

INTELLIGENT PATH PLANNING AND OPTIMISATION FOR
REAL-TIME COOPERATIVE FLIGHT

JORDAN HAMILTON

ACADEMIC YEAR 2020–2021

CRANFIELD UNIVERSITY

Cranfield Defence and Security

PhD

Academic Year 2020–2021

Jordan Hamilton

Intelligent Path Planning and Optimisation for Real-Time
Cooperative Flight

Supervised by

Dr. A. Saddington, Dr. D. Galvão Wall and Dr. J. Economou

Abstract

UNMANNED aerial vehicles (UAVs) have seen an increase in military and commercial use over the past two decades. Whilst the majority of UAVs remain remotely operated, there is a desire amongst many for highly-autonomous aircraft capable of performing solo or cooperative missions with little human interaction. This work presents a novel path planning framework using an evolutionary algorithm to train a deep neural network for generating dynamically-achievable, continuously-flyable, and manoeuvrable paths for a large-scale fixed-wing military autonomous combat aircraft during three-dimensional cooperative flight missions. This autonomous flight control system has been demonstrated to provide robust and reliable path generation during solo and cooperative flight missions with online path planning.

The robustness of the solution is demonstrated on a computational model representative of the F-16 Fighting Falcon during both static and dynamic waypoint flight simulations, with online path re-planning. The computational model utilises a novel flight control system governs rotational motion of the aircraft, utilising multiple hyper-surfaces of optimised controller gains to ensure optimal dynamic performance and closed-loop stability throughout the flight envelope.

Acknowledgements

I would like to express thanks to Cranfield University for funding this research, and to my supervisory team: Dr Alistair Saddington, Dr David Galvão Wall and Dr John Economou, for providing guidance and feedback throughout this project.

I extend gratitude to my fellow researchers James Norris and Ayden Clay, whom provided me with constant enthusiasm, support, and invaluable advice.

—— Alan Turing ——

"If a machine is expected to be infallible, it cannot also be intelligent"

Contents

Contents	vii
List of Figures	xiii
List of Tables	xxv
1 Introduction	1
1.1 Motivation	1
1.2 UAV Designs	2
1.3 Requirements of an Autonomous System	4
1.4 Challenges	5
1.5 Research Aim and Objectives	6
1.6 Methodology	6
1.6.1 Limitation of the Methodology	7
1.7 Implementation	7
1.7.1 Limitation of the Implementation	9
1.8 Outline of the Thesis	10
2 Literature Review	13
2.1 Path Finding Algorithms	13
2.2 Path Smoothing Algorithms	16
2.2.1 Dubins Paths	17
2.2.2 Bézier Curves	20
2.3 Summary of Path Generation Algorithms	25
2.4 Artificial Intelligence	26
2.4.1 Neural Networks	26
2.4.2 Reinforcement Learning	28

2.5	Summary	35
2.6	Research Objectives	37
3	Computational Flight Physics Model	39
3.1	Euler Angles	39
3.2	Angular Velocity Transformation	42
3.3	Equations of Motion	46
3.3.1	Translational Motion	47
3.3.2	Rotational Motion	48
3.3.3	Forces and Moments	51
3.4	Navigation Equations	54
3.5	Quaternions	55
3.6	Aerodynamic Model	55
3.7	Engine Model	58
3.8	Aircraft Model Parameters	60
3.9	Atmospheric Model	60
3.10	Control Surfaces	61
3.11	Flight and Operational Envelope	62
3.12	Summary	64
4	Flight Control System	65
4.1	Controller Architectures	65
4.1.1	Background	65
4.1.2	Gain Scheduling	68
4.2	Design Specification	70
4.3	Optimisation of PID/PIV Controllers	72
4.3.1	Heuristic Methods	72
4.3.2	Meta-heuristic Algorithms	74
4.3.3	Summary	96
4.4	Angular Rate Controllers	96
4.4.1	Optimisation Algorithm	97
4.4.2	High Precision Gain Surface	101
4.4.3	Pitch Rate Optimisation	104
4.4.4	Roll Rate Controller	120

4.4.5	Yaw Rate	126
4.5	Airspeed Controller	129
4.6	Summary	135
5	Path Planning Algorithm and Guidance	137
5.1	Path Planning Algorithm	137
5.1.1	Problem Formulation	137
5.1.2	Bézier Mathematics	143
5.1.3	Second-Order Bézier Curves	144
5.1.4	Third-Order Bézier Curve	148
5.1.5	Higher-Order Bézier Curves	154
5.2	Guidance Control System	157
5.3	Summary	162
6	Constant-Altitude Path Planning	165
6.1	Introduction	165
6.2	Fitness Function	168
6.3	Control Point Optimisation	173
6.4	Classification Neural Network	178
6.4.1	Preliminary Network	181
6.4.2	Observation Interval Size Analysis	183
6.5	Classification Implementation	195
6.6	Distance Controller	198
6.7	Classification Testing	202
6.7.1	Static Waypoint Analysis	202
6.7.2	Dynamic Waypoint Analysis	206
6.8	Discussion	225
7	Variable-Altitude Path Planning	229
7.1	Variable Altitude Problem	229
7.2	Climb Rate	230
7.3	Fitness Function	247
7.4	Control Point Optimisation	254
7.4.1	The Curse of Dimensionality	258

7.5	Regression Deep Neural Network	263
7.5.1	Preliminary Study	263
7.5.2	North-East (NE) Model	265
7.5.3	North-East-Heading Model (NEH)	276
7.5.4	Final Solution	280
7.6	Model Implementation and Testing	288
7.6.1	Static Waypoint Testing	288
7.6.2	Dynamic Waypoint Testing	296
7.6.3	Discussion	318
8	Conclusions	321
8.1	Conclusions from Chapter 2	321
8.2	Conclusions from Chapter 3	322
8.3	Conclusions from Chapter 4	323
8.4	Conclusions from Chapter 5	323
8.5	Conclusions from Chapter 6	324
8.6	Conclusions from Chapter 7	326
8.7	Limitations of the Methodology	329
8.8	Future Work	331
8.8.1	Flight Physics Model	331
8.8.2	Path Planning Framework	332
	References	335
A	Computational Model	355
A.1	Aircraft Axis Systems	355
A.1.1	Inertial Axis Systems	355
A.1.2	Body Axes	356
A.1.3	Wind Axes	357
A.2	Equations of Motion	357
A.2.1	Velocity and Acceleration Analysis	358
A.2.2	Coriolis Acceleration	365
A.2.3	Translational Motion	366
A.2.4	Angular Motion	368

A.2.5	Forces and Moments	371
A.3	Engine Model Data	380
B	Optimisation Test Functions	381
C	Autopilot Controller Gains	385
C.1	Pitch Rate Controller Gains	386
C.1.1	High Precision Gain Surface	386
C.1.2	Primary Positive Gain Surface	387
C.1.3	Primary Negative Gain Surface	388
C.1.4	Neutral Surface	389
C.2	Roll Rate Controller Gains	390
C.2.1	High Precision Gain Surface	390
C.2.2	Primary Gain Surface	391
C.2.3	Neutral Gain Surface	392
C.3	Yaw Rate Controller Gains	393
C.3.1	High Precision Gain Surface	393
C.3.2	Primary Gain Surface	394
D	Classification Network - Dynamic Waypoint Testing	395
D.1	Mission 1	395
D.1.1	Test 1	396
D.1.2	Test 2	397
D.1.3	Test 3	398

List of Figures

1.1	MQ-1 Predator, armed with AGM-114 Hellfire missiles, piloted by Lt. Col. Scott Miller on a combat mission over southern Afghanistan (US Air Force Photo / Lt. Col. Leslie Pratt)	3
1.2	Aircraft model structure overview	8
2.1	Construction of Voronoi diagram for UAV path planning using objects detected through radar [26].	15
2.2	An example of a collision-free path (blue) generated using a path finding algorithm; this is then adapted (red) using a path smoothing algorithm [37].	17
2.3	Medium and high altitude changing Dubins trajectories proposed by Owen et al. [44].	18
2.4	Primitive turn rotations about a torus based on the flight path vector [47].	19
2.5	Bezier curve convex hull defined by four control points	20
2.6	Generated trajectories for a three-vehicle system with spatial separation constraints imposed [51].	21
2.7	Collision scenario between a UAV and linearly-pathing obstacle in [53].	22
2.8	Illustration of the exact detour curve calculated and added to the trajectory [54].	23
2.9	Deep neural network architecture comprised of three hidden layers [69].	27

2.10	Geometry for waypoint navigation [72]. ρ represents the Euclidean distance from the vehicle to the waypoint, θ indicates the bearing angle to the waypoint, ψ indicates the angle to waypoint heading direction, ϕ represents the vehicle banking angle and v represents the aircraft's airspeed.	28
2.11	A non-exhaustive, but useful taxonomy of modern reinforcement learning algorithms [78].	29
2.12	Path planning comparison between IQL, CQL and A* in a static grid-world environment [35].	32
2.13	Structure of DQN mobile robotic path planning technique in a densely populated environment [87]	33
2.14	DQN UAV successfully arriving at target point with acceptable safe trajectory during dynamically changing environmental conditions [89].	34
3.1	body-axis right-hand orthogonal axis system rotations [91]	40
3.2	Euler angle rotations about each axis	40
3.3	Translational motion schematic between inertial and body-axes	47
3.4	Variation of inverse of thrust time constant with incremental power command. Adapted from Nguyen et al. [16]	58
3.5	First order engine model tree, adapted from Nguyen et al. [16]	59
3.6	Analytically derived flight operational envelope	63
4.1		70
4.2	Pitch rate response for a 10 deg/s demand conducted at various proportional gain values whilst the integral and derivative gains are set at zero	73
4.3	Bat algorithm performance analysis for variable population sizes analysed with a fixed number of 50 movements per bat.	80
4.4	Bat algorithm number of movements and runtime analyses for various population sizes conducted on all optimisation test functions.	82
4.5	Genetic Algorithm selector method population diversity study conducted on the Holder-Table optimisation function.	88

4.6	Population exploration for all four selector methods conducted on the Holder-Table function after 25 generations.	93
4.7	Block diagram for the angular rate controllers [19]	97
4.8	Three-dimensional topology profile for the search space of the angular rate controllers. Each subplot shows a singular controller value which has been held constant whilst the other two controllers indicate the fitness over the entire range	100
4.9	Angular velocity analysis using zero-demand surface gains for roll, pitch and yaw rate controllers. Analysis conducted at an altitude of 4 km with an airspeed of 200 m/s	103
4.10	Sub-optimal manual pitch rate gain analysis to determine the maximum sustainable pitch rate for airspeed 40 m/s with an altitude of 0 metres	106
4.11	Primary positive gain surface reference signal used for pitch rate optimisation	107
4.12	Optimised pitch rate response for an altitude of 0 metres with an airspeed of 40 m/s and a demand of 7 deg/s	108
4.13	Time response comparison for negative pitch rate demand applied using positive pitch rate gains against optimised negative pitch rate gains	110
4.14	Primary positive gain surface resulting in overshoot upon removal of the input demand. Airspeed 100 m/s, Altitude: 0 metres	111
4.15	Optimised time response from Figure 4.14 with comparison to the implementation of an optimised and scheduled neutral surface . .	112
4.16	Steady-state response improvements using the high-precision gain surface. Simulation conducted with an airspeed of 200 m/s at an altitude of 0 metres. The input demand was a 30 deg/s pitch rate prior to being removed	113
4.17	Steady-state response for input demand of 30 deg/s with an airspeed of 300 m/s at an altitude of 0 metres. Simulation has been conducted at three different time-steps to highlight time-sensitive issues within the controller	115

4.18	Normalised primary-positive surface engaging with the secondary surface (neutral surface) over a normalised elevator input demand [19]	116
4.19	Demonstration of scaling the primary-positive surface based on the input demand	117
4.20	Controller gain comparison between the use of the precision gain surface applied during the manoeuvre conducted in Figure 4.15 . .	118
4.21	Roll rate time response for a demanded roll rate of -120 degrees-per-second. The non-linear roll/yaw coupling induces a deceleration to occur as the yaw rate controller attempts to damp the induced yaw rate.	121
4.22	Time response for 220 degree-per-second demand for airspeed 150 m/s and altitude 0 metres. Simulation was conducted using only the primary optimised surface.	123
4.23	Comparison between the roll rate time response seen in Figure 4.22 against the time response using the neutral surface.	125
4.24	Steady-State error time comparison between roll rate controller using primary/secondary gain surfaces and a roll rate controller using primary/secondary with precision gain surfaces	126
4.25	Angular velocity controllers producing counteracting aerodynamic moments due to passive engagement	127
4.26	Optimised responses for maintaining an airspeed of 100 m/s with optimisation performed at 0 metres altitude.	130
4.27	Optimised 100 m/s airspeed gains used various airspeeds and altitude levels.	131
4.28	Airspeed controller performance during airspeed changing manoeuvre	132
4.29	Time response for PID controller when compared to PD/PID controller	133
4.30	Demonstration of PD/PID airspeed controller during accelerations, decelerations and hold	134
5.1	Path planning problem formulation in two-dimensions	139

5.2	Demonstration of the aircraft and waypoints unit vectors described in the inertial frame	141
5.3	Line-of-sight heading vectors used to satisfy the path planning orientation constraint for both position and waypoint unit vectors	142
5.4	Second-order Bézier curve with intersecting control point placed at the intersection of the heading vectors	145
5.5	Second-order Bézier curve with intersecting control point placed at the intersection of the heading vectors	146
5.6	Second-Order three-dimensional Bézier curve with unsatisfied orientation constraints	147
5.7	Third-order Bézier curve in three-dimensions with variable pitch angle	149
5.8	Symmetrical control point scaling for second-order Bézier curves using the globally defined variable.	150
5.9	Curvature profile in the x - y plane for Bézier path in Figure 5.8 with a scaling factor of 0.5.	152
5.10	Radius of curvature profile in the x - y plane for Bézier path in Figure 5.8 with a scaling factor of 0.5.	153
5.11	Second-order Bézier curve for large heading change resulting in a large cusp	154
5.12	Higher-Order Bézier curve control point schematic	155
5.13	High heading change at low magnitude distance comparison for third, fourth and fifth-order Bernstein polynomials	156
5.14	Guidance block diagram interaction with the angular rate controller	158
5.15	Pure pursuit line-of-sight illustration	158
5.16	Path planning guidance initial scenario in the xy plane	159
5.17	Small and large look ahead illustrations for pure pursuit guidance [135]	160
5.18	Look ahead distance comparison for flight simulation conducted at 300 m/s.	161
6.1	Control points optimisation. The red line indicates the optimal solution with the black line representing the control points and convex hull	166

