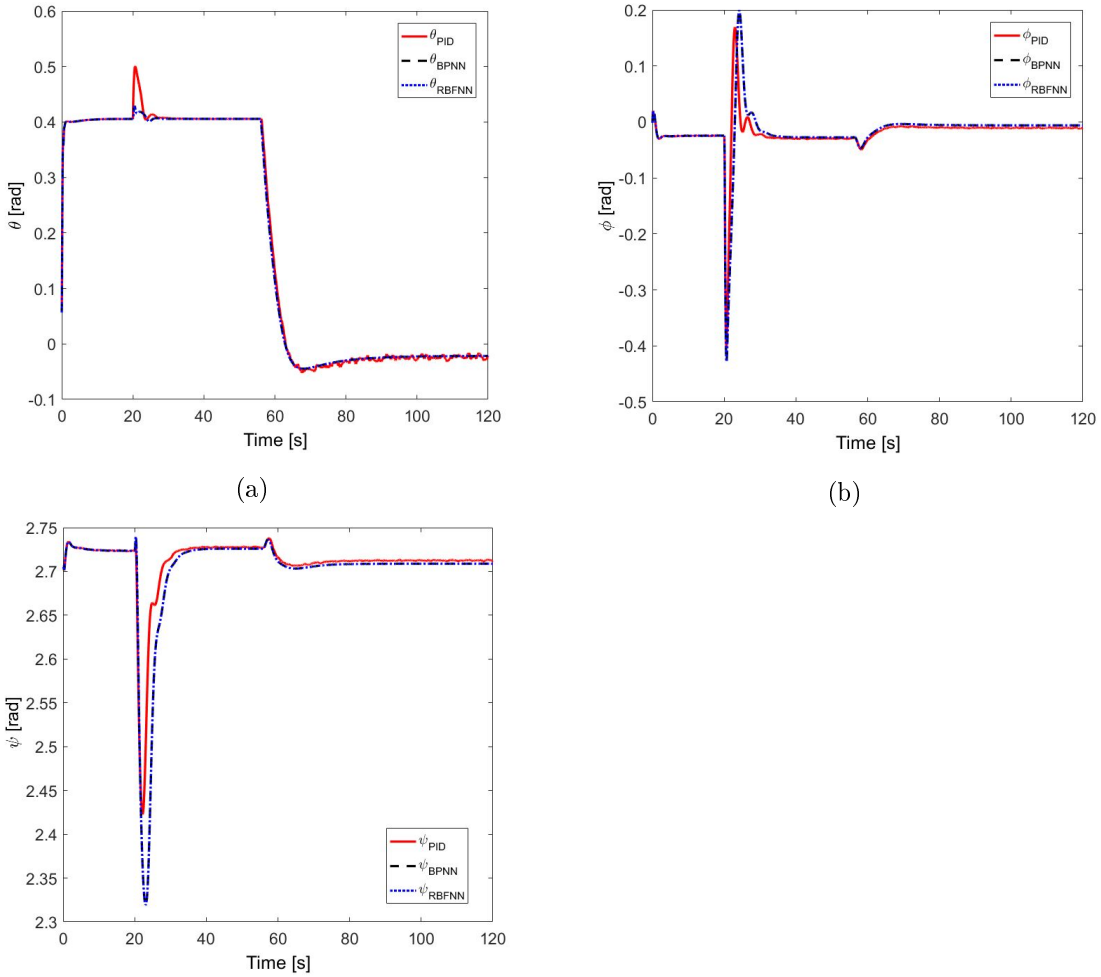
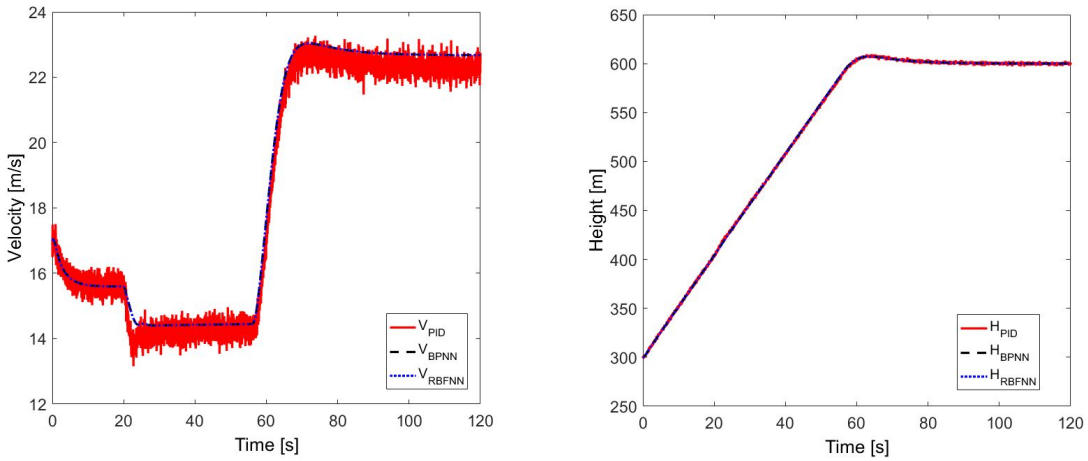


Figure 4.59: Simulated time history of middle loop pitch (a), bank (b), and heading (c) angles (100% hardover right aileron failure)

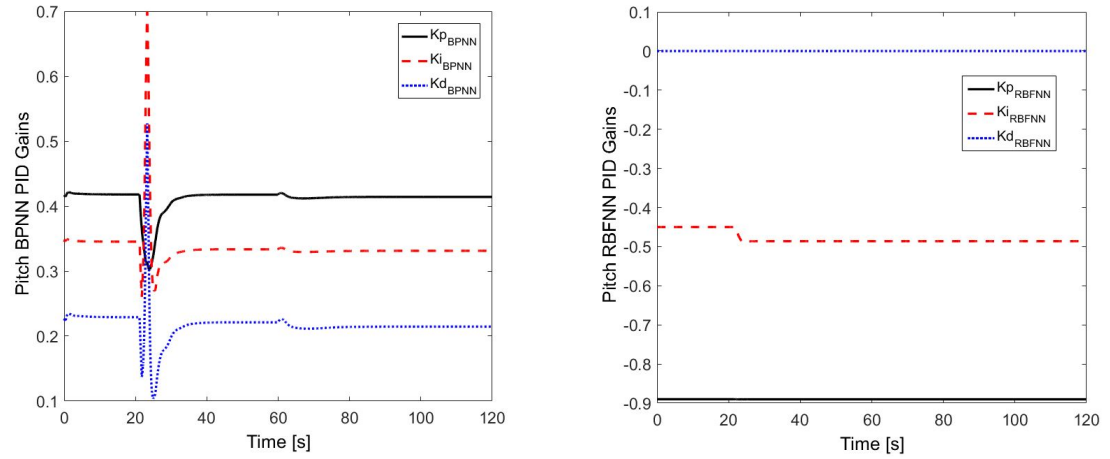
Figure 4.60 shows the time history of the airspeed and altitude. There is a drop in the airspeed at failure while the altitude performance remains the same.



(a) (b)

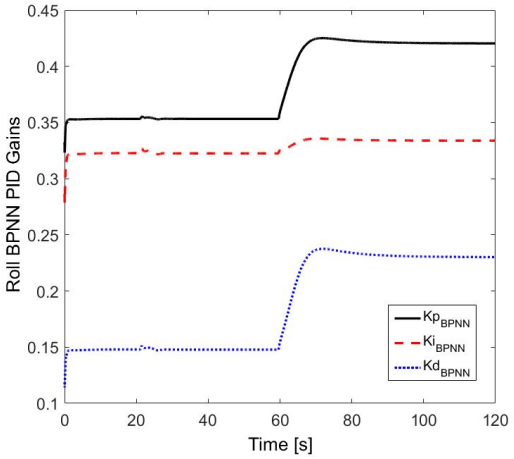
Figure 4.60: Simulated time history of the airspeed (a), and altitude (b) (100% hardover right aileron failure)

Figure 4.61 shows the variation in PID controllers' gains for all control loops. The BPNN-based controller pitch gains spike when failure occurs while the RBFNN-based controller integral gains decreases slightly. The roll, velocity and altitude gains are similar to that of 50% hardover aileron failure simulation. The BPNN-based controller heading gains show a sharp drop before they settle after failure.

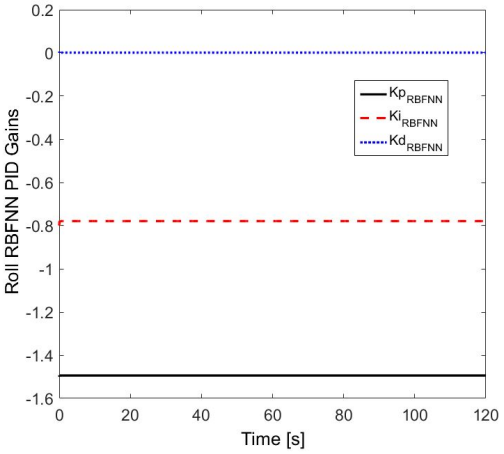


(a)

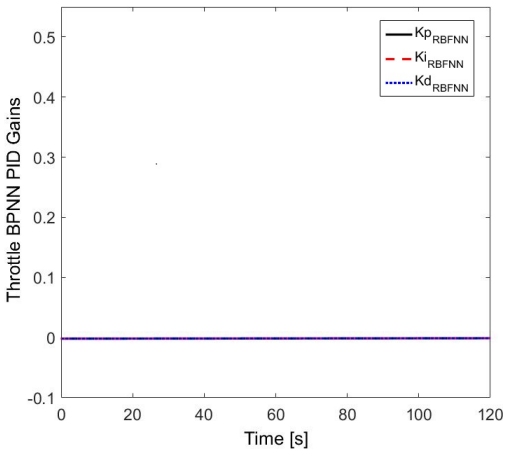
(b)



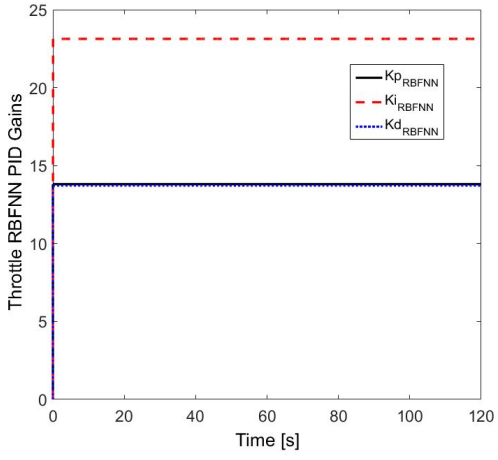
(c)



(d)



(e)



(f)

Figure 4.62 (a) to (f) shows the time history of the control deflections during the simulation. Figure 4.62 (b) shows a rapid increase in the left aileron deflection which then decreases once stability is achieved. It can be observed that the deflection of the control surfaces are well within their limits.

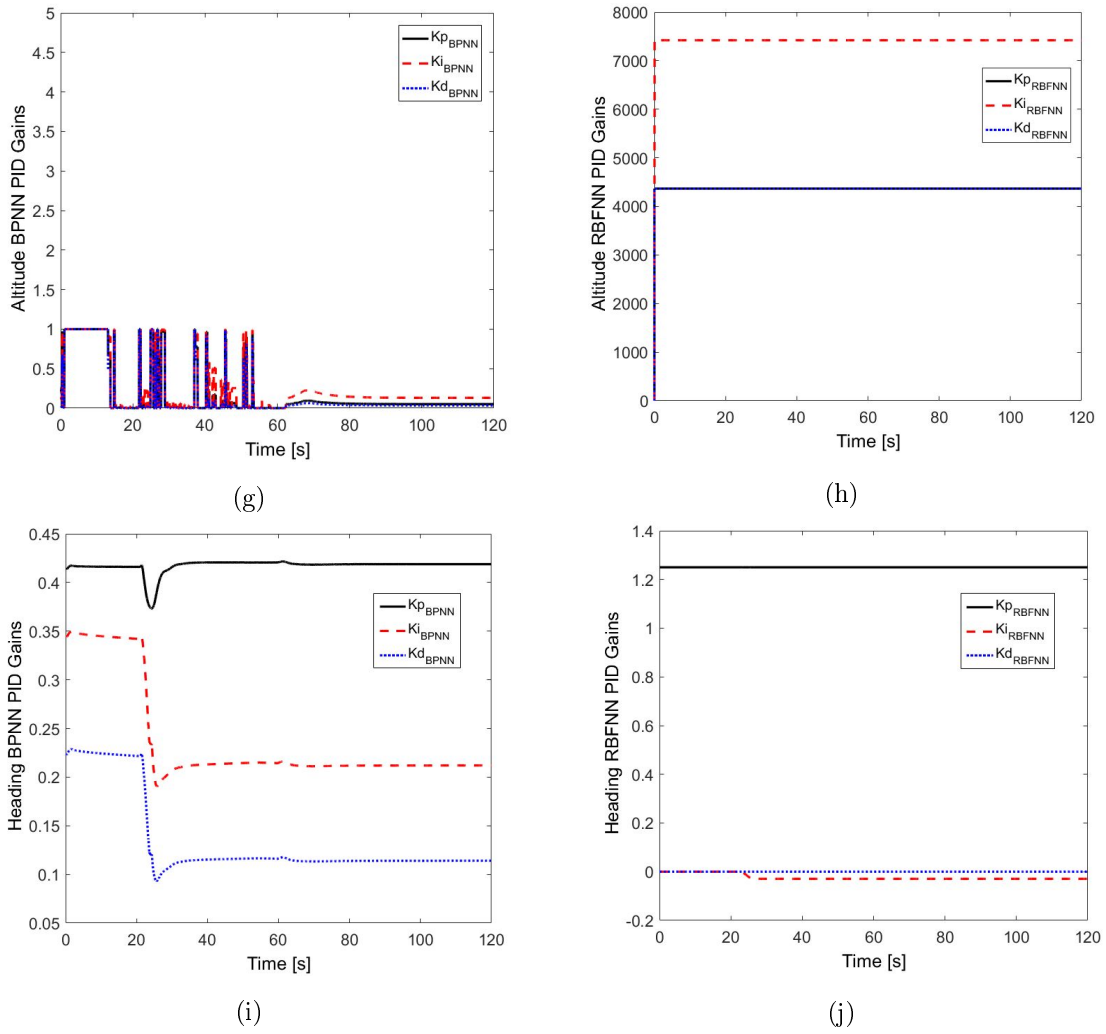


Figure 4.61: Simulated time history of the pitch (BPNN-based (a), and RBFNN-based (b)), and roll (BPNN-based (c), and RBFNN-based (d)), velocity (BPNN-based (e), and RBFNN-based (f)), altitude (BPNN-based (g), and RBFNN-based (h)), and heading (BPNN-based (i), and RBFNN-based (j)) PID gains with 100% hardover right aileron failure

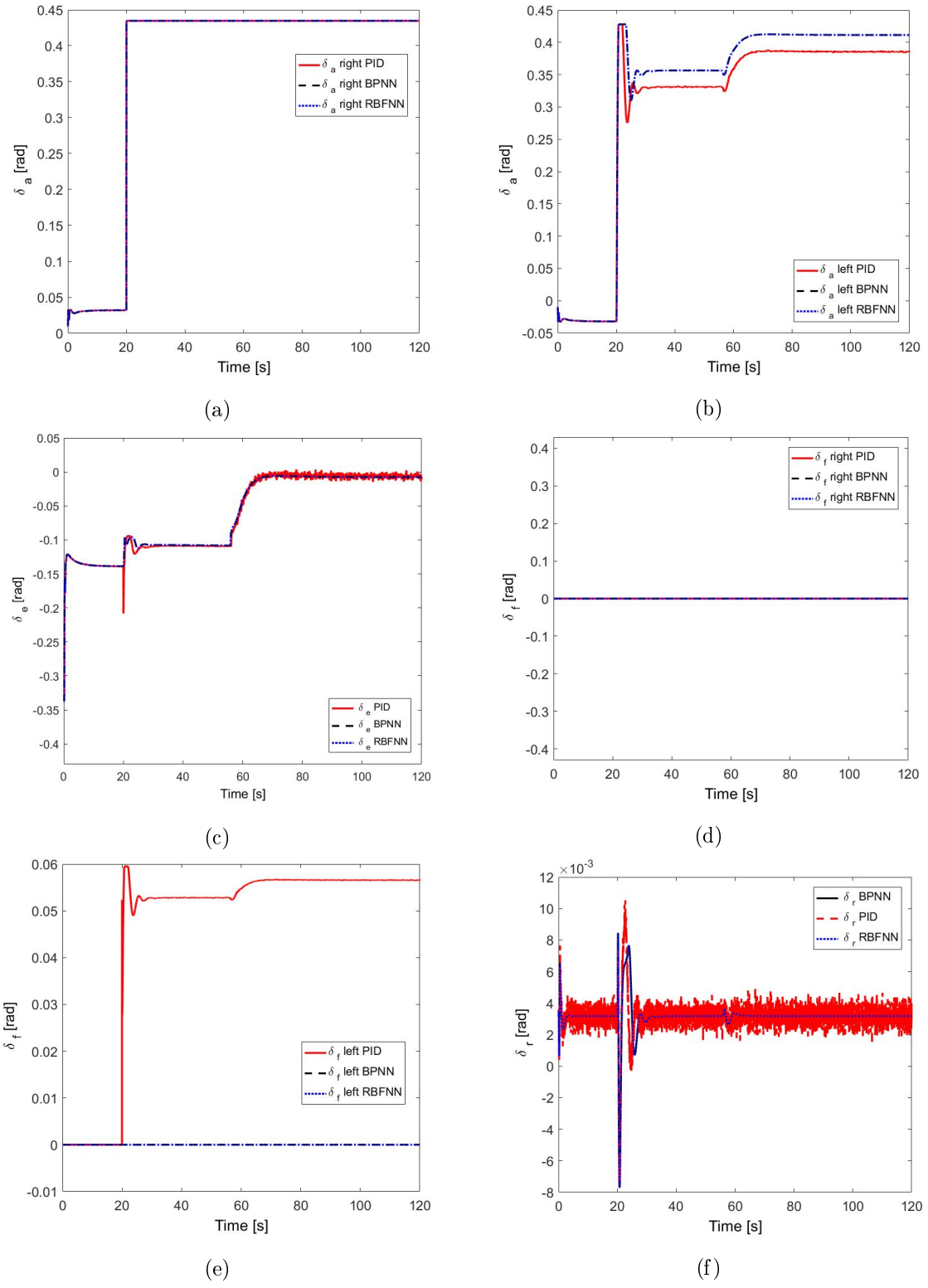


Figure 4.62: Simulated time history of right aileron (a), left aileron (b), elevator (c), right flap (d), left flap (e), and rudder (f) deflections (100% hardover right aileron failure)

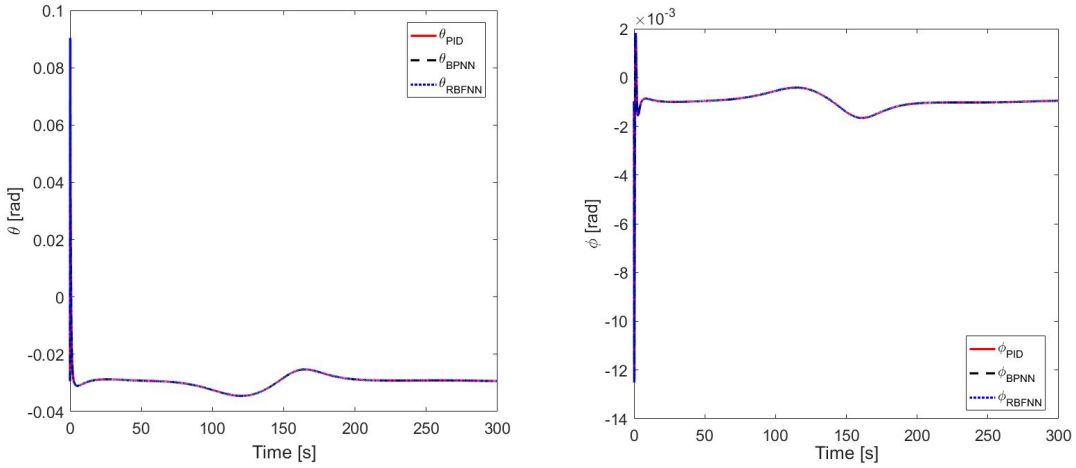
4.4.3 Inner Control Loop Testing - Microburst Windshear

The inner loop of the autopilot was also tested in the presence of a microburst windshear. The simulation was done during descent and the trim position determined.

To display the results of all aircraft state data would be considerable: hence for each test in this section the primary point of interest will be pitch and roll angles as these are the controlled parameters of the inner flight loop. The control deflection data, the PID gain coefficient data as well as the controller effort will also be supplied. The noise due to the sensors previously observed in the simulations without windshear was turned off during these simulations to ensure the effects of the windshear were clearly visible.

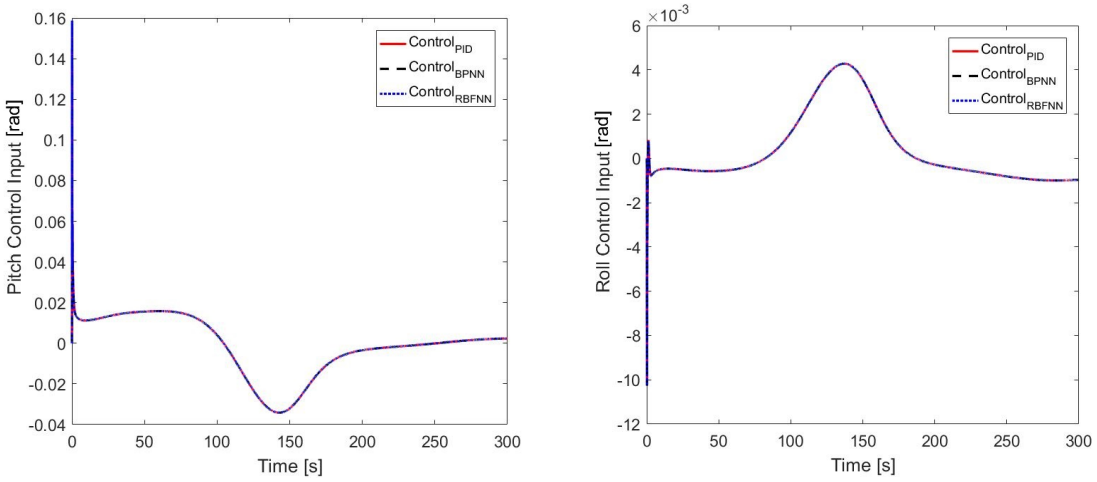
4.4.3.1 Control - No Failures in the presence of Microburst Windshear

The simulation was run without inducing any failures, both with the PID-based and neural network-based controllers in the presence of microburst windshear. Figure 4.63 shows the time history of the pitch and bank angle, the effects of the mircoburst windshear are clearly visible in the shape of the pitch angle plot and the spike at the start of the simulation. Though the desired bank angle is 0 *rad* this slight deviation is still negligible as it is unlikely to be noticed in flight, the effects on bank angle are similar to that of the pitch angle, however they are on a smaller scale. Figure 4.64 illustrates the controller effort required and shows that there is a some variation in control effort required for both cases that corresponds to the response plots.



(a) (b)

Figure 4.63: Simulated time history of inner loop pitch and bank angles in the presence of microburst windshear without failures



(a) (b)

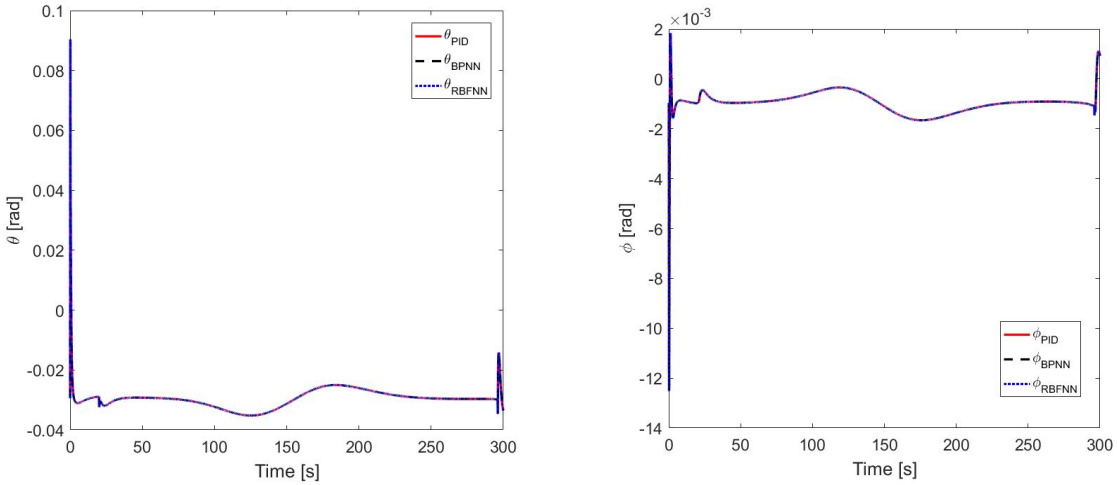
Figure 4.64: Simulated time history of the pitch (a), and roll (b) controller effort required in the presence of microburst windshear without failures

The performance of these controllers were deemed to be satisfactory for the purposes of this research.

4.4.3.2 Elevator Failure - *0 rad* Deflection in the presence of Microburst Windshear

The simulation was run with 0 *rad* elevator deflection failure. As in the previous set of simulations, both the left and right elevators move from their current position, at the time of failure to their neutral position. For this flight position and aircraft configuration, a slight down elevator trim is required. Hence, as the elevator returns to its neutral position, a nose up disturbance is created. Figure 4.65 clearly shows the point at which the elevator failed which is observed by the slight diversion of the roll and pitch angles.

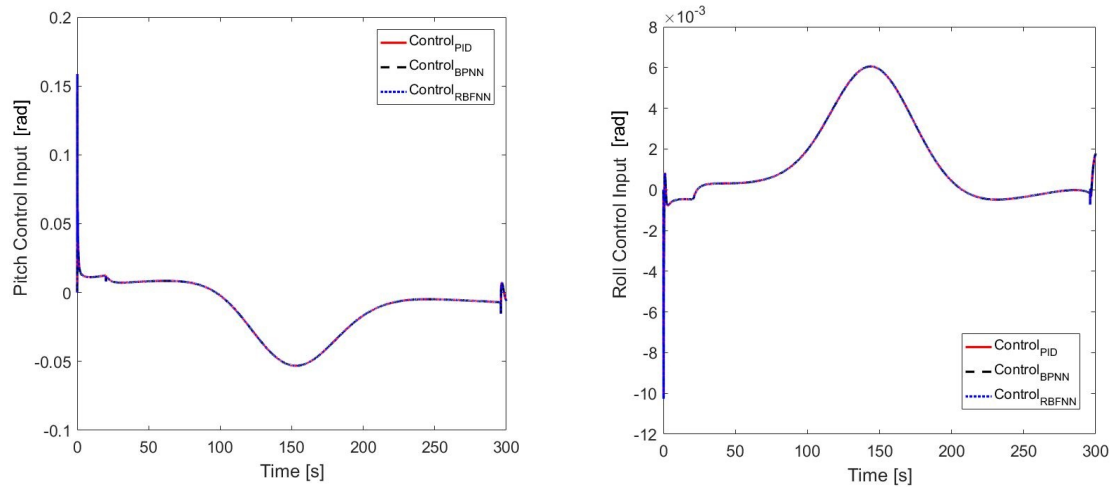
Figure 4.65 (a) shows that, the pitch change is again of the order of 0.01 *rad* (0.57 deg). The pitch change is in the form of a slight dip followed by a smooth return to the desired setpoint. Figure 4.65 (b) shows a small peak in the bank angle at failure. With the neural networks enabled there is not a large difference in control performance noted at this point. The control inputs shown in Figure 4.66 show an increase in the roll control input and decrease in the pitch control input at failure.



(a) (b)

Figure 4.65: Simulated time history of pitch and bank angles (0 *rad* elevator failure) in the presence of microburst windshear

Figure 4.66 (b) shows the roll disturbance created as the elevator is moved to the neutral position. The effect was small and no noticeable difference can be observed between the PID-based and NN-based cases.



(a) (b)

Figure 4.66: Simulated time history of pitch (a) and roll (b) control inputs (0 *rad* elevator failure) in the presence of microburst windshear

Figure 4.67 shows the time history of the PID controllers' gains during 0 *rad* elevator failure. Figure 4.67 (a) shows that for the BPNN-based pitch controller there is a slight change in the derivative gain while the proportional and integral gains remain constant. For the RBFNN-based case (Figure 4.67 (b)) the gains remain constant. In Figure 4.67 (c) and (d) show a

hardly noticeable change in the roll control gains for the BPNN-based case in beginning of the simulation while the RBFNN-based controllers' gains remain constant.

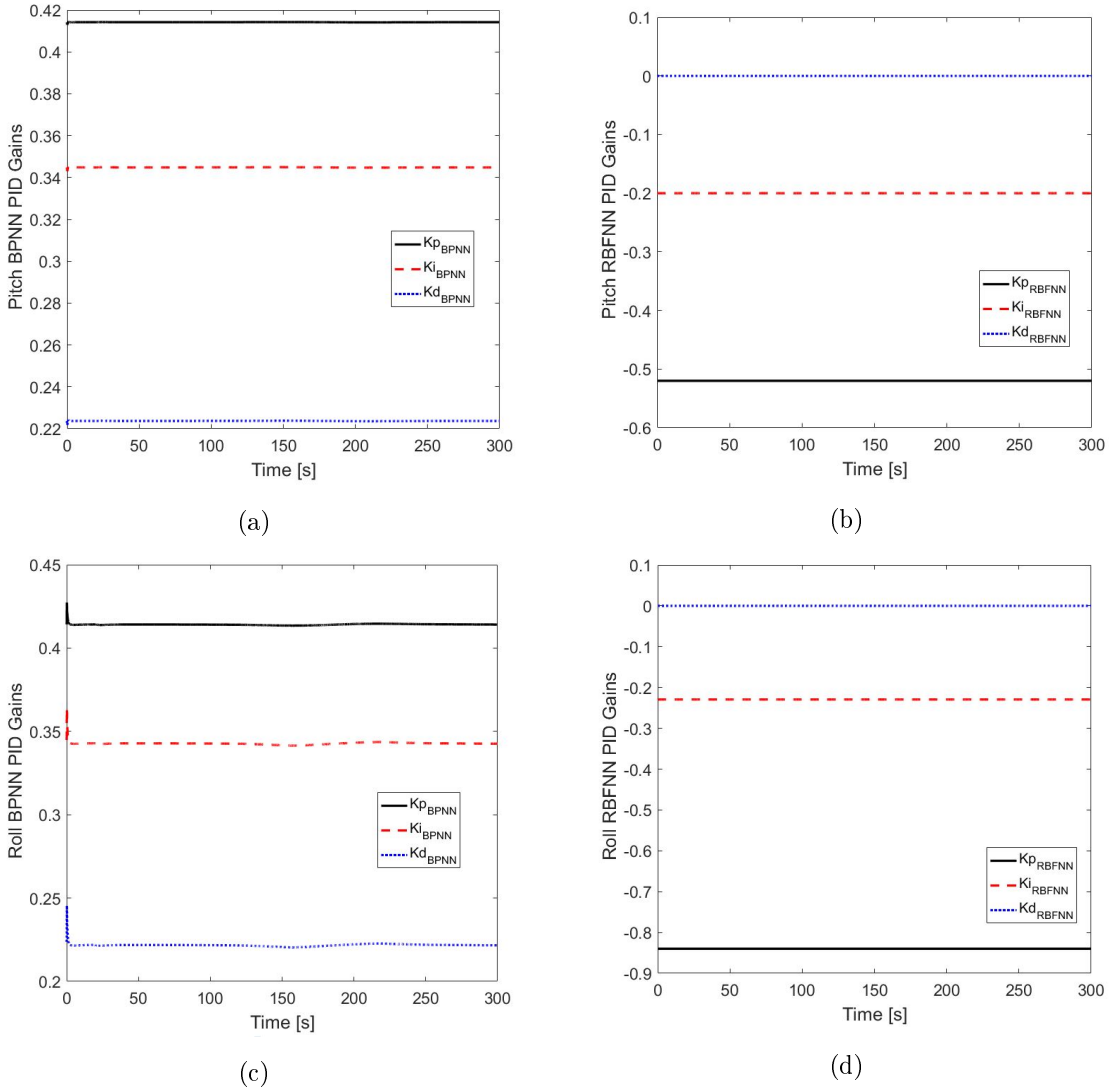


Figure 4.67: Simulated time history of pitch (BPNN-based (a), and RBFNN-based (b)), and roll (BPNN-based (c), and RBFNN-based (d)) gains with 0 *rad* elevator failure in the presence of microburst windshear

4.4.3.3 Elevator Failure - 50% Hardover in the presence Microburst Windshear

The simulation was run with a more severe failure of 50% hardover elevator failure in the presence of microburst windshear. In this simulation the CA algorithm was used to force the UAV to carry on descending (see Figure 4.68a). The altitude plot has been included here to illustrate that UAV descends at a faster rate to get through the microburst. However the BPNN-based controller which was simulated without CA enabled wants to increase altitude to fly out of the worst of the microburst region. Figure 4.68 shows the results of this simulation

in the form of a time history plot of pitch and roll angles. There is some oscillation (which dampens out) experienced by the BPNN-based controller without CA. It can be observed in Figure 4.68 that the elevator failure caused a large change in the nose-up direction. There is a very sharp rise in the pitch angle, followed by a drop below the setpoint before a gradual rise toward the setpoint for the controllers with CA enabled. There is very little difference between the RBFNN-based and PID-based cases. For the roll angle there is also a sharp jump for the controllers combined with CA.