6.2	Path fitness comparison between a short and unfeasible path and a long and feasible path	172
6.3	Path planning intermediate reference frame moving through the inertial axis system with the aircraft fixed at the centre	180
6.4	Constant altitude path plan mirroring	181
6.5	Control point scaling functions for a north coordinate of 2 km and varying east locations and heading values	182
6.6	Control point scaling solutions for north observation of 1 km, east observation of 1 km and heading angle of 0 degrees.	184
6.7	Optimised control point scaling solutions for the north-east interval study. Optimised was conducted for an airspeed of 150 m/s with the orientation of the waypoint target constant at 0 degrees. .	186
6.8	Intermediate waypoint classification path against the optimised path produced through use of the Genetic Algorithm	188
6.9	Comparison between the optimised solution and the output path based on different heading angle intervals.	190
6.10	Path planning optimisation for a north coordinate of 2000 metres, east coordinate of 6000 metres and heading angle of 0 degrees for various airspeeds	192
6.11	Path planning intermediate axis system formation	193
6.12	Neural network classification training accuracy and loss metrics .	195
6.13	Inertial axis representation of the intermediate reference frame for constant-altitude path planning	196
6.14	Classification model mirrored waypoint configuration two orientation angles	197
6.15	Distance controller block diagram schematic	198
6.16	Distance controller simulation optimisation	201
6.17	Classification network static waypoint analysis optimal and sub-optimal solutions displayed for north and east observations with respect to the intermediate reference frame	205
6.18	Online Path re-planning example conducted every 500 metres for four intervals. This example was conducted on a static waypoint target	207

6.19	Partial path re-planning demonstration for moving waypoint target. Partial path re-planning is demonstrated between two time intervals $t = 1$ and $t = 2$	208
6.20	Classification network dynamic waypoint testing <i>Mission 1</i> flight path	210
6.21	<i>Mission 1</i> : separation distance and airspeed time responses	212
6.22	<i>Mission 1</i> : Test 3 - Time sampled inertial position. Red dotted line indicates historic position data for the wingman with the blue dotted line indicating the historic position data for the lead aircraft. The black line indicates the flight path for the lead aircraft with the magenta dotted line indicating the current path for the wingman developed by the classification network.	214
6.23	Classification network with dynamic waypoint path re-planning <i>Mission 2</i>	216
6.24	<i>Mission 2</i> Time sampled inertial position. Red dotted line indicates historic position data for the wingman, blue dotted line indicates historic position data for manned aircraft, black line indicates the path of the manned aircraft and the purple dotted line indicates the current path for the wingman	217
6.25	<i>Mission 2</i> time sampled inertial axis positions for second test flight with partial replanning every 0.5 seconds and fully replanning every three seconds	218
6.26	<i>Mission 2</i> time sampled inertial axis positions for second test flight with partial replanning every 0.5 seconds and fully replanning every three seconds	220
6.27	Classification DNN with dynamic waypoint path replanning <i>Mission 2</i>	221
6.28	<i>Mission 3</i> time sampled inertial axis position for first test flight . .	222
6.29	<i>Mission 3</i> time sampled inertial axis position for second test flight with offset waypoint target of 3,000 metres.	224
6.30	Classification network dynamic waypoint analysis - <i>Mission 2</i> performed with waypoint offset of 3 km ahead of lead aircraft	227
7.1	Climb rate control system block diagram	231

7.2	Climb rate optimised response for an airspeed of 80 m/s starting with an initial altitude of 0 m and a demanded climb rate of 40 m/s	233
7.3	Climb rate comparison conducted at 80 m/s for input demands 20 m/s, 40 m/s and 60 m/s	234
7.4	Climb rate optimised responses for all optimisation flight speed trim points starting with an altitude of 0 metres	235
7.5	Initial climb rate manoeuvre conducted with an airspeed of 80 m/s at an initial altitude of 0 metres	237
7.6	Specific excess power analysis for an airspeed of 200 m/s with an initial altitude of 0 metres for various climb rate demands	238
7.7	Specific excess power analysis for a climb rate demand of 20 m/s with an airspeed of 200 m/s. This analysis was conducted at three different initial altitudes, which are indicated in the legend	240
7.8	SEP simulation analysis for an airspeed of 200 m/s conducted with an initial altitude of 3.5 km. The time responses plotted represent the manoeuvre conducted at an initial altitude of 0 metres and then at 3.5 km.	243
7.9	SEP analysis for a climb rate of 70 m/s with an initial altitude values of 0 km and 4 km and an airspeed of 200 m/s	245
7.10	Specific excess power prediction analysis on three-dimensional altitude changing paths for an airspeed of 200 m/s	246
7.11	Variable altitude changing path performing a loop-like manoeuvre to avoid the dynamics constraints of an x - y path	252
7.12	Intermediate control point placement in the y axis indicating the climb rate variances in the $y - z$ plane	253
7.13	Optimisation algorithm and fitness function testing for variable-altitude path planning. Blue markers indicate solutions which were dynamically achievable, with red markers indicating solutions that were not dynamically achievable	258
7.14	Optimised 3D path originating from position (North: 0 km, East: 0 km, Altitude: 0 km) to a waypoint target (N: 3 km, E: 3 km, Alt: 2 km)	259

7.15 Optimised three-dimensional path originating from position (North: 0 metres, East: 0 metres, Altitude: 6 km) to a waypoint target (North: 3 km, East: 3 km, Altitude: 8 km)	260
7.16 Five optimised control point scaling factors for a simply regression model.	264
7.17 North-East model position/waypoint configuration	265
7.18 Control point scaling solutions for NE model where the waypoint north coordinate is located between 0 km and 1 km. Optimisation was performed for a velocity of 200 m/s with an initial and final heading angle of 0° and an initial and final altitude of 0 km. Data points assigned as blue markers have been identified as outliers using a z-score measurement and removed from the training sample.	268
7.19 Optimised control point scaling functions for north coordinates between 8 km and 9 km. Optimisation was performed for a velocity of 200 m/s with an initial and final heading angle of 0° and an initial and final altitude of 0 km. Data points assigned as blue markers have been identified as outliers using a z-score measurement and removed from the training sample.	270
7.20 North-East DNN model surface prediction for the first control point scaling function	272
7.21 North-East DNN predicted paths against optimised paths for various north and east coordinate waypoints	275
7.22 North-East-Heading model observational configuration	276
7.23 North-East-Altitude DNN model surface prediction for the second control point scaling function for heading angles 170° to 180° . . .	278
7.24 Waypoint - North: 3233 metres, East: 4620 metres, Heading: 4.25 radians - Path classified as sub-optimal but dynamically feasible .	279
7.25 Final model observations analysis conducted with 100,000, 250,000 and 500,000 observations on five-layer, 250 neuron DNN	282
7.26 Various network structures for final model trained with 500,000 observations	284

7.27	Network performance analysis for various number of hidden layers, where each layer was comprised of 1,000 fully connected neurons. Training was performed for 500,000 observations.	286
7.28	Variable-altitude DNN optimised space in the north/east axes with respect to the inertial frame.	289
7.29	Distribution of dynamically unachievable solutions for static waypoint analysis with no path replanning	291
7.30	<i>Mission 1</i> flight plan	299
7.31	<i>Mission 1</i> inertial representation at Waypoint 2	300
7.32	<i>Mission 1</i> inertial representation at Waypoint 3	301
7.33	<i>Mission 1</i> inertial representation at Waypoint 4	302
7.34	<i>Mission 1</i> inertial representation at Waypoint 5	303
7.35	<i>Mission 1</i> final inertial flight paths	304
7.36	Variable-altitude model, dynamic waypoint target analysis <i>Mission 2</i>	305
7.37	<i>Mission 2</i> inertial representation at Waypoint 2	306
7.38	<i>Mission 2</i> inertial representation at Waypoint 3	307
7.39	<i>Mission 2</i> inertial representation at Waypoint 4	308
7.40	<i>Mission 2</i> inertial representation at Waypoint 6	309
7.41	<i>Mission 2</i> inertial representation at Waypoints 7 and 8	310
7.42	<i>Mission 2</i> inertial	311
7.43	Variable-altitude model, dynamic waypoint target analysis <i>Mission 3</i>	312
7.44	<i>Mission 3</i> inertial representation at Waypoint 2	313
7.45	<i>Mission 3</i> inertial representation at Waypoint 4	314
7.46	<i>Mission 3</i> inertial representation at Waypoints 5 and 6	315
7.47	<i>Mission 3</i> inertial representation at Waypoints 8	316
7.48	<i>Mission 3</i> inertial flight paths	317
A.1	Aircraft body-axis system adapted from Wiki Creative Commons [142]	355
A.2	Multi-axis system made up of an inertial frame, Earth-axis and body-axis system [91]	358
A.3	Translational motion schematic between inertial and body-axes . .	367

A.4	Thrust vectoring schematic for the F-16 aircraft [20]	373
A.5	Free-body diagram for moments induced through thrust vectoring (Images Modified from: [149])	375
D.1	Deep neural network (classification model) dynamic waypoint tar- get Mission 1	395
D.2	Mission 1: Test 1 - Time history	396
D.3	Mission 1: Test 2 - Time history	397
D.4	Mission 1: Test 3 - Time history	398

List of Tables

3.1	Aircraft model weight, mass, and moments of inertia used in simulation [16]	60
3.2	Wing dimensions and centre of gravity location using in simulation [16]	60
3.3	Control surface first-order dynamic [16]	62
4.1	Bat algorithm optimisation of test functions with default settings (Population Size 20)	78
4.2	Bat algorithm optimisation of test functions with default settings (Population Size 50)	79
4.3	Convergence Accuracy (%) for Genetic Algorithm Selector Methods	92
4.4	Average number of Generations to converge for GA Selector Method Performance Analysis	94
4.5	Average runtime to converge for GA Selector Method Performance Analysis	95
4.6	Optimisation local control law boundaries for high-precision gain surface	102
4.7	Maximum sustainable pitch rates (deg/s) for varying flight conditions throughout the operating envelope	109
4.8	Roll rate primary surface maximum input demands in deg/s . . .	122
4.9	Maximum instantaneous yaw rate demands (deg/s) used during optimisation of the primary surface for the yaw rate controller . .	128
6.1	Constant altitude heading size interval analysis	190
6.2	Constant-altitude path planning observations	194
7.1	Compensated offset equations from Figure 7.6	239

7.2	Specific excess power values (m/s) for nominal flight conditions at various airspeeds and altitudes. These values were obtained by flying the aircraft at the specified states with a climb rate of 0 m/s for 50 seconds. For any flight conditions that were not attainable, the SEP has been left blank	241
7.3	Observation breakdown for constant-altitude path planning	259
7.4	Observation breakdown for variable-altitude path planning given the same observational area and intervals used for constant-altitude path planning	261
7.5	Variable-altitude static waypoint analysis for operation in all identified regions depicted in Figure 7.28	295
7.6	Mission parameters for dynamic waypoint analysis of the variable-altitude path planner	297
A.1	F-16 Turbofan and afterburner thrust data (N) [16]	380
B.1	Optimisation Functions	382
B.2	Optimisation Functions Continued	383
B.3	Optimisation Functions Continued	384
C.1	Pitch rate controllers high precision surface optimised gains. . . .	386
C.2	Pitch rate controllers high precision surface optimised gains. . . .	387
C.3	Pitch rate controllers primary negative optimised gain surface. . .	388
C.4	Pitch rate controllers primary negative optimised gain surface. . .	389
C.5	Pitch rate controllers high precision surface optimised gains. . . .	390
C.6	Roll rate controllers primary gain surface optimised gains	391
C.7	Roll rate controllers neutral gain surface optimised gains.	392
C.8	Yaw rate controllers high precision surface optimised gains.	393
C.9	Yaw rate controllers primary gain surface optimised gains	394

Nomenclature

Throughout the thesis, various different nomenclature styles are used to symbolise various meanings, either descriptively or mathematically. All nomenclature listed below is standardised throughout the thesis and relates more appropriately to specific applications. However, at various instances within the thesis, the nomenclature listed below is used in a different context to describe different properties. Whenever this occurs, the meaning in the current instance is described in the body of the thesis for both clarity and brevity.

Roman

b	Wingspan [m].
$\bar{c}$	Mean aerodynamic chord length [m].
$a_{b,i}$	Translational acceleration in the body-axis with respect to the inertial frame [m/s].
$B_{i,n}(t)$	Bernstein Polynomial.
C_D	Coefficient of drag.
C_g	Centre of Gravity.
$C_{l,t}$	Total aerodynamic moment coefficient acting about the body x -axis.
C_L	Coefficient of lift.
$C_{m,t}$	Total aerodynamic moment coefficient acting about the body y -axis.
$C_{n,t}$	Total aerodynamic moment coefficient acting about the body z -axis.
$C_{x,t}$	Total aerodynamic force coefficient acting about the body x -axis.
$C_{y,t}$	Total aerodynamic force coefficient acting about the body y -axis.