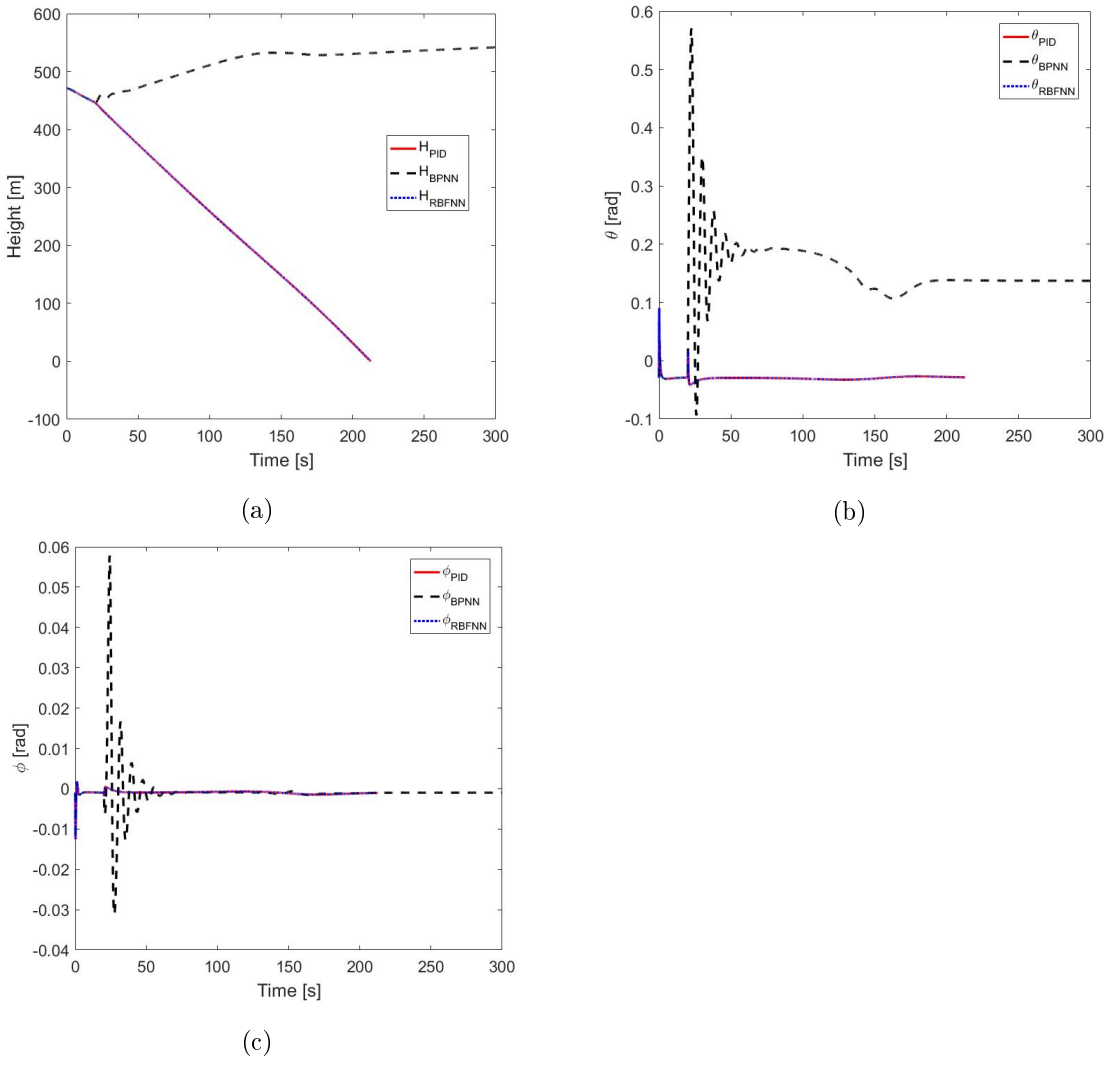
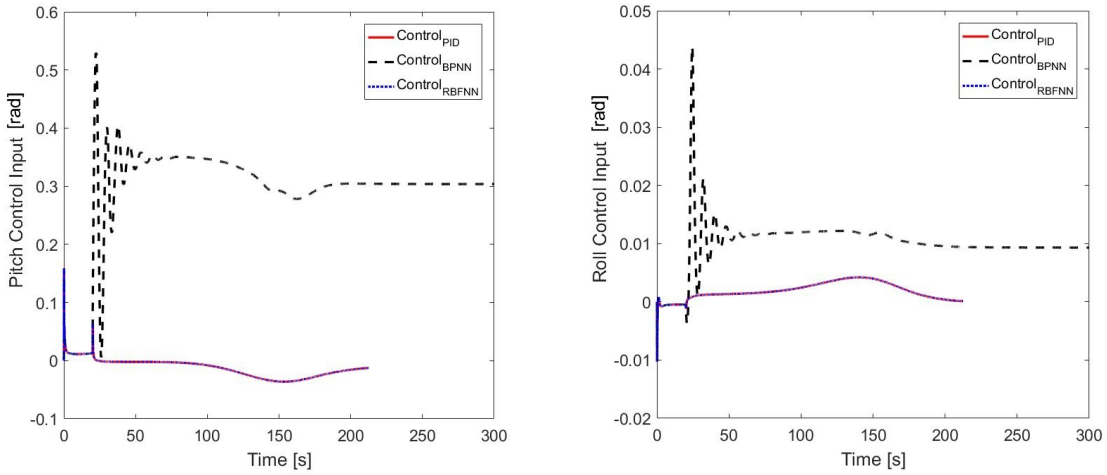


Figure 4.68: Simulated time history of altitude (a), and pitch (b) and bank (b) angles(50% hardover) elevator failure in the presence of microburst windshear

Figure 4.68 shows a spike in the pitch control at the time of failure. In the case of roll control there is a jump in the roll control effort at failure. There are oscillations corresponding to the BPNN-based controller response.



(a) (b)

Figure 4.69: Simulated time history of pitch (a) and roll (b) control inputs (50% hardover) elevator failure in the presence of microburst windshear

Figure 4.70 shows the time histories of the PID controllers' gains during 50% hardover elevator failure. For the pitch gains (Figure 4.70 (a) and (b)) in the BPNN-based controller case there is some small oscillation in the gains at the time of failure while the RBFNN-based controller gains remain constant. In contrast to this the roll gains (Figure 4.70 (c) and (d)) show oscillation in the proportional gain for the BPNN-based case and in the RBFNN-based case the gains remain constant.

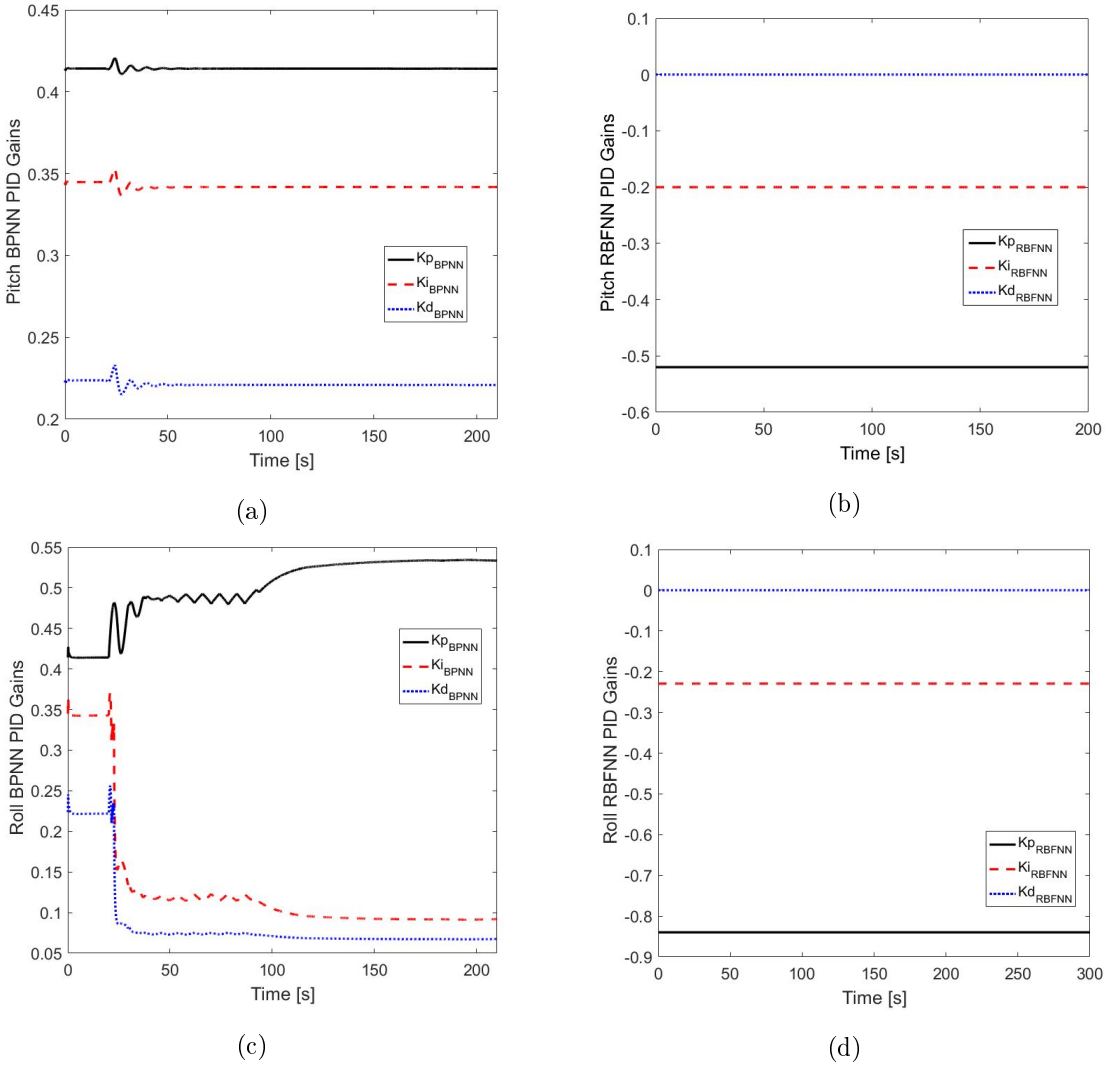


Figure 4.70: Simulated time history of pitch (BPNN-based (a), and RBFNN-based (b)), and roll (BPNN-based (c), and RBFNN-based (d)) gains with 50% hardover elevator failure in the presence of microburst windshear

Figures 4.71 (a) to (f) show the time history of the control deflections during the simulation. As is illustrated in Figure 4.9 (c) both the right and left elevator once again move to an angle of $-0.17\,rad\,(-9.74\,\text{deg})$ (50% of $0.34\,rad$ (19.48 deg))*andremainatthatpositionforthedurationofthesimulatio*
basedcontrollercase.Onecanseethattheflapsremainun−deflectedfortheBPNN−basedcontrollercase.

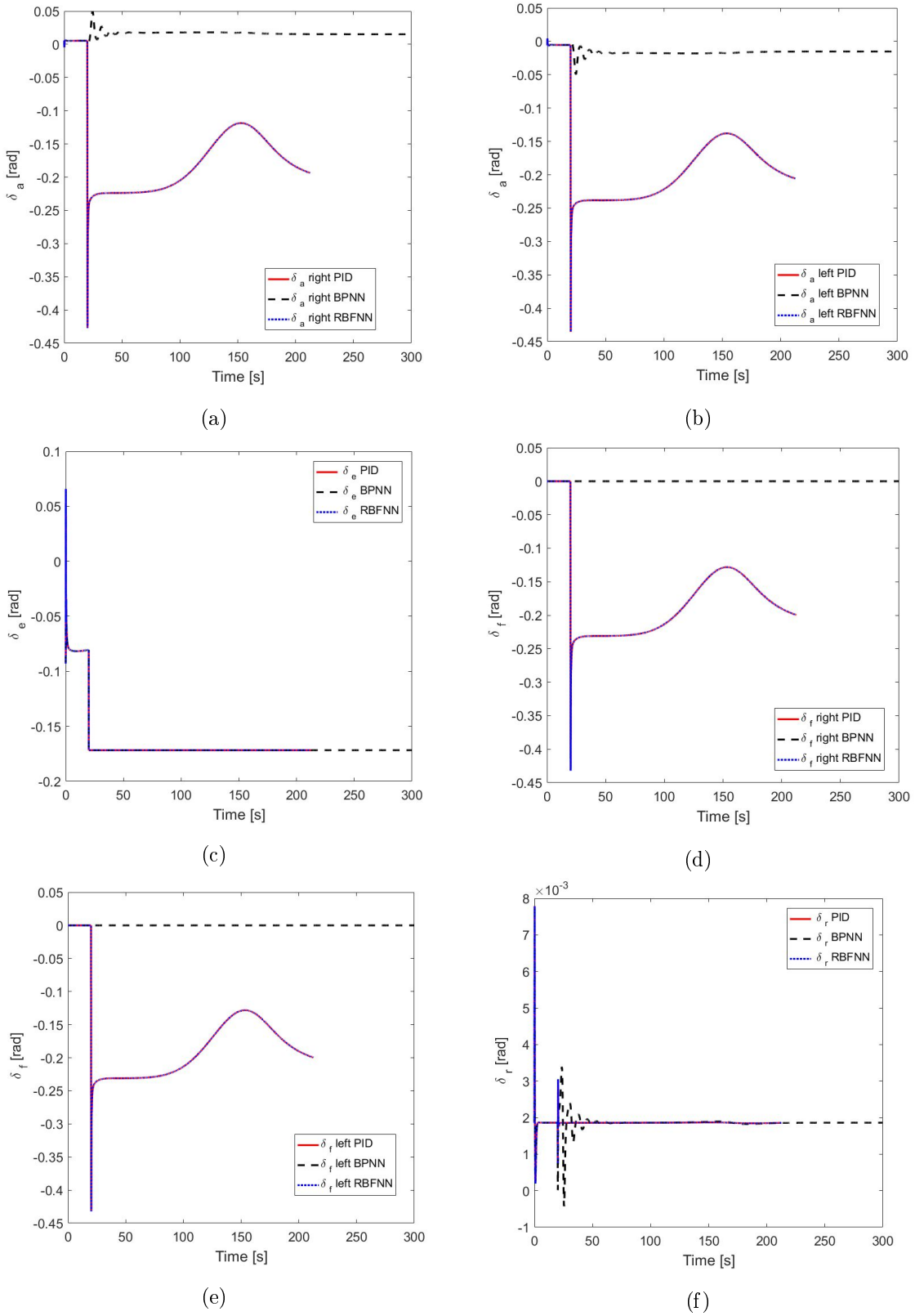
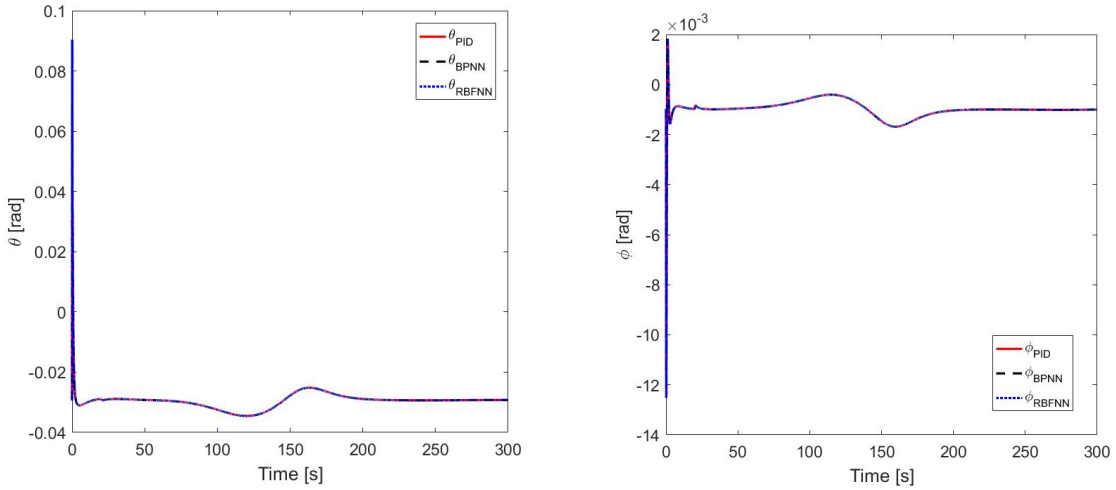


Figure 4.71: Simulated time history of right aileron (a), left aileron (b), elevator (c), right flap (d), left flap (e), and rudder (f) deflections (50% hardover) elevator failure in the presence of microburst windshear

4.4.3.4 Right Aileron Failure - *0 rad* Deflection in the presence of Microburst Windshear

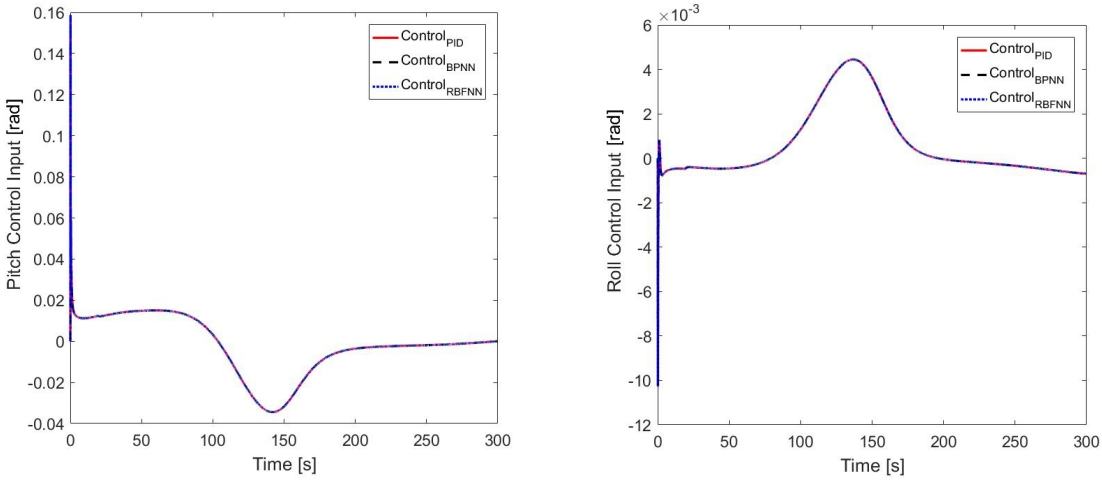
The simulation was run with a 0 *rad* aileron deflection failure at 20 seconds after the start of the simulation as done previously. Figure 4.72 shows a very small dip in the pitch angle at failure which is hardly noticeable, a similar response can be observed in the bank angle plot. Figure 4.73 does show a slight disturbance in the pitch angle control effort. The roll control effort illustrates a small increase in the form of a hardly discernible jump at failure.



(a) (b)

Figure 4.72: Simulated time history of pitch (a) and bank (b) angles (0 *rad* right aileron deflection failure) in the presence of microburst windshear

Figures 4.73 (a) and (b) show a small change in the control input for both pitch and roll controllers.



(a) (b)

Figure 4.73: Simulated time history of pitch (a) and roll (b) control inputs (0 *rad* right aileron) deflection failure in the presence microburst windshear

Figure 4.74 shows the time history of the PID controllers' gains during 0 *rad* right aileron failure. In Figure 4.74 the BPNN-based and RBFNN-based controller gains remain constant.

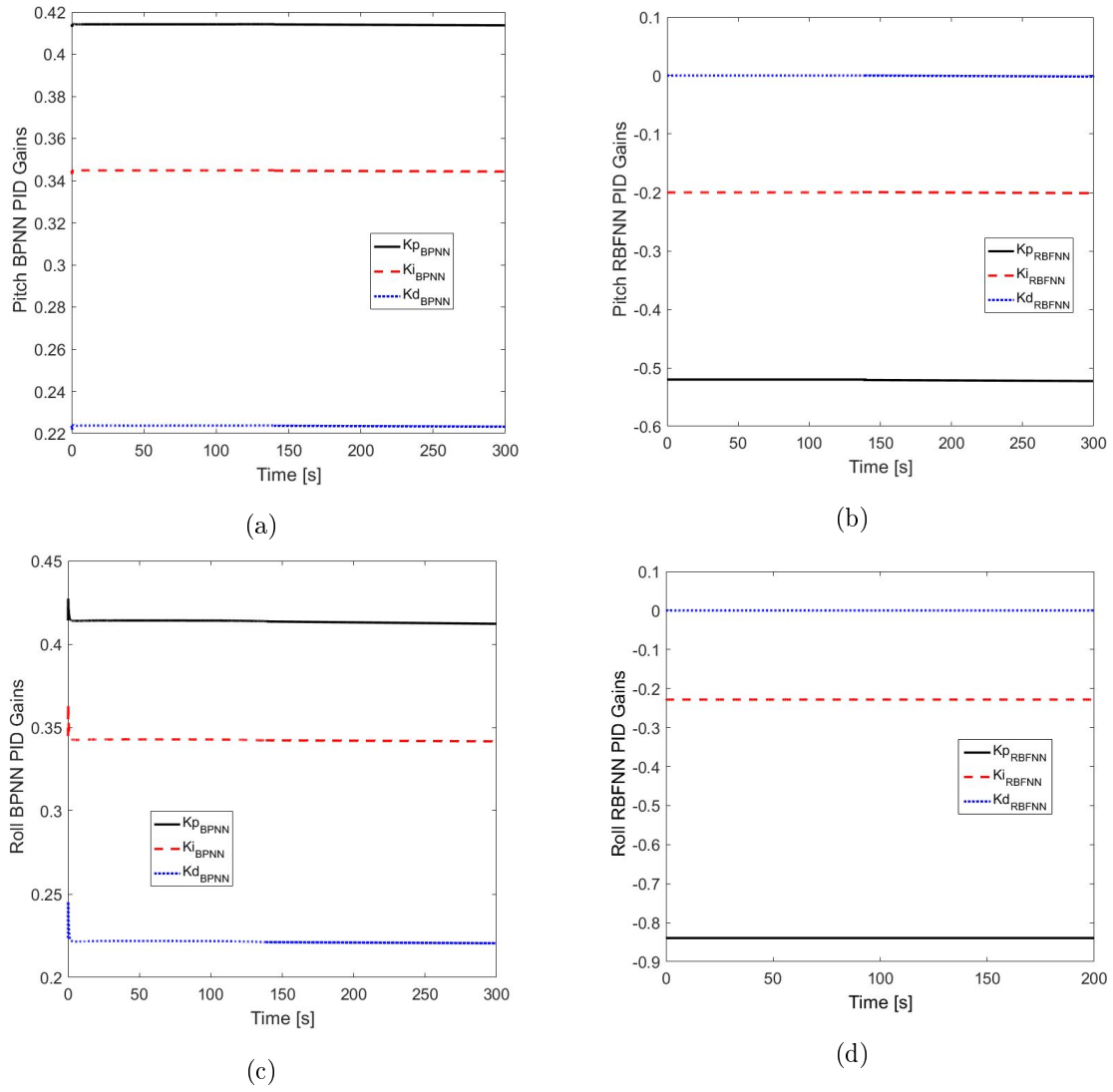
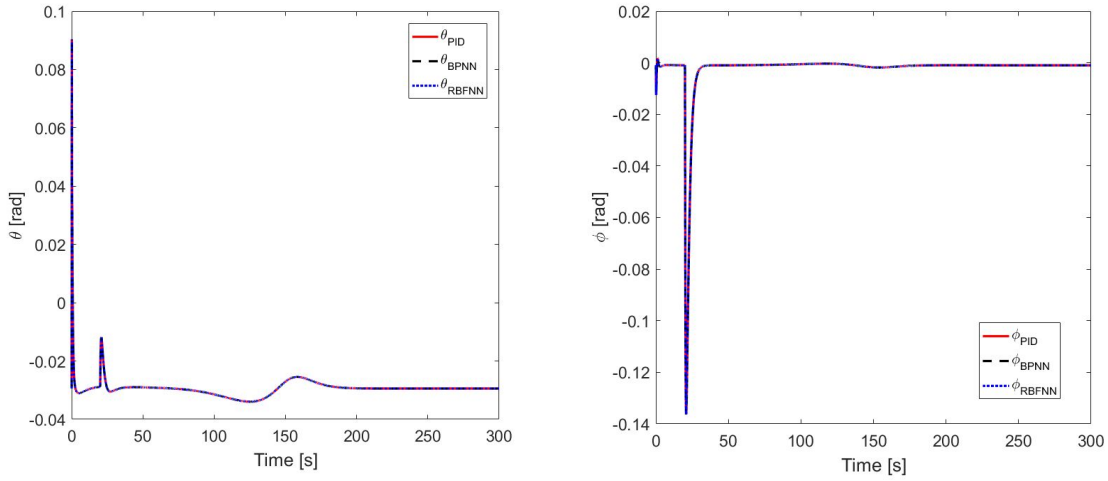


Figure 4.74: Simulated time history of pitch (BPNN-based (a), and RBFNN-based (b)), and roll (BPNN-based (c), and RBFNN-based (d)) gains with 0 *rad* right aileron failure in the presence of microburst windshear

4.4.3.5 Right Aileron Failure - 50% Hardover in the presence of Microburst Windshear

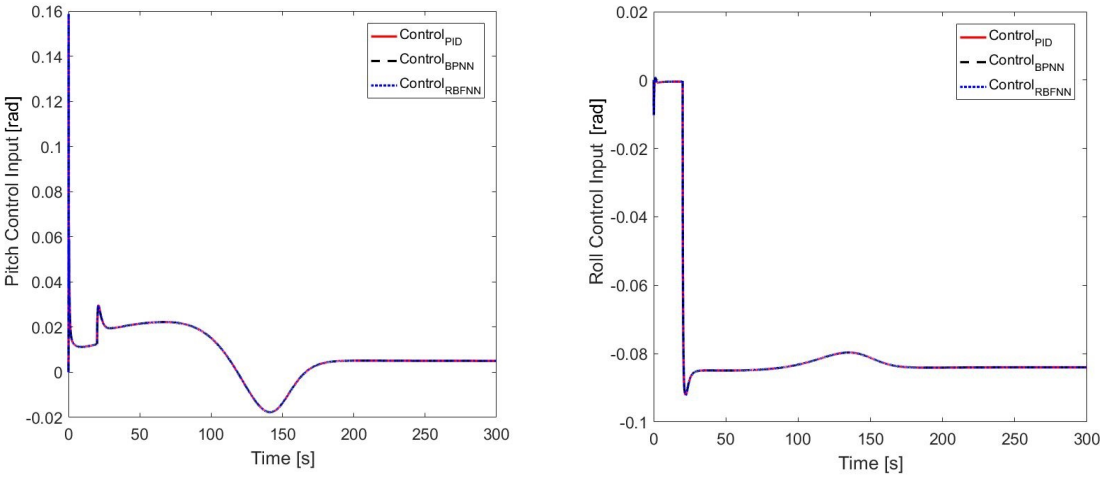
A 50% hardover failure of the right aileron was modelled, the results of which can be found in Figure 4.75. Figure 4.75 shows that there is a rapid decrease in bank angle followed by an equally rapid return to level flight as the controller compensates for this error. The maximum bank angle is found to be -0.136 *rad* (-7.79 deg). A small jump can be observed in the pitch angle at failure. Upon examination of Figure 4.76 it can be observed that there is a rapid change in the control input to mitigate the change in bank angle. There is also a small rapid change in the pitch angle control input in an attempt to mitigate failure.



(a) (b)

Figure 4.75: Simulated time history of pitch (a) and bank angles (b) (50% right aileron hardover failure) in the presence of microburst windshear

Figure 4.76 shows that the control effort for the bank angle shows the PID-based controller case required much less control effort to maintain stability when compared to the NN-based controller cases.



(a) (b)

Figure 4.76: Simulated time history of pitch (a) and roll (b) control inputs (50% right aileron hardover failure) in the presence microburst windshear

Figure 4.77 shows the time history of the PID controllers' gains during 50% right aileron hardover failure. The roll gains (Figures 4.77 (c) and (d)) show a small disturbance in the BPNN-based case while the RFNN-based case show a barely discernible change in the proportional and integral gains. For the pitch control gains (Figure 4.77 (c) and (d)) show a

sharp drop at failure in the BPNN-based controller case while the RBFNN-based controller gains remain constant.

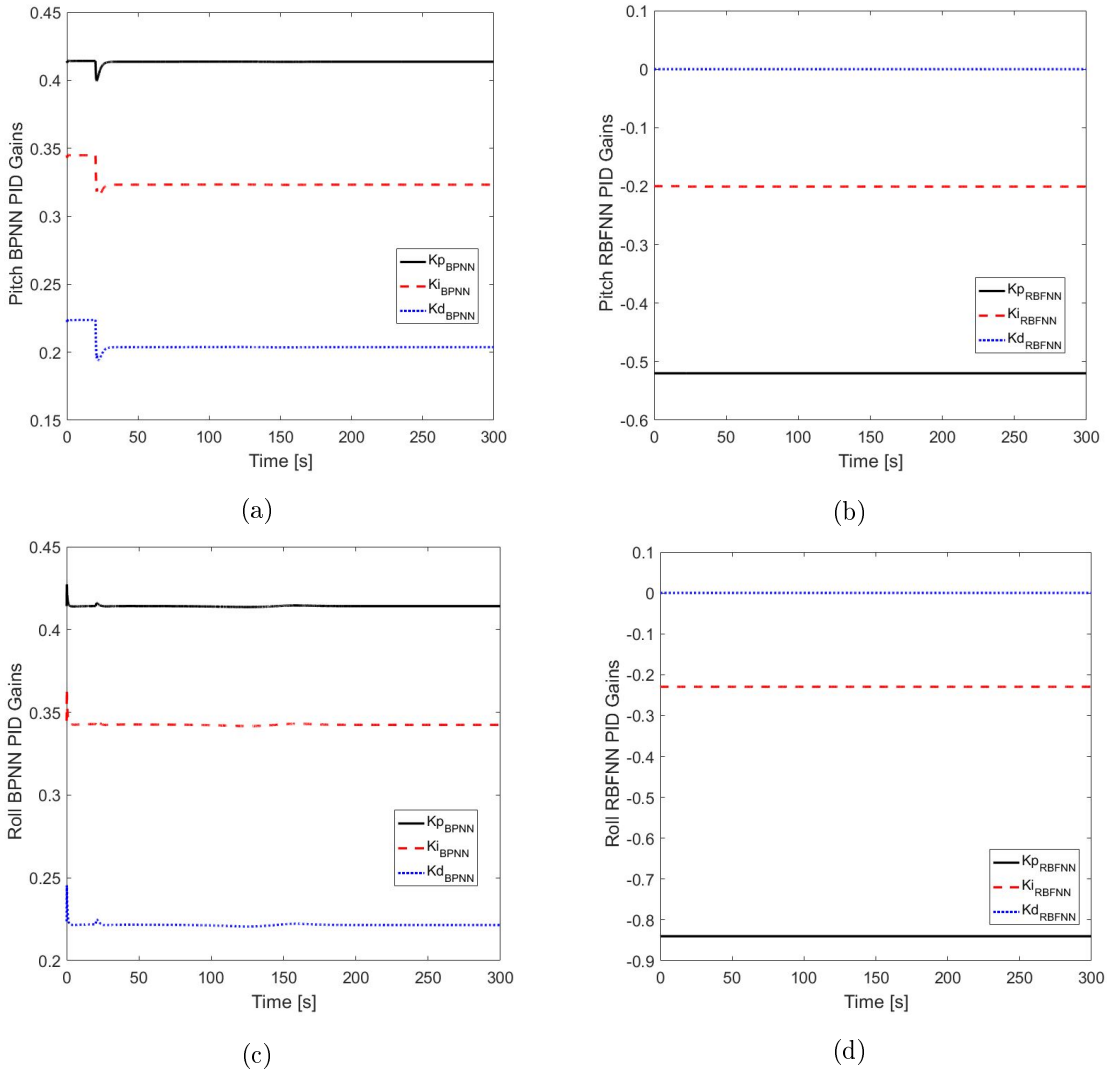


Figure 4.77: Simulated time history of pitch (BPNN-based (a), and RBFNN-based (b)), and roll (BPNN-based (c), and RBFNN-based (d)) gains with 50% right aileron hardover failure in the presence of microburst windshear

Figure 4.78 (a) shows that the right aileron moves to 0.22 *rad* (12.61 deg) and remains in that position until the end of the simulation while the left aileron moves to compensate the induced rolling moment and achieves an angle slightly lower (just above 0.14 *rad* (8.02 deg)) than the right aileron. One also notes a sharp jump in elevator deflection to compensate for the failure (see Figure 4.78 (c)).

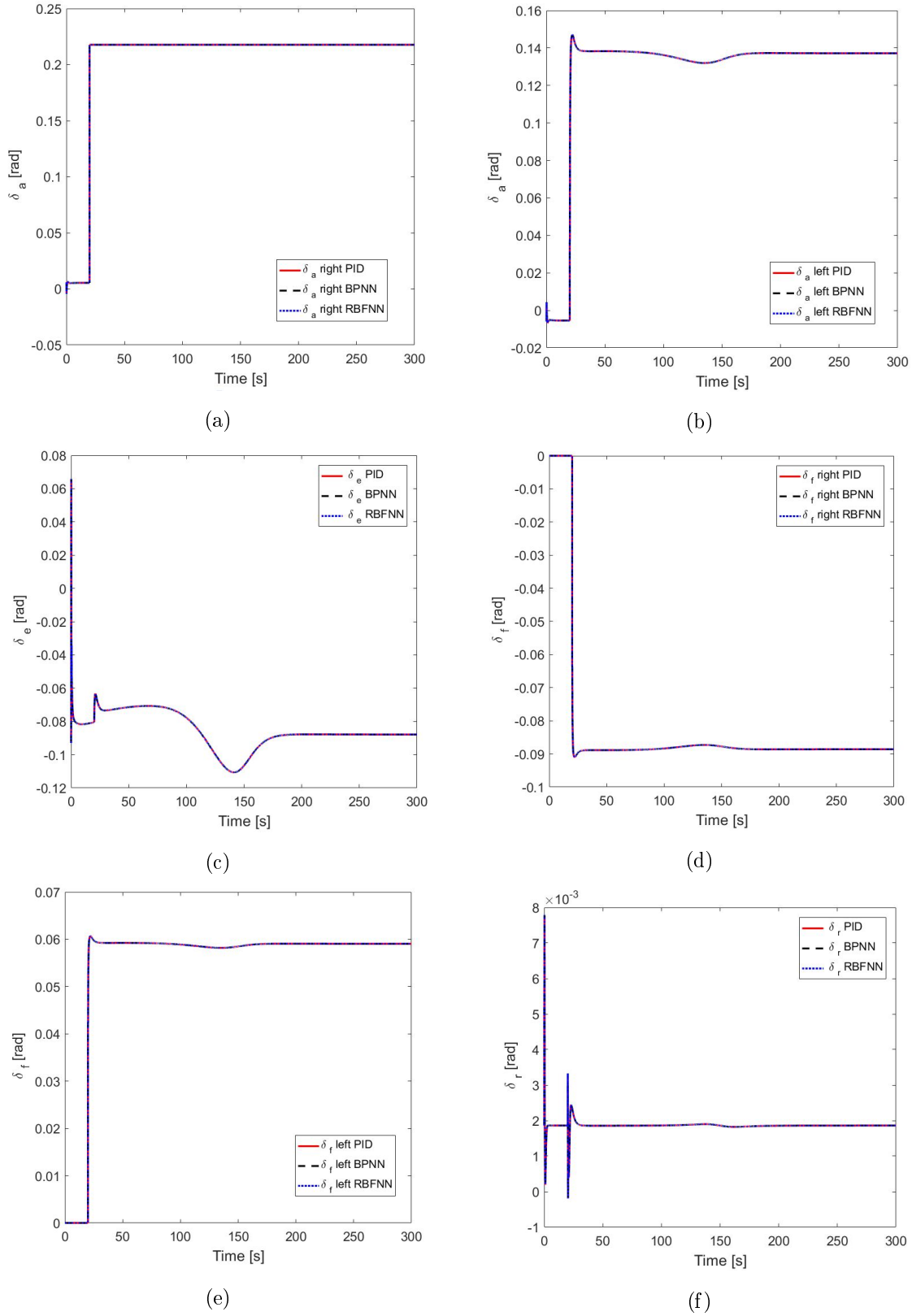
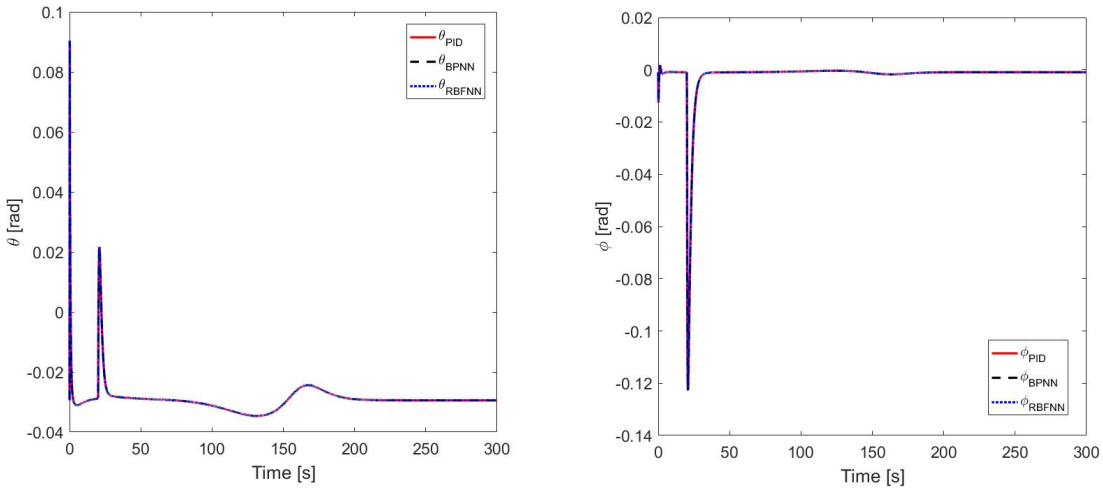


Figure 4.78: Simulated time history of right aileron (a), left aileron (b), elevator (c), right flap (d), left flap (e), and rudder (f) deflections (50% hardover right aileron failure) in the presence of microburst windshear

4.4.3.6 Right Aileron Failure - 100% Hardover Failure in the presence of Microburst Windshear

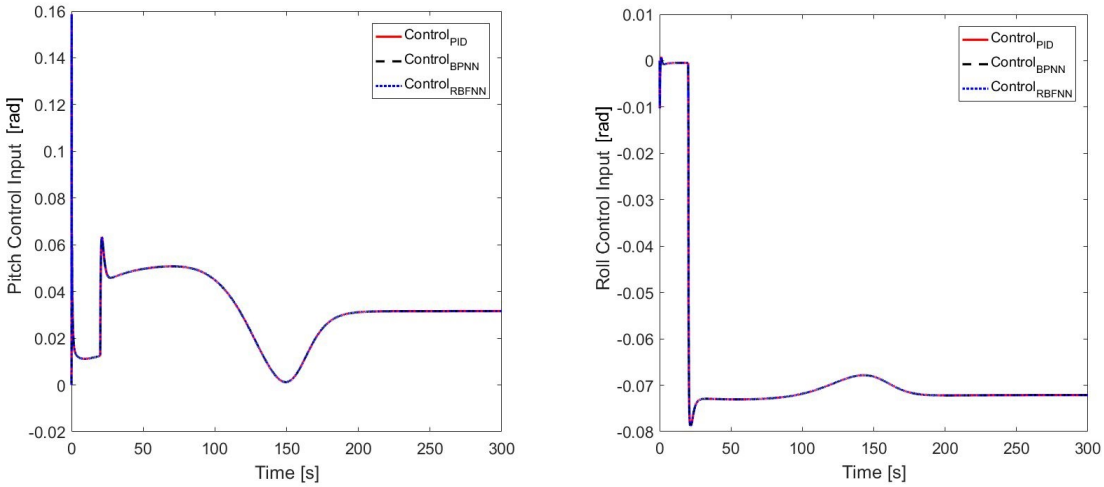
The simulation was run for a 100% hardover failure of the right aileron. The failure causes a strong rolling moment to the right with a small nose-up pitching moment due to change in the effective wing camber. Figure 4.79 shows the time history of the pitch and roll angles. Figure 4.79 shows the sharp increase in the pitch angle at the time of failure, this has increased with the severity of failure. The roll angle performance is similar for both the PID-based and NN-based controller cases (there is a sharp decrease followed by a rapid increase back to the desired setpoint).



(a) (b)

Figure 4.79: Simulated time history of pitch and bank angles (100% right aileron hardover failure) in the presence of microburst windshear

It can be observed in Figure 4.80 (a) that the control input for pitch experienced a jump at failure. A similar response was found in the decrease off the roll control input (Figure 4.76 (b)). The control efforts for both the PID-based and NN-based cases are similar.



(a) (b)

Figure 4.80: Simulated time history of pitch (a) and roll (b) control inputs (100% right aileron hardover failure) in the presence microburst windshear

Figure 4.81 shows the time history of the PID controllers' gains during 100% right aileron hardover failure. The figure shows a drop in the BPNN-based controller pitch gains while the RBFNN-based controller gains do not show any notable changes. The roll gains for the BPNN-based controller case show a slight change while the RBFNN-based controller gains remain constant.

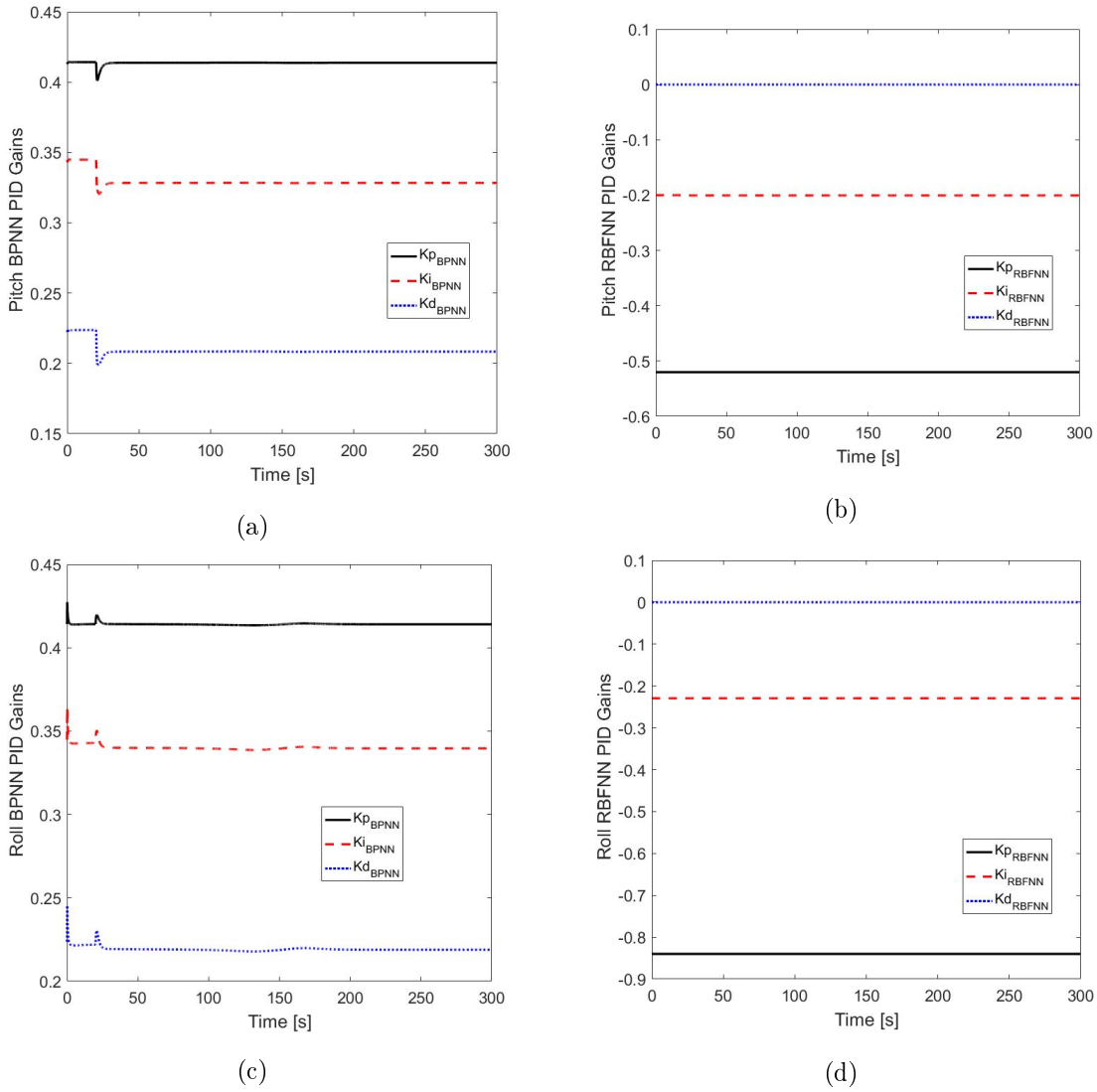


Figure 4.81: Simulated time history of pitch (BPNN-based (a), and RBFNN-based (b)), and roll (BPNN-based (c), and RBFNN-based (d)) gains with 100% right aileron hardover failure in the presence of microburst windshear

Examining Figure 4.82 (a) shows the deflection of the right aileron to maximum. There is a sharp spike in the flap deflections at failure (see Figure 4.23 (d) and (e)).

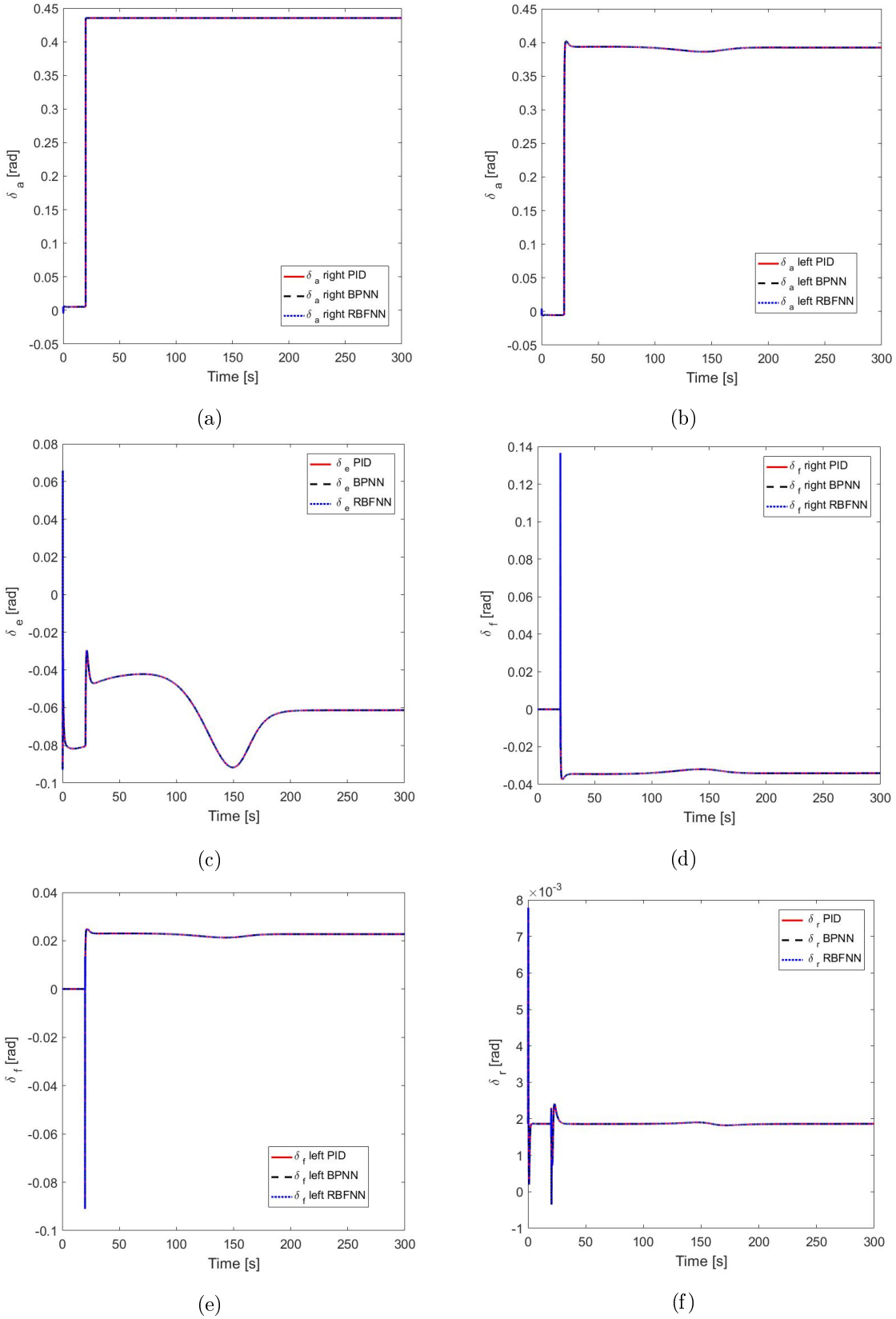
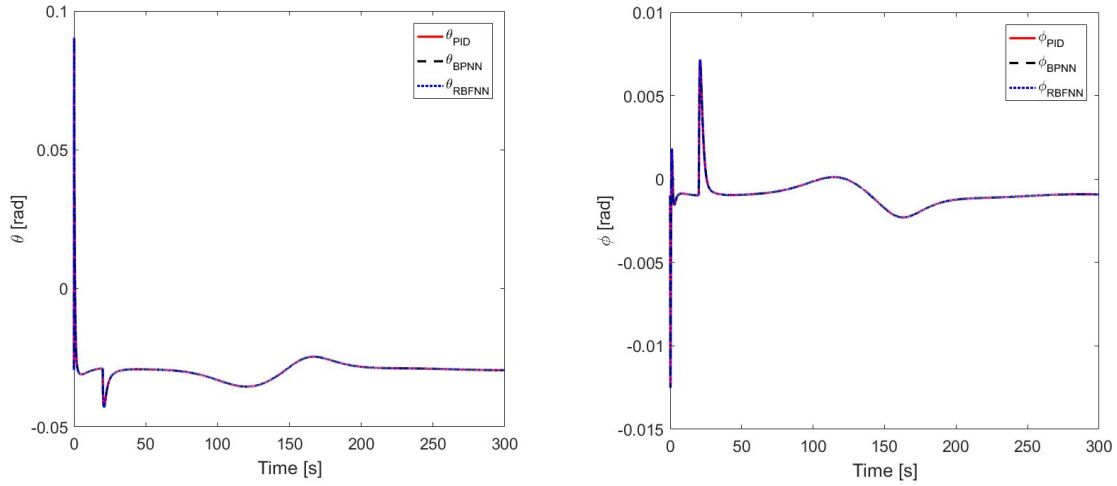


Figure 4.82: Simulated time history of right aileron (a), left aileron (b), elevator (c), right flap (d), left flap (e), and rudder (f) deflections (100% hardover right aileron failure) in the presence of microburst windshear

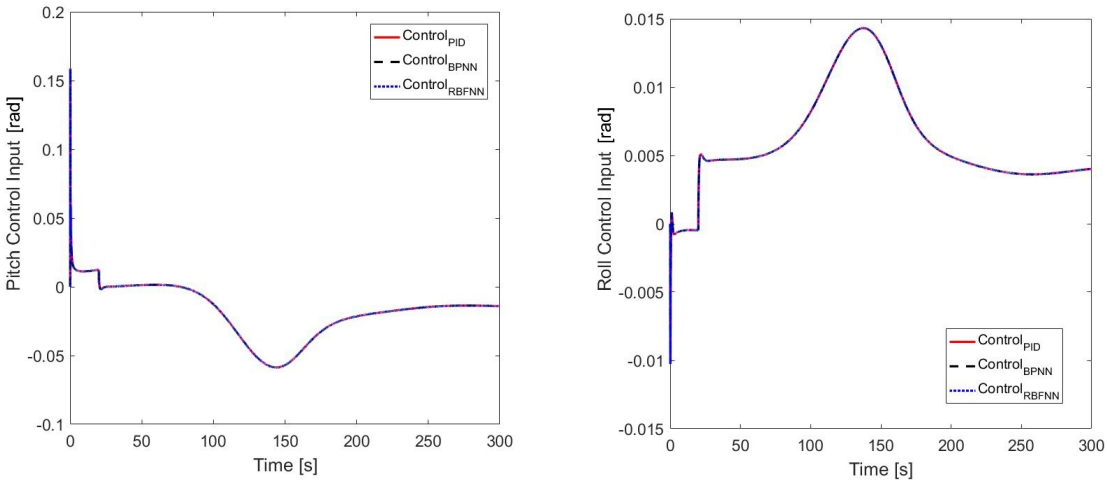
4.4.3.7 Elevator and Aileron combined Failure - *0 rad* Deflection in the presence of Microburst Windshear

Zero *rad* failure was again induced but in this case failing the right aileron and elevators simultaneously. the result of this simulation was a combination of the two previous cases: there was a change in both roll and pitch (see Figure 4.83). Similar to the previous cases, the aircraft was able return to the desired setpoint (the response of the no fault case) in both the PID-based and NN-based controller cases after the initial peak at failure. There was an increase in the roll control input to return to a stable level as shown in Figure 4.84, while the pitch control decreased.



(a) (b)

Figure 4.83: Simulated time history of pitch (a) and bank (b) angles (0 *rad* elevator and right aileron deflection failure) in the presence of microburst windshear



(a) (b)

Figure 4.84: Simulated time history of pitch (a) and roll (b) control inputs (0 *rad* elevator and right aileron deflection failure) in the presence of microburst windshear

Figure 4.85 shows the time history of the PID controllers' gains during 0 *rad* elevator and right aileron deflection failure. There is a subtle change in the BPNN-based controller pitch gains while the RBFNN-based controller pitch gains remain the same. Some slight dip can be observed in the BPNN-based controller roll gains.

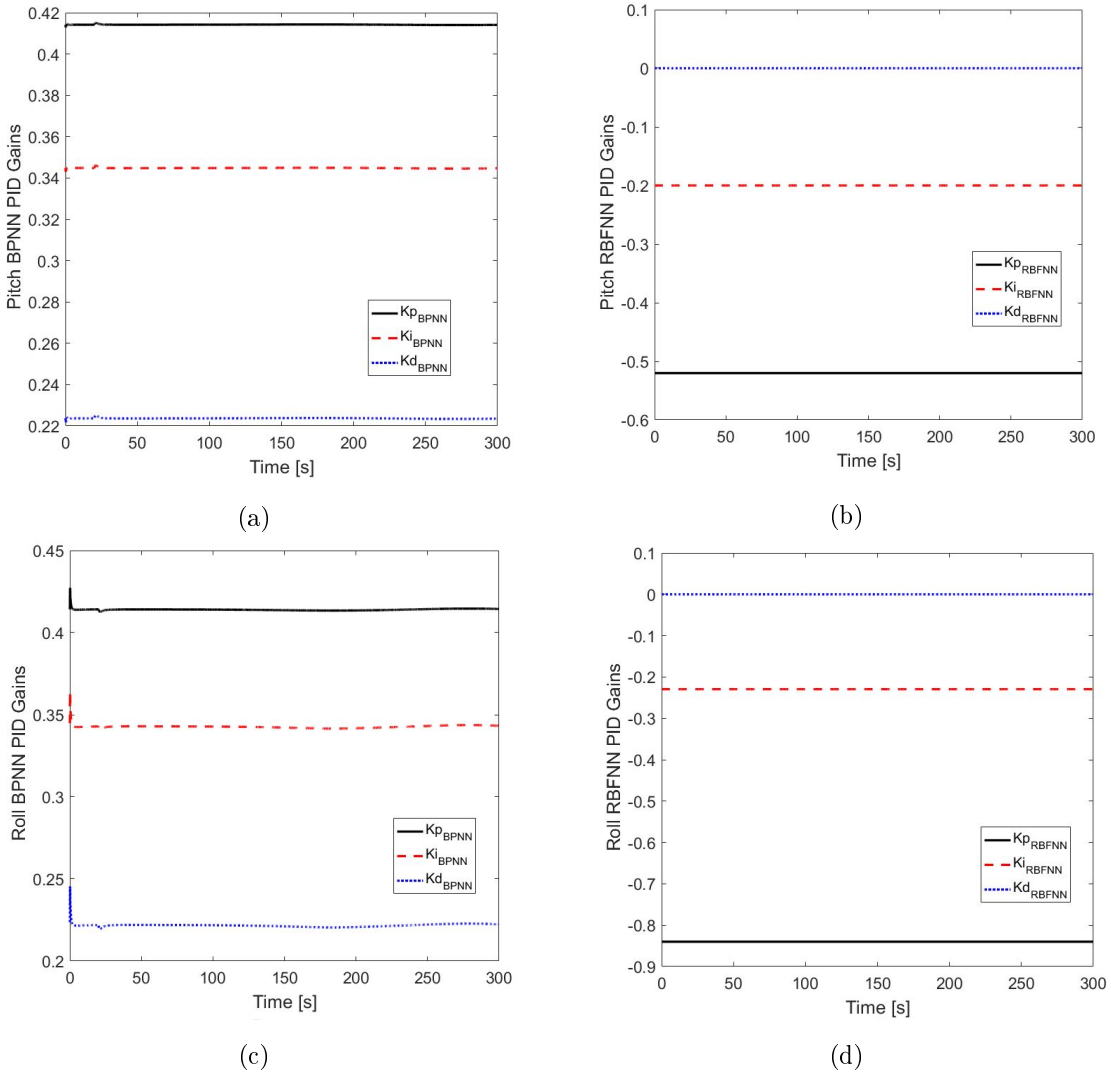


Figure 4.85: Simulated time history of pitch (BPNN-based (a), and RBFNN-based (b)), and roll (BPNN-based (c), and RBFNN-based (d)) gains with 0 *rad* elevator and right aileron failure in the presence of microburst windshear

Figure 4.86 illustrates the control deflections during the simulation.

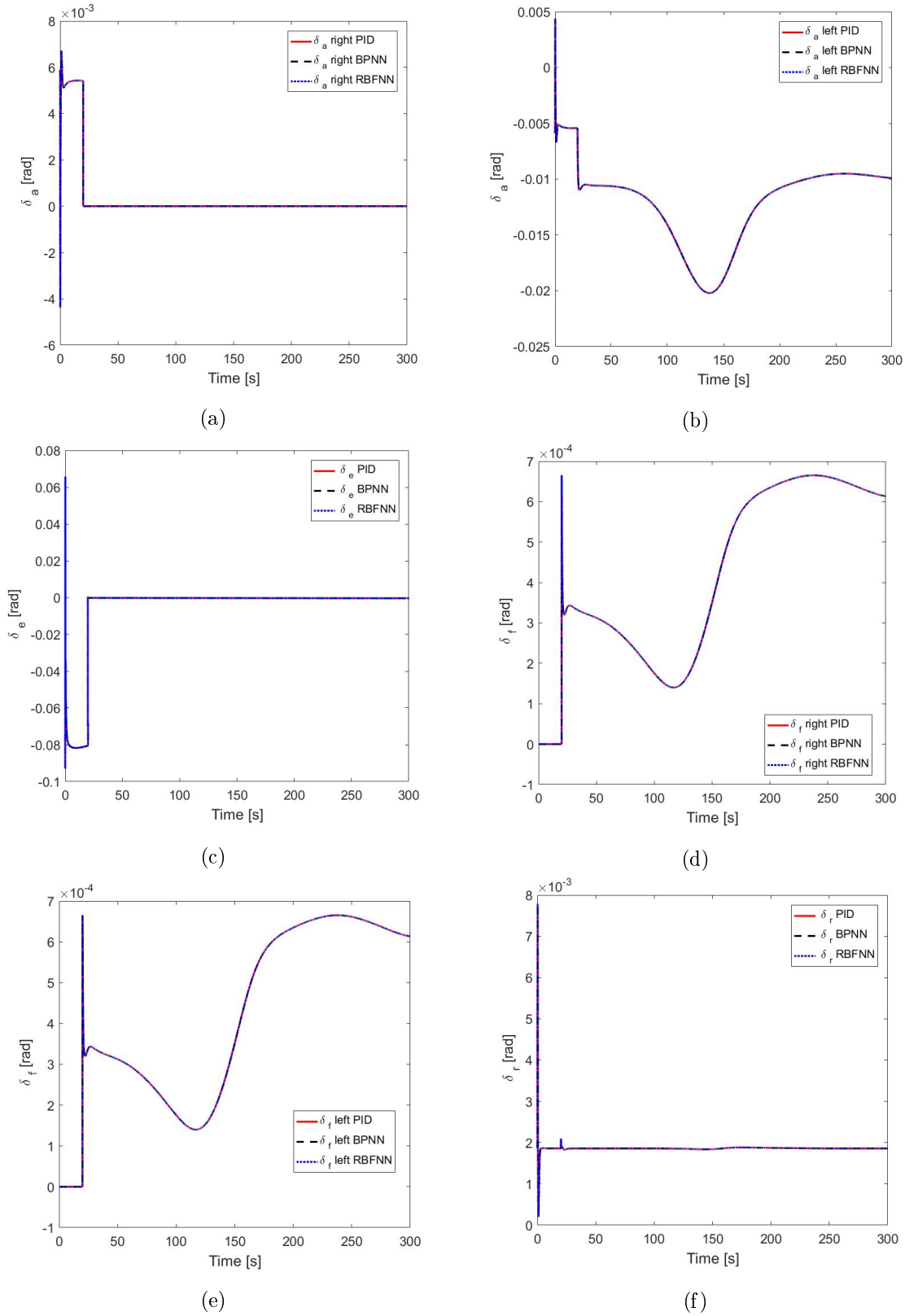
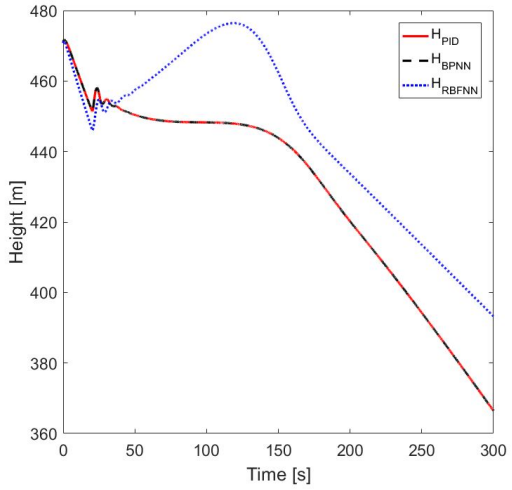


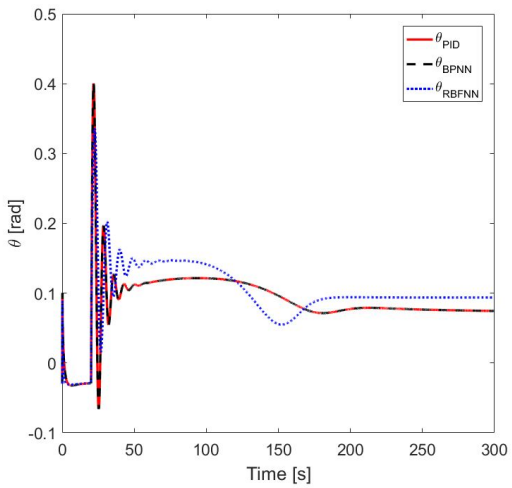
Figure 4.86: Simulated time history of right aileron (a), left aileron (b), elevator (c), right flap (d), left flap (e), and rudder (f) deflections (0 *rad* elevator and right aileron deflection failure) in the presence of microburst windshear

4.4.3.8 Elevator and Aileron combined Failure - 50% Hardover in the presence of Microburst Windshear

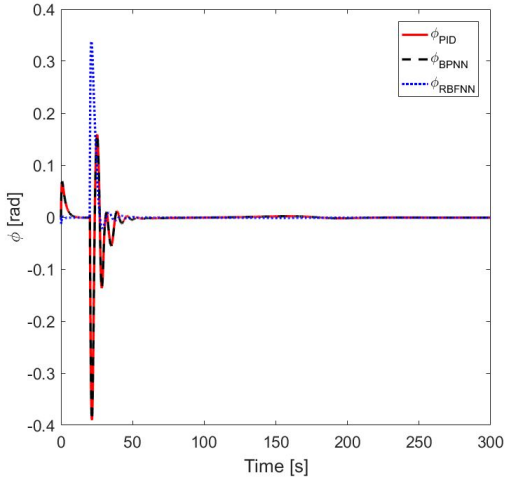
The simulation was run, but with a 50% hardover failure of both elevators and right aileron in the presence of microburst windshear. During this simulation the RBFNN-based controller was run without CA enabled, this was done to compare the controllers' responses. It should be noted that the right aileron failure was negative (opposite to that of the inner-loop right aileron-only failure simulations). Figure 4.87 shows the time history of the pitch and roll angles in the presence of microburst windshear and illustrates a large pitch and roll disturbance. The altitude is also shown here and one can observe the reduced rate of descent that occurs at failure. The pitch disturbance was found to be similar in the PID-based and NN-based controller cases. There is also some oscillation that occurs in the angles' response which dampens out. The RBFNN-based PID controller's oscillations are smaller in magnitude. The effects of microburst are more prevalent in this more severe failure case.