$C_{z,t}$	Total aerodynamic force coefficient acting about the body z -axis.
$\dot{x}$	First derivative of a path along the x -axis.
$\dot{y}$	First derivative of a path along the y -axis.
$\dot{z}$	First derivative of a path along the z -axis.
$\ddot{x}$	Second derivative of a path along the x -axis.
$\ddot{y}$	Second derivative of a path along the y -axis.
$\ddot{z}$	Second derivative of a path along the z -axis.
$F_{x,i}$	Total force acting along the body x -axis when with respect to the inertial frame.
$F_{y,i}$	Total force acting along the body y -axis when with respect to the inertial frame.
$F_{z,i}$	Total force acting along the body z -axis when with respect to the inertial frame.
g	Acceleration due to gravity [9.80665 ms^{-2}].
h	Altitude [m].
H	Momentum [kg ms^{-1}].
H_e	Engine angular momentum [$\text{kgm}^2\text{s}^{-1}$]
I_x, I_y, I_z	Moment of inertia about x_b, y_b , and z_b [kgm^2].
I_{xy}	Product of inertia with respect to Oxy [kgm^2].
I_{yz}	Product of inertia with respect to Oyz [kgm^2].
I_{zx}	Product of inertia with respect to Ozx [kgm^2].
k	Curvature profile
k_d	Derivative controller gain.
k_i	Integral controller gain.
k_p	Proportional controller gain.
L	Lifting Force (N).
l_g	Gravitational moment arm [m]
$\bar{L}$	Rolling moment about the Ox axis [Nm].
l_g	Moment induced by gravity [Nm].
M	Pitching moment about the Oy axis [Nm].
N	Yawing moment about the Oz axis [Nm].
O	Co-ordinate system origin and location of the centre of gravity.
Ox_b, y_b, z_b	Body-axis coordinate system

Ox_i, y_i, z_i	Inertial frame coordinate system
p	Angular velocity about the Ox (roll) axis [ms^{-1}].
p_1	Initial point in space for path planning.
p_2	Waypoint in space for path planning.
p_e	Navigation frame east position [m].
p_n	Navigation frame north position [m].
p_s	Static pressure [Nm^{-2}]
P_i	Set of B�zier Control Points; First point along a path.
P_f	Final point along a path.
P_s	Specific excess power [m/s].
$P_{s,\text{nom}}$	Nominal specific excess power at a predetermined flight level [m/s].
$P_{s,\text{alt}}$	Offset specific excess power for unknown flight level derived from $P_{s,\text{nom}}$ [m/s].
q	Angular velocity about the Oy (pitch) axis [ms^{-1}].
$\bar{q}$	Dynamic pressure [kgm^{-1}].
r	Angular velocity about the Oz (yaw) axis [ms^{-1}].
r_{xy}, r_{xz}, r_{yz}	Radius of curvature about the xy , xz , and yz planes.
r_0	Position unit vector.
S	Reference area [m^2]; Laplace Variable [1/t].
T	Instantaneous thrust [N].
T_A	Thrust available [N].
T_{idle}	Idle thrust [N].
T_{max}	Maximum thrust [N].
T_{mil}	Military thrust [N].
T_R	Thrust required [N].
u	Component of velocity along the Ox axis [ms^{-1}].
$u(t)$	Controller output signal.
u_d	Demanded airspeed value [m/s].
V	Velocity [ms^{-1}].
v	Component of velocity along the Oy axis [ms^{-1}].
w	Component of velocity along the Oz axis [ms^{-1}].
X	Component of force along the Ox axis [N].

Y	Component of force along the Oy axis [N].
Z	Component of force along the Oz axis [N].
x, y, z	Cartesian co-ordinates relating to the body axes [m].
x_{cg}	reference centre-of-gravity.

Greek

α	Angle of attack [radians].
β	Side-slip angle [radians].
δ_a	Aileron deflection angle [rad].
δ_h	Elevator deflection angle [rad].
δ_{lef}	Leading edge flap deflection angle [rad].
δ_m	Particle mass.
δ_r	Rudder deflection angle [rad].
δ_{sb}	Speed brake deflection angle [rad].
δ_t	Throttle stick setting (%).
$\delta_{t_{dem}}$	Throttle stick setting demand (%).
ϵ	Internal controller position error.
$\epsilon(t)$	Controller error signal
θ	Angular displacement (pitch) about the Oy axis [radians].
κ_{xy}	Piecewise curvature of paths in the xy plane.
κ_{xz}	Piecewise curvature of paths in the xz plane.
κ_{yz}	Piecewise curvature of paths in the yz plane.
λ	Reference length [m].
Λ	Quaternion transformation matrix.
ρ	Density of air [kgm^{-3}].
τ_T	First-order time constant [s].
ϕ	Angular displacement (roll) about the Ox axis [radians].
ϕ_d	Demanded angular displacement (roll) about the Ox axis [radians].
ψ	Angular displacement (yaw) about the Oz axis [radians]
ψ_d	Demanded angular displacement (yaw) about the Oz axis [radians].

ω	Angular velocity [rad/s].
ω_{avg}	Average angular velocity [rad/s].
ω_{max}	Maximum angular velocity [rad/s].
Ω_e	Angular velocity of the Earth within an Earth axis reference frame.

Subscripts

a	Aerodynamic.
b	Co-ordinates relating to the Body axes.
b,i	body-axis when observed from the inertial frame
e	Co-ordinates relating to the Earth axes.
i	Co-ordinates relating to the Inertial axes.
g	Gravitational.

Superscripts

$\langle \cdot \rangle$	Time rate of change (first derivative).
$\langle \ddot{\cdot} \rangle$	Time rate of change (second derivative).
$\langle ' \rangle$	Axis of a system after first translation.
$\langle '' \rangle$	Axis of a system after second translation.
$\langle ''' \rangle$	Axis of a system after third translation.
$\langle \rightarrow \rangle$	Axis of a system after third translation.
$\langle ^\wedge \rangle$	Unit vector

Abbreviations

2D	Two dimensions
3D	Three dimensions
6-DoF	Six-degrees-of-freedom
ACO	Ant Colony Optimisation
AI	Artificial Intelligence
ALFUS	Autonomous Levels for Unmanned Systems
AVR	Automatic Voltage Regulator

ANN	Artificial Neural Network
BA	Bat Algorithm
BFO	Bacteria Foraging Optimisation
BTT	Bank-To-Turn
BS	Best Selector
CNN	Convolutional Neural Network
CPU	Centre Processing Unit
DNN	Deep Neural Network
DoD	Department of Defence
DQN	Deep Q-Network
EQL	Extended Q-Learning
GA	Genetic Algorithm
GUI	Graphical User Interface
HS	Harmony Search; Hybrid Selection
IQL	Improved Q-Learning
IRF	Intermediate Reference Frame
LANCA	Lightweight Affordable Novel Combat Aircraft
LEF	Leading Edge Flap
LOS	Line-of-Sight
MDPs	Markov Decision Processes
MOD	Ministry of Defence
NATO	North Atlantic Treaty Organization
NE	North-East
NEH	North-East-Heading
NIST	National Institute of Standards and Technology
NN	Neural Network
PD	Proportional-Derivative
PID	Proportional-Integral-Derivative
PIV	Proportional-Integral-Velocity
PSO	Particle Swarm Optimisation
RAF	Royal Air Force
RK4	Runge-Kutta fourth-order integration
RL	Reinforcement Learning

RS	Rank Selection
RWS	Roulette Wheel Selection
SSE	Sum-Squared-Error
TS	Tournament Selection
UAV	Unmanned Aerial Vehicle
UAS	Unmanned Aerial Systems
wrt	with respect to
WSSE	Weighted Sum-Squared-Error

Chapter 1

Introduction

This chapter discusses the motivation for conducting the research by exploring the background of autonomous flight and how advancement in technology continues to drive research and development of highly autonomous unmanned aerial vehicles (UAVs). Included is an introduction to traditional UAV designs, highlighting specific use cases, from military and commercial to hobbyists and flight enthusiasts. This is followed by an overview of currently proposed rules and regulations relating to various levels of autonomous flight, before identifying some of the current challenges associated with manned/unmanned flight operations. The chapter concludes by outlining the research focus and defining the research aim before outlining the structure of the thesis.

1.1 Motivation

Since the early 2000s, the use of unmanned aerial vehicles (UAVs), unmanned aerial systems (UAS) or drones, has seen a dramatic increase within military applications. Whilst these applications often existed as small unarmed drones deployed by soldiers on the ground for surveillance [1], the use of remotely operated drones armed with explosive payloads, has risen significantly over the last 20 years due to the *War on Terrorism*. Between January 2017 and January 2020, the United States conducted 4557 confirmed drones strikes over Afghanistan, an increase of over 250% from previous years [2].

This increase in UAV activity has contributed to a desire for more complex and intelligent autonomous systems. The US Army predicts that by the year 2035, all armed reconnaissance, attack and surveillance missions will be con-

ducted by autonomous and unmanned systems [3], with the US allocating over \$1.6 billion on UAS research projects in 2005 [4]. In 2019, the US Air Force announced plans to implement, by 2021, an artificial brain codenamed Skyborg into an autonomous fleet of AI-enabled XQ-58 Valkyrie drones [5]. This research is expected to expand through the implementation of the same artificial intelligence into a combat aircraft, with the F-16 being a primary platform for such an endeavour.

The United Kingdom has also expressed its desire for a fully autonomous loyal wingman, with the RAF awarding Boeing, Callen-Lenz and Blue Bear Systems Research, contracts for the completion of a preliminary design, codenamed Project Mosquito. The requirements demand a lightweight, affordable, novel, combat aircraft (LANCA), which is unmanned and capable of operating alongside manned aircraft during future aerial combat missions [6]. The Defence Science and Technology Laboratory (dstl), recently announced it had completed the initial one-year design phase for Project Mosquito, with the LANCA expected to be completed in 2023 [7]. Additionally, BAE Systems, Rolls Royce, Leonardo and MBDA formed Team Tempest, dedicated to developing the UK's next generation fighter, alongside the RAF and Ministry of Defence (MOD) [8]. Tempest is expected to be in service by 2035, replacing the Typhoon, with the aircraft described as a "plug and play fighter, where hardware and software can be easily interchanged depending on the mission" [9].

In 2019, Northwestern Polytechnic University in Xi'an China unveiled an autonomous jet-powered drone, LJ-1 at the MAKS Air Show in Russia. LJ-1 was designed to imitate third and fourth generation fighters with the objective of flying and cooperating alongside manned fighters, and self-sacrifice if required. Furthermore, LJ-1 can be sent on one-way solo missions, essentially acting as a cruise missile with explosive payload [10].

1.2 UAV Designs

Generally, there are two main types of UAV design; multi-rotor and fixed-wing. Multi-rotor designs have the ability to take-off and land vertically, whilst also occupying a smaller footprint during transportation and storage. In addition, they

have better stability and controllability, due to their hover mechanics and the ability to instantaneously manoeuvre in any direction, making them favourable amongst commercial and aviation enthusiasts.



FIGURE 1.1 MQ-1 Predator, armed with AGM-114 Hellfire missiles, piloted by Lt. Col. Scott Miller on a combat mission over southern Afghanistan (US Air Force Photo / Lt. Col. Leslie Pratt)

Despite multi-rotor UAVs having many desirable characteristics, they are very rarely considered for medium-to-large-scale military applications, due to a limited operating range, low speed and low altitude flight envelope. Since military UAVs are often launched from an already established military compound, or from a naval vessel, the increased footprint, take-off and landing distances associated with fixed-wing designs are not considered major drawbacks. Additionally, since many of these launch sites are often located in remote areas, the endurance capabilities and airspeed superiority of fixed-wing design are one of the deciding factors [11]. Finally, during a loss of power, a fixed-wing UAV has the potential to glide and be recovered, whereas a multi-rotor design will simply fall out of the sky. For these reasons, many prominent US military UAVs such as the MQ-1 Predator (Figure 1.1) and MQ-9 Reaper are all fixed-wing designs, as is the Chinese LJ-1 and the development of Tempest and LANCA by the UK.

1.3 Requirements of an Autonomous System

An autonomous systems framework commissioned by the North Atlantic Treaty Organization (NATO), outlines various levels of autonomy, and describes the complexity of the required subsystems based on mission specific applications. This framework indicates that any UAV operating autonomously and with little human interaction, must incorporate a mission planning framework capable of performing online trajectory optimisation [12], whilst also considering specific mission objectives. For example, a UAV operating as fly-by-signature, must perform online trajectory optimisation with a consideration for reducing exposure to hostile sensors [12]. In addition, the aircraft must have a complex mission management system, enabling re-planning of mission profiles whilst also maintaining a contingency plan should the dynamics of the mission change. This contingency plan must also consider external phenomena, such as a dynamically changing environment or changes associated with situational awareness and collision avoidance [12].

Similarly, the National Institute of Standards and Technology (NIST) proposed the *Autonomous Levels for Unmanned Systems* framework known as ALFUS. This framework categorises four different levels of autonomy based on mission complexity and environmental difficulty, as well as the required level of human interaction. ALFUS provides an extensive and exhaustive overview of mission profiles and applications, including the coordination, collaboration, and cooperation between manned and unmanned systems, and how they can work together by sharing data. This level of data sharing could be coordinates for manoeuvre(s) or the acquisition of intelligence to enable a mission to be performed efficiently and synergistically [13]. The four levels of autonomy described by ALFUS are as follows.

1. Fully autonomous - the unmanned system is expected to accomplish a specific mission without requiring any human interaction.
2. Semi autonomous - coordination of an unmanned systems mission profile is conducted by a human operator before being applied directly to the mission planner onboard the vehicle.

3. Teleoperation - a human operator has access to and is monitoring data acquired by the autonomous system. This data could be video feedback or onboard sensory data relating to the status of the mission.
4. Remote control - a human operator directly influences the control surfaces of the aircraft on a continuous basis.

1.4 Challenges

The US Department of Defence (DoD) has outlined future objectives and challenges for the field of autonomy when applied to unmanned systems. The objectives exploit the rapidly growing field of artificial intelligence, and how improvements to machine learning techniques can aid in the development of enhanced navigation, perception, obstacle detection, avoidance and swarming behaviours [14]. The challenges associated with obtaining these objectives, focuses on increasing the trust factor between autonomous systems and humans, through improvements to guidance, navigation and validation of the online decision making process [14]. By implementing artificial intelligence into the decision making process during mission planning, navigation and guidance, the DoD aims to incorporate fully unmanned aerial operations by the year 2029, including leader-followers task, with highly-autonomous armed-wingman/teammate interactions expected by 2042 [14].

When considering any UAV or UAS with an emphasis towards achieving a high level of autonomy, both frameworks in Protti and Barzan [12] and NIST [13] emphasise online mission planning as a primary objective, with path planning and trajectory optimisation the initial focus. In addition, based on future autonomous system roadmaps produced by the DoD [4, 3, 14], autonomy is the future of ground-based and aerial-warfare, with the main challenge being to increase the human trust factor. The human trust factor can be increased by reassuring humans that autonomous systems operate safely and reliably, particularly during manned/unmanned interaction. Ensuring an autonomous system can navigate safely and reliably, is a function of path planning as well as the mission profile's online decision making process. These factors all emphasise the importance of developing autonomous control systems with robust path

planning, navigation and guidance.

1.5 Research Aim and Objectives

Based on the content in Section 1.1-1.4, the demand for highly autonomous systems is increasing. Autonomous regulatory frameworks govern the various levels of applied autonomy based on; the mission specifications of an autonomous system; the device requirements for the control system; and the level of necessary human interaction. These frameworks emphasise the importance of a robust mission planning framework for highly autonomous systems, capable of performing online path planning during dynamically changing missions. Furthermore, the challenges outlined by the US DoD indicate that for humans and autonomous systems to work with synchronicity, the autonomous system must demonstrate safe, reliable and robust operation to increase the human trust factor. To address these issues, this thesis has the following aim.