(a)



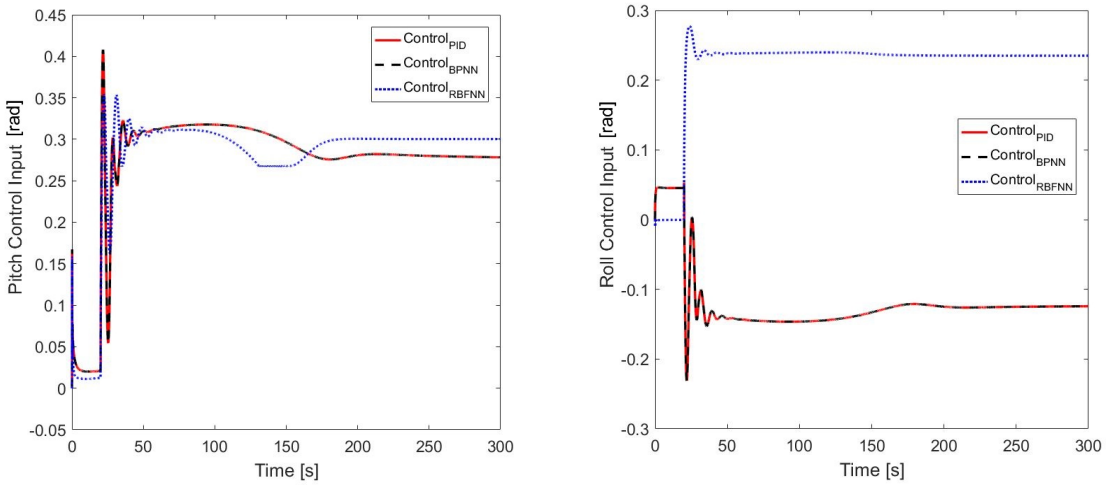
(b)



(c)

Figure 4.87: Simulated time history of altitude, and pitch and bank angles (50% elevator and right aileron hardover failure) in the presence of microburst windshear

Figures 4.88 (a) and (b) show the changes in control input for the pitch and roll angles. The inputs rise sharply with slight oscillation for the pitch to a higher value or lower value in the case of the bank angle for the cases with CA enabled. After some oscillation this then becomes stable. The RBFNN-based controller puts in roll control effort in the opposite direction to that of the other controllers which corresponds to the bank angle response plot. The reason for the control input being in the opposite direction is not clear.



(a) (b) (b) (b)

Figure 4.88: Simulated time history of pitch (a) and roll (b) control inputs (50% elevator and right aileron hardover failure) in the presence microburst windshear

Figure 4.89 shows the time history of the PID controllers' gains during 50% elevator and right aileron hardover failure. There is some oscillation in both the pitch and roll gains (see Figures 4.89 (a) and (c)) for the BPNN-based controllers' case. There is a large peak in the BPNN roll integral and derivative gains at $\pm 170 seconds$. The RBFNN-based controller's integral gain shows a gradual incline from the time failure occurs to the end of the simulation.

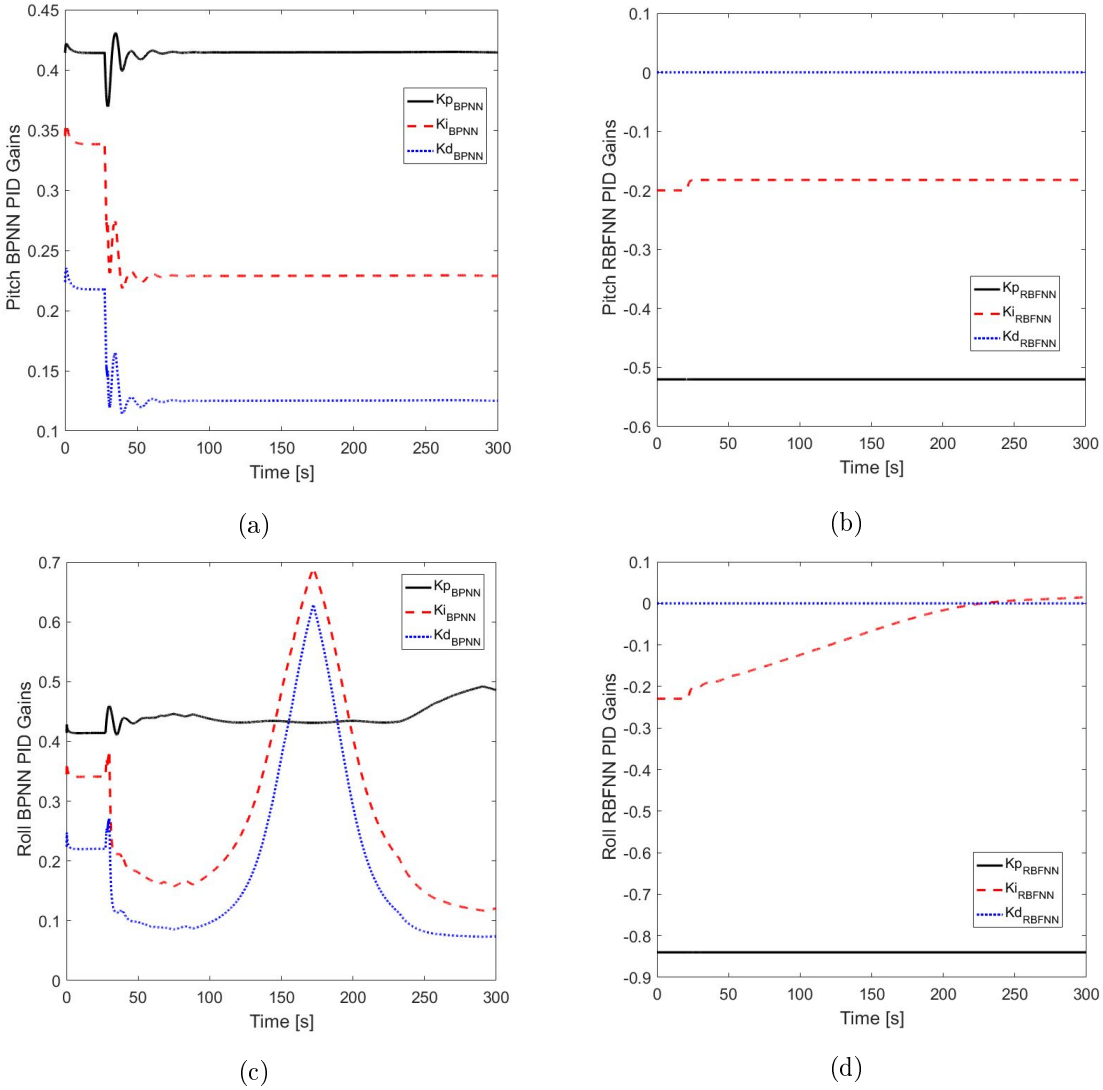


Figure 4.89: Simulated time history of pitch (BPNN-based (a), and RBFNN-based (b)), and roll (BPNN-based (c), and RBFNN-based (d)) gains with 50% elevator and right aileron hardover failure in the presence of microburst windhsear

Figures 4.90 (a) to (f) show the control deflections during the simulation and shows the point at which the elevators and aileron failed. The figure shows that the RBFNN-based case flaps are not deflected at failure.

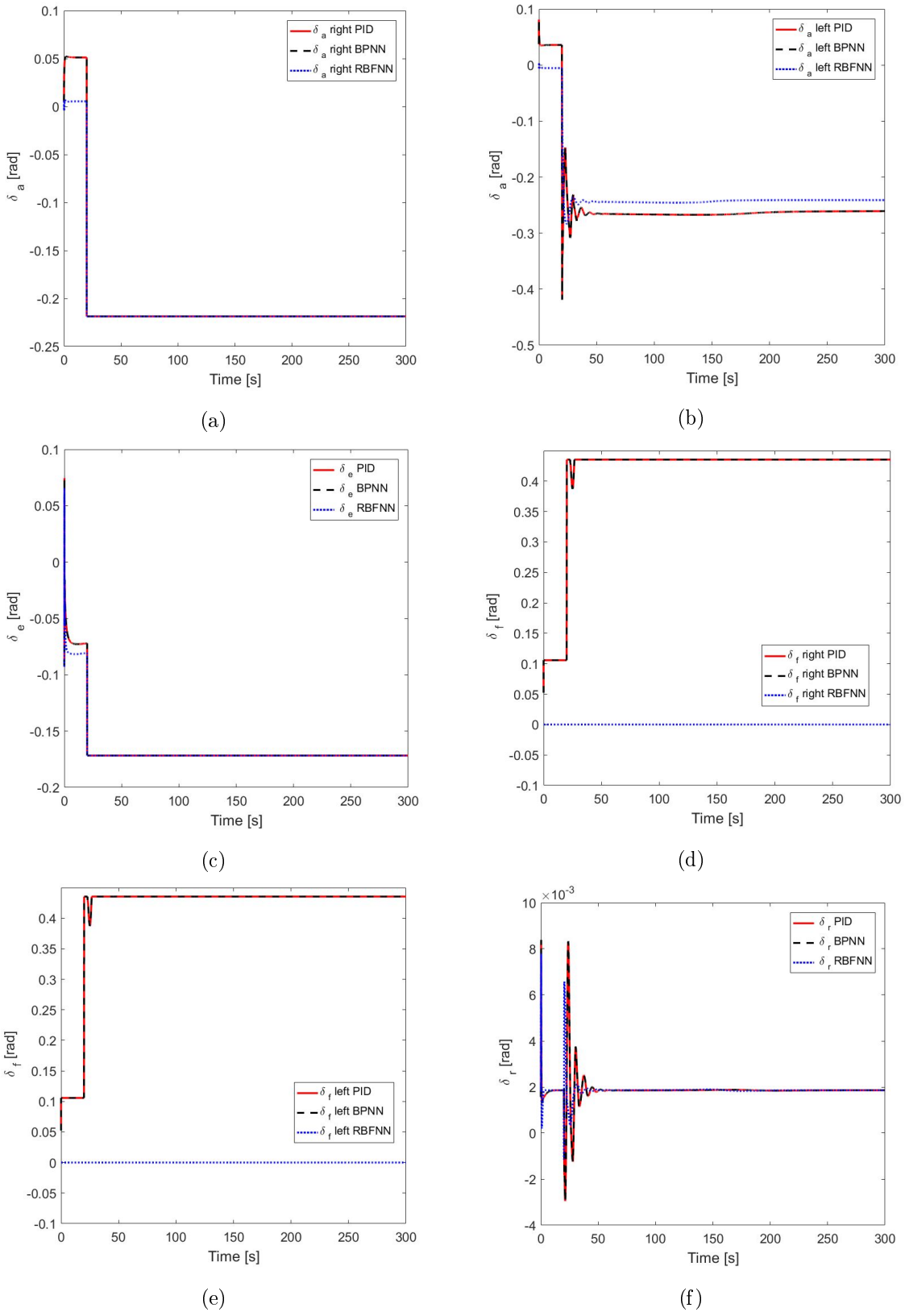


Figure 4.90: Simulated time history of right aileron (a), left aileron (b), elevator (c), right flap (d), left flap (e), and rudder (f) deflections (50% elevator and right aileron hardover failure) in the presence of microburst windshear

4.4.4 Middle Control Loop Simulation - Microburst Windshear

The middle control loop was tested in combination of first climb then the level flight mode. This flight mode control airspeed through throttle input and altitude and heading through use of pitch and roll command respectively. For these tests, the airspeed was initially set to a value of $18m/s$ and altitude $25m$ above sea level. The final altitude for this simulation is above $400m$. This is higher than where microburst simulations usually take place. This is because the UAV is very small and requires maximum throttle to climb, so if it experiences the more intense part of the microburst (which it passes over during the level flight mode) it will be very difficult for it to recover in the presence of failure. During climb the microburst speed is still increasing so its effects are not as large as when it flies through it during level flight (it is during this time that the microburst reaches its maximum speed). The no-fault plots that follow show that the effects of the microburst windshear are still experienced by the UAV. The interaction between the inner-loop and outer-loop controllers also plays a factor in running the simulation at a higher altitude. The sensor noise was enabled in the middle loop simulation.

4.4.4.1 Control - No Failure in the presence of Microburst Windshear

A control simulation was run to determine the response of the aircraft under these conditions without failures. The simulation was run for both the PID-based and NN-based controller cases. The responses between the cases showed no noticeable differences. Figure 4.92 shows the pitch, bank, and heading angle responses in the no fault condition. One can observe an initial overshoot each time the setpoint changes (at 0 and 60 seconds) before it moves to the setpoint value for the heading and bank angle responses while the pitch angle shows a gradual move to the setpoint. The effects of microburst windshear are shown in the bank and heading angle responses after the setpoint changes.

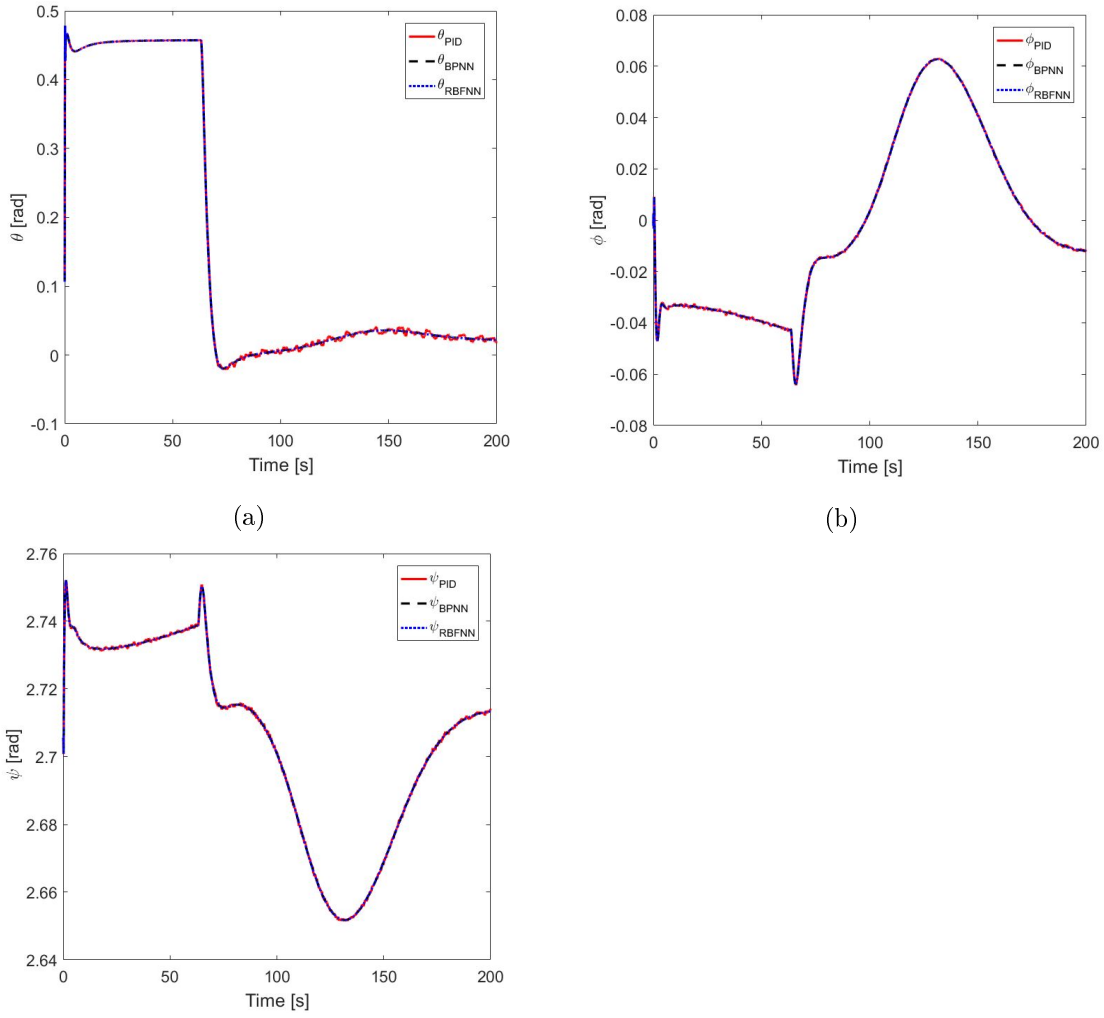
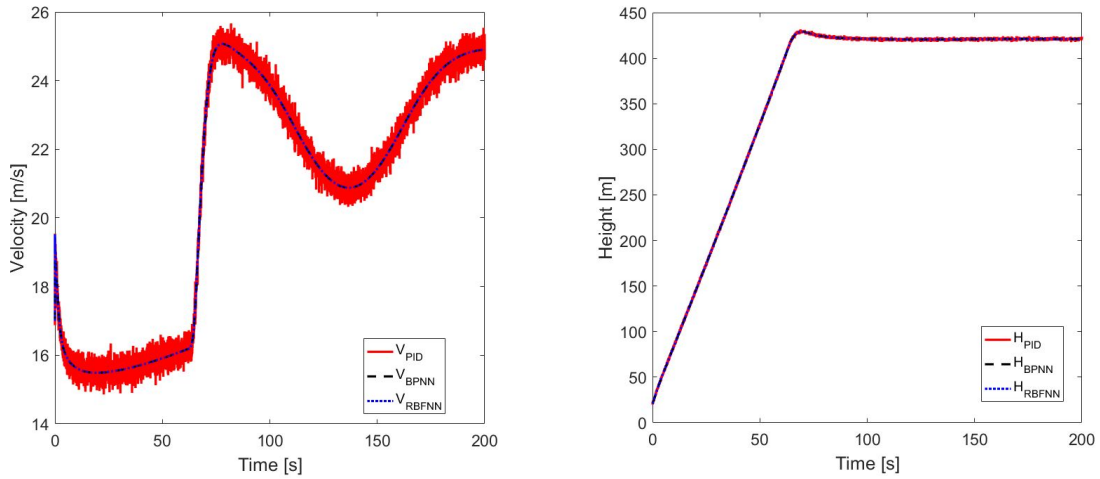


Figure 4.91: Simulated time history of middle loop pitch (a), bank (b) and heading (c) angles in the presence of microburst windshear without failures

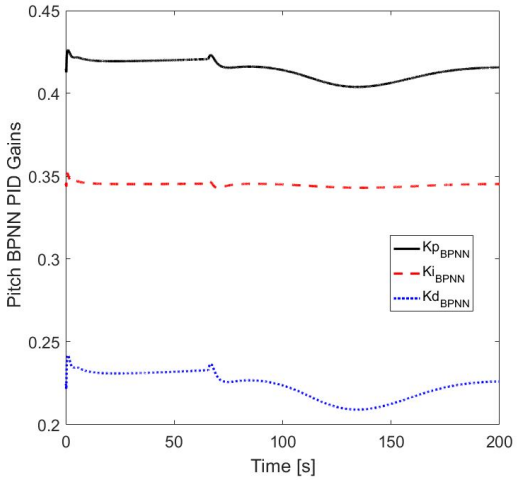
Figure 4.92 (a) shows the large effect of microburst windshear on the velocity (there large dip after the setpoint changes is the most notable). Figure 4.92 (b) shows the UAV maintaining the desired climb rate of $5m/s$ until it is required to maintain level altitude at 60 seconds.



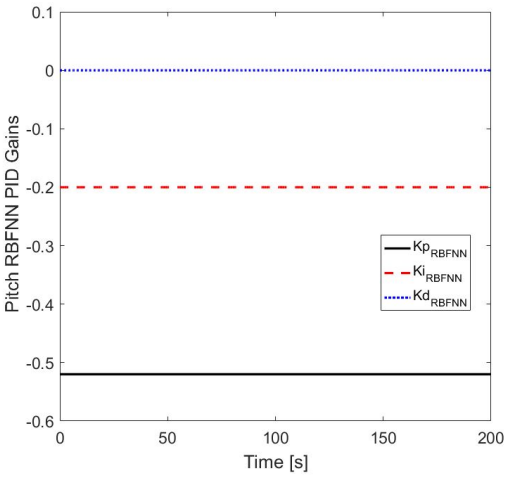
(a) (b)

Figure 4.92: Simulated time history of the airspeed (a), and altitude (b) in the presence of microburst windshear without failures

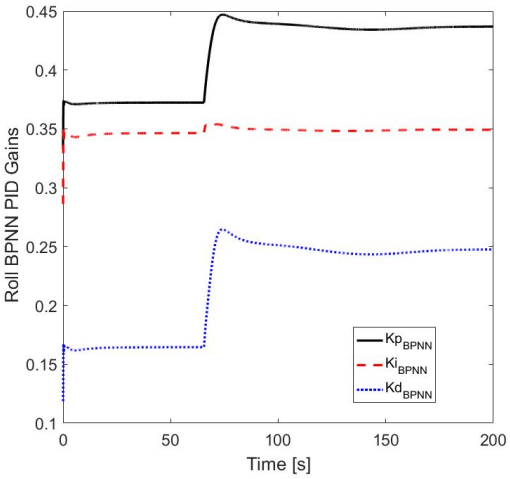
Figure 4.93 shows the variation in PID controllers' gains for all control loops. The pitch gains (see Figures 4.93 (a) and (b)) show that the BPNN-based controller gains have a slight change at the time when the setpoint changes and dip which corresponds to the microburst while the RBFNN-based controller gains remain constant as before. In the roll gains (see Figures 4.93 (c) and (d)) a similar pattern is found in the RBFNN-based controller gains while the BPNN-based controller gains increase then almost level out when the setpoint changes, the proportional and derivative gains experience a much larger increase. The velocity gains (see Figures 4.93 (e) and (f)) for the BPNN-based controller remain constant while for the RBFNN-based controller case the integral gain gradually decreases halfway through the simulation. The altitude gains on the other hand (see Figures 4.93 (g) and (h)) remain constant for all cases. Lastly, in the heading gains (see Figures 4.93 (i) and (j)) in particular the BPNN-based controller gains there is a disturbance when the setpoint changes. There is also a disturbance due to microburst windshear observed after the setpoint changes. The RBFNN-based controller heading gains remain constant.


Pitch BPNN PID Gains
Time [s]
Kp_BPNN
Ki_BPNN
Kd_BPNN

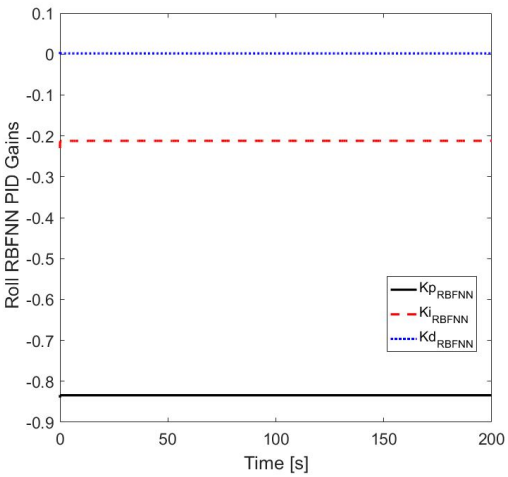
(a)


Pitch RBFNN PID Gains
Time [s]
Kp_RBFNN
Ki_RBFNN
Kd_RBFNN

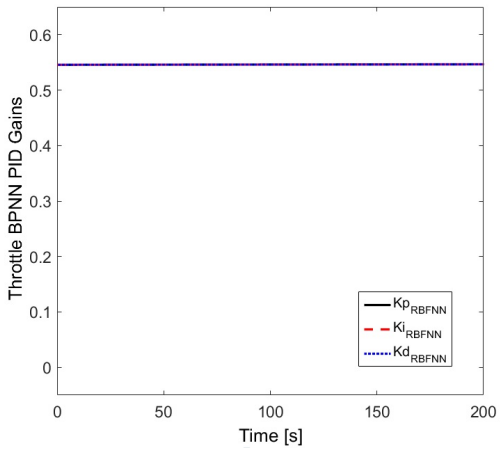
(b)


Roll BPNN PID Gains
Time [s]
Kp_BPNN
Ki_BPNN
Kd_BPNN

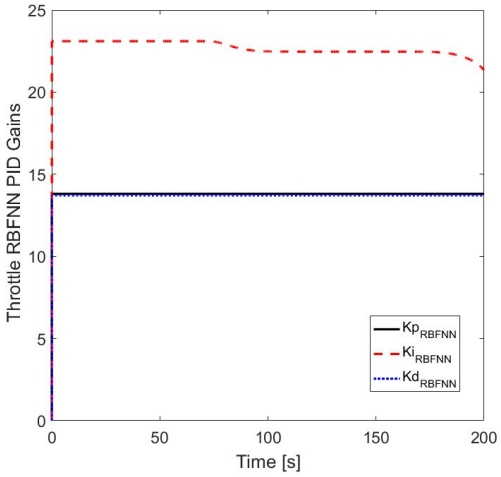
(c)


Roll RBFNN PID Gains
Time [s]
Kp_RBFNN
Ki_RBFNN
Kd_RBFNN

(d)


Throttle BPNN PID Gains
Time [s]
Kp_RBFNN
Ki_RBFNN
Kd_RBFNN

(e)


Throttle RBFNN PID Gains
Time [s]
Kp_RBFNN
Ki_RBFNN
Kd_RBFNN

(f)

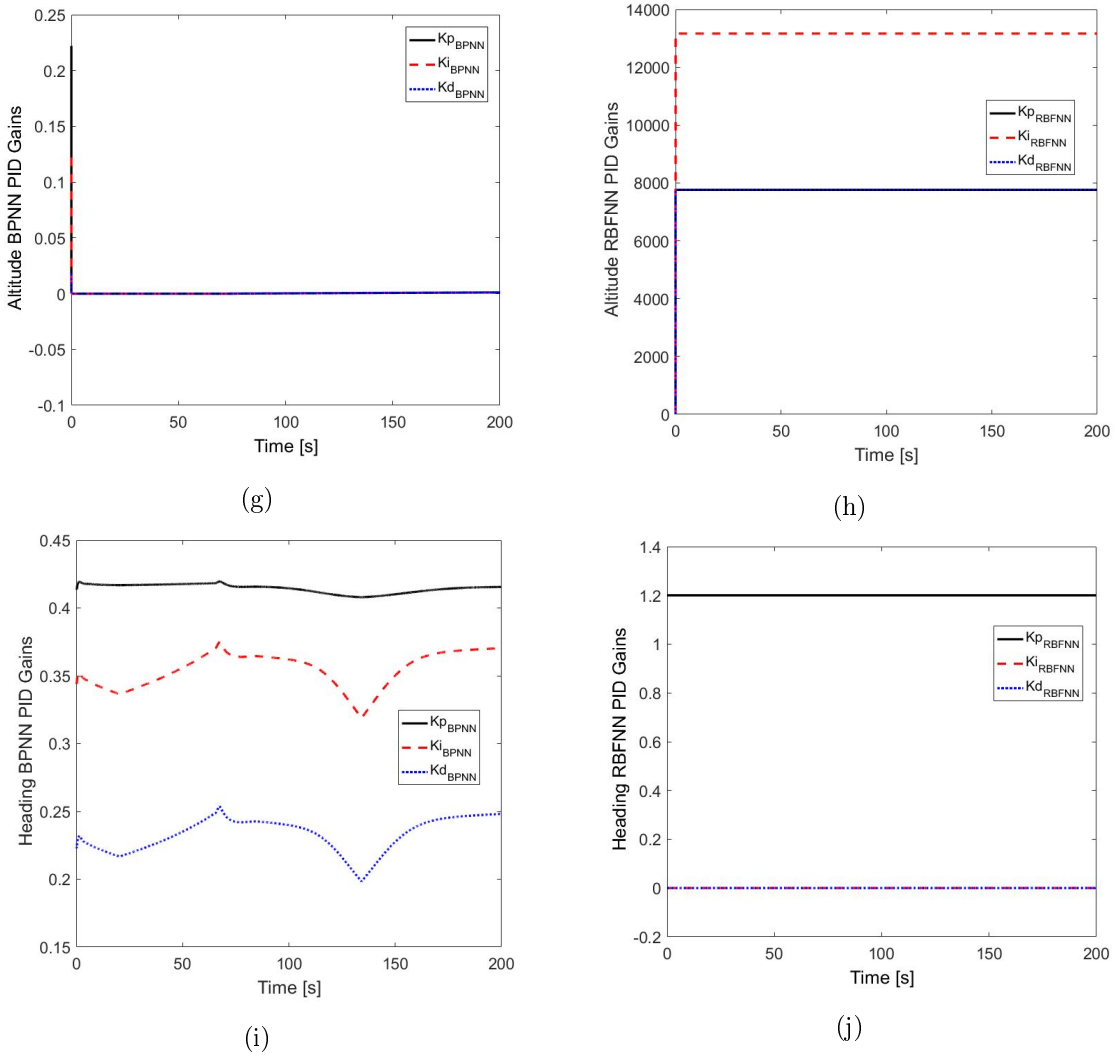


Figure 4.93: Simulated time history of the pitch (BPNN-based (a), and RBFNN-based (b)), and roll (BPNN-based (c), and RBFNN-based (d)), velocity (BPNN-based (e), and RBFNN-based (f)), altitude (BPNN-based (g), and RBFNN-based (h)), and heading (BPNN-based (i), and RBFNN-based (j)) PID gains without failure in the presence of microburst windshear

4.4.4.2 Elevator Failure - *0 rad* Deflection in the presence of Microburst Windshear

The simulation was run with 0 *rad* elevator deflection failure. That is, the elevator moves from its current position, at the time of failure to its neutral position. To maintain consistency with the previous simulations, the failure was once again induced at 20 seconds. Figure 4.94 shows the time history of the pitch, roll and heading angles. The pitch (Figure 4.94 (a)) response is similar for all three cases, namely, it results in a drop in the pitch angle when the failure occurs. The heading angle increases slightly while the roll angle shows a slight dip.

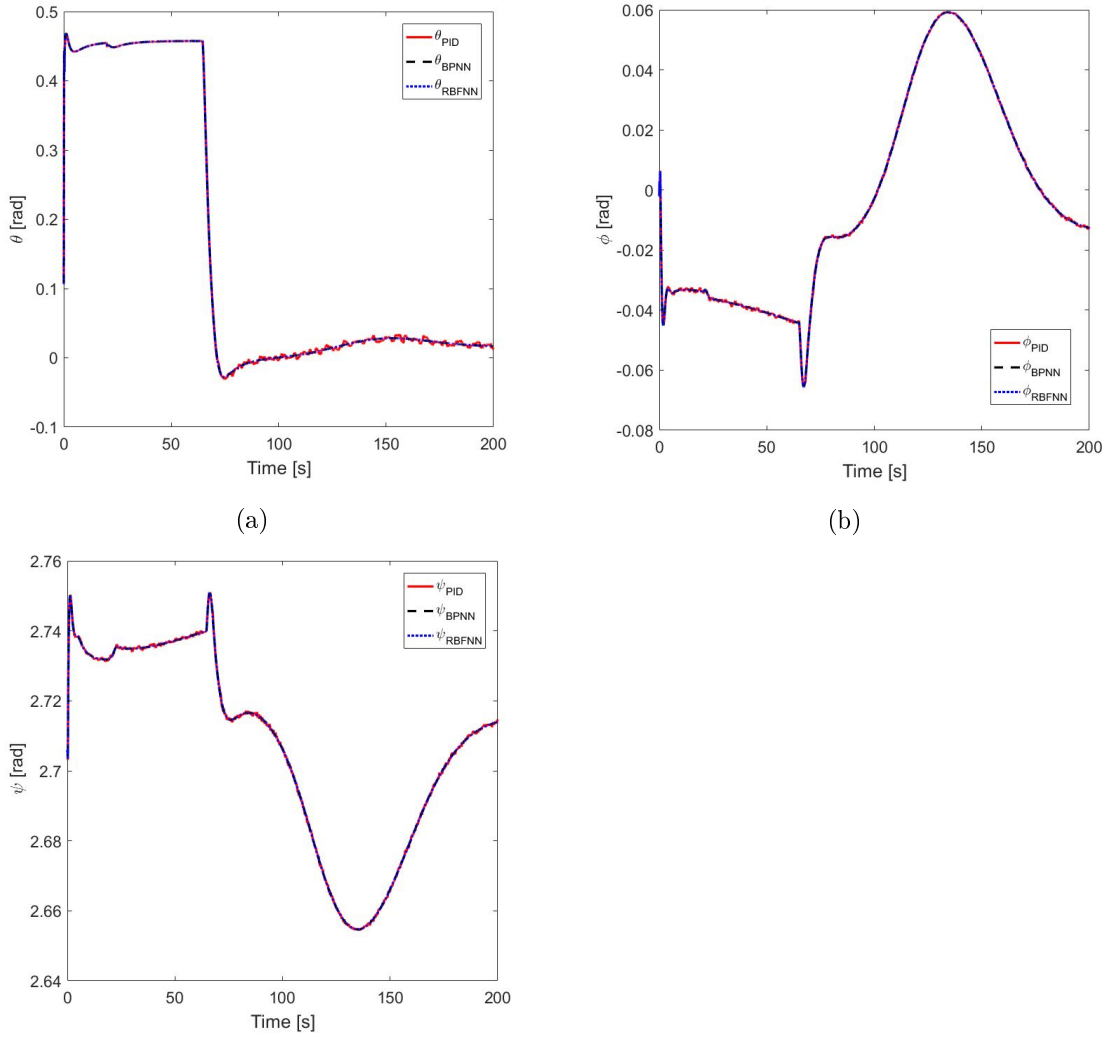
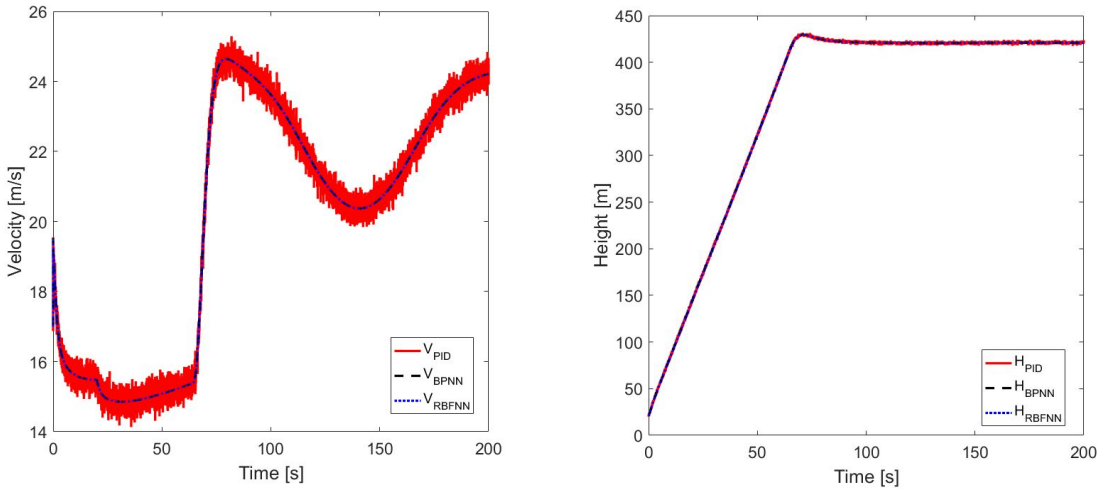


Figure 4.94: Simulated time history of middle loop pitch (a), bank (b) and heading (c) angles (0 *rad* deflection) in the presence of microburst windshear

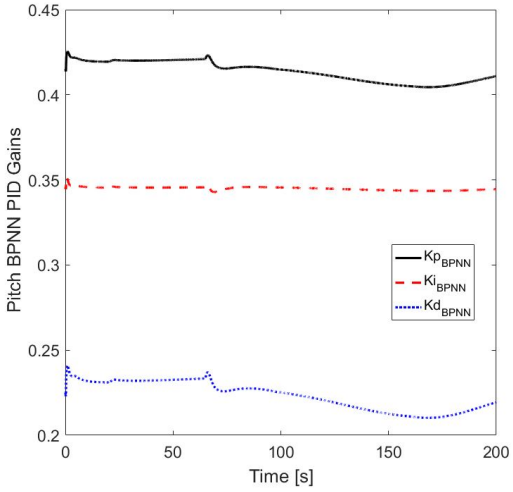
Figure 4.95 shows the time history of the airspeed, there is a slight drop in the speed at failure. Figure 4.95 (b) shows there is no discernible difference in altitude due to failure.



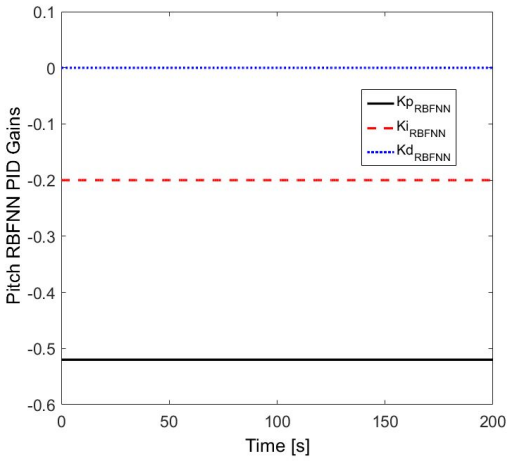
(a) (b)

Figure 4.95: Simulated time history of the airspeed (a), and altitude (b) (0 *rad* deflection) in the presence of microburst windshear

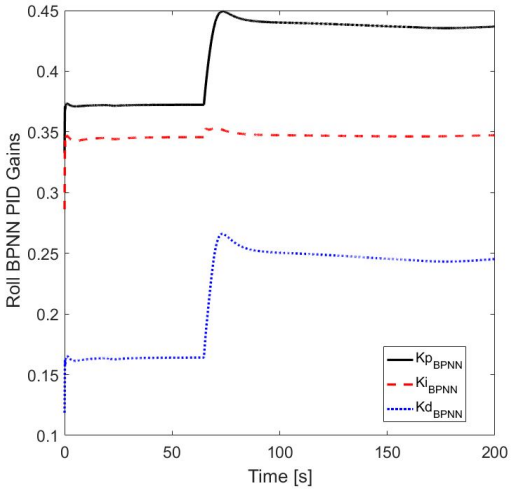
Figure 4.96 shows the variation in PID controllers' gains for all control loops. The gains are similar to that of the no fault cases with a hardly discernible disturbance in some of the BPNN-based controllers gains. The BPNN-based controller altitude gains showed a step-like behaviour at failure and only settled after the setpoint changed. The BPNN-based throttle gains overlap.


Pitch BPNN PID Gains
Time [s]
Kp BPNN
Ki BPNN
Kd BPNN

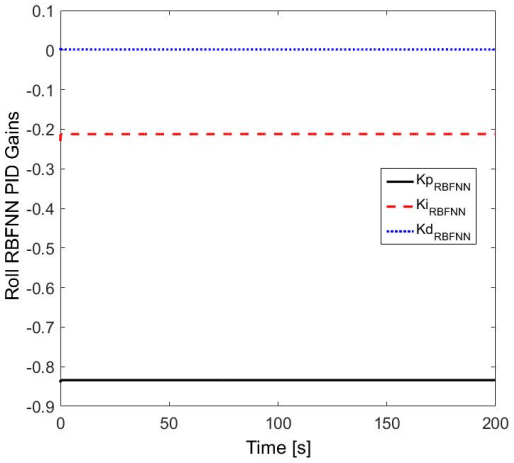
(a)


Pitch RBFNN PID Gains
Time [s]
Kp RBFNN
Ki RBFNN
Kd RBFNN

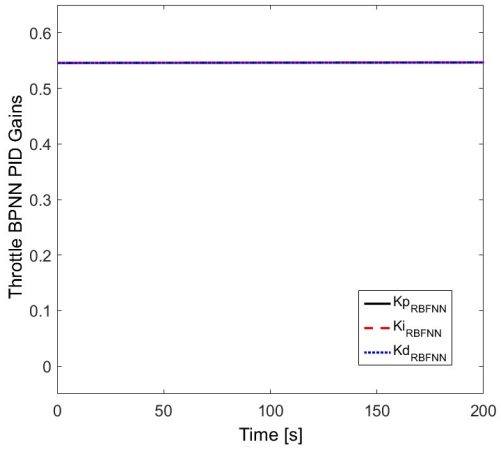
(b)


Roll BPNN PID Gains
Time [s]
Kp BPNN
Ki BPNN
Kd BPNN

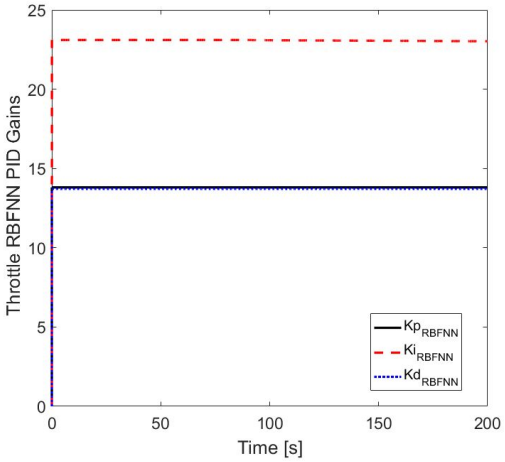
(c)


Roll RBFNN PID Gains
Time [s]
Kp RBFNN
Ki RBFNN
Kd RBFNN

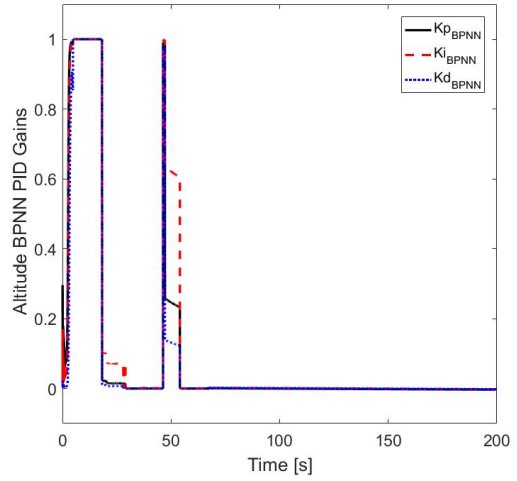
(d)


Throttle BPNN PID Gains
Time [s]
Kp RBFNN
Ki RBFNN
Kd RBFNN

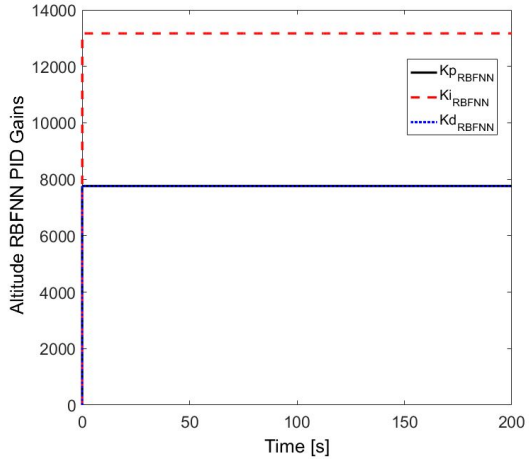
(e)


Throttle RBFNN PID Gains
Time [s]
Kp RBFNN
Ki RBFNN
Kd RBFNN

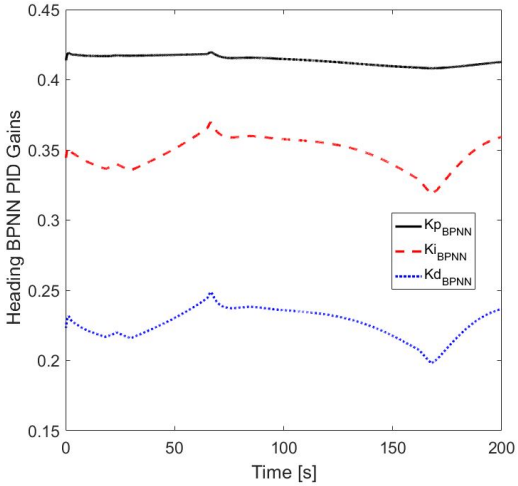
(f)



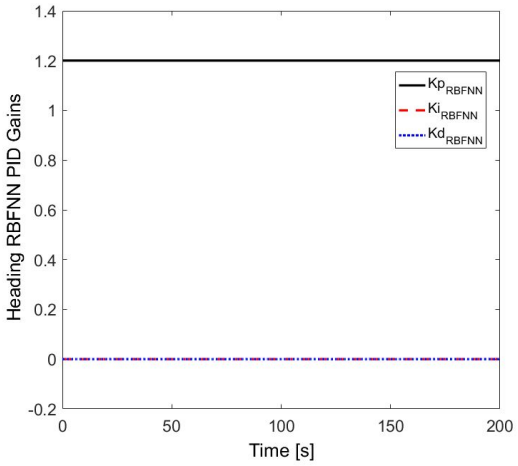
(g)



(h)



(i)



(j)

Figure 4.96: Simulated time history of the pitch (BPNN-based (a), and RBFNN-based (b)), and roll (BPNN-based (c), and RBFNN-based (d)), velocity (BPNN-based (e), and RBFNN-based (f)), altitude (BPNN-based (g), and RBFNN-based (h)), and heading (BPNN-based (i), and RBFNN-based (j)) PID gains with 0 *rad* deflection in the presence of microburst windshear

4.4.4.3 Elevator Failure - 50% Hardover in the presence of Microburst Windshear

The simulation was run with a more severe failure of 50% hardover elevator failure. Figure 4.97 shows the time history of the pitch, roll and heading angles. The response is similar to the 0 *rad* failure case with a sharp peak being added to the pitch angle response at failure.

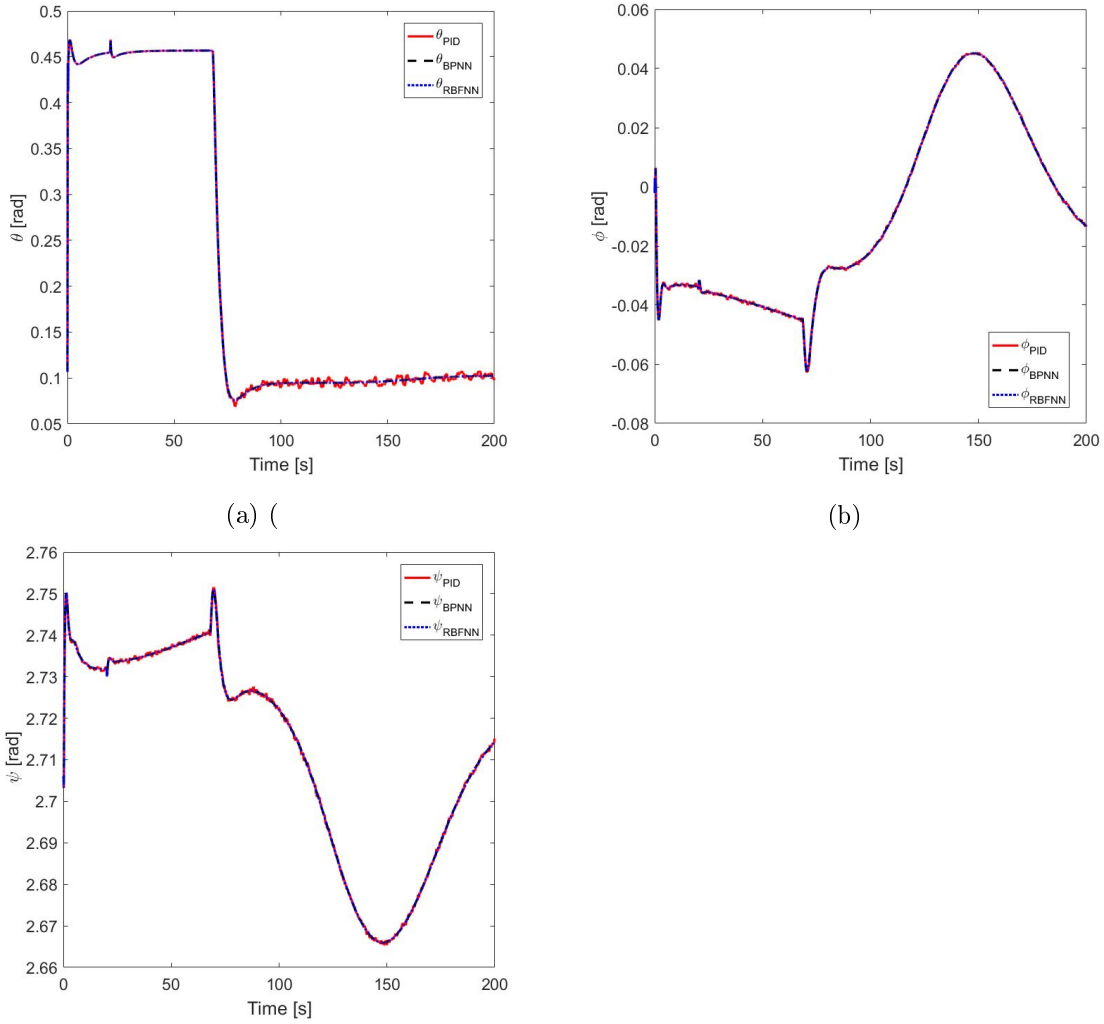
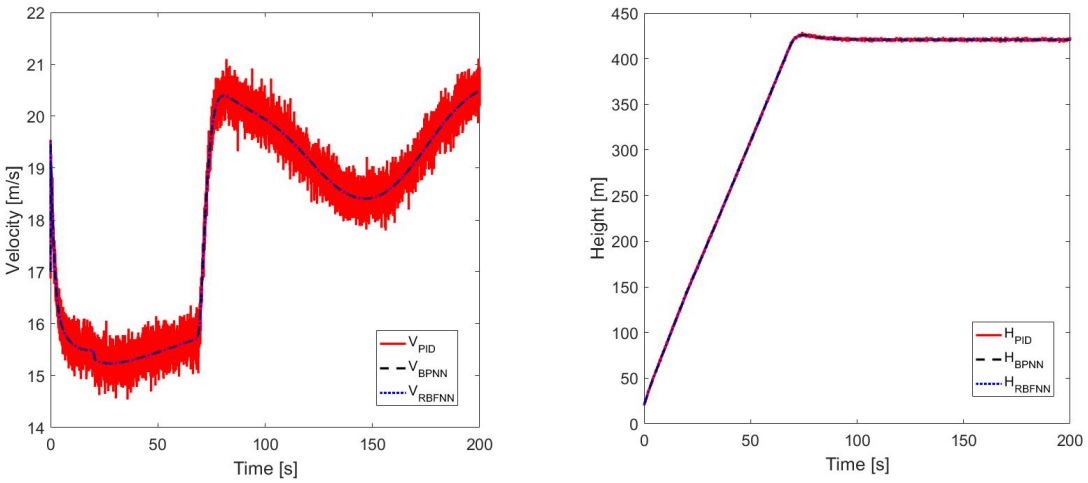


Figure 4.97: Simulated time history of middle loop pitch (a), bank (b) and heading (c) angles (50% hardover elevator failure) in the presence of microburst windshear

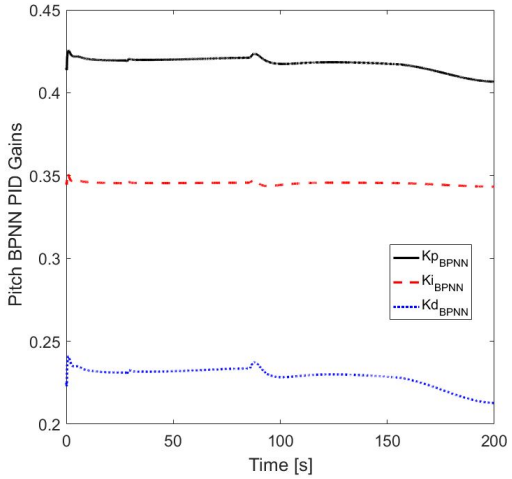
Figure 4.98 shows the time history of the airspeed and altitude. Figure 4.98 (a) shows a slight drop in the airspeed at when failure occurs which quickly returns back to a trend similar to that of the fault-free case. Figure 4.98 (b) shows no change in the altitude response.



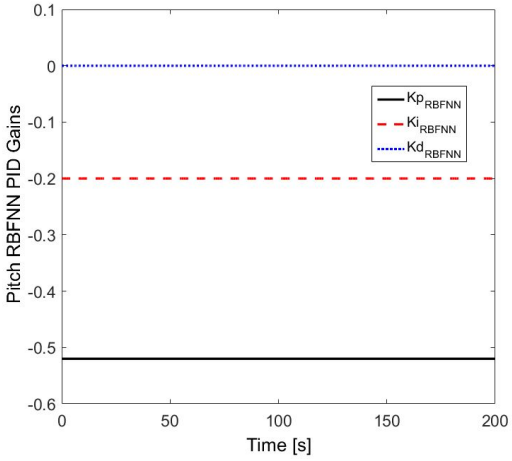
(a) (b)

Figure 4.98: Simulated time history of the airspeed (a), and altitude (b) (50% hardover elevator failure) in the presence of microburst windshear

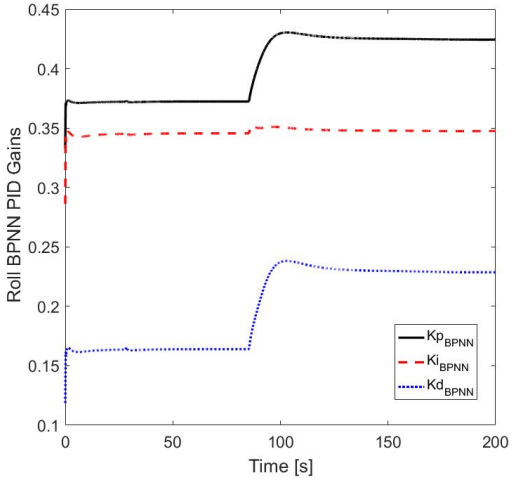
Figure 4.99 shows the variation in PID controllers' gains for all control loops. The pitch gains (see Figure 4.99 (a) and (b)) are similar to those of the case with 0 *rad* failure with the difference being a slight disturbance, which occurs in the BPNN-based controller gains when failure occurs. The roll gains (see Figures 4.99 (c) and (d)) show no change for the RBFNN-based case while the BPNN-based controller gains show a slight jump when failure occurs. The velocity and altitude gains (see Figures 4.99 (e) to (h)) remain constant during the simulation. The heading gains (see Figures 4.99 (i) to (j)) follow the same trends as the no-fault case with a slight jump when failure occurs as with the pitch gains.



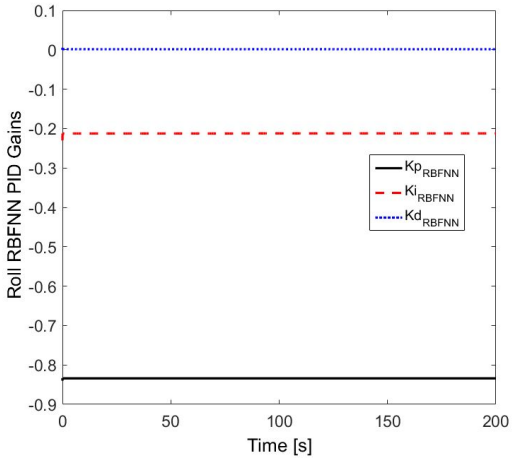
(a)



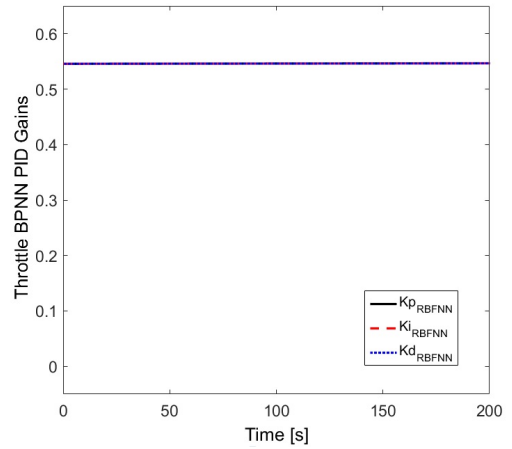
(b)



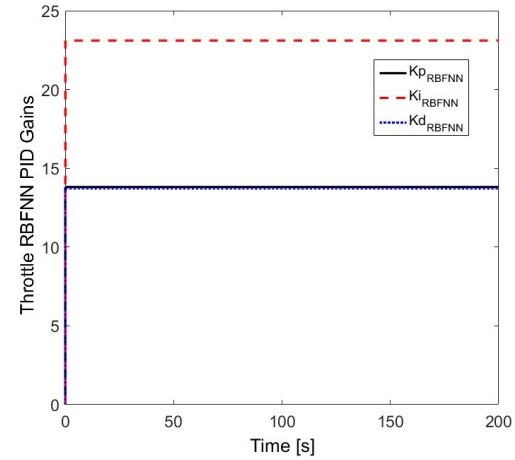
(c)



(d)



(e)



(f)

Figures 4.100 (a) to (f) show the time history of the control deflections during the simulation. It shows a sharp drop in aileron and flap deflection as the required control input is reallocated when failure occurs, this drops again when the setpoint changes. The effect of microburst

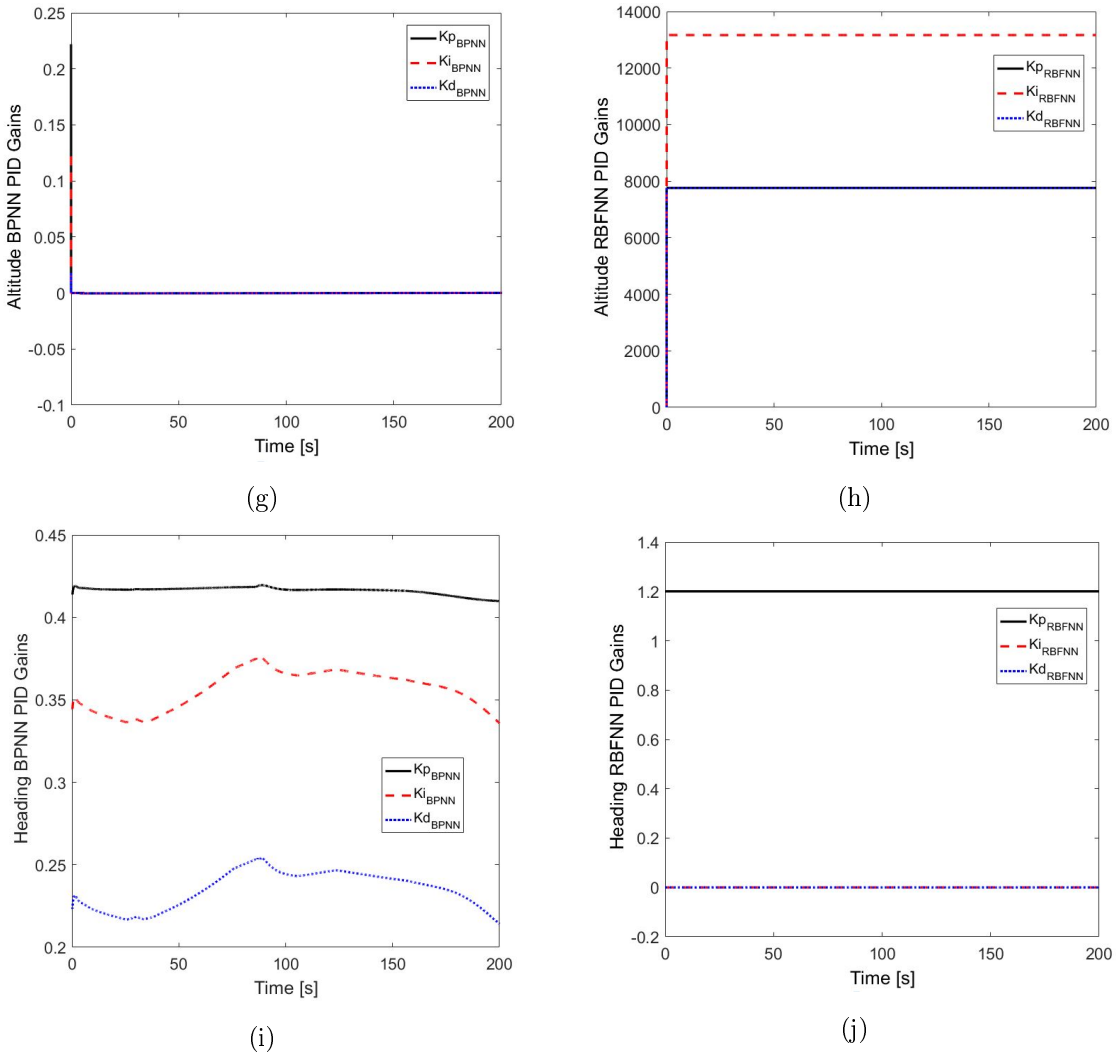


Figure 4.99: Simulated time history of the pitch (BPNN-based (a), and RBFNN-based (b)), and roll (BPNN-based (c), and RBFNN-based (d)), velocity (BPNN-based (e), and RBFNN-based (f)), altitude (BPNN-based (g), and RBFNN-based (h)), and heading (BPNN-based (i), and RBFNN-based (j)) PID gains with 50% hardover elevator failure in the presence of microburst windshear

windshear is also illustrated in the shape of these deflection curves. The noisy control surface deflections are due to controller issues with the noise modelled in the sensors.

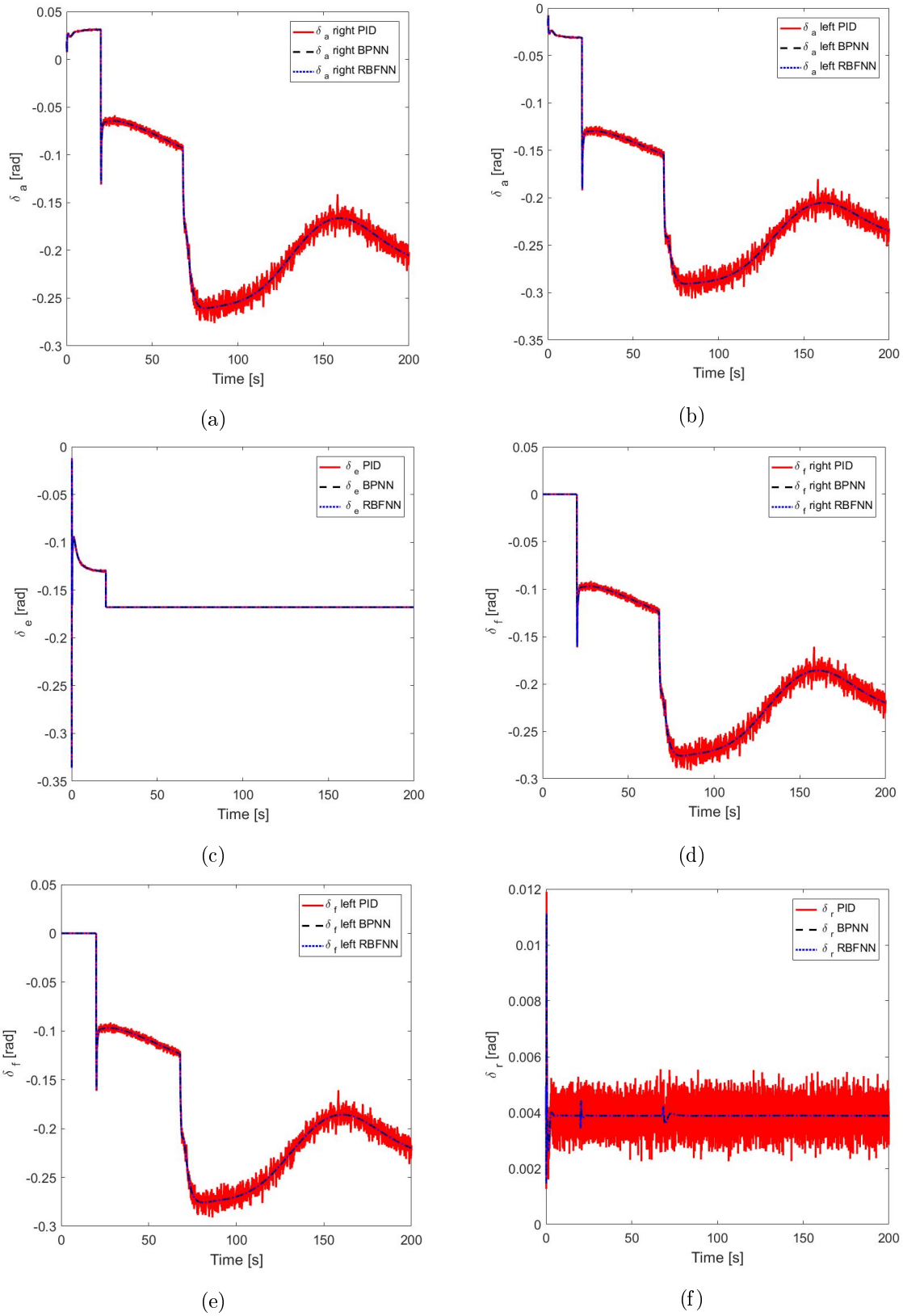


Figure 4.100: Simulated time history of right aileron (a), left aileron (b), elevator (c), right flap (d), left flap (e), and rudder (f) deflections (50% hardover elevator failure) with microburst windshear

4.4.4.4 Right Aileron Failure - *0 rad* Deflection in the presence of Microburst Windshear

The simulation was again run with a 0 *rad* aileron deflection failure at 20 seconds after the start of the simulation. The angular responses are shown in Figure 4.101. There is a slight change in the pitch angle and not a really discernible change in the heading angle at failure. The bank angle experiences a slight increase when failure occurs.