Develop an online path planning algorithm enabling continuously flyable paths to be developed for a fixed-wing autonomous fighter jet during cooperative flight.

Prior to outlining the objectives of the thesis, a review of literature is performed in Chapter 2 to understand current limitations in path finding and path planning for highly autonomous systems. The objectives of the thesis are outlined in Section 2.6.

1.6 Methodology

The research presented in this thesis develops a path planning solution for a fixed-wing highly-maneuvrable fast jet. Since the path planning algorithm accounts for the dynamics of the aircraft, it was important that the computational model used for validation was able to replicate the dynamics of the aircraft with high-fidelity. For this reason, a non-linear computational model of the F-16 Fighting Falcon was created programmatically in MATLAB. However, due

to the non-linear dynamics and non-linear aerodynamics, obtaining an analytical solution to the equations of motion is a complex problem [15], meaning a numerical simulation-based methodology was implemented, with integration of the equations of motion performed using a fourth-order Runge-Kutta algorithm.

Validation of the path planning solution was quantitatively analysed for static waypoint targets using statistical validation through Monté Carlo simulation, with a qualitative analysis performed for dynamically moving waypoints, where specific mission profiles were carefully formulated.

1.6.1 Limitation of the Methodology

One of the main limitations from the proposed methodology occurs for guaranteeing dynamically feasible path planning solutions. Since an analytical solution is not derived for the equations of the motion or during path planning, the accuracy of the solution obtained through numerical simulation can be attributed to the size of the time-step used during numerical integration.

Secondly, when measuring the performance of the aircraft during dynamic waypoint simulations, the results were qualitatively analysed as opposed to quantitatively. This meant that performing an almost exhaustive overview of simulation scenarios was not possible, with a finite number of mission profiles generated to test specific aircraft path planning characteristics. Whilst this statistical analysis does provide a comprehensive overview of system certainty and performance, it cannot be guaranteed with 100% certainty without a systematic probabilistic modelling technique.

1.7 Implementation

To conduct this research, it was discussed in Section 1.6 that a computational model representative of the F-16 Fighting Falcon was created. The F-16 was selected as the control surface deflections and engine models are published by Nguyen et al. [16], together with a series of non-linear subsonic aerodynamic datasets. In addition, this aircraft model has been well researched and validated [17, 18, 19, 20]. An abstracted overview of the aircraft model can be seen in Figure 1.2, where the model has been considered as four parts: (i) settings and

configuration of the simulation, aircraft and environment, (ii) aircraft dynamics, (iii) flight control system, and (iv) the flight systems interface.

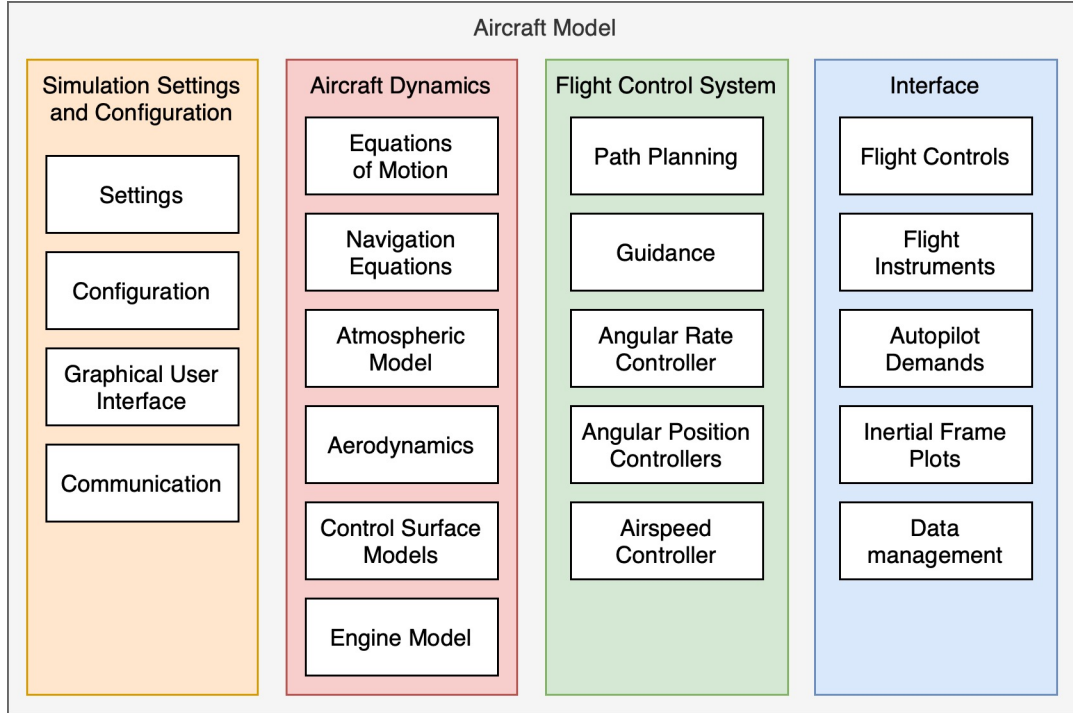


FIGURE 1.2 Aircraft model structure overview

The aircraft model was developed programmatically in MATLAB and did not use any Simulink integration or processes. Since the purpose of the research focuses on real-time flight, using MATLAB's built-in toolboxes was not possible, since these toolboxes contain many unnecessary features that aid usability at the cost of increased computational overheads. Therefore computational processes such as numerical integration of the equations of motion, control surface model deflection angles, and engine model thrust integration, were performed using a custom Runge-Kutta fourth-order (RK4) solver. Additionally, processes that required multi-dimensional linear interpolation algorithms, such as the aerodynamics, atmospheric data, approximated trim values and the specific excess power analysis utilised a custom multi-dimensional linear interpolation algorithm which followed logic-based matrix operations as opposed to computationally expensive sequential iterative statements. Additionally, development of any tools used for the purpose of this research, such as meta-heuristic optimisation algorithms were all programmatically developed and did not utilise any pre-built MATLAB toolboxes or functions. Not only did this enable

the codebase to run at a higher frequency, it also aids in compatibility between software releases, as well as transitions towards other programming languages.

To enable real-time piloted inputs, a manual flight control system was implemented which interfaced directly with a set of flight controls (yoke, throttle and rudder pedals) or a PlayStation 4 controller. Inputs are received and applied directly to the angular rate controllers (discussed in Chapter 4), with a graphical user interface (GUI) developed to aid the pilot. Although not discussed in the thesis, the GUI consisted of a 2D artificial horizon with full 360 degree wrapping of the roll and pitch axes, multiple custom dials that use circular mechanics to move the needle, a heading indicator, an inertial frame representation of the environment, a radar system with live data readouts from communicating aircraft, and a flight control systems panel for interacting and adjusting the onboard autopilots.

1.7.1 Limitation of the Implementation

During the derivation of the equations of motion, the aircraft was assumed to be of constant mass and rigid-body. This indicates that the mass of the aircraft does not change during flight and the body does not deform or change shape, meaning the location of the centre of the gravity (C_g) and the products and moments of inertia remain constant. By assuming the aircraft to be of constant mass and rigid-body, the derivation of the equations of motion was performed using Newton's second law of motion with constant mass, simplifying the expansion and component buildup of the forces and moments.

Although these assumptions are valid, they directly impact the practicality and the validity of the methods proposed in this thesis, whilst not disregarding their significance. For example, the flight control system was developed based on a single configuration, where the reference centre-of-gravity (x_{cg}) location was 35% of the mean aerodynamic chord ($\bar{c}$) [16]. The location of the centre-of-gravity is an important measurement, as this is the point where the mass of the body is assumed concentrated, and where all forces and moments act about. Since the longitudinal dynamics of the aircraft model are controlled using an elevator control surface mounted on the tail of the aircraft, shifting the C_g further aft of the reference location increases elevator authority due to a reduced moment

arm, whereby creating a more manoeuvrable aircraft at the expense of reduced stability. Since the flight control system was developed under the assumption of a constant mass and rigid-body, with a fixed x_{cg} location, if the value of x_{cg} was to be adjusted it is unknown whether the flight control system can guarantee closed-loop stability and adequate dynamic performance throughout the entire operating envelop for this flight regime. Furthermore, the fitness function used during path optimisation required knowledge of the aircraft's dynamic limits. If changes are made to the x_{cg} , the dynamical limits of the aircraft applied to the fitness function will no longer be representative of the true values, meaning that optimisation of the paths produced cannot be guaranteed to be dynamically achievable.

1.8 Outline of the Thesis

Due to the length of the thesis and the multi-disciplinary nature of its components, this thesis is written as a linear timeline of events, where each chapter represents an individual body of work. Whilst this may result in an unorthodox structure at various points, the aim of this is to give the reader clear and concise information when making decisions and creating solutions.

The remainder of the thesis discusses the follow topics:

Chapter 2. Review of Literature

Introduces the topic of path planning with an emphasis towards path finding and path generation algorithms, before discussing how artificial intelligence can be incorporated into obtaining a solution. The chapter ends by discussing the research focus and outlining the objectives of the research.

Chapter 3. Flight Physics Model

Introduces the flight physics model used for computational studies throughout the thesis. This includes axis translations and derivation of the translational and rotational equations of motion. The aerodynamic, control surface, engine and atmospheric environment models are described together with the operational flight envelope.

Chapter 4. Flight Control System

Introduces the inner loop flight control system used for controlling the translational and rotational motion of the flight physics model. This includes a literature review into proportional-integral-derivative (PID) controller architecture, with an analysis into heuristic and meta-heuristic optimisation algorithms for controller tuning. The autopilot systems discussed in this chapter refer to low level controllers which interface directly with the control surface models and engine models. Therefore, at times throughout the thesis additional autopilot systems are introduced. For example, the distance controller in Chapter 6 and climb rate controller in Chapter 7. These autopilots have not been included in Chapter 4 since they were created for the purpose of research explained in later chapters in the thesis and are not considered to be part of the inner loop flight control system.

Chapter 5. Path Planning Algorithm

Introduces the path planning algorithm, and how the selected solution is appropriate for determining continuously flyable and manoeuvrable paths. At the end of the chapter, the guidance control system, which allows the computational aircraft model to autonomously follow a three-dimensional path, is discussed.

Chapter 6. Constant-Altitude Path Planning

Introduces constant-altitude path optimisation. This includes the method of optimisation and the developed fitness function, allowing paths to be quantitatively measured for constant-altitude flight. In addition, a classification deep neural network is developed and trained using the path optimisation algorithm. This allows constant-altitude path planning to be conducted online during real-time simulations.

Chapter 7. Variable-Altitude Path Planning

Introduces variable-altitude path optimisation. Optimisation is performed using the same methodology outlined in Chapter 6. However, a climb rate analysis is performed, allowing specific excess power-based climb predictions to be obtained for altitude-changing paths. In addition, a regression

deep neural network is trained using the path optimisation algorithm. This allows variable-altitude path planning to be conducted online during real-time simulations.

Chapter 8. Conclusions

Highlights the conclusions made from each of the individual chapters, forming the closing statement and summing the overall findings of the thesis, before introducing future work.

Due to the nature of each chapter containing individual bodies of work, a discussion and summary is performed at the end of each chapter which concludes the findings from within the chapter. In addition, where appropriate, a chapter begins by introducing relevant literature by performing a small overview of current techniques. This enables the literature review in Chapter 3 to focus solely on the literature relating to the fundamental principles of the thesis, with relevant literature introduced within individual chapters.

Publications arising from this work

Jordan Hamilton, David Galvão Wall, Alistair J. Saddington, and John T. Economou. Hypersurface normalised gain-scheduled controller for a non-linear 6-DOF fast jet. *Aerospace Science and Technology*, 106:106155, November 2020. doi: 10.1016/j.ast.2020.106155. URL <https://doi.org/10.1016/j.ast.2020.106155>

Chapter 2

Literature Review

This chapter reviews the literature relevant to achieving the research aim. It initially focuses on path planning frameworks, specifically how to determine collision-free paths using path finding algorithms and then how collision-free paths are dynamically and kinematically constrained using path smoothing algorithms. An analysis into each method is conducted individually, allowing a more comprehensive overview of current problems to be identified. The computational burdens associated with current methodologies are identified and an assessment of various artificial intelligence algorithms is performed. The chapter concludes with a summary, highlighting current problems identified within the literature review, followed by the objectives of the thesis required to achieve the research aim.

2.1 Path Finding Algorithms

Any autonomous vehicle that requires mobility, needs to perform path planning [21]. Path planning is a subsystem of an autonomous system's mission planner, and requires consideration of path finding and collision avoidance algorithms, as well as a path generation or path smoothing algorithm.

Path finding algorithms are a vital aspect for any autonomous system, with the internal characteristics of the algorithm tailored towards specific applications. For example, low-altitude or ground-based autonomous vehicles are likely to encounter static objects, such as buildings, as well other mobile objects, such as manned and unmanned systems. For conventional autonomous systems, there exists a need to navigate around these obstacles to avoid a collision.

This means that path finding and collision avoidance algorithms must utilise geographic information relating to the operating environment, or develop an overview of the environment using information obtained from hardware located onboard or off-board the system. Autonomous systems operating at mid-to-high altitude, such as military jets and weaponised or reconnaissance UAVs, are less likely to encounter static objects during conventional missions, with the exception being during take-off and landing. Therefore, path finding and collision avoidance algorithms focus on tracking dynamically moving objects for the purpose of producing spatially de-conflicted paths. Additionally, path finding algorithms can also be used to determine optimal paths through unknown environments, with an emphasis towards reducing the systems exposure to hostile sensors or dangerous territories.

In the field of robotics, path finding and collision avoidance algorithms are used to determine the ideal and collision-free path from one location to another. Popular path finding algorithms include the A-Star algorithm [22], Dijkstra's algorithm [23], and Bellman Ford's algorithm [24]. Generally, these algorithms are applied to node-based networks derived from grid-world-like environments. However, techniques exist such as Voronoi diagrams [21] using Delaunay triangulation [25] for developing node-based roadmaps on non-grid-world environments. Considering the operating environment as a node-based roadmap enables paths generated between identified nodes, allowing a vehicle to navigate from point A to point B, potentially using many intermediate points along the way.

Voronoi diagrams are used for developing node-based roadmaps of environments. By identifying objects within an environment, Delauney triangulation is used to identify common locations known as Voronoi vertices. These Voronoi vertices are then connected creating a partitioned roadmap, as depicted in Figure 2.1. This technique was used by Bortoff [26] for path planning of a fixed-wing UAV, where the objects forming the Voronoi diagram are obtained from radar data. However, one of the limitations associated with path planning using a Voronoi diagram, occurs due to the lack of dynamic consideration applied when computing the Voronoi vertices, resulting in rough-cut paths. For these reasons, Bortoff [26] assigned virtual masses to each object, which in turn, attracted and

repelled the path, inducing curvature. Similarly, Qu et al. [25] describes how artificial potential field theory was applied for Voronoi path smoothing, whilst Liu and Zhang [27] utilised an iterative path smoothing algorithm and Judd and McLain [28] used cubic spline interpolation to connect each of the Voronoi line segments. Rather than smoothing a rough-cut path between the Voronoi vertices, Ho and Liu [29] used these vertices as locations for the internal control points of a piecewise linear composite Bézier curve, ensuring the paths were curvature bounded.

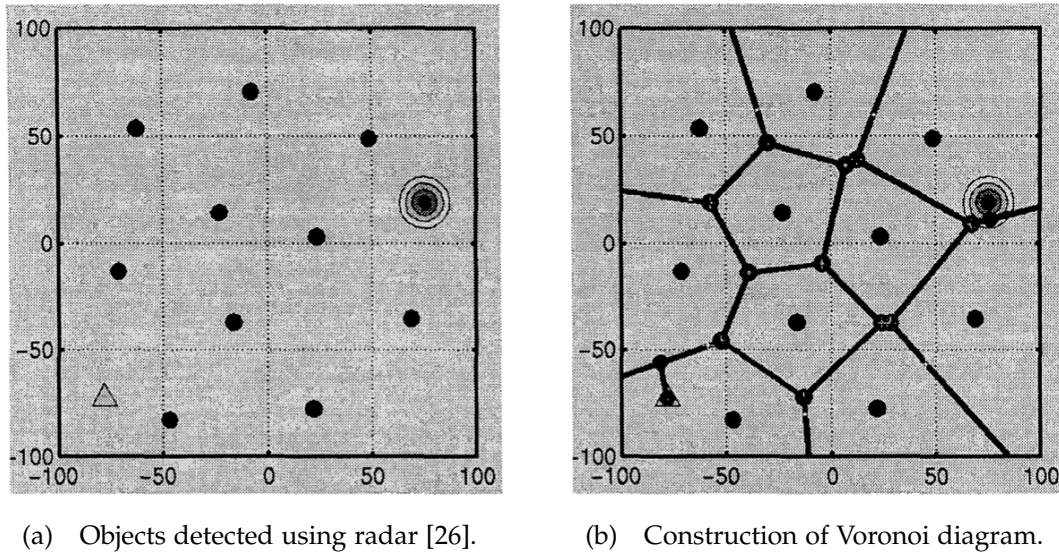


FIGURE 2.1 Construction of Voronoi diagram for UAV path planning using objects detected through radar [26].

Whilst Qu et al. [25] and Ho and Liu [29] implemented Dijkstra's algorithm for locating the shortest collision-free path through an environment, by factoring additional environmental parameters paths can be produced based on many characteristics. This was demonstrated by Deng et al. [30] for path planning of a ground-based transportation vehicle operating in uncertain environments. Using fuzzy logic, uncertainty weightings were applied to each node corresponding to environmental conditions, allowing the most suitable paths to be found as opposed to simply finding the path with the shortest distance. Similarly, in the work of Wang and Zhou [31] nodes are assigned probabilistic measures of an aircraft's survival chances, enabling Dijkstra's algorithm to produce an optimal path that reduced exposure to enemy radar.

Although Voronoi diagrams and Dijkstra's algorithm are amongst the most established path finding algorithms in the field of mobile robotics, additional algorithms exist capable of performing similar actions. For example, a genetic algorithm (GA) was used by Cui et al. [32] for trajectory optimisation of a multi-rotor UAV conducting power transmission line inspections and by Cui et al. [32] for solving the global optimisation problem for a fixed-wing UAV, whereas Hao et al. [33] used graph theory and a multi-colony ant algorithm for aircraft path planning in a grid-world environment.

Based on the analysed literature, it is evident that path finding algorithms are a well established technique, and although they are primarily aimed at ground-based vehicles [25, 29, 30], in many cases they have been successfully adapted for aerial-based applications [31, 32, 33]. In addition, by implementing artificial intelligence, path finding can be simultaneously produced for multiple autonomous systems working together to accomplish a common goal [34, 35].

Regardless which path finding algorithm is used, there is no guarantee that the path produced is flyable, drivable or manoeuvrable by an autonomous system [21], particularly when considering fixed-wing UAVs. For a path to be continuously flyable, the path must adhere to the kinematic and dynamic constraints of the autonomous vehicle, whilst also relying on the curvature of the path to gradually rotate the motion vector, allowing for smooth and continuous flight. For these reasons, the focus of this review transitioned from path finding algorithms towards path generation and path smoothing algorithms.

2.2 Path Smoothing Algorithms

Path generation and path smoothing algorithms are applied after a collision-free path has been identified, as shown in Figure 2.2. Techniques such as potential field theory [25], virtual mass manipulation [26], and cubic splines [28] are potential and interesting solutions. However, in recent years the use of Bézier curves has seen an increase in popularity, as well a geometric approach to geodesic path generation, known as Dubins paths [36].

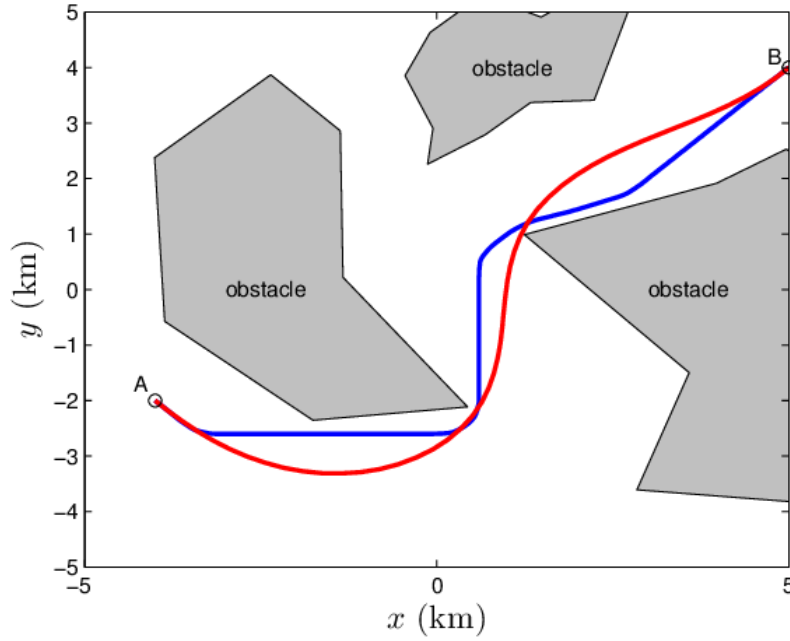
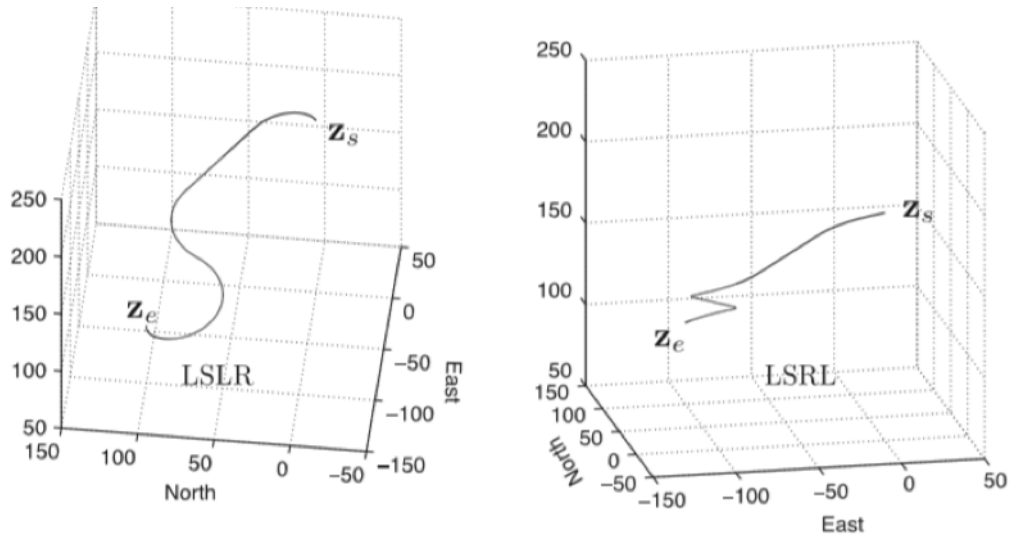


FIGURE 2.2 An example of a collision-free path (blue) generated using a path finding algorithm; this is then adapted (red) using a path smoothing algorithm [37].

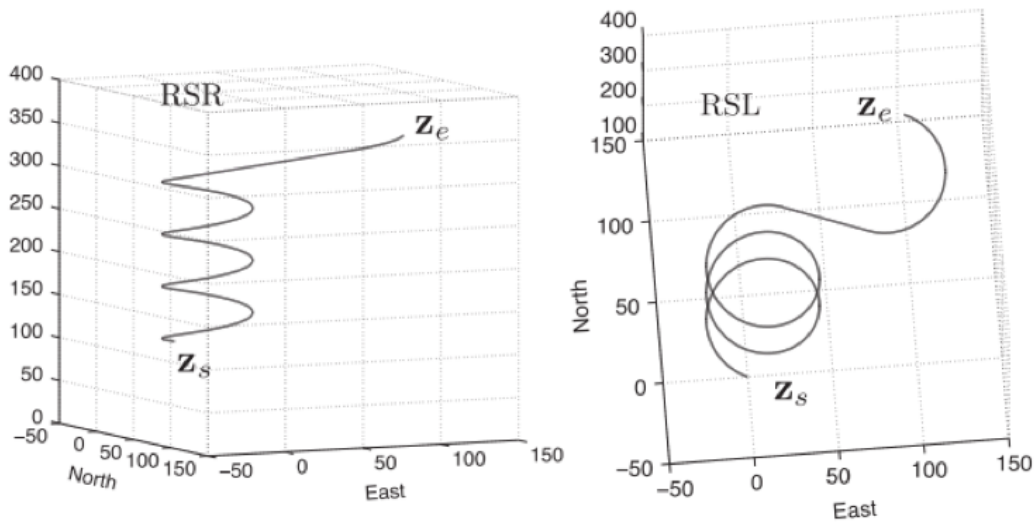
2.2.1 Dubins Paths

Dubins paths, first proposed by Lester Dubins [36], are a geometric approach for determining the geodesic trajectories between two points. The geometric properties of Dubins paths make them desirable for ground-based robots which can only move forwards [38, 39, 40]. As these applications were largely conducted in the x - y plane, this gave rise to Dubins paths often being referred to as a “Dubins car” [41]. Although initially developed for vehicles that can only move forwards, Dubins paths have also been used for path planning of ground-based vehicles that can move forwards and backwards [42, 43], with a focus towards automated car parking applications.

Although the majority of early research into Dubins paths was conducted on ground-based vehicles, many researchers have modified the path generation process to incorporate altitude-changing manoeuvres for UAV applications [21, 41, 45]. Owen et al. [44] converted a two-dimensional trajectory into three-dimensions by considering changes to the altitude as three categories: (i) low-altitude changes; (ii) medium-altitude changes; (iii) high-altitude changes. Dur-



(a) Medium altitude change manoeuvre.



(b) High altitude change manoeuvre with helical climb.

FIGURE 2.3 Medium and high altitude changing Dubins trajectories proposed by Owen et al. [44].

ing low-altitude-changing manoeuvres a constant climb was administered over the duration of the entire path. This constant-climb approach was also implemented by Cai et al. [46] for use on an underwater autonomous vehicle. However, if the required climb rate was not feasible, an intermediate helical climb was applied. This helical climb was only applied to high-altitude change scenarios, allowing the aircraft to increase altitude until a low-altitude changing manoeuvre could be dynamically achieved (Figure 2.3b). Finally, a medium altitude-change manoeuvre was classified if the total altitude change could not

be achieved by performing a low-altitude manoeuvre but was not large enough to complete a single full helical turn climb. Therefore, a third circular section was introduced, aligned at the common cotangent to the initial Dubins radial section, but in the opposite direction (Figure 2.3a).

Although the research conducted by Owen et al. [44] was extensive and provided a robust solution to all possible scenarios, the paths produced are dynamically restrictive, with rotations occurring solely about the x - y plane. When considering the dynamic capabilities of a highly manoeuvrable fast jet, performing a multi-turn helical climb for the sole purpose of increasing the elevation level to allow a secondary low-altitude change manoeuvre, could be considered as an inefficient use of energy. Additionally, G^1 continuity concerns were raised by Cai et al. [46] for low-altitude change manoeuvres due to the applied constant climb. This concern could cause paths deemed achievable in two-dimensions to no longer be achievable when converted to three dimensions.

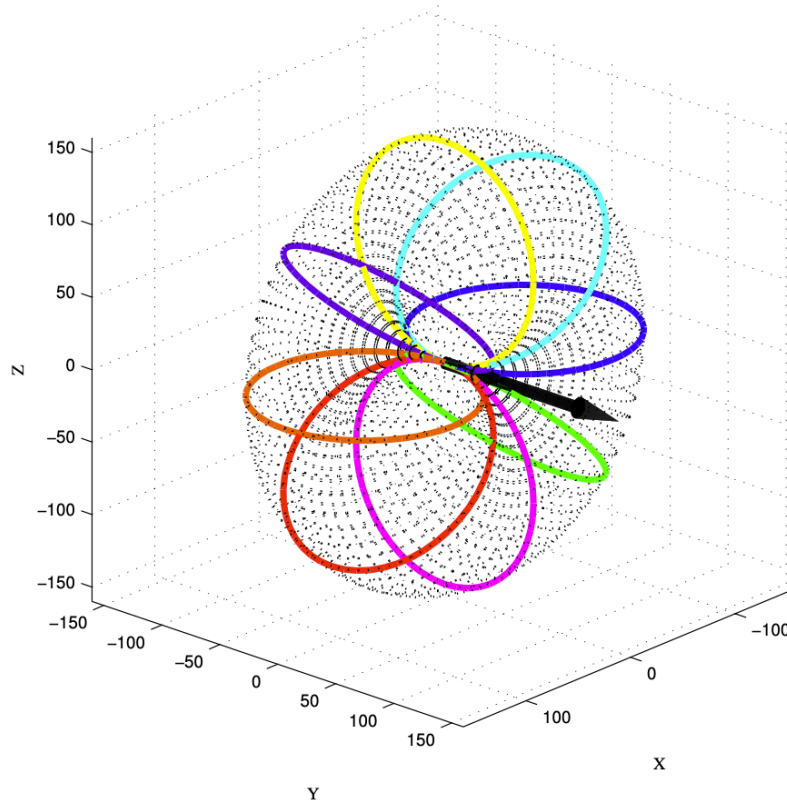


FIGURE 2.4 Primitive turn rotations about a torus based on the flight path vector [47].

Shanmugavel [21] used an initial coplanar rotation to align the UAV onto a two-dimensional manoeuvrable plane. The rotation was defined by the waypoint

target's tangential vector, based on the sight line vector. Whilst this method produces more desirable three-dimensional trajectories when compared to Owen et al. [44], the application is focused towards multi-rotor UAVs that possess the ability to rotate the motion vector whilst maintaining altitude and position. Rather than performing an initial coplanar rotation, Hurley et al. [47] described the radial sections of the Dubins path as a three-dimensional torus, aligned to the motion vector of the aircraft (Figure 2.4). Although this method does solve many manoeuvrability issues, the solution is computationally expensive due to the infinite number of potential outcomes.

2.2.2 Bézier Curves

Bézier curves are a parametric curve, computed by solving the Bernstein polynomial [48]. The Bernstein polynomial, developed by Sergei Natanovich Bernstein in the early 1900s, is a constructive proof of Weierstrass theorem [49] from 1885. This theorem states that *"any continuous function on a finite subinterval of the real line can be uniformly approximated by a polynomial"*. Due to limitations in computational methodologies in the early 20th century, Bernstein polynomials remained theoretical until two French engineers, Paul de Faget de Casteljaeu and Pierre Etienne Bézier, devised a method for automotive designers to develop "free-form" shapes [50]. The work by Casteljaeu and Bézier led to the Bernstein form.

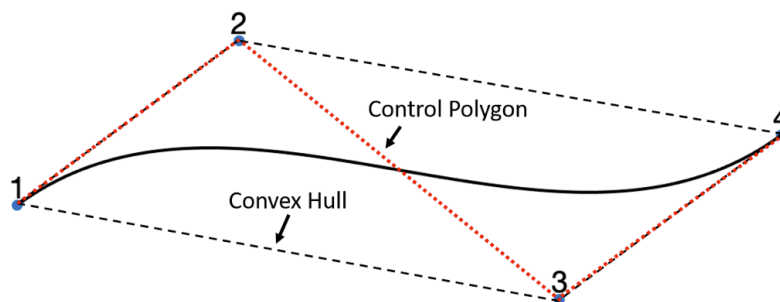
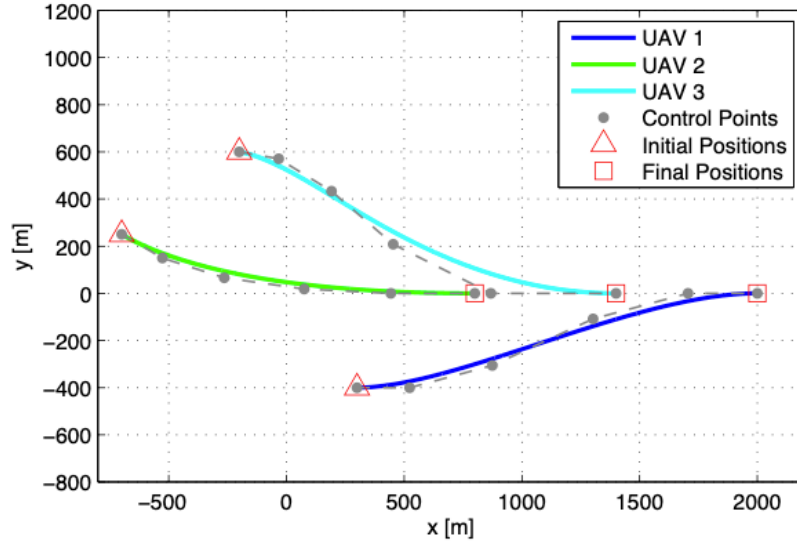


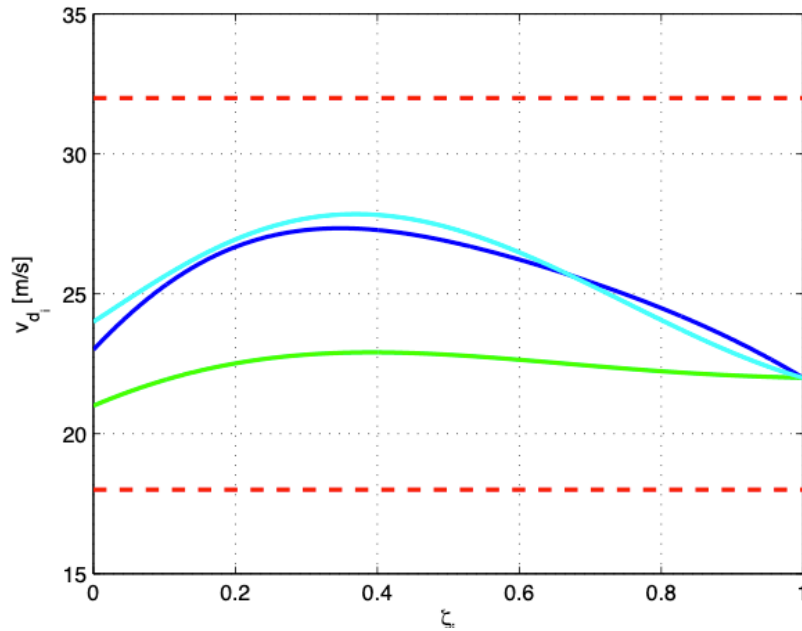
FIGURE 2.5 Bézier curve convex hull defined by four control points

In more recent years, Bézier polynomials have been used in computer graphics, animations and during the design of type fonts [51] due to their constructive properties. One of the most important properties being that the polynomial is

formed completely within a convex hull defined by its own control points (Figure 2.5), where the first and last control point are defined as the initial and final conditions of the curve. These properties are extremely attractive for path planning as the control points are able to not only manipulate the curvature profile, but also apply bounded limits to where the polynomial can be generated.



(a) Two-dimensional flight paths.



(b) Speed profiles of the UAVs.

FIGURE 2.6 Generated trajectories for a three-vehicle system with spatial separation constraints imposed [51].

Over the past decade, publications from the University of Illinois have utilised the desirable characteristics of Bézier curves and the Bernstein polynomial for the development of continuously flyable trajectories for UAV applications. Choe et al. [51] developed a trajectory generation framework for three low-speed UAVs operating in an urban environment. The environment brought the UAVs within close proximity to humans on the ground, emphasising the importance of ensuring that the UAVs remain stable during operation by producing paths that are continuously flyable, and with spatial deconfliction (Figure 2.6a). Despite the authors making numerous references to deriving the acceleration profile of the proposed paths, paths were deemed dynamically achievable and continuously flyable based solely on the velocity profile (Figure 2.6b). One noticeable concern with the results published by Choe et al. [51] occurs due to an invalidated timing law for the third UAV, denying the reader the ability to accurately replicate the results. The issue regarding the timing law discrepancy was addressed and amended by [52].

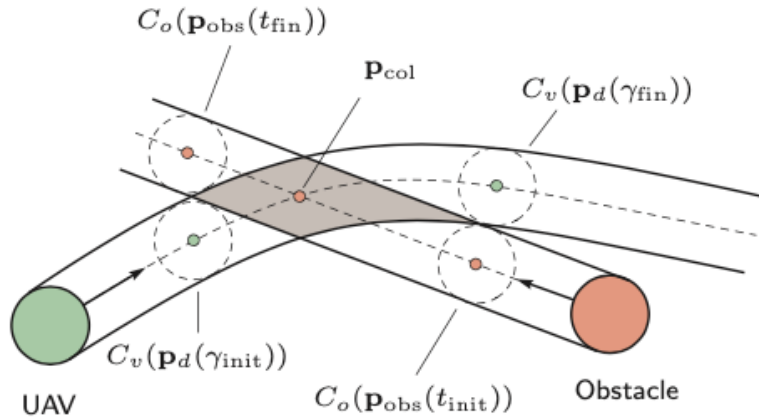


FIGURE 2.7 Collision scenario between a UAV and linearly-pathing obstacle in [53].

Cichella et al. [53] extended the framework developed by Choe et al. [51] allowing UAVs with paths not spatially or temporally de-conflicted to perform an online velocity adjustment. The applied collision avoidance algorithm worked in three stages: (i) detection; (ii) analysis; (iii) avoidance. An impending collision was detected if a UAV's flight path, and the flight path of an unknown obstacle, converge on the same region in space during the same period (Figure 2.7). To avoid a collision, the UAV reduces its velocity, allowing the obstacle

to pass through the conflicted space ahead of the UAV. However, the collision detection algorithm was only valid if the obstacle's path was observable and predictable, with demonstrations only conducted on linearly-pathing and constant-velocity obstacles. In addition, when a velocity change was performed by a UAV, the same velocity change occurred for all cooperating UAVs, indicating that individual UAVs were not operating fully-autonomously, with all computations being performed and administered by a single processor.

Although Choe et al. [51] and Cichella et al. [53] both produced trajectory generation and collision avoidance frameworks for UAVs, the trajectories were limited to constant-altitude flight. This was remedied by Choe et al. [52], where three fixed-wing UAVs were demonstrated to have altitude-changing path planning capabilities. However, the demonstrated UAV scenarios imply that the headings of each UAV are locked, indicating trajectory optimisation was performed longitudinally. Additionally, similarly to Choe et al. [51] and Cichella et al. [53], trajectory optimisation remained offline, meaning that paths between each UAV were assured to be spatially and temporally de-conflicted a priori, with no online path re-planning or velocity-adjusting collision avoidance manoeuvres performed.

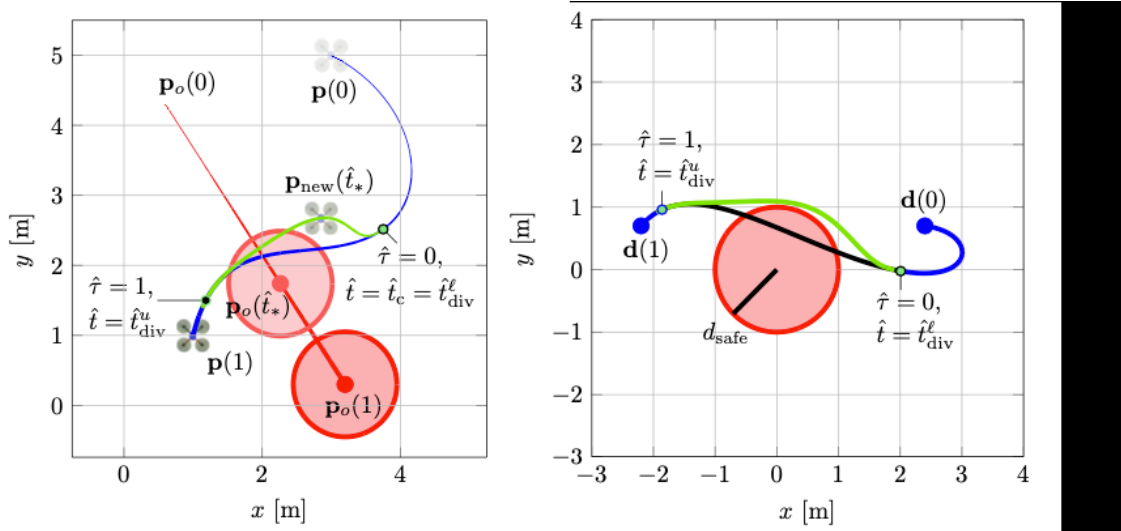


FIGURE 2.8 Illustration of the exact detour curve calculated and added to the trajectory [54].

Partial path re-planning was performed by Mehdi et al. [54, 55] for a multi-rotor UAV, using de Casteljaou's algorithm to sub-divide the originally offline-

generated trajectory and implement a detour around an identified object. However, similarly to Cichella et al. [53], it was assumed that the obstacle's direction of travel and velocity was predictable and constant. Although the authors didn't make it clear whether or not the applied detour is generated online, this method demonstrated the capabilities of de Casteljau's algorithm to sub-divide a Bézier curve, allowing some form of online manipulation (Figure 2.8).

Whilst Choe et al. [51, 52, 56] demonstrated the capabilities of Bézier curves for UAV path planning, the solutions have failed to achieve many of the requirements for autonomous systems outlined in Section 1.3. Both Protti and Barzan [12] and National Institute of Standards and Technology [13] indicate the mission planner for an autonomous system must incorporate online optimisation with full path planning, as well as maintaining a contingency mission plan. Additionally, the proposed solutions were only demonstrated on simple and routine manoeuvres, with no measure of the solution robustness during conventional flight missions. Finally, when determining whether a path is dynamically feasible and continuously flyable, the velocity profile was the only state dynamic used, with no association to the vehicle's turn-rates or acceleration limits. Although an element of online re-planning was performed by Mehdi et al. [54, 55], the initial trajectories were optimised offline and limited to constant-altitude flight, with full path re-planning not achievable.

When considering using a Bézier curve for path planning, it is apparent that there is not one configuration of Bézier curve that produces optimal solutions for every application. A second-order Bézier curve was used by Song et al. [57], third-order by Yang and Sukkarieh [58] and fifth-order by Choe et al. [51]. To understand the differences between various orders of Bézier curves, a study was performed by Bae et al. [59] using third- and fifth-order Bézier curves as input signals for the steering controller of an autonomous ground-based vehicle. The research focused on how the curvature profile of the Bézier impacts the performance of the controller during two-dimensional manoeuvres, measured in the lateral-axis. Although the authors indicated that visually the third-order Bézier was not too dissimilar from the fifth-order, differentiation of an already low-order polynomial resulted in abrupt changes to the initial and end conditions. When applied to the steering controller, the lateral movement was deemed

“not suitable for comfortable driving” [59].