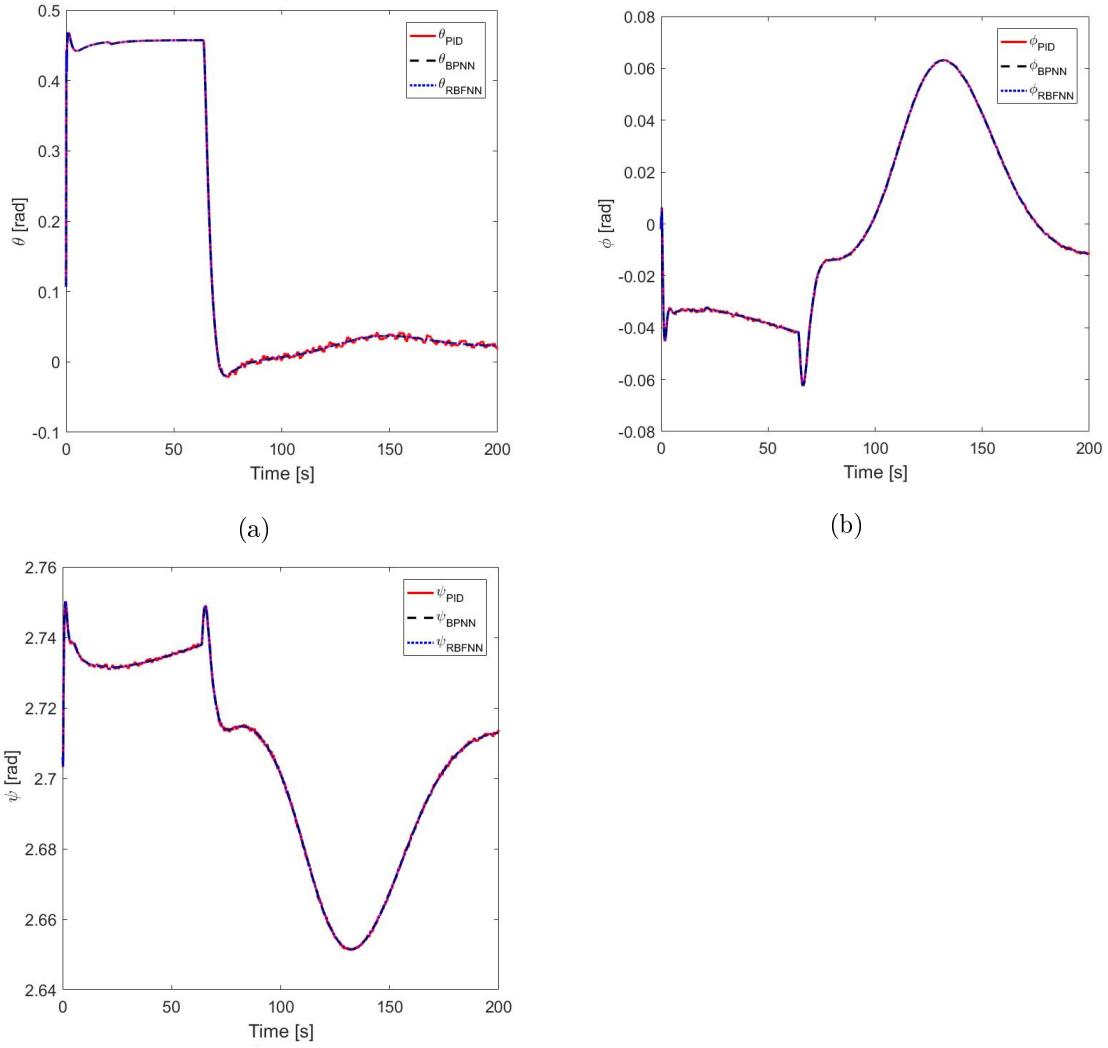
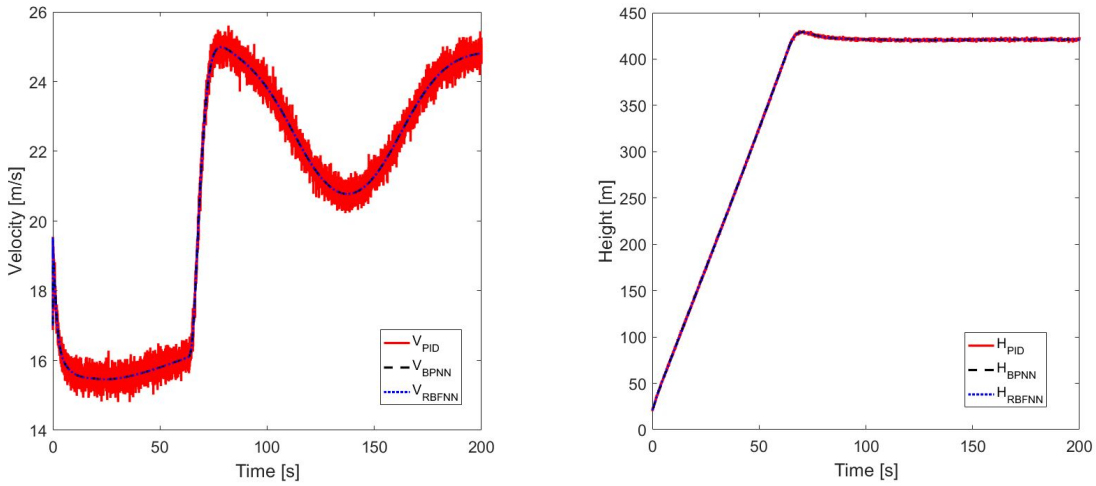


Figure 4.101: Simulated time history of middle loop pitch (a), bank (b) and heading (c) angles (0 *rad* right aileron deflection failure) in the presence of microburst windshear

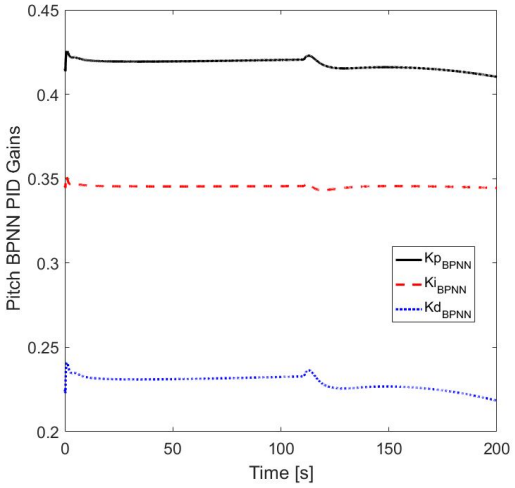
Figure 4.102 shows the time history of the airspeed and altitude. The is no noticeable difference in the airspeed and altitude responses when compared to the no-fault plots.



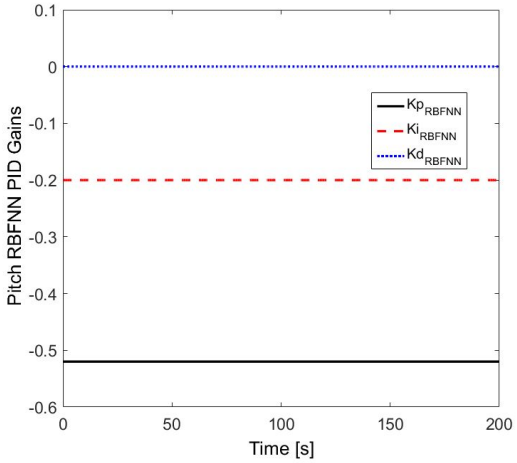
(a) (b) (b)

Figure 4.102: Simulated time history of the airspeed (a), and altitude (b) (0 *rad* right aileron deflection failure) in the presence of microburst windshear

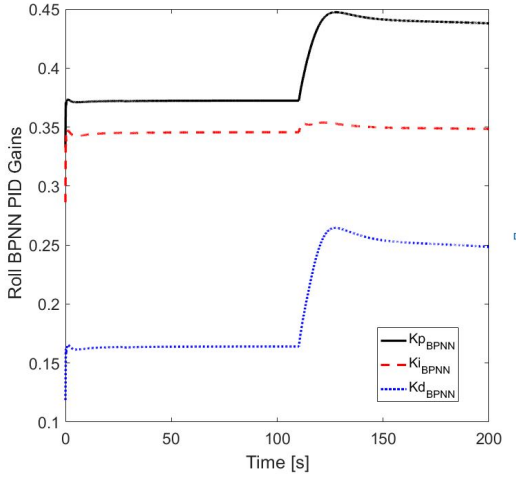
Figure 4.103 shows the variation in PID controllers' gains for all control loops. There is not really discernible change in the gains observed at failure.


Pitch BPNN PID Gains
Time [s]
Kp BPNN
Ki BPNN
Kd BPNN

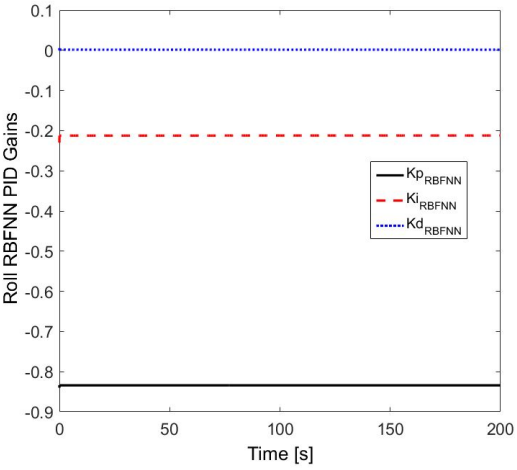
(a)


Pitch RBFNN PID Gains
Time [s]
Kp RBFNN
Ki RBFNN
Kd RBFNN

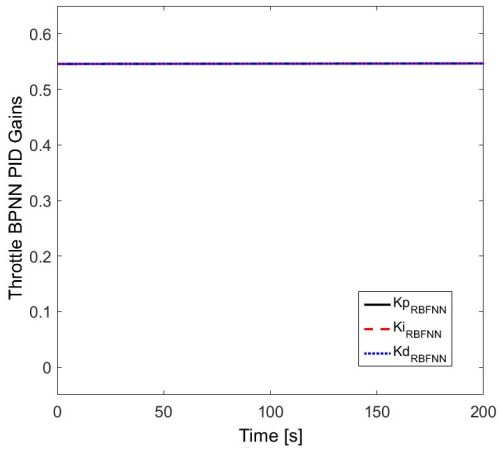
(b)


Roll BPNN PID Gains
Time [s]
Kp BPNN
Ki BPNN
Kd BPNN

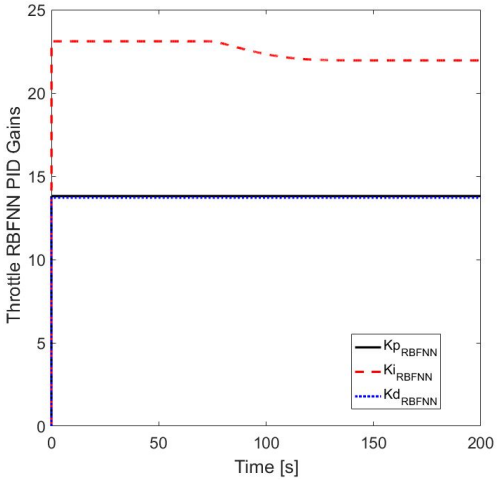
(c)


Roll RBFNN PID Gains
Time [s]
Kp RBFNN
Ki RBFNN
Kd RBFNN

(d)


Throttle BPNN PID Gains
Time [s]
Kp RBFNN
Ki RBFNN
Kd RBFNN

(e)


Throttle RBFNN PID Gains
Time [s]
Kp RBFNN
Ki RBFNN
Kd RBFNN

(f)

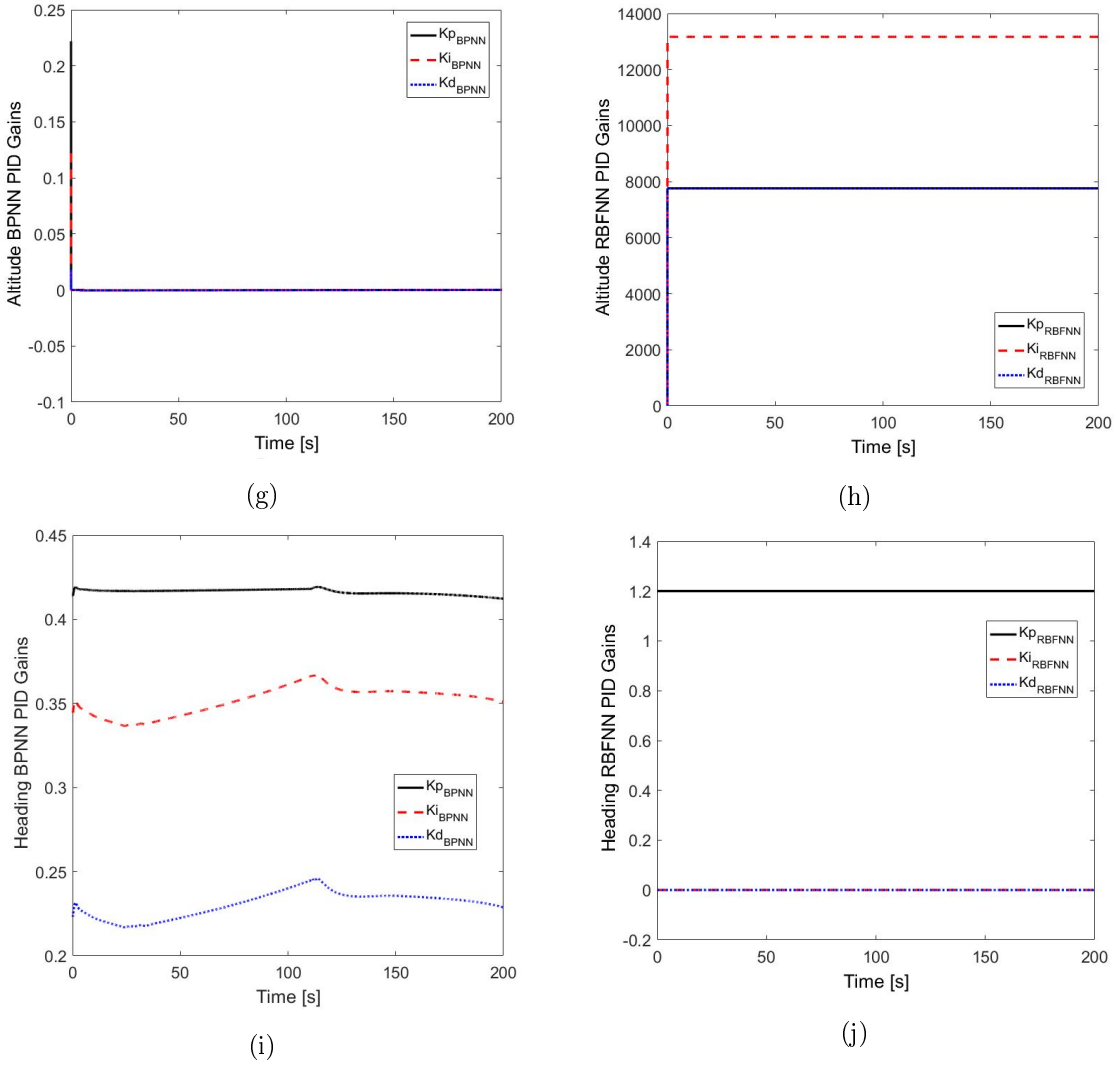


Figure 4.103: Simulated time history of the pitch (BPNN-based (a), and RBFNN-based (b)), and roll (BPNN-based (c), and RBFNN-based (d)), velocity (BPNN-based (e), and RBFNN-based (f)), altitude (BPNN-based (g), and RBFNN-based (h)), and heading (BPNN-based (i), and RBFNN-based (j)) PID gains with 0 *rad* right aileron deflection failure in the presence of microburst windshear

Figures 4.104 (a) to (f) show the time history of the control deflections during the simulation. They show the control reallocations at failure and effects of the change in setpoint and microburst windshear.

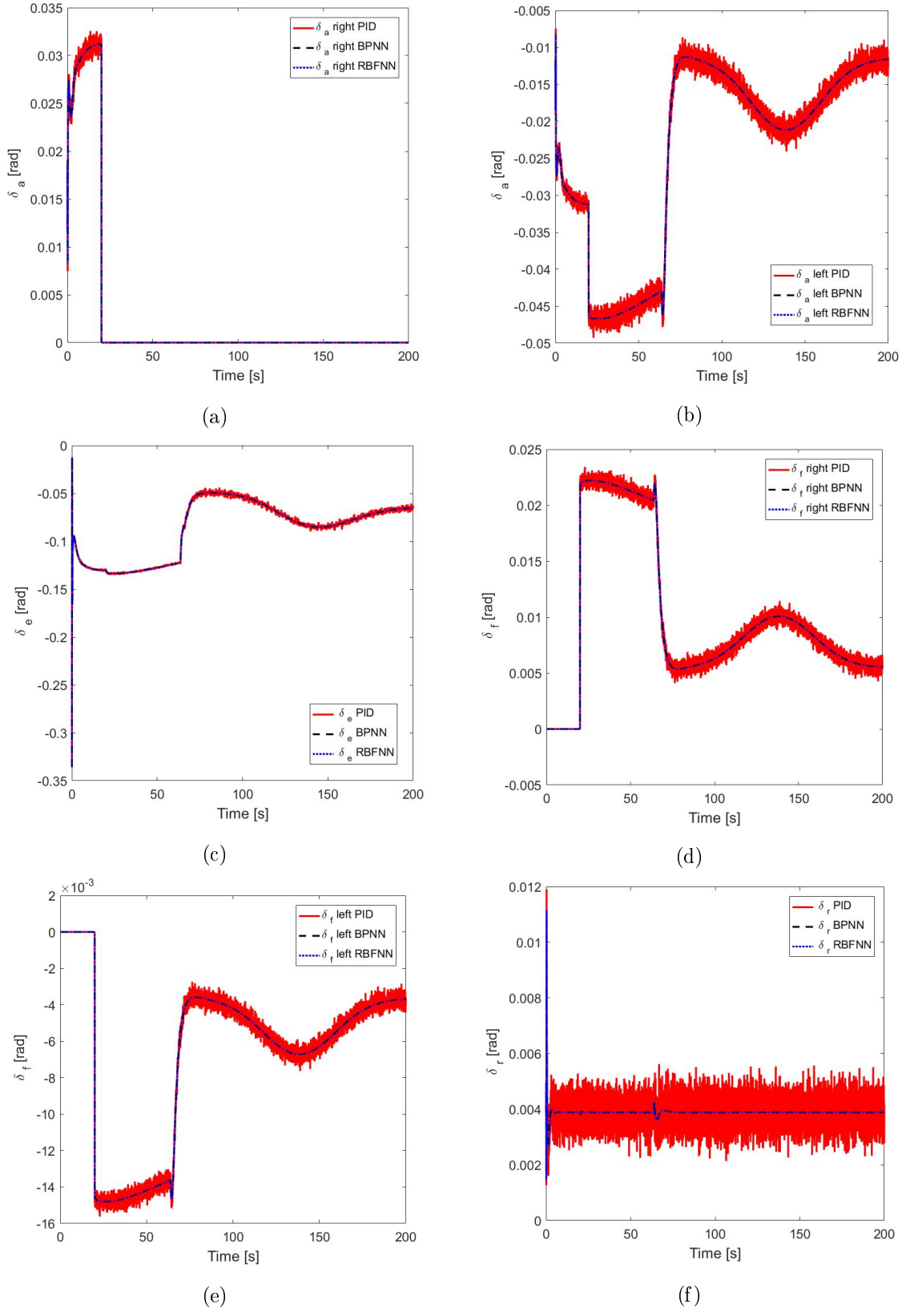


Figure 4.104: Simulated time history of right aileron (a), left aileron (b), elevator (c), right flap (d), left flap (e), and rudder (f) deflections (0 *rad* right aileron deflection failure) in the presence of microburst windshear

4.4.4.5 Right Aileron Failure - 50% Hardover in the presence Microburst Windshear

A 50% hardover failure of the right aileron was modelled, the results of which can be found in Figure 4.105. The CA algorithm was not enabled for the NN-based controller cases. There is a peak in the pitch angle (see Figure 4.105 (a)) as well as a shift of the plot shown in the response of the NN-based cases. The heading angle for the NN-based controller cases shows a sharper drop compared to the PID-based controller case. Some bit of oscillation is observed in the bank angle (Figure 4.55 (b)) with the peaks of the NN-based controller cases being of a higher magnitude.

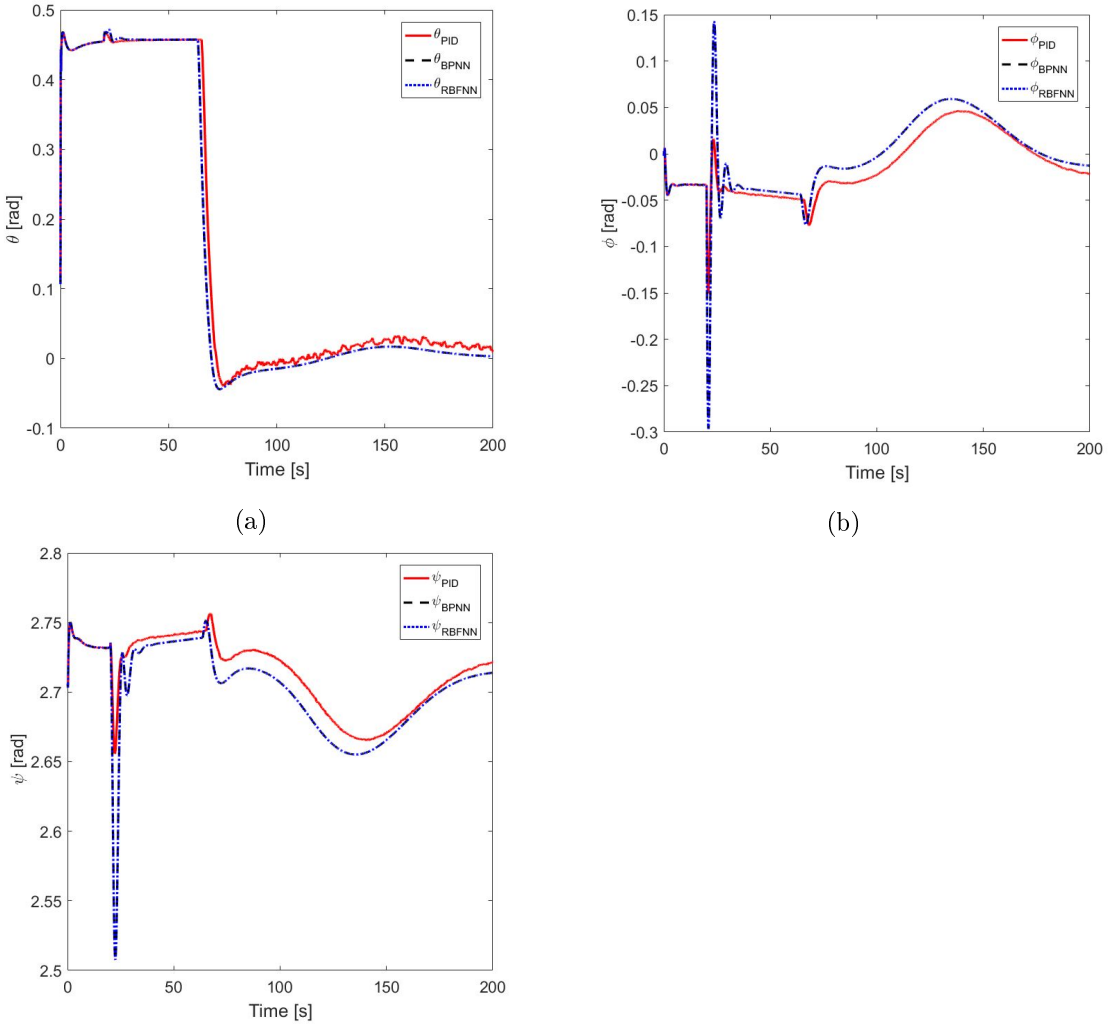
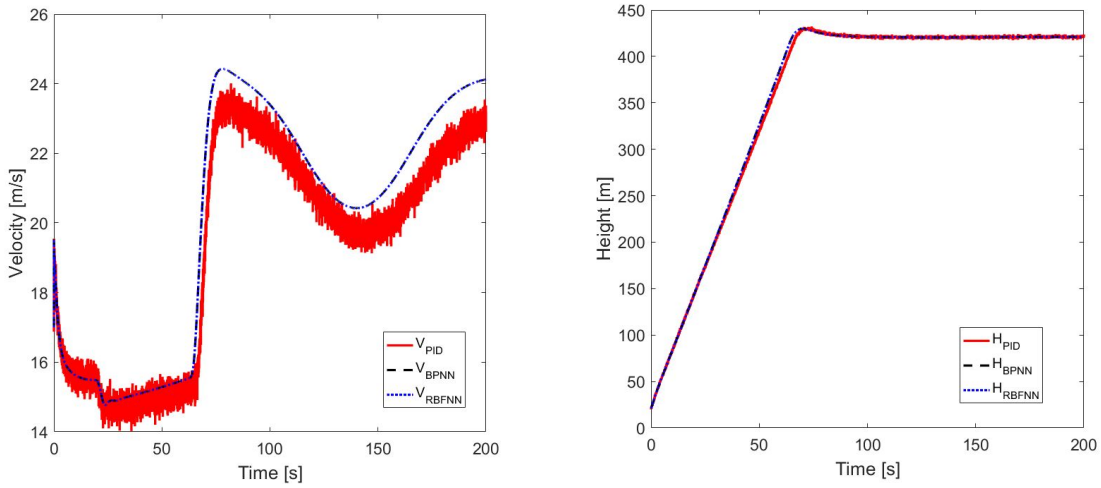


Figure 4.105: Simulated time history of middle loop pitch (a), bank (b) and heading (c) angles (50% hardover right aileron failure) in the presence of microburst windshear

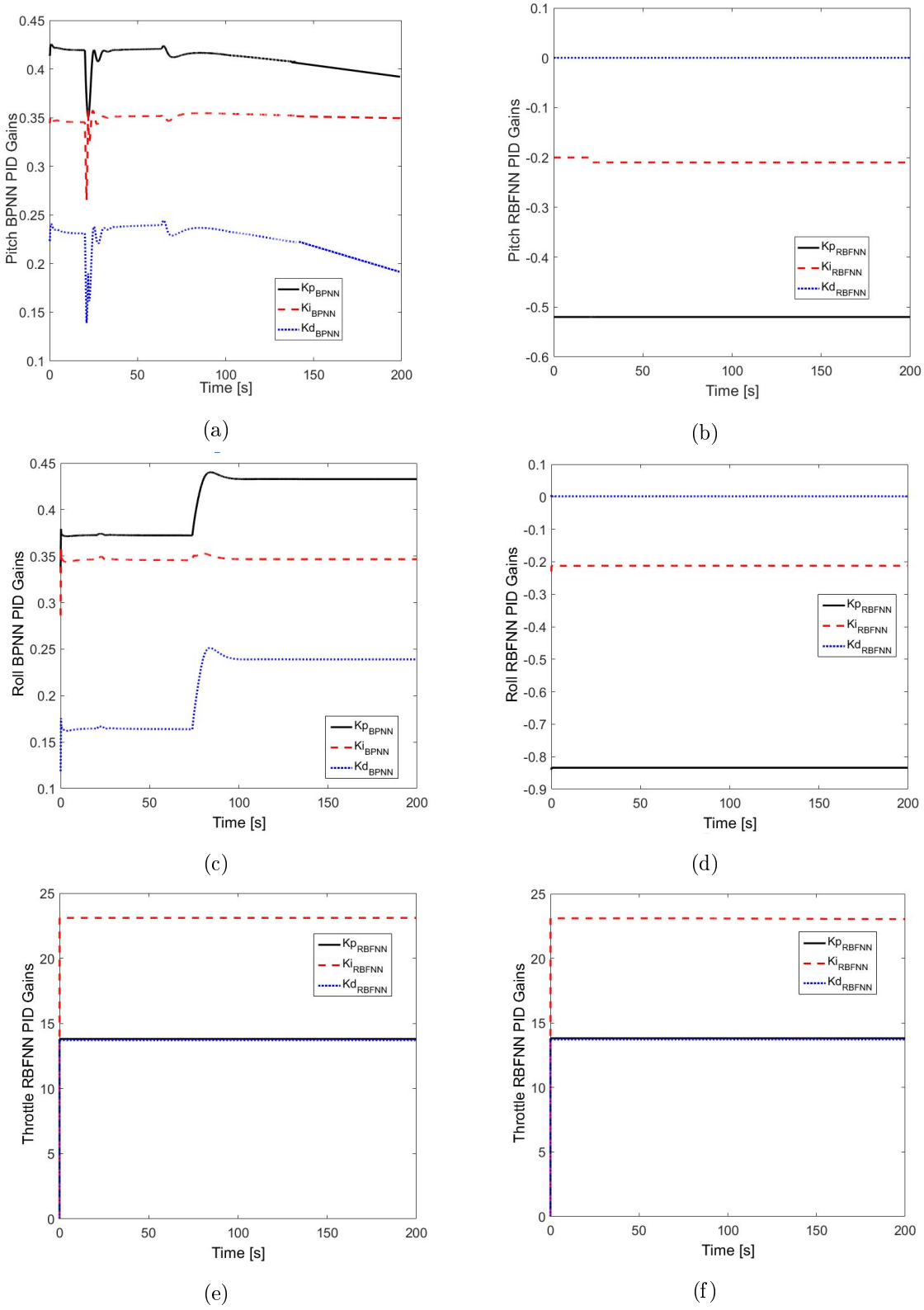
Figure 4.106 shows the time history of the airspeed and altitude. There is a drop in the airspeed at failure and in the second half of the altitude in the NN-based controller cases reaches its maximum before the PID-based controller case.



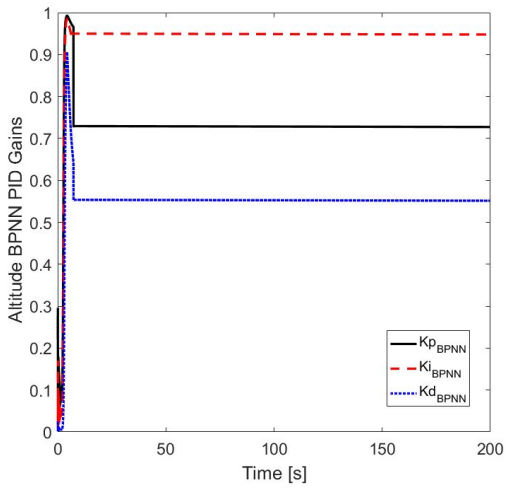
(a) (b)

Figure 4.106: Simulated time history of the airspeed (a), and altitude (b) (50% hardover right aileron failure) in the presence microburst windshear

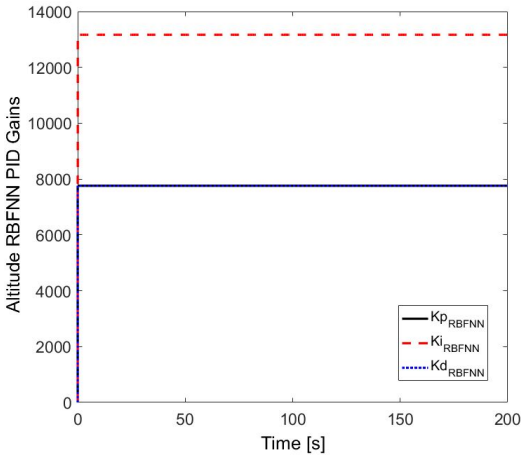
Figure 4.107 shows the variation in PID controllers' gains for all control loops. There is a drop then rapid increase in the BPNN-based controller pitch and heading gains when failure occurs, while the roll gains experiences a slight jump as with the simulations without the microburst windshear. The altitude gains of the BPNN-based controller shower some disturbance at the start of the simulation and then remain constant for the remainder of the simulation. Majority of the RBFNN-based controller gains remains constant throughout the simulation.



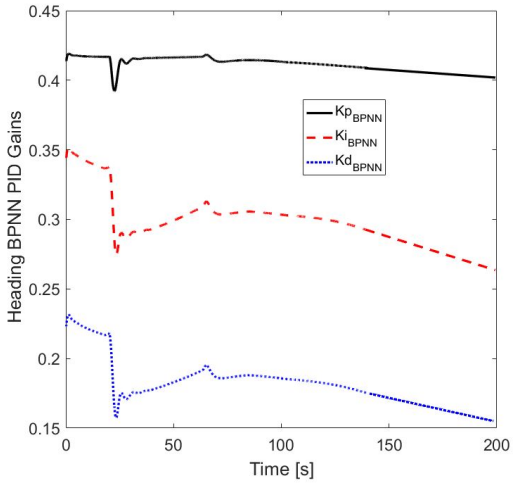
Figures 4.108 (a) to (f) show the time history of the control deflections during the simulation. The left aileron is used more in the NN-based cases. The flaps are not deflected for both the NN-based cases. There elevator is deflected slightly less for the PID-based cases.



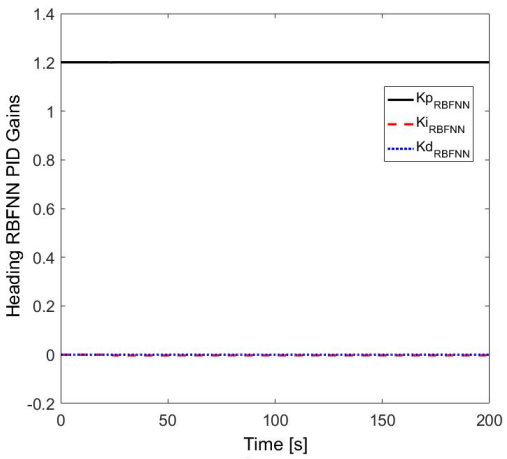
(g)



(h)



(i)



(j)

Figure 4.107: Simulated time history of the pitch (BPNN-based (a), and RBFNN-based (b)), and roll (BPNN-based (c), and RBFNN-based (d)), velocity (BPNN-based (e), and RBFNN-based (f)), altitude (BPNN-based (g), and RBFNN-based (h)), and heading (BPNN-based (i), and RBFNN-based (j)) PID gains with 50% hardover right aileron failure in the presence of microburst windshear

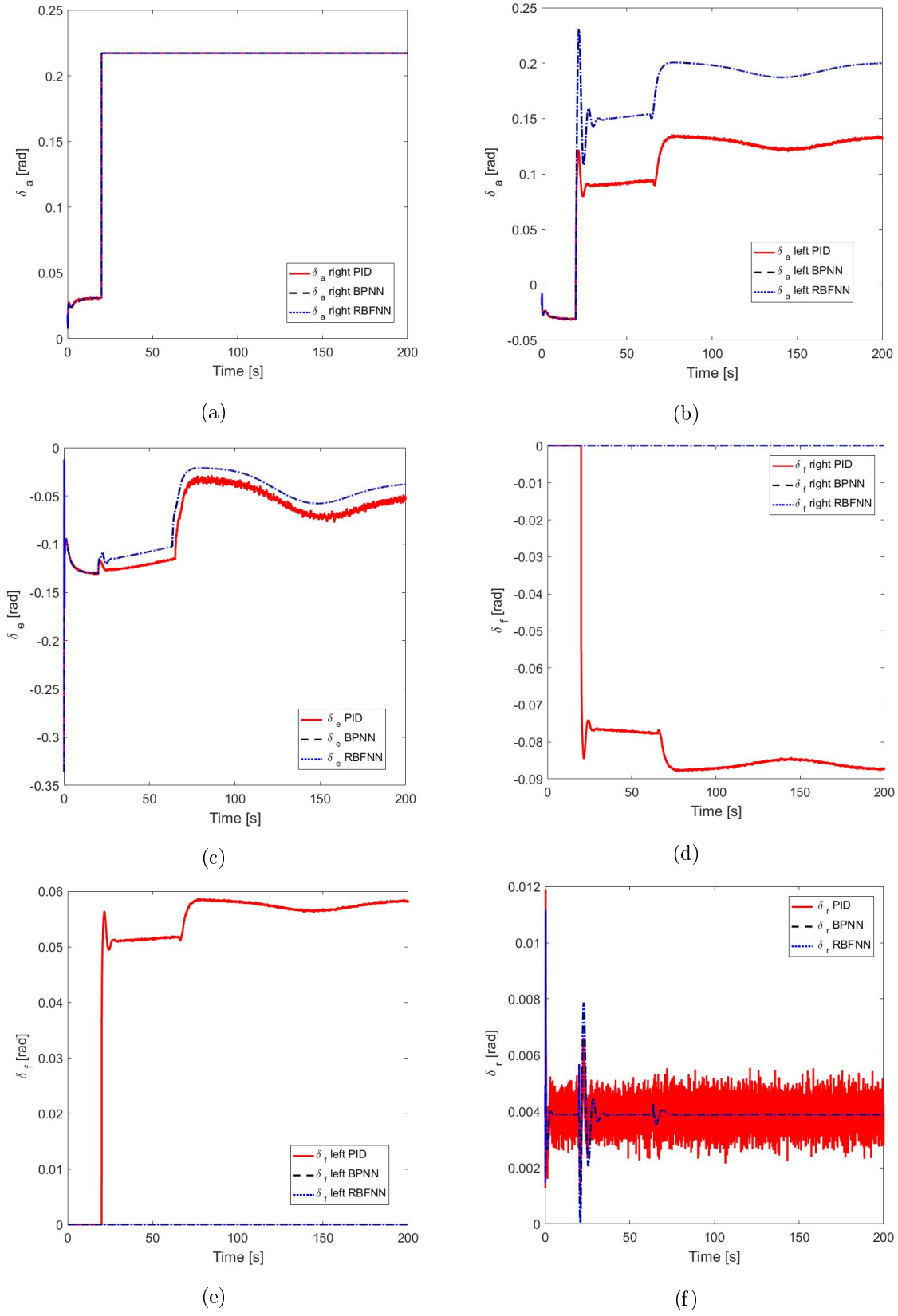


Figure 4.108: Simulated time history of right aileron (a), left aileron (b), elevator (c), right flap (d), left flap (e), and rudder (f) deflections (50% hardover right aileron failure) in the presence of microburst windshear

4.4.4.6 Right Aileron Failure - 100% Hardover in the presence Microburst Windshear

A 100% hardover failure of the right aileron in the presence of microburst windshear was modelled, the results of which can be found in Figure 4.109. There is a peak in the pitch angle at failure. For the bank angle there is a sharp dip followed by a peak in the response. There is a sharp dip in the heading angle.

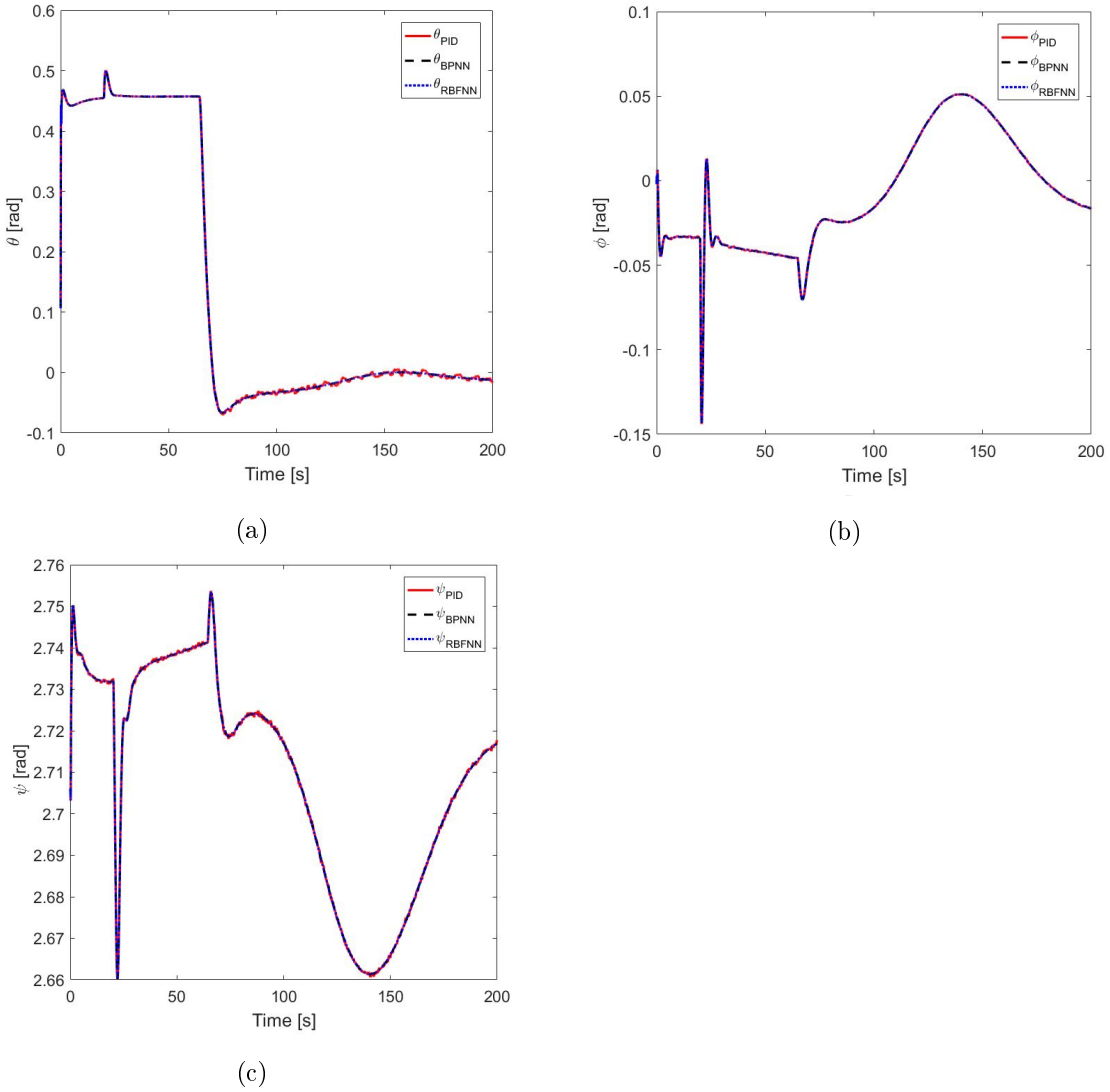
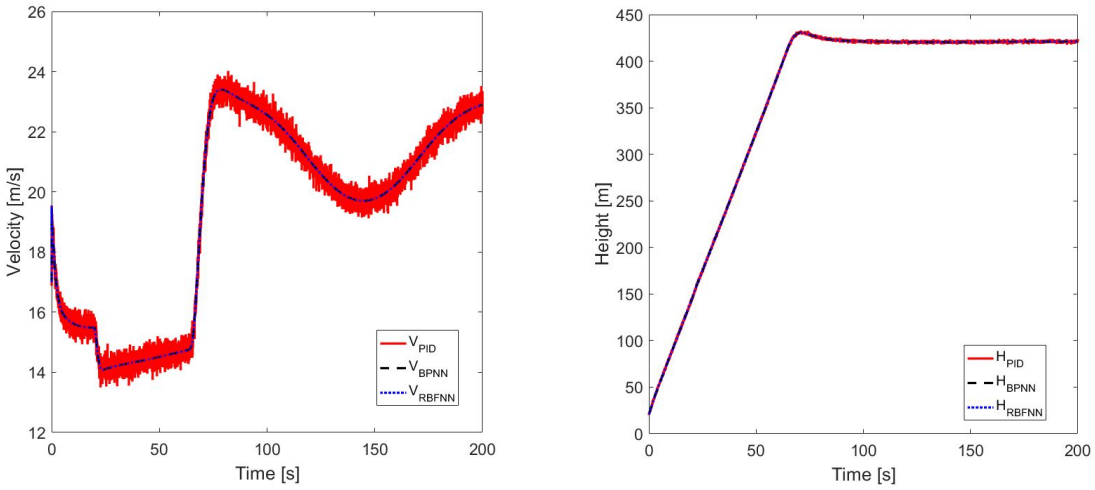


Figure 4.109: Simulated time history of middle loop pitch (a), bank (b) and heading (c) angles (100% hardover right aileron failure) n the presence of microburst windshear

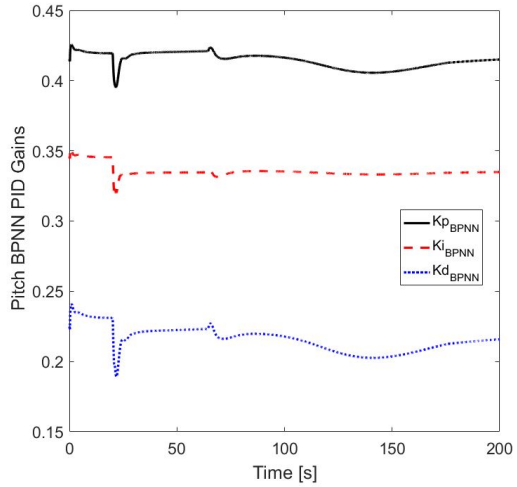
Figure 4.110 shows the time history of the airspeed and altitude. There is a drop in the airspeed at failure while the altitude performance remains the same.



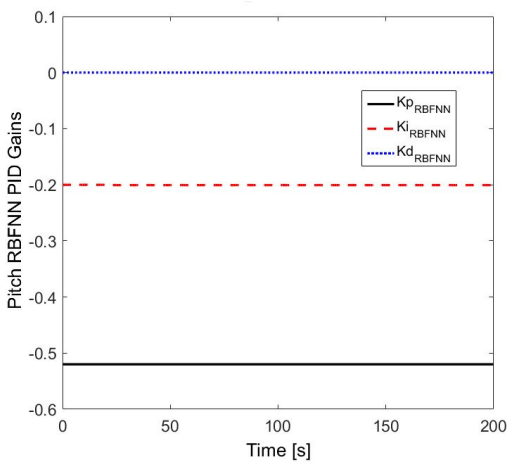
(a) (b)

Figure 4.110: Simulated time history of the airspeed (a), and altitude (b) (100% hardover right aileron failure) in the presence of microburst windshear

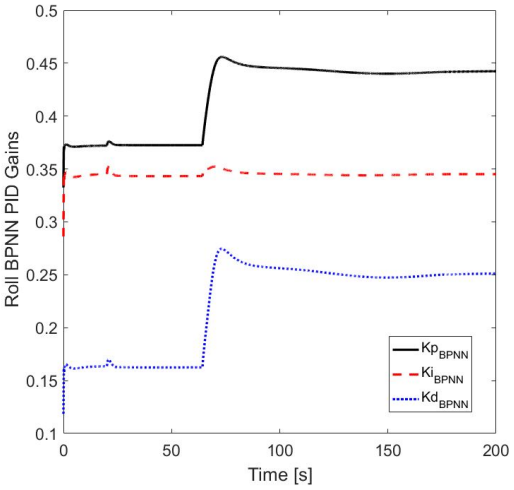
Figure 4.111 shows the variation in PID controllers' gains for all control loops. The BPNN-based controller pitch gains show a rapid drop then increase when failure occurs while the RBFNN-based controller gains remain constant. The roll, velocity and altitude gains are similar to that of 50% hardover aileron failure simulation. The BPNN-based controller heading gains show a sharp drop with the dip after failure.



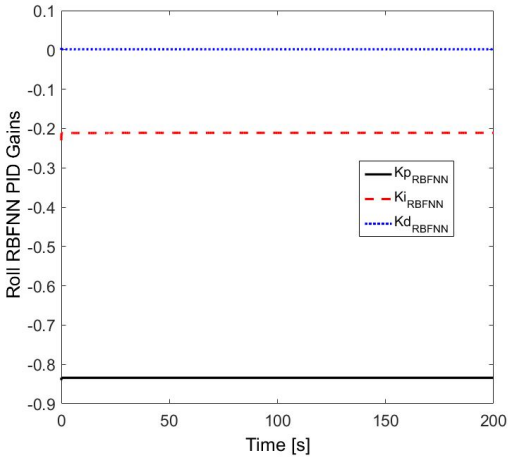
(a)



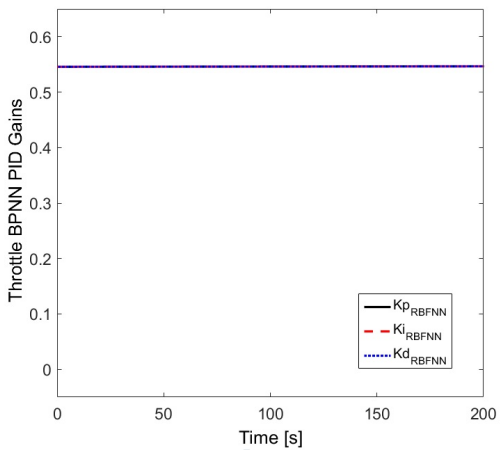
(b)



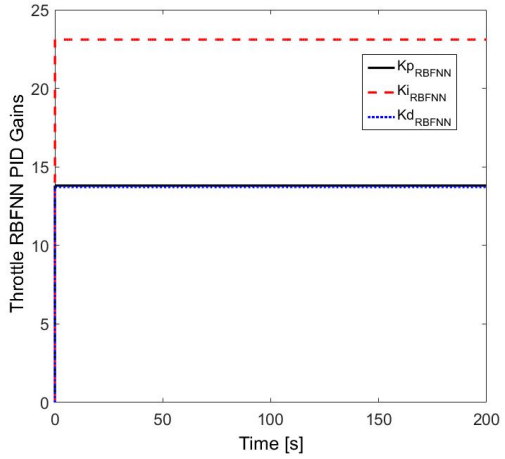
(c)



(d)

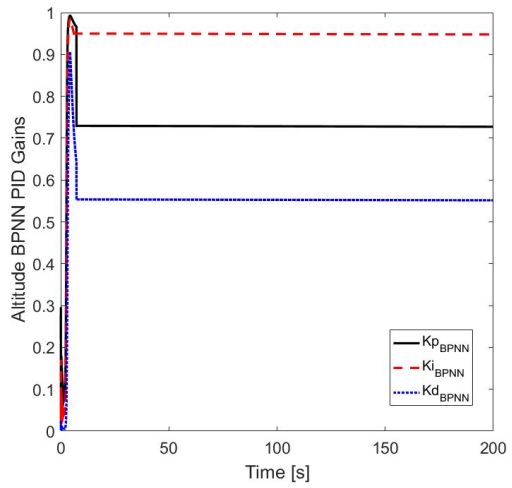


(e)

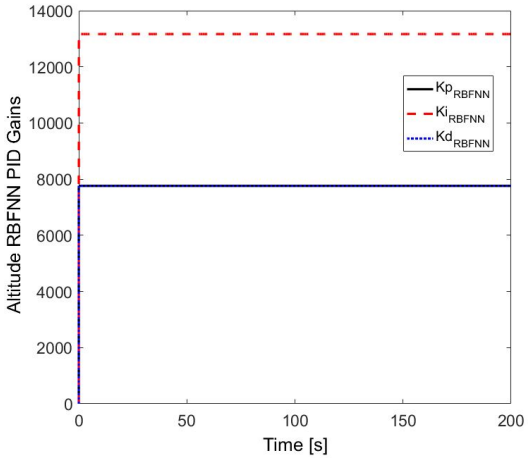


(f)

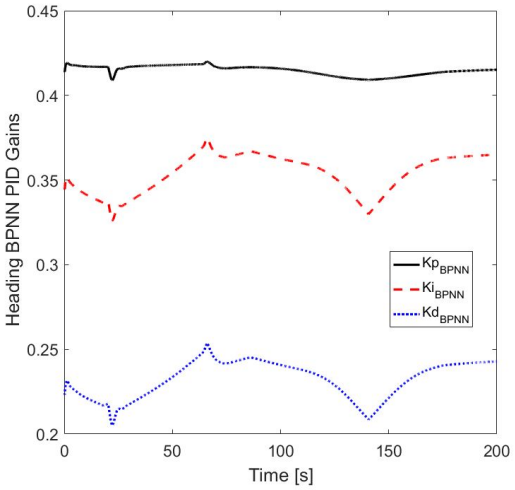
Figures 4.112 (a) to (d) show the time history of the control deflections during the simulation. Figure 4.112 (a) shows a rapid increase in the left aileron deflection which then decreases once stability is achieved. It can be observed that the deflection of the healthy control surfaces are



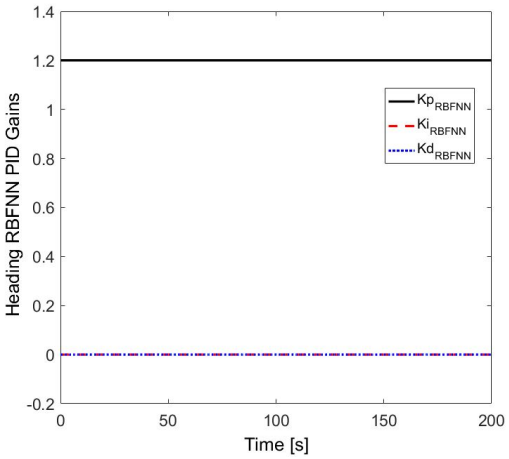
(g)



(h)



(i)



(j)

Figure 4.111: Simulated time history of the pitch (BPNN-based (a), and RBFNN-based (b)), and roll (BPNN-based (c), and RBFNN-based (d)), velocity (BPNN-based (e), and RBFNN-based (f)), altitude (BPNN-based (g), and RBFNN-based (h)), and heading (BPNN-based (i), and RBFNN-based (j)) PID gains with 100% hardover right aileron failure in the presence of microburst windshear

well within their limits, making the controller well suited to this type of failure case. There is also a rapid increase in elevator deflection (see Figure 4.112 (c)). There is a spike in the flap deflections at failure as observed in previous simulations (see Figures 4.112 (d) and (e)).

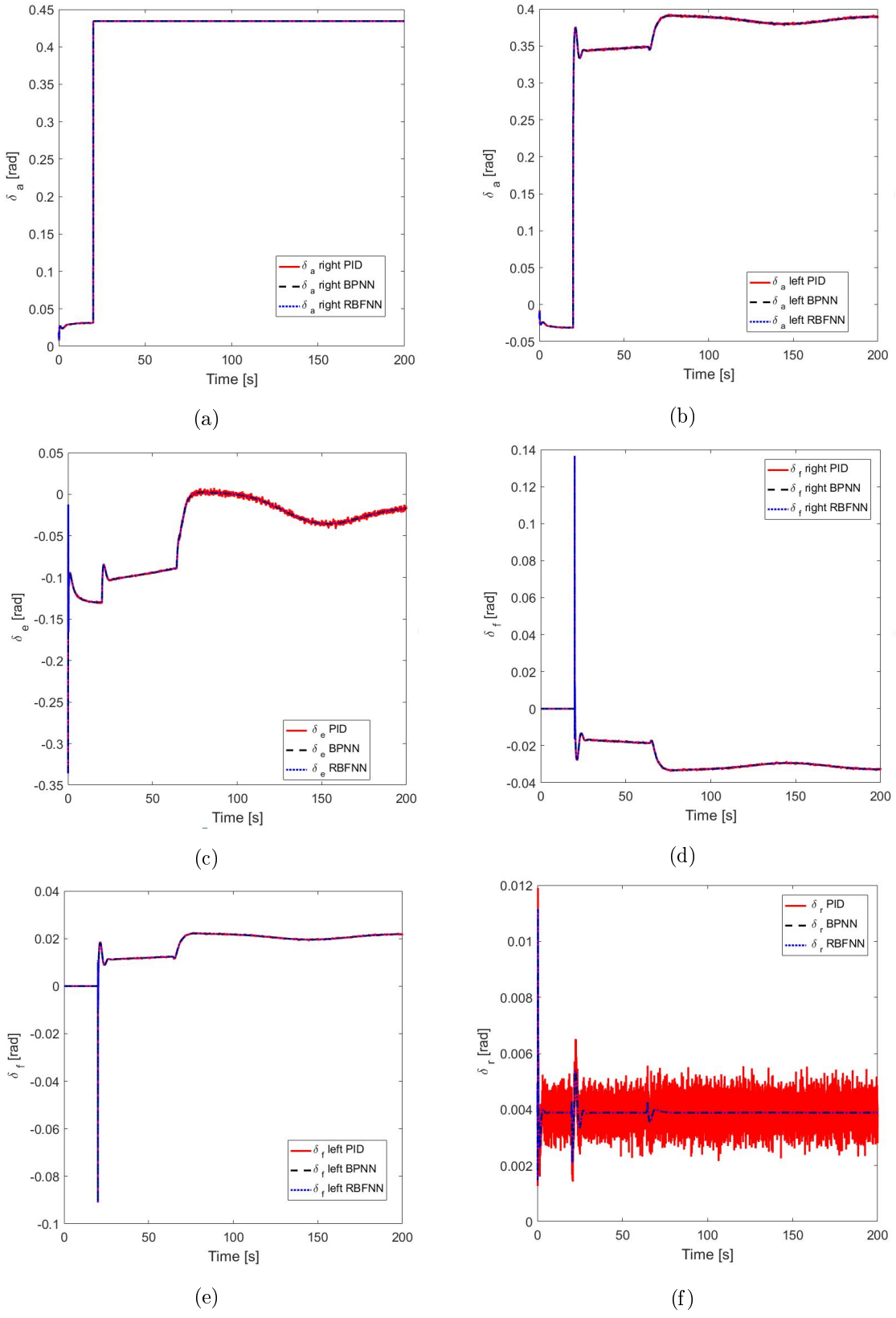


Figure 4.112: Simulated time history of right aileron (a), left aileron (b), elevator (c), right flap (d), left flap (e), and rudder (f) deflections (100% hardover right aileron failure) in the presence of microburst windshear

5 Analysis of Results

5.1 Chapter Outline

This chapter presents a review of the objectives of the research as well as the results of Chapter 4. The success or failure of the proposed fault-tolerant control strategy is discussed and conclusions on the suitability of the proposed fault-tolerant control system are drawn. Finally, the recommendations for future work on this topic are presented.

5.2 Research Objectives

The primary objective of this research, as outlined in Chapter 1.3, was to research, design and simulate a fault-tolerant control system, while making use of artificial intelligence techniques.

This objective was expanded into the following detailed objectives:

- actuator malfunction;
- partial loss of control surface;
- structural damage; and
- aircraft model uncertainties.

The simulated tests concentrated on a number of different actuator failures. These included a 0 *rad* deflection failure, a 50% hardover failure, and a 100% hardover failure. Both the left and right elevators, as well as the right aileron were individually faulty. Simulations where combined aileron and elevator failure occurred simultaneously were also conducted.

5.3 Inner Loop Simulations

The inner loop simulations provided valuable insight into the feasibility of the neural network in providing a measure of fault tolerance. The failures were tested for a commanded flight maneuver of straight and level flight. The neural networks were tested against the PID autopilot, in each test with the control allocation algorithm enabled. The effects of the microburst windshear are visible in the fault-free simulation (in particular in the pitch and bank angle plots). Without the microburst windshear, the system response is constant throughout the simulation as compared to the fault-free simulation in the presence of microburst windshear, which shows a dip (the angle temporarily decreases due to the microburst and then increases again) in the response of these angles as expected. This effect is also observed in the control surface deflection and control input plots.

5.3.1 Elevator Failures

The first failure tested was a 0 *rad* deflection failure, which generated a nose-up pitching moment. This failure showed that the autopilot, in its current form and without neural network supervision, is tolerant to this type of failure. There was no noticeable difference in performance with the neural network supervision. The response of the system in the presence of microburst windshear is similar to that of the system without microburst windshear at failure, but with the inclusion of the disturbance caused by the microburst windshear.

The 50% deflection failure showed a marked increase in pitch and roll error even though it was demonstrated that the unmodified autopilot was unable to tolerate this failure. It was noted that there was a significant improvement in the control performance with the neural network supervision. A significant improvement in control performance was observed in the roll response while the improvements were observed in the pitch response with the system reaching a stable pitch angle. Little difference was observed between the two NN-based controllers (BPNN and RBFNN). For the system simulated in the presence of microburst windshear with PID-based and NN-based controllers, the response was found to be similar between the three cases in the middle loop simulations. There was a small disturbance at failure which gradually returns to the setpoint. The limitations of the PID-based controller in the presence of microburst windshear were shown here as the PID-based controller was able to return to the desired setpoint but the effects of failure were not mitigated fast enough. In the case where CA was not enabled it was found that the BPNN-based controlled had a tendency to guide the UAV away from the microburst centre rather than descend into it.

The 100% elevator failure, which was expected to induce difficulties for the autopilot, was not effectively controlled. This was observed in the autopilot with and without neural network supervision was unable to maintain the desired setpoint for the pitch angle. The PID controller responded better in pitch, but worse in roll. Significant improvement in the neural network's performance was observed in the roll angle, while larger oscillations were induced in the pitch angle.

Of note, was the sharp spike in derivative BPNN gain and the RBFNN integral gain for the 100% elevator failure case as the neural network attempted to regain control. This presents a potential problem with the neural network. Should a control surface reach its physical limits and saturate (as observed with the pitch control input), the neural network has no mechanism to detect or resolve the resulting lack of control authority. This was an unintended consequence that developed, which has the potential to cause a catastrophic failure of the system. The loss of control authority is highly undesirable as the purpose of neural network supervision is to improve fault tolerance and not to induce further difficulties. This result highlights the need for the development of a simple control allocation technique to be used in conjunction with the neural network supervision.

The simulations showed that, for the inner control loop, the proposed control methodology of combining the neural network supervision with the control allocation yielded a successful tolerance against for neutral and small to medium deflection failure as well as combined failures, but was unable to deal effectively deal will full elevator hardover failures.

5.3.2 Aileron Failures

The first aileron failure to be tested was the 0 *rad* deflection failure of the right aileron, which yielded only a small roll disturbance. Because the disturbance was so small, only a small opportunity existed for the neural network supervision to make changes to the PID gains. As a result, only a small difference was noted in bank and pitch angle control performance for level flight. The same was observed for the system in the presence of microburst windshear as there was very little difference observed between this and the fault-free simulation for microburst windshear.

The more aggressive 50% hardover failure simulation yielded significant results. There was a large roll disturbance that was adequately controlled by the PID autopilot. No significant difference in roll angle was found between the PID-based and NN-based controllers. The maximum pitch error was reduced with the neural network supervision. When failure occurred a far more visible change was observed in the BPNN-based controller gains. The response of the system in the presence of microburst windshear was similar to that of the system without

it at failure with the addition of microburst windshear effects. The sharp dip in the bank angle response was of a lower magnitude for the simulation with microburst windshear.

The 100% hardover failure of the right aileron yielded an interesting set of results. It was thought that this failure, similar to elevator 100% hardover failure case, would result in the failure of the autopilot to maintain the set pitch and bank angles. However, this was not the case. The autopilot, without neural network supervision, was able to level the wings of the aircraft with the aid of the rolling moment supplied by the rudder. This rudder input was generated by a feedforward loop commonly used on remotely piloted aircraft. The same success was found with the NN-based controllers. A large decrease was observed in the maximum error of the pitch angle. The trend of the response of the system in the presence of microburst windshear is similar to that previously observed without microburst windshear at failure with the inclusion of microburst windshear effects in the response plots.

5.3.3 Combined Failures

A combined failure of the aileron and elevator was performed to test the ability of the proposed control laws to tolerate a combination of control surfaces failures. The first combination of failures was a 0 rad deflection failure of both elevators and right aileron. This failure caused a larger pitch and roll error, as would be expected, than the single control surface type cases. The test showed that the autopilot, even without neural network supervision, was able to tolerate this failure; however, as in the previous cases, there was no noticeable difference in pitch and roll performance between the PID-based and NN-based controllers. In the presence of microburst windshear the system showed a different response to this error as there was a dip in the pitch response and large peak in the bank angle response, unlike previously where there was a bit of oscillation.

The 50% hardover failure of the elevator and aileron were also tested. The responses were again larger than individual control surface type failures. The autopilot without adaption was tolerant to this failure and restored the aircraft to the required setpoints in both pitch and roll. The effects of the microburst windshear were observed in the large oscillations in the system response. The pitch angle did not return to the desired setpoint though the system did stabilise. In addition to this it was observed that once this failure occurred the control system struggled to get the UAV to carry on descending both with and without the CA algorithm enabled as the UAV descended at a very slow rate and did not reach the desired end of at least $25m$ at the end of the simulation time.

Lastly, the simulation was run with a 100% hardover failure of both elevators and right aileron. This resulted in the successful return of the aircraft to the desired setpoint for the roll angle.

Though the pitch angle was stabilised, there was still a constant error of 0.2 *rad* (11.46 deg) which remained. The control allocation algorithm did, however, reach the limits of its ability to control this failure and a number of oscillations in pitch and roll angle response occurred. The flaps, being the only remaining control surfaces to provide both pitch and roll control, were unable to provide enough control authority and reached their limits of travel.

The results of the combined control failures show that, for the inner loop, the autopilot was tolerant to neutral and small to medium deflection failures and even combined failures.

5.4 Middle Loop Simulations

The middle loop simulations provided additional insight into the suitability of the proposed algorithms in the higher flight modes of the autopilot. These tests are important for two reasons: Firstly, it tests the neural network supervision to track changing setpoint values and secondly, it tests the coupling effect of the neural network supervisions.

The inner loop simulations showed that the NN-based controllers were able to track a constant setpoint with little interference from each other. In the middle loop tests, the pitch PID controller received commands from the altitude controller, both of which have their own neural network supervisions in place. These simulations provided means to determine if the coupling of the three PID-based controllers would initiate a series of instabilities or if the system would be enhanced by the neural network supervisions. Of primary importance, however, is the objective of this research in providing a system that is tolerant to various control actuation failures: hence, the middle loop simulations provided a small step forward in determining the suitability of this proposed control system. This small step allowed for a detailed examination of the airspeed and altitude controllers and allowed any final tuning of the neural network supervisions.

5.4.1 Control

The simulation was run in a combination of climb and level flight mode with no failures being induced in the system. This allowed for the confirmation of stable operation with the coupling of the various PID controllers. It also provided a baseline result with which to compare the failure test results.

It was found that there was good tracking of airspeed and altitude with the neural networks enabled, which is promising considering the coupling of the PID controllers. The results also

showed that the adaptation algorithm was able to track a moving setpoint value. There was some slight overshoot in the bank angle response when the setpoint was changed.

The effects of microburst windshear were observed in the spikes, which occurred at the start of the simulation as well as the dips observed in the responses when the setpoint is changed during the simulation.

5.4.2 Elevator Failures

The first elevator failure modelled was the 0 *rad* deflection failure. This failure had little effect on the aircraft in this flight mode, the only difference being observed in a subtle diversion of the bank angle. Little effect was observed on the airspeed and almost no effect was observed on altitude and the neural network was able to maintain pitch tracking. Thus, the improvements observed in the control were maintained and the neural network showed improved tracking of the altitude. In the presence of microburst windshear the effects of failure were more pronounced in the system response.