2.3 Summary of Path Generation Algorithms

When selecting a path generation and path smoothing algorithm, a number of factors need to be considered. Since many autonomous vehicles have limited computing capabilities, they require a computationally-efficient path planning algorithm [52]. Although Bézier curves are regarded as being a computationally-efficient algorithm [51, 52], they require trajectory optimisation to ensure paths satisfy the kinematic and dynamic constraints of the vehicle. This causes many issues for online path planning applications, since bio-inspired or meta-heuristic optimisation techniques are generally not fast enough to be considered online [25, 60]. In addition, the dimensionality of optimisation is dependant on the order of the Bernstein polynomial, since optimisation occurs by optimally placing the internal control points. As the order of the polynomial increases, so too does the dimensionality of the optimisation problem. This was the reason why offline trajectory optimisation of fifth-order Bézier curves was performed by Choe et al. [51, 56].

Since Dubins paths are a geometric approach to path planning, they do not require trajectory optimisation because the radial sections of the path have dimensions determined by the dynamic limits of the vehicle. This makes Dubins paths an ideal solution for producing dynamically achievable paths for autonomous systems. However, Dubins paths are formed by concatenating multiple individual segments and hence discontinuities can be present at joining intersections. For this reason, Shanmugavel [21], Shanmugavel et al. [45], Kito et al. [61], and Fraichard and Scheuer [62] combined Dubins paths with clothoids, allowing a smoother transition whilst maintaining a continuous curvature. However, the computational costs associated with clothoids is higher than obtaining a solution to a parametric curve, since they require a solution to the Fresnel integrals [58].

Based on the reviewed literature, it is evident that path finding algorithms such as Dijkstra, A-Star and Voronoi diagrams are a well defined and established area of research. Although these path finding algorithms enable collision-free

paths to be identified online and in cluttered environments, there is no guarantee that paths are dynamically achievable, leading many researchers to incorporate additional path smoothing or path generation algorithms. Although path smoothing techniques such as Dubins paths and Bézier curves do enable paths to be dynamically achievable and continuously flyable, the reviewed literature has not demonstrated robust operation during conventional flight scenarios. In addition, trajectory optimisation is largely performed offline due to computational limitations. However, advancements in the field of artificial intelligence and machine learning has enabled complex mathematical systems to be represented by artificial intelligence algorithms, allowing applications to be performed online.

2.4 Artificial Intelligence

Artificial intelligence is a component of computer science, focusing on how machines demonstrate intelligence [63]. Generally, intelligence is implemented using mathematical algorithms, often describing behaviour characteristics exhibited by biological entities. Many popular artificial intelligence algorithms include: artificial neural network (ANNs); reinforcement learning (RL); support vector machines; and fuzzy logic.

2.4.1 Neural Networks

Artificial neural networks are systems motivated by the computational parallelism of the brain, and its ability to successfully perform complex recognition and classification tasks [64]. A simple neural network (NN) consists of many artificially connected nodes called neurons, where each neuron is defined by a real-value activation function. Neurons achieve activation through the passing of information along weighted connections, stemming from previously activated neurons [65]. By introducing additional layers between the input and output, known as hidden layers, the resultant network is considered a deep neural network (DNN) (Figure 2.9).

Although NNs are often used to perform complex recognition and classification tasks [66], they can also be used to solve problems involving approximating

a continuous non-linear function. This allows data structures such as lookup tables to be represented by a NN, with interpolation between known data points provided with no additional computational effort [67]. This was performed by Gautam and Verma [68], eliminating the need to perform online trajectory optimisation for path replanning of a fixed-wing UAV. By performing trajectory optimisation over various flight conditions, the authors were able to train a NN on the optimised dataset, allowing path planning of a low-velocity fixed-wing UAV to be performed faster.

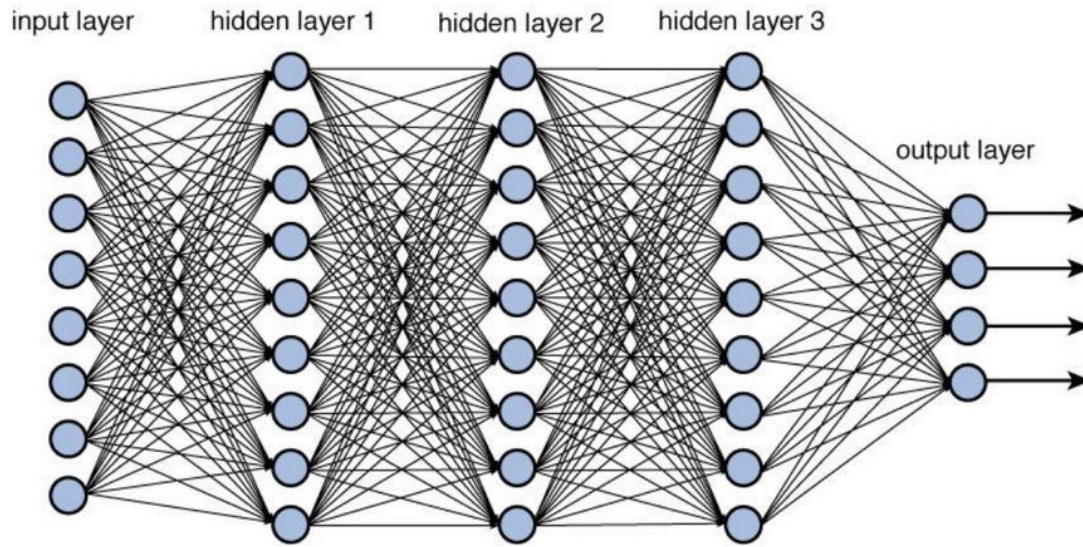


FIGURE 2.9 Deep neural network architecture comprised of three hidden layers [69].

When considering NNs for path finding and obstacle avoidance algorithms, a NN-based path planner was demonstrated by Li et al. [70] for a multi-robot system located in an environment with moving obstacles. The aim of this research was to allow multiple ground-based mobile robots to communicate and achieve a common goal. Similarly, Yang and Meng [71] used a NN for determining collision-free paths for robot manipulators in real-time. The research enabled path planning to be performed in an environment with non-stationary objects, based on three levels of clearance: (i) no clearance or the shortest path; (ii) a moderate clearance or comfortable path; and (iii) a strong clearance or the safest path.

Rather than using a NN for path finding, a NN was used by Julian and Kochenderfer [72] as a guidance autopilot for a small-scale UAV, limited by the availability of onboard memory. The NN was trained based on a compressed dis-

crete state-action table framed as a Markov decision process. Similar to Gautam and Verma [68], training of the NN allowed a lookup table comprised of discrete data to be represented as a continuous function, with the system states representing five dynamic variables (Figure 2.10). Although the NN allowed state-actions to be modelled as continuous functions, due to memory limitations the output from the NN represented only two discrete actions, which were to either increase or decrease the UAV's bank angle. The research conducted by Julian and Kochenderfer [72] is highly promising as demonstrations of the NN guidance autopilot were conducted on physical hardware. The test flights indicated how a NN can be applied to memory-restricted flight control systems, where large multi-dimensional lookup tables are deemed unfeasible.

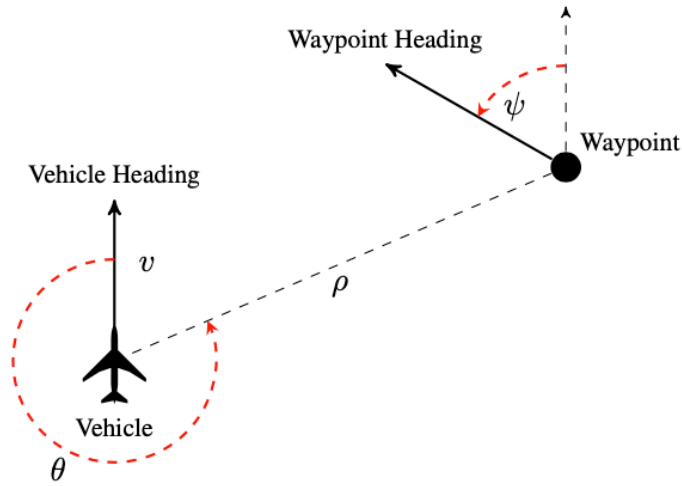


FIGURE 2.10 Geometry for waypoint navigation [72]. ρ represents the Euclidean distance from the vehicle to the waypoint, θ indicates the bearing angle to the waypoint, ψ indicates the angle to waypoint heading direction, ϕ represents the vehicle banking angle and v represents the aircraft's airspeed.

2.4.2 Reinforcement Learning

Reinforcement learning (RL) is a subset of machine learning by which an agent is trained to make a sequence of decisions based on achieving an ultimate goal [73]. During reinforcement learning the agent is initialised within an environment, capable of performing numerous actions. For each step within a training episode, the agent performs an action based on its interpretation of the enviro-

onment, known as an observation. For each action selected by the agent, the environment provides a reward, usually represented by a scalar value [74].

When attempting to solve a problem using reinforcement learning, a dilemma is present between exploration and exploitation [75]. For example, as an agent selects an action, the corresponding action determines whether or not the agent should gain new information by exploring the environment, or whether the agent should exploit past experience with the aim of maximising the reward. This means that reinforcement learning can be considered as a form of trial-and-error learning of an agent based on interactions with a dynamic environment [76]. Modern day reinforcement learning algorithms are considered as part of two classes: model-free; and model-based reinforcement learning [77] (Figure 2.11).

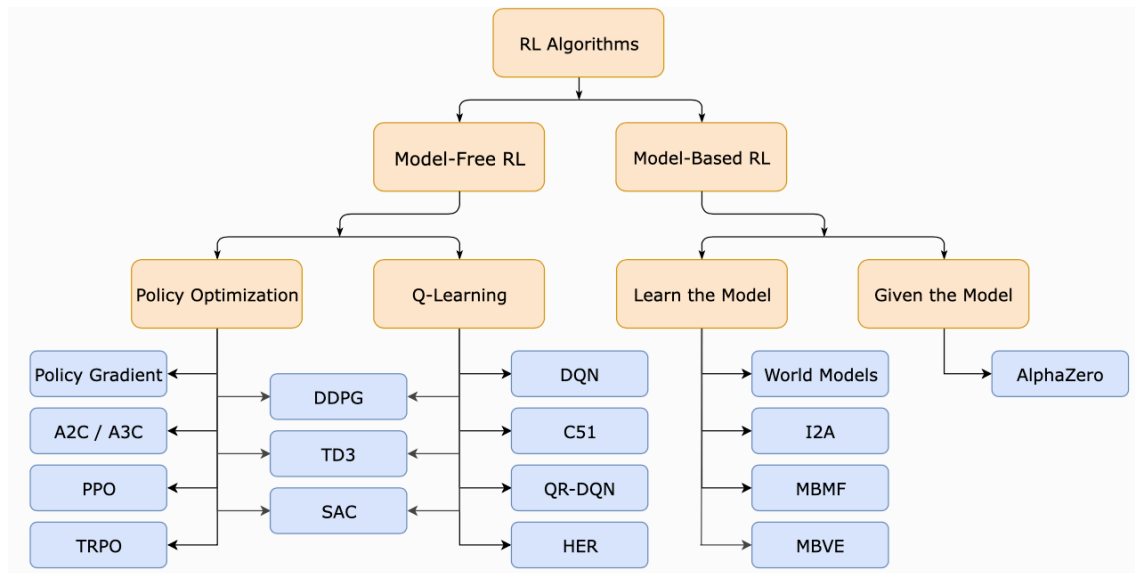


FIGURE 2.11 A non-exhaustive, but useful taxonomy of modern reinforcement learning algorithms [78].

Model-based reinforcement learning algorithms rely on developing a model of the environment using historic experience obtained through training. The agent then uses prior experience and understanding of the environment to select the optimal policy. On the other hand, model-free reinforcement learning algorithms do not develop an overall model of the environment, with the agent selecting a specific policy based on estimating the current state [74]. This means that model-free methods are generally easier to implement in terms of

online decision-making since the agent is solely focused on maximising its reward based on its current state or action. However, model-free methods are significantly less efficient since training involves a lot of trial-and-error experience to ensure the agent performs the correct action. Additionally, model-free methods are unable to account for changes occurring within the environment, such as a change to the overall goal or introducing additional obstacles, without undergoing additional trial-and-error learning based on the new outcome [77].

Model-based methods are described as trying to navigate from point A to point B whilst using a map, where the map represents historic knowledge corresponding to the overall route. Whereas model-free methods navigate from point A to point B without taking prior experience of the overall route into account, with the agent simply reacting locally at specific intersections in an attempt to maximise the reward signal [77]. Since model-free reinforcement learning is considered to be easier to implement for online decision making [77] and contains noticeably more algorithmic methodologies, as shown in Figure 2.11, understanding the internal properties of popular approaches and how they could potentially be employed as potential solutions, was considered a priority.

Q-Learning

Q-Learning, is a model-free off-policy reinforcement learning algorithm that provides the agent with the capabilities of learning to act optimally by collecting experience based on a sequence of actions in Markovian domains [79]. The term off-policy makes reference to the fact that during Q-learning the agent seeks to find the best action given its current state, allowing the agent to create its own policy by maximising the total reward [80].

Traditionally, Q-learning is performed by initialising an array populated with Q-values, known as a Q-table. The Q-table acts as a database of observations and actions for the agent. As the agent observes the environmental states, a Q-value exists for all possible actions, for which the agent selects the largest Q-value. Upon selecting an action, the agent is met with a reward indicating how good the outcome of performing the action was. The Q-value updates following every action based on the previous Q-value, learning rate, reward signal, discount factor, and additional coefficients that estimate the optimal future Q-value. This

means that over the course of many training episodes, the Q-values correspond to the likelihood that performing a specific action produces the best reward signal given a specific environmental observation [80].

Q-learning is considered to be a primitive form of learning [81] since the agent applies a trial-and-error action configuration for specific state observations with the sole aim of maximising the overall reward [81]. Over long training episodes, it can be expected that the agent will have attempted all actions for all observational states, allowing a long-term reward judgment to be made regarding which action was perceived to be the best for any specific observation [79].

Since the Q-table acts as a large multi-dimensional look-up table, the corresponding Q-values represent discrete actions and observations, preventing the agent from performing continuous actions over a continuous observational environment. This often limits Q-learning to applications described by grid-world or node-based environments, making Q-learning an alternative approach to many of the path finding algorithms discussed in Section 2.1.

Although Q-learning has been shown to provide optimal solutions for many Markov Decision Processes (MDPs), the overall performance of Q-learning can be poor in stochastic or complex MDPs, with slow convergence due to large estimations of the action values [82]. To improve learning performance, an extended Q-learning (EQL) algorithm was implemented by Goswami et al. [83] for path planning of a mobile robot operating in a grid-world environment. The extended portion of the Q-learning algorithm focused on improving the efficiency during training and avoiding unnecessary computations by only storing the Q-value corresponding to the best action. Li et al. [35] performed an investigation to establish how the performance of the EQL implemented by Goswami et al. [83] compared to CQL and an A-star path finding algorithm. The authors also demonstrated their own interpretation of an improved Q-learning (IQL) algorithm, which employed a new exploratory strategy, allowing the agent to assign a trade-off between exploration and exploitable. It was concluded by Li et al. [35] that not only did their IQL pose advantages when considering time-complexity, when compared to CQL, EQL and A-star, but was also able to function in dynamically changing grid-world environments, whilst EQL and A-star

were limited to static environments. An example of the paths produced by Li et al. [35] can be seen in Figure 2.12.

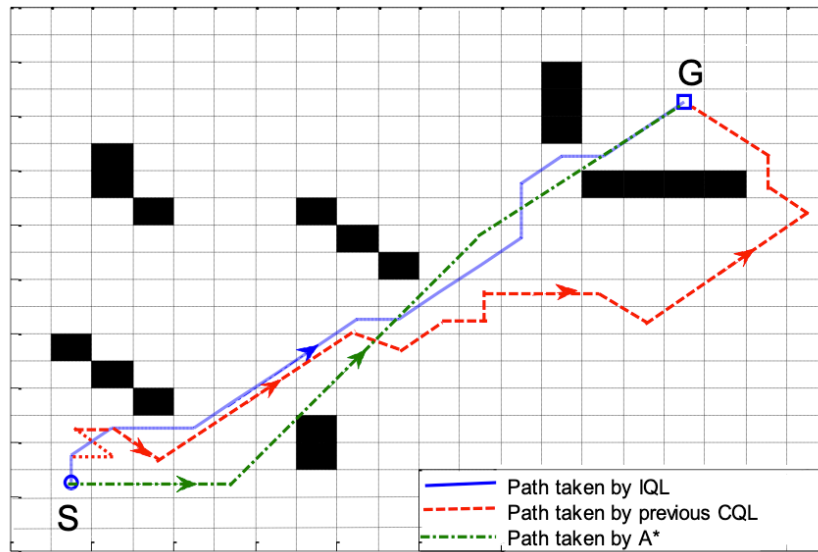


FIGURE 2.12 Path planning comparison between IQL, CQL and A* in a static grid-world environment [35].

One of the main issues associated with Q-learning stems from how the Q-table is initially populated with Q-values. Generally, the Q-table is initiated randomly assigned Q-values, meaning that during early training episodes the agent experiences random movements. This random assignment of the Q-values increases the required number of training episodes to achieve convergence towards the optimal solution. To overcome slow convergence, a meta-heuristic optimisation algorithm was used by Low et al. [84] to initialise the Q-values more appropriately. By using the flower pollination optimiser, the authors were able to accelerate convergence by up to 13% when producing collision-free paths for a mobile robot located in multiple unknown and cluttered environments. Additionally, a small NN was used by Yijing et al. [85] to improve the rate of training. The use of a NN not only improved the rate at which the agent trained, but also allowed the agent to learn with some continuance from past experiences.

Deep Q-Learning (DQN)

Deep Q-Learning (DQN) is model-free off-policy learning algorithm, combining reinforcement learning and deep learning. Whilst DQN learning follows the

same algorithmic procedure as Q-learning, the implementation of deep learning replaces the Q-table with a deep neural network, used to approximate the Q-value [86].

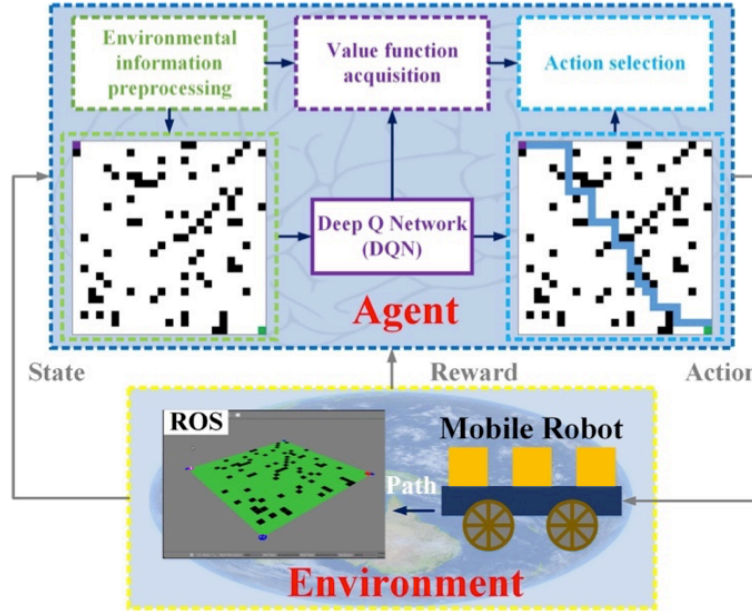


FIGURE 2.13 Structure of DQN mobile robotic path planning technique in a densely populated environment [87]

Zhou et al. [87] used DQN for a mobile robotic path planner, with the DQN agent evaluating state information before selecting an action, as depicted in Figure 2.13. Despite this structure being designed for a specific path finding application, it demonstrates how a NN is embedded into the learning process for an autonomous path finding algorithm. Although combining reinforcement learning with deep learning is not a new concept, the potential of DQN was first demonstrated by Google's DeepMind project, where a DQN network was trained to play a popular 1980s Atari video game [88]. By incorporating a Q-learning algorithm with a convolutional neural network (CNN), the authors were able to extract features from individual frames, allowing Atari to be played using pure image input through learning [87].

One advantage of implementing a NN, is the addition of an experience replay. Experience replay allows an agent to store past experiences, which is then pooled over many episodes into replay memory [88]. This enables training to utilise samples of randomly drawn experience, enhancing the agent's ability to

select and execute better actions in accordance to the applied ϵ -greedy policy. Use of an experience and replay buffer also improves data efficiency, since each step during a training episode can be used for many of the internal network weight updates. Additionally, techniques where learning is performed based on consecutive samples of data can result in inefficient solutions, due to a strong correlation between known samples. Use of a replay and experience buffer can aid in alleviating these issues since experience is loaded into the replay buffer randomly, preventing strong data correlations from overfitting the data [88].

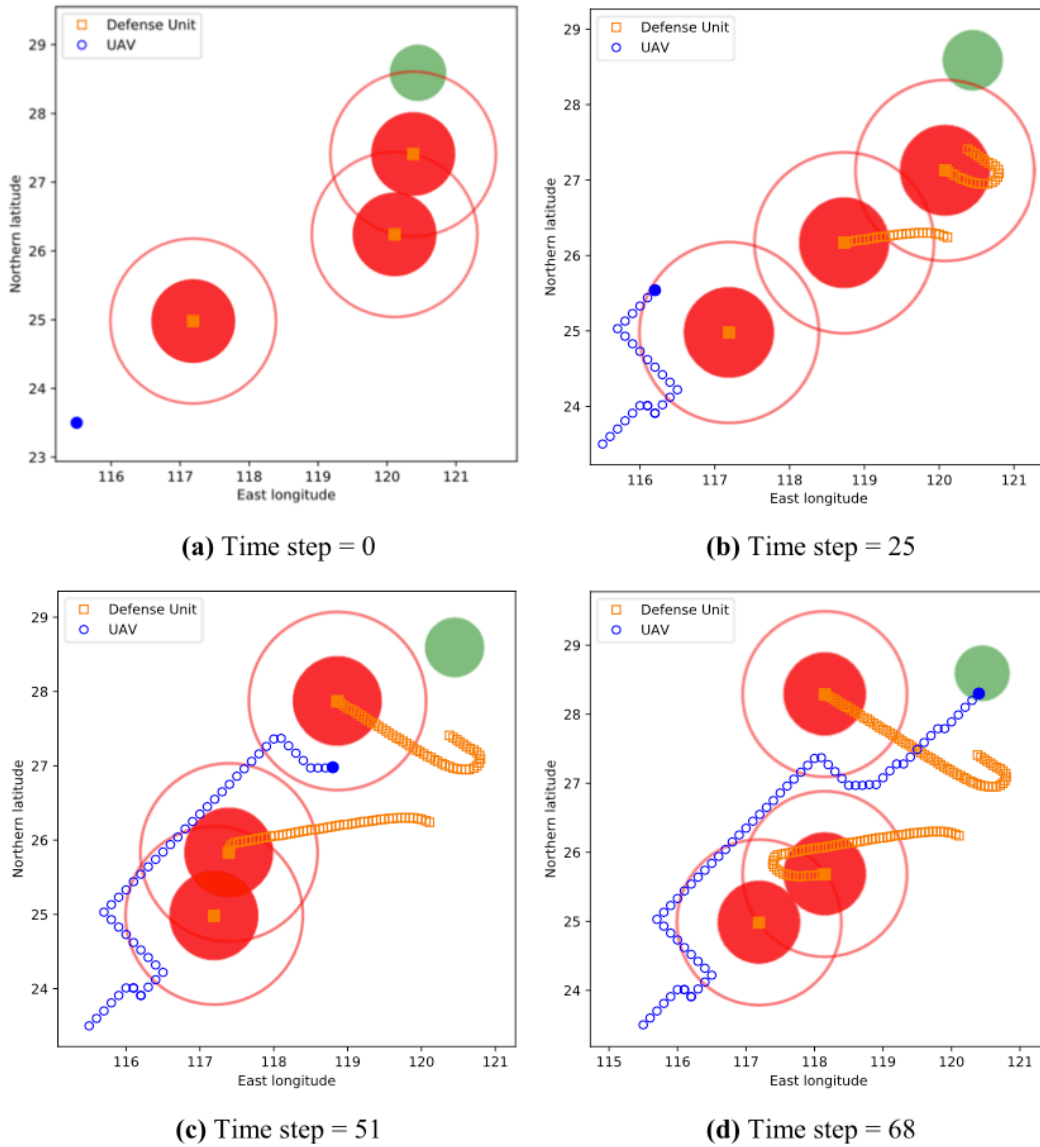


FIGURE 2.14 DQN UAV successfully arriving at target point with acceptable safe trajectory during dynamically changing environmental conditions [89].

Despite the main change to DQN being the implementation of a NN, the use-cases remain largely similar to Q-learning, with an emphasis on path finding algorithms or applications with discrete action spaces. Zhou et al. [87] trained a DQN agent for path finding of a micro multi-rotor UAV operating in cluttered environments. This research is particularly interesting because the authors directly compared DQN path finding to traditional algorithms, such as A-star and Dijkstra. Not only was it demonstrated that the DQN network displayed a higher level of exploration, but was able to converge onto a solution in less than 0.15 seconds after post-training, compared to the 14.6 seconds for A-Star and 16 seconds for Dijkstra's algorithm. Yan et al. [89] used a DQN network for path finding of a UAV with an emphasis on survivability as opposed to shortest-distance trajectories. This research is interesting because the DQN agent displayed an ability to path find despite being exposed to dynamically moving energy threats (Figure 2.14).

2.5 Summary

Based on the literature reviewed, it is evident that path finding algorithms are a well defined and well established area of research for both aerial and ground-based autonomous systems. Algorithms such as A-Star [22], Voronoi [25, 26, 27] and Dijkstra's [29, 30, 32] have been demonstrated to provide collision-free trajectories in simple and complex environments. Additionally, NNs have been used for determining collision-free paths for mobile robotic applications [71, 90], as well as coordinated movement of multiple robots [34, 70]. Although deep reinforcement learning has become a hotspot for robotic path planning in recent years with proven benefits over traditional path finding techniques [87, 89], these applications are still limited to discrete observational/action spaces, making them undesirable for high-fidelity fixed-wing flight application. Furthermore, regardless which method is used for determining a collision-free path, these methods do not guarantee that paths are continuously flyable, drivable or manoeuvrable [21], since they do not factor the kinematic and dynamic constraints of the vehicle. Therefore, path generation and path smoothing algorithms are required, such as Dubins paths or Bézier curves.

When considering path generation, it is evident that the area is not as well developed for the application of online path re-planning for large-scale fixed-wing aircraft. The literature indicates benefits of using Dubins paths when compared to parametric curves, since this geometric approach does not necessarily require trajectory optimisation, allowing trajectories to be replanned more efficiently. However, when considering a military-style high-performance aircraft, Dubins paths do not allow the manoeuvrability and performance characteristics to be utilised, since manoeuvres are generally performed on a single plane. Additionally, the formation of Dubins paths produces discontinuous transitions between joining sections, leading many researchers to combine Dubins paths with clothoids [21, 45, 61, 62], allowing a smoother transition whilst maintaining a continuous curvature. However, the computational costs associated with clothoids is higher than obtaining a solution to a parametric curve, since they require a solution to the Fresnel integrals [58]. When considering high performance and manoeuvrable flight, parametric curves such as Bézier curves provide a more desirable solution. They have been demonstrated to be computationally efficient to produce, whilst also allowing a high level of curvature manipulation through placement of the internal control points.

Although researchers have focused on developing paths using Bézier curves for cooperative flight missions, the demonstrated solutions often consist of benign manoeuvres conducted in two-dimensions or at low velocities, with no robust solution demonstrated for generic flight mission profiles. In addition, optimisation of the paths is largely conducted offline, meaning that online path re-planning is only partially considered or left unsolved, providing a dissatisfying solution for the mandated autonomy frameworks [12, 13]. Whilst solutions have been presented in the literature indicating that paths are dynamically flyable and manoeuvrable for a fixed-wing UAV, this dynamic analysis often only considers a small quantity of state dynamics, analysed in two-dimensions. Whilst paths have been demonstrated to factor a dynamic and kinematic analysis, this analysis often only accounts for minimal state dynamics, such as a velocity profile