The 50% hardover failure of the elevators provided good results. The autopilot, without neural network supervision, showed minor changes to airspeed results when compared to the control case and the performance, while not exceptional, was adequate. The altitude controller showed good tracking performance. Based on this test, it was observed that there was very little change in the PID gains within the NN-based controller cases. There was a spike observed in the angle responses in the presence of microburst windshear. In addition, to this little change observed in the altitude response.

Following this test, the 100% hardover failure of the right elevator test was simulated. The settings determined in the 50% deflection simulations were maintained. It was observed that the controller was unable to return to the desired setpoint though the system was able to maintain stability. This can be attributed to large control deflections in the aileron and flaps, which resulted in degraded airspeed and altitude tracking performance when the setpoint was changed. It can be observed from the roll gains that there was some attempt with the neural network to apply more control when the setpoint changed, but due to the control surfaces being saturated this was not a possibility.

It can be concluded that the proposed control system is tolerant of less severe elevator failure cases.

5.4.3 Aileron Failures

The first aileron failure to be tested for the middle control loop was the 0 *rad* deflection failure test. This case showed little to no effect on the system. This was true for the system both with and without the presence of microburst windshear. The airspeed and altitude were not visibly affected by the aileron failure. This is not surprising, as only a small change in pitch angle occurs at the moment of failure, which is easily controlled by the autopilot both with and without neural network supervision. The only real evidence of failure lies in the change in bank angle. This change was similar in trend to the changes observed in the inner loop simulation, which is unsurprising as the bank angle is controlled by the inner loop with no influence by the middle control loop. Thus, with the rest of the aileron failures, a result similar to that attained in the inner loop simulation was demonstrated with there being a difference in an overshoot in the bank angle. Of note, however, was the ability of the autopilot to be tolerant to all aileron failures tested. The ability of the system to tolerate these failures is largely due to the control allocation algorithm that made use of the flaps to compensate for the reduced roll authority. Great improvement was observed with the neural network in the pitch results.

The 50% and 100% hardover failure errors were slightly increased when the neural network was enabled. In the microburst windshear case for 50% hardover failure there was a difference observed in the magnitude of the spike at failure in the bank angle response. This may be due to the NN-based system not using the flaps but rather increasing left aileron deflection to mitigate failure. This shows that the NN-based controller system does not always need to reallocate to the remaining healthy control surfaces to reduced the effects of failure. It was concluded that the proposed system is tolerant and well suited to aileron failures and that there was a large performance advantage to be gained by using the neural network and control allocation algorithms. It allows the system greater flexibility in the method used to mitigate the effects of failure.

The RBFNN-based controller was found to be very robust throughout the simulations as it did not show large variations in the controller gains set at the start of the simulation. This proves to be advantageous, as if one did not have large computational ability during flight this controller would be ideal. If one chooses the controller gains beforehand the less severe faults could be handled without the need for too much extra computation.

6 Conclusions and Recommendations

6.1 Conclusions

The current research has shown considerable potential in the proposed design of an intelligent fault-tolerant control system for an unmanned aerial vehicle. It has shown that the proposed system, consisting of either a BPNN-based or RBFNN-based adaptive controller and a sequential least squares control allocation algorithm can be used successfully to tolerate a range of actuator failures, loss of a control surface as shown by the 0 *rad* deflection failures, and uncertainties within the aircraft model as shown by the use of the secondary and tertiary control surfaces. The system was tested under a number of different conditions in a number of different flight modes and, in the majority of cases, the system performed well, yielding safe and fault-tolerant flight.

This was made particularly evident when comparisons were made between a PID autopilot and an autopilot with the proposed control architecture. The PID controller alone was found to be unable to handle any faults without the control allocation algorithm, therefore it was always simulated with the CA enabled during failure (it was able to handle neutral and small to medium control surface failures as well as combined failures with CA enabled). The NN-based PID controllers were able to handle some failures without the use of CA. This shows the improved performance of the intelligent controllers. The system was also found to be able to handle most of the same failure conditions in the presence of microburst windshear. In this respect, this research and the proposed intelligent fault-tolerant control system can be considered successful. The RBFNN-based controller was found to be very robust.

However, there are several concerns and issues that were identified. A number of problems were encountered in the research that may limit the ability of this system to function under some circumstances. This was observed in the neural network's trade off between bank angle and pitch angle performance. However, the system was always able to dampen out oscillations overtime (the initial peak of these oscillations were still large at times) and return to a stable flight mode. There was also uncertainty on the performance of the proposed system for

different airframes; and autopilot control structures. There may well be issues surrounding changes in aircraft characteristics with changes in airspeed and altitude.

Furthermore, the complexity of a practical autopilot made the determination of the optimum settings difficult. There were numerous limits, correcting factors and adjustments that could be applied to a greater or lesser extent that all contribute to the successful simulation of fault tolerance. To understand fully the nuances of the system and to determine with confidence that this system could be implemented into a UAS airframe was difficult and time consuming due to the many different combinations of parameters that could be set. This complexity was further increased by the introduction of the proposed system which increased the number of parameters to be adjusted. Thus, while a successful result was demonstrated, it was difficult to say with confidence that the proposed system was fully fault-tolerant in all conditions.

6.2 Future Recommendations

This research has provided insight into the use of an artificial intelligent adaptive controller combined with a control allocation algorithm to create a control strategy that is tolerant to several control actuation failures. However, in this research a number of concerns were raised, and a number of recommendations are made.

One of the areas of concern is that at times, the neural network could improve one parameter such as bank angle and one would see a degradation in the pitch performance. It is recommended that a study and review on the interaction of the control allocator and neural network algorithm be done in the form of the control matrix weights being adjusted continually during the simulation rather than over the entire simulation done in this study to ascertain whether the reallocation could be improved. The effectiveness of the proposed control strategies could also be further explored through the use of a Monte Carlo simulation of a wide range of uncertain parameters and failures.

Other issues surround the ability of the aircraft to manoeuvre, after the failure of a control surface. This was not observed often in these tests especially in the higher flight modes, however with the reduced control authority that develops from the control failure, an introduction of a flight envelope restriction should be applied. This may be in the form of reduced pitch and bank angles, an increase in stall speed, or a reduction in maximum altitude.

It is recommended that a system be developed, that detects that the control authority is reduced and prevents the neural network from increasing the PID gains uncontrollably. This would prevent a situation where the controller introduces large oscillations, as would occur with this system.

It is recommended that a form of fault detection and diagnosis (FDD) be introduced into the system in the form of intelligence where information on the health condition of each control surface is gathered. This information could then be used to aid the neural networks and allocation algorithms in controlling the aircraft. This could be accomplished in several ways. The first and perhaps simplest method of control surface monitoring would be the direct measurement of the control actuator deflections. While this technique would be simple, it adds to the hardware complexity and in fact may decrease the fault tolerance of the system as the additional sensors could fail and provide incorrect or spurious data.

The other potential solution lies in the use of a neural network monitoring algorithm that compares the behaviour of the system to a set of predefined control inputs. The control surfaces could move into an input sequence whose output is known, if the output is outside of certain boundaries then a potential fault exists. This method requires a detection system which is very good at determining the difference between transient errors and actual faults within the system. Time could be included as a factor here.

It would also be advantageous to do a more in depth study into the limits of the intensity of the microburst windshear the UAV is able to sustain as this plays a big role in the types of faults the control system is able to handle.

A brief investigation of real-world implementation issues is also recommended. These issues could take the form of uncertainties resulting from the state reconstruction based on imperfect sensors (noise, calibration errors, bias errors, drift, alignment errors and position errors) as well as time-delays associated with estimating the aircraft state from these imperfect measurements; the impact of computational time-delays, lags and other dynamics; in addition to actuator dynamics and uncertainties.

Finally, it is recommended that the flight dynamic model be validated through the use of flight tests and that the proposed control strategy in its current form is flown. This flight should be conducted with a “man-in-the-loop" that would allow a pilot to regain control should the proposed control strategy becomes unstable.

7 References

Abdulrahim, M., & Lind, R. (2005). Control and Simulation of a Multi-role Morphing Micro Air Vehicle. *AIAA Guidance, Navigation and Control Conference and Exhibit*, no. 2005-6481. San Francisco: California. 15-18 August.

Akyurek, S., Gunes, U., Elbir, O., Kasnakoglu, C., & Kaynak, U. (2017). Rudder loss recovery autopilot system for a small fixed-wing aircraft. *Proceedings of 10th International Conference on Electrical and Electronics Engineering (ELECO)* (pp. 752 - 757). Bursa, Turkey: IEEE.

Alam, M., Das, B., & Pant, V. (2015). A Comparative Study of Metaheuristic Optimization Approaches for Directional Overcurrent Relays Coordination, *Electric Power Systems Research* 128 pp.39-52.

Alwi, H., & Edwards, C. (2008). Fault Tolerant Control Using Sliding Modes With On-line Control Allocation. *Automatica*, 44, 1859-1866.

Bateman, F., Noura, H., & Ouladsine, M. (2011). Fault Diagnosis and Fault-Tolerant Control Strategy for the Aerosonde UAV. *IEEE Transactions on Aerospace and Electronic Systems* , 47(3), 2119 - 2137.

Bobbitt, R. B., (1991) *Escape Strategies for Turboprop Aircraft in Microburst Windshear*, PhD Thesis, Naval Postgraduate School, Monterey, California, USA.

Brockhaus, R. (1987). A Mathematical Multi-point Model for Aircraft Motion in Moving Air, Zeitschrift fÃijr Flugwissenschaften und Weltraumforschung, ZfW, 11, 174-184.

Chamseddine, A., Zhang, Y., Rabbath, C., Fulford, C., & Apkarian, j. (2011). Model Reference Adaptive Fault Tolerant Control of a Quadrotor UAV. *Proceedings of the Infotech@Aerospace Conference* . St. Louis, Missouri, 29 - 31 March: AIAA.

Chamseddine, A., Amoozgar, M. H., & Zhang, Y. M. (2015). Experimental Validation of Fault Detection and Diagnosis for Unmanned Aerial Vehicles. *Handbook of Unmanned Aerial Vehicles*. Springer. 1123-1155. ISBN: 9789048197071.

Che, Z., Chiang, T., & Che, Z. (2011). Feed-forward Neural Network Training: A Comparison Between Genetic Algorithm and Back-Propagation Learning Algorithm. *International Journal of Innovative Computing, Information and Control*, 7(10), 5839-5850.

Chollom, T., Ofodile, N., & Ubadike, O. (2016). Application Techniques of Multi-objective Particle Swarm Optimization: Aircraft flight control. *Proceedings of the 2016 UKACC 11th International Conference on Control (CONTROL)*. Belfast, 31st August - 2nd September: UKACC.

Chowdhary, G., Johnson, E. N., Chandramohan, R., Kimbrell, M. S., & Calise, A. (2013). Guidance and Control of Airplanes Under Actuator Failures and Severe Structural Damage. *Journal of Guidance, Control, and Dynamics*, 36(4), 1093-1104.

Clabaugh, C., Myszewski, D., & Pang, J. (2000). Feed-Forward networks. Retrieved September 23, 2016, from Stanford University: https://cs.stanford.edu/people/eroberts/courses/soco/projects/neural-networks/Architecture/feedforward.html

Cook, M. V. (2007). *Flight Dynamics Principles*. Oxford: Elsevier. ISBN: 978-0-7506-6927-6.

Coopmans, C., Podhradsky, M., & Hoffer, N. V. (2015). Software- and Hardware-in-the-loop Verification of Flight Dynamics Model and Flight Control Simulation of a Fixed-wing Unmanned Aerial Vehicle. *Education and Development of Unmanned Aerial Systems (RED-UAS)* , 115-122. Cancun: 23-25 November.

Dadian, O., Bhandari, S., & Raheja, A. (2016). A recurrent neural network for nonlinear control of a fixed-wing UAV. *Proceedings of the American Control Conference (ACC)* (pp. 1341 - 1346). Boston: IEEE.

Danilin, S., Makarov, M., & Shchanikov, S. (2014). Numerical Simulation of Neural Network Components of Controlling and Measuring Systems. *2014 International Conference on Mechanical Engineering, Automation and Control Systems (MEACS)*. IEEE. Tomsk: Russia. 27âĂŞ29 October.

Davidson, J., Lallman, F., & Bundick, T. (2001). Real-time adaptive control allocation applied to high performance aircraft. *Proceedings of the 5th SIAM Conference on Control & its Applications*. San Diego: California. 9-13 July.

Defense Industry Daily Staff. (2011, December 22). Israeli Manufacturers Turkish UAV Contract. Retrieved August 17, 2016, from Defense Industry Daily: http://www.defenseindustrydaily.com/israeli-manufacturers-win-150m-turkish-uav-contract-updated-0389/

Dorobantu, A., (2013). *Test Platforms for Model-Based Flight Research*. PhD Thesis. University of Minnesota. MN, USA.

Drozeski, G. R. (2005). A Fault-Tolerant Control Architecture for Unmanned Aerial Vehicles. PhD Thesis .*Georgia Institute of Technology, School of Electrical and Computer Engineering*. Georgia.

Ducard, G., Rudin, K., Omari, S., & Siegwart, R. (2014). Strategies for Sensor-Fault Compensation on UAVs: Review, Discussions & Additions. *Proceedings of the 13th European Control Conference (ECC)* (pp. 1963-1968). Strasburg: 24-27 June 2014: IEEE.

Durham, W., Bordignon, K., & Beck, R. (2017). *Aircraft Control Allocation*. Wiley & Sons.

Elferink, J., & Visser, H. (2001). Optimal and near-optimal take-off maneuvers in the presence of windshear. *Proceedings of Institute of Mechanical Engineers*, 215(5), 257-268.

Etkin, B. (1959). Dynamics of Atmospheric Flight. New York: John Wiley & Sons Inc.

Etkin, B. & Reid, L. (1996). Dynamics of Flight: Stability and Control. New York: John Wiley & Sons Inc.

Feng, L., Wang, Y., & Qu, X. (2015). The Small UAV Longitudinal Control Law Design Based on Genetic Algorithms. *Proceedings of the 2nd International Conference on Electrical, Computer Engineering and Electronics (ICECEE 2015)*. Jinan: 29-31 May 2015: Atlantis Press.

Fiuzy, M., Haddadnia, J., & Mashhadi, S. K. (2013). Designing an Optimal PID Controller for Control the Planes Height, Based on Control of Autopilot by using Evolutionary Algorithms. *Journal of Mathematics and Computer Science*, 6(4), 260 - 271.

Freeman, P., & Balas, G. J. (2014). Actuation Failure Modes and Effects Analysis for a Small UAV. *Proceedings of the American Control Conference* (pp. 1292-1297). Portland: 4-6 June: IEEE.

Fuggetti, G., Ghetti, A., & Zanzi, M. (2015). Safety Improvement of Fixed Wing Mini-UAV Based on Handy FDI Current Sensor and a Fail-safe Configuration of Control Surface Actuators. *Proceedings of the IEEE Metrology for Aerospace (MetroAeroSpace)*, 356-361. Benevento: 4-5 June 2015: IEEE.

Garcia, G. A., & Keshmiri, S. (2011). Nonlinear Model Predictive Controller for Navigation, Guidance and Control of a Fixed-Wing UAV. *Proceedings of the AIAA Guidance, Navigation, and Control Conference*. Portland: 8-11 August 2011.

Gautam, D., & Ha, C. (2013). Control of a Quadrotor Using a Smart Self-Tuning Fuzzy PID Controller. *International Journal of Advanced Robotic Systems*, 10(11).

Gavilan, F., Vazquez, R., & Camacho, E. F. (2015). An Iterative Model Predictive Control Algorithm for UAV Guidance. *IEEE Transactions on Aerospace and Electronic Systems*, 51(3), 2406-2419.

Ge, S., Hang, C., Lee, T., & Zhang, T. (2002). Stable Adaptive Neural Network Control. *The Kluwer International Series on Asian Studies in Computer and Information Science*; 13. Springer US. New York.

Ghazzawi, K. Y. (2011). Robust PID Controller Design for an UAV. *Proceedings of the World Congress on Engineering and Computer Science 2011*, 2. San Francisco: 19-21 October 2011.

Gibson, A., & Patterson, J., (2017). *Deep Learning: A Practitioner's Approach*, O'Reilly Media, Inc. USA.

Girish, C. V., Emilio, F., Jonathan, H. P., & Hugh, L. (2015). Nonlinear Flight Control Techniques for Unmanned Aerial Vehicles. *Handbook of Unmanned Aerial Vehicles*, 577-612. Springer.

Gomez, J. F., & Jamshidi, M. J. (2011). Fuzzy Adaptive Control for a UAV. *Journal of Intelligent & Robotic Systems*, 62(2), 271-293.

Gripp, J. A., & Sampaio, U. P. (2014). Automatic Landing of a UAV using Model Predictive Control for the Surveillance of Internal Autopilot's Controls. *Proceedings of the International Conference on Unmanned Aircraft Systems (ICUAS)*; (pp. 1219-1224). Orlando: 27-30 May 2014: IEEE.

Grymin, D. J. (2013). Two-Step System Identification and Primitive-Based Moyion Planning for Control of Small Unmanned Aerial Vehicles. PhD Thesis. *Virginia Polytechnic Institute and State University, Aerospace Engineering Department*. Blacksburg.

Guo, Y., Zhang, Y., & Jiang, B. (2010). Multi-model-based Flight Control System Reconfiguration Control in the Presence of Input Constraints. *Proceedings of the 8th World Congress on Intelligent Control and Automation (WCICA)*, (pp. 5819-5824). Jinan: 7-9 July: IEEE.

Hafez, A. T., & Kamel, M. A. (2016). Fault-Tolerant Control for Cooperative Unmanned Aerial Vehicles Formation Via Fuzzy Logic. *Proceedings of the International Conference on Unmanned Aircraft Systems (ICUAS)*. Arlington: 7-10 June: ICUAS.

Hamidi, J. (2012). Control System Design Using Particle Swarm Optimization (pso). *Intelligent Journal of Soft Computing and Engineering*, 1(6), 116-119.

Hansen, S., & Blanke, M. (2014). Diagnosis of Airspeed Measurement Faults for Unmanned Aerial Vehicles. *IEEE Transactions on Aerospace and Electronic Systems*, 50(1), 224-239.

Harkegard, O. (2002). Efficient Active Set Algorithms for Solving Constrained Least Squares Problems in Aircraft Control Allocation. *Proceedings of the 41st IEEE Conference on Decision and Control* , 1295-1300. Las Vegas: 10-13 December.

Haupt, R., & Haupt, S. (2004). Practical Genetic Algorithmns. Hoboken: John Wiley & Sons, Inc.

Hervas, J. R., Kayacan, E., Reyhanoglu, M., & Tang, H. (2014). Sliding Mode Control of Fixed-wing UAVs in Windy Environments. *Proceedings of the 13th International Conference on Control Automation Robotics & Vision (ICARCV)* (pp. 986 - 991). Singapore: 10-12 December 2014: IEEE.

Higgs, D. (2014). Design of an Intelligent Fault-tolerant Control System for Unmanned Aerial Vehicle. MSc Dissertation. University of the Witwatersrand, School of Mechanical, Industrial, and Aeronautical Engineering. *Johannesburg.*

Hoffer, N. V. (2014). System Identification of a Small Low-cost Unmanned Aerial Vehicle using Flight Data from Low-cost Sensors. PhD Thesis. Utah State University, Mechanical and Aerospace Engineering Department. *Logan.*

Hoffer, N. V., Coopmans, C., Chen, Y., & Fullmer, R. R. (2014). Small Low-Cost Unmanned Aerial Vehicle System Identification: Brief Sensor Survey and Data Quality, Consistency Checking, and Reconstruction. *Proceedings of the International Conference on Unmanned Aircraft Systems (ICUAS)* (pp. 477-482). Orlando: 27-30 May 2014: IEEE.

Hoffer, N. V., Coopmans, C., Fullmer, R. R., & Chen, Y. (2015). Small Low-Cost Unmanned Aerial Vehicle System Identification by Error Filtering Online Learning (EFOL) Enhanced Least Squares Method. *Proceedings of the International Conference on Unmanned Aircraft Systems (ICUAS)*, 1355-1363. Denver: 9-12 June 2015: IEEE.

Huang, C.J., & Jan, S.S. (2014, May 2). UAV Shipboard Landing with RTK. Retrieved August 22, 2016, from GPS World: http://gpsworld.com/uav-shipboard-landing-with-rtk/

Ismail, S., Pashilkar, A. A., Ayyagari, R., & Sundararajan, N. (2014). Improved Neural-aided Sliding Mode Controller for Autolanding Under Actuator Failures and Severe Winds. *Aerospace Science and Technology*, 33(1), 55-64.

ISO 2533:1975, (1975). Standard Atmosphere. Retrieved March 20, 2018, from International Organization for Standardization: https://www.iso.org/standard/7472.html

Jiang, B., Gao, Z., Shi, P., & Xu, Y. (2010). Adaptive Fault-Tolerant Tracking Control of Near-Space Vehicle Using TakagiâĂŞSugeno Fuzzy Models. *IEEE Transactions on Fuzzy Systems*, 18(15), 1000 - 1007.

Johansen, T., & Fossen, T. I. (2013). Control allocationâĂŤA survey. *Automatica*, 49(5), 1087-1103.

Kapela, R., Swietlicka, A., Kolanowski, K., Pochmara, J., & Rybarczyk, A. (2017). A Set of Dynamic Artificial Neural Networks for Robot Sensor Failure Detection. *Proceedings of the 11th International Workshop on Robot Motion and Control* (pp. 199-204). Wasowo Palace, Poland, 3-5 July: Romoco.

Kaufman, H., Barkana, I., & Sobel, K. (1998). *Direct Adaptive Control Algorithms: Theory and Applications*, Communications and Control Engineering Series. New York: Springer.

Khan, R. (2016). Fault Tolerant Flight Control System Design for Unmanned Aerial Vehicles. PhD Thesis. *RMIT University, School of Aerospace, Mechanical and Manufacturing Engineering*. Australia.

Lavretsky, E. & Wise, K. A. (2013). Robust and Adaptive Control with Aerospace Applications. London: Springer.

Lee, D., Kim, H. J., & Sastry, S. (2009). Feedback Linearization vs. Adaptive Sliding Mode Control for a Quadrotor Helicopter. *International Journal of Control, Automation, and Systems*, 7(3), 419-428.

Limnaios, G. (2014). Current Usage of Unmanned Aircraft Systems (UAS) and Future Challenges: A Mission Oriented Simulator for UAS as a Tool for Design and Performance Evaluation. *Journal of Computations & Modelling*, 4(1), 167-188.

Lin, C. T., & Chung, C. (2014). Intelligent Control System Design for UAV using a Recurrent Wavelet Neural Network. *Neural Computing and Applications*, 24(2), 487-496.

Lombaerts, T. V., Chu, P., & Mulder, J. (2011). Adaptive Nonlinear Flight Control and Control Allocation Failure Resilience. *Advances in Aerospace Guidance, Navigation and Control*, 41-53. Springer

Ma, L. (2011). Development of Fault Detection and Diagnosis Techniques with Applications to Fixed-wing and Rotary-wing UAVs. PhD Thesis. Concordia University, Mechanical and Industrial Engineering Department. *Montreal*.

Mac, T., Copot, C., Duc, T., & De Keyser, R. (2016). AR.Drone UAV Control Parameters Tuning Based on Particle Swarm Optimization Algorithm. *Proceedings of the*

IEEE International Conference on Automation, Quality and Testing, Robotics - THETA 20th edition (AQTR). Cluj-Napoca, Romania, 19-21 May: IEEE.

Mahmoud, M. S., & Xia, Y. (2013). Analysis and Synthesis of Fault-Tolerant Control Systems. Chichester: John Wiley & Sons.

Meirovitch, L. (2003). Methods of Analytical Dynamics. New York: McGraw-Hill Book Company.

Mick, J. (2010, May 14). Boeing Unveils Phantom Ray, an Autonomous, Unmanned Stealth Fighter . Retrieved August 17, 2016, from Daily Tech: http://www.dailytech.com/Boeing+Unveils+Phantom+Ray+an+Autonomous +Unmanned+Stealth+Fighter/article18381.htm

Milhim, A., Zhang, Y., & Rabbath, C. A. (2012). Fault-tolerant Fuzzy Gain-Scheduled PID for a Quadrotor Helicopter Testbed in the Presence of Actuator Faults. *Proceedings of the 2nd IFAC Conference on Advances in PID Control*. Brescia: 28-30 March 2012.

Moller, M. (1993). A Scaled Conjugate Gradient Algorithm for Fast Supervised Learning. *Neural Networks*, 6(4), 525-533.

Norton, M., Khoo, S., & Kouzani, A. (2015). Multi-surface Sliding Control of MIMO Autonomous Flight Systems. *2015 10th Asian Control Conference (ASCC)*. Kota Kinabalu, Malaysia: 28 May 2015 - 3 June 2015.

Norton, M., Khoo, S., Kouzani, A., & Stojcevski, A. (2015). Adaptive Fuzzy Multi-surface Sliding Control of Multiple-Input and Multiple-Output Autonomous Flight Systems. *IET Control Theory & Applications*, 9(4), 587 - 597.

Owens, B., Cox, D., & Morelli, E. (2006). Development of a Low-cost Sub-scale Aircraft for Flight Research: The FASER Project. *Proceedings of the AIAA Aerodynamic Measurement Technology and Ground Testing Conference*, no. 2006-3306. San Francisco, California: 5-8 June.

Pascanu, R., Gulcehre, C., Cho, K., & Bengio, Y. (2014). How to Construct Deep Recurrent Neural Networks. *arXiv preprint arXiv:1312.6026v5*.

Passino, K. M. (2005). *Biomimicry for Optimization, Control, and Automation*. London: Springer.

Paw, Y. C. (2009). Synthesis and Validation of Flight Control for UAV, PhD Thesis. University of Minnesota. MN, USA.

Paw, Y. C., & Balas, G. (2008). Uncertainty Modeling, Analysis and Robust Flight Control Design for a Small UAV System. *AIAA Guidance, Navigation and Control Conference and Exhibit*. Honolulu, Hawaii: 18-21 August.

Pedro, J. O., (1992). Numerical Simulations of Transport Aircraft in Variable Wind Field, PhD Thesis, Faculty of Mechanical, Power and Aeronautical Engineering, Warsaw University of Technology, Warsaw, (in Polish, unpublished).

Pedro, J. O., (2004). Aircraft Gaudance Strategies in Microburst Windshear Environment. *Proceedings of 6th Portuguese Conference on Automatic Control*, Faro: 7-9 June.

Peni, T., Vanek, B., Szabo, Z., & Bokor, J. (2014). Supervisory Fault Tolerant Control of the NASA AirStar Aircraft. *2014 American Control Conference* (pp. 666-671). Portland: 4-6 June 2014 :IEEE.

Qian, M., Jiang, B., & Xu, D. (2012). Fault Tolerant Tracking Control Scheme for UAV Using Dynamic Surface Control Technique. *Circuits, Systems, and Signal Processing*, 31(2), 1713-1729.

Rahimi, M. R., Hajighasemi, S., & Sanaei, D. (2013). Designing and Simulation for Vertical Moving Control of UAV System using PID, LQR and Fuzzy logic. *International Journal of Electrical and Computer Engineering*, 3(5), 651-659.

RP Defence. (2013, March 12). U.S. Army UAVs Frozen In Time. Retrieved August 17, 2016, from RP Defence: http://rpdefense.over-blog.com/article-u-s-army-uavs-frozen-in-time-116113908.html

Saad, M. S., Jamaluddin, H., & Darus, I. Z. (2012). PID Controller Tuning Using Evolutionary Algorithms. *WSEAS Transactions on Systems and Control* 2012, 7(4), 139-149.

Sadeghzadeh, I. (2015). Fault Tolerant Flight Control of Unmanned Aerial Vehicles. PhD Thesis. Concordia University, Mechanical Engineering Department. *Montreal.*

Sadeghzadeh, I., & Zhang, Y. (2013). Actuator fault-tolerant control based on Gain-Scheduled PID with application to fixed-wing Unmanned Aerial Vehicle. *Proceedings of the 2nd Conference on Control and Fault-Tolerant Systems (SysTol)* (pp. 342 - 346). Nice: 9-11 October 2013: IEEE.

Shin, Y. (2005). *Neural network based adaptive control for nonlinear dynamic regimes*. PhD Thesis, Georgia Institute of Technology, School of Mechanical Engineering.

Siyang, D., Guang-Xun, D., Quan, Q., & Kai-Yuan, C. (2014). Tracking control of a small fixed-wing UAV based on Additive-State-Decomposition-Based dynamic inversion

method. *Proceedings of the 33rd Chinese Control Conference* (pp. 3625-3631;). Nanjing: 28-30 July 2014: IEEE.

Sundararajan, N., Saratchandran, P., & Li, Y. (2002). Fully Tuned Radial Basis Function Neural Networks for Flight Control. *The Kluwer International Series on Asian Studies in Computer and Information Science*, 12. Springer.

Tang, Y. (2013). Fault Tolerant Control for Nonlinear Aircraft Based on Feedback Linearization. PhD Thesis. University of Hull, Department of Engineering, United Kingdom.

Valavanis, K. P., Oh, P. Y., & Piegl, L. A. (2008). Unmanned Aircraft Systems: International Symposium on Unmanned Aerial Vehicles, UAV '08. Orlando: 23-25 June 2008: Springer.

Vishal, & Ohri, J. (2014). GA Tuned LQR and PID Controller for Aircraft Pitch Control. *IEEE 6th India International Conference on Power Electronics (IICPE)*. Kurukshetra, India: 8-10 December.

Viudez-Moreiras, D., Martin, I., & Martin-Sanchez, J. M. (2014). A New Pitch Angle Adaptive Control Design. *Proceedings of the International Conference on Unmanned Aircraft Systems (ICUAS)*, (pp. 928-935). Orlando: 27-30 May 2014: IEEE.

Voos, H. (2009). Nonlinear Control of a Quadrotor Micro-UAV using feedback-linearization. *Proceedings of the IEEE International Conference on Mechatronics*. Malaga: 14 -17 April 2009.

Wang, S., Zhen, Z., Jiang, J., & Wang, X. (2016). Flight Tests of Autopilot Integrated with Fault-Tolerant Control of a Small Fixed-Wing UAV. *Mathematical Problems in Engineering*. 2016.

Wang, J., Song, X., Jiang, J., & Wang, X. Y., (2016). Modeling and Simulation of China C Series Large Aircraft with Microburst. *Mathematical Problems in Engineering*. 2016.

Wilburn, B. K., Perhinschi, M. G., & Wilburn, J. N. (2014). A Modified Genetic Algorithm for UAV Trajectory Tracking Control Laws Optimization. *International Journal of Intelligent Unmanned Systems*, 2(2), 58-90.

Witczak, M., & Pazera, M. (2016). Fault-Tolerant Control: Solutions and Challenges. *Pomiary Automatyka Robotyka*, 20(1), 5-15.

Yadav, A., & Gaur, P. (2014). AI-based Adaptive Control and Design of Autopilot System for Nonlinear UAV. *Sadhana*, 39(4), 765-783.

Yang, S. (2014). Intelligent PID Controller Based on Fuzzy Logic Control and Neural Network Technology for Indoor Environment Quality improvement. PhD Thesis. University of Nottingham.

Yoon, S., Kim, S., Bae, J., Kim, Y., & Kim, E. (2015). Experimental Evaluation of Fault Diagnosis in a Skew-Configured UAV Sensor System. *Control Engineering Practice*, 19(2), 158-173.

Yu, X., Liu, Z., & Zhang, Y. (2016, August). Fault-Tolerant Formation Control of Multiple UAVs in the Presence of Actuator Aaults. *International Journal of Robust and Nonlinear Control*, 26(12), 2668-2685.

Zaki, H., Unel, M., & Yildiz, Y. (2017). Trajectory control of a quadrotor using a control allocation approach. *International Conference on Unmanned Aircraft Systems (ICUAS)*. Miami, FL, USA, 13-16 June, (pp. 533-539).

Zhang, Y. M. (2015). Fault Detection and Diagnosis for NASA GTMUAV with Dual Unscented Kalman Filter. *Handbook of Unmanned Aerial Vehicles*, 1157-1181. Springer.

Zheng, F., Gong, H., & Zhen, Z. (2015). Adaptive constraint backstepping fault-tolerant control for small carrier-based unmanned aerial vehicle with uncertain parameters. *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering*, 230(3), 407-425.

Zhenxing, G., Hongbin, G., & Hui, L., (2009). Real-Time Simulation of Large Aircraft Flying Through Microburst Wind Field, *Chinese Journal of Aeronautics*, (22)5, 459-466.

Zhou, Y., Chen, M., & Jiang, C. (2016). Robust Tracking Control of Uncertain MIMO Nonlinear Systems with Application to UAVs. *IEEE/CAA Journal of Automatica Sinica*, 2(1), 25-32.

Zhou, W., Yin, K., Wang, R., & Wang, Y. (2014). Design of Attitude Control System for UAV Based on Feedback Linearization and Adaptive Control, *Mathematical Problems in Engineering*, vol. 2014, Article ID 492680, 8 pages, doi:10.1155/2014/492680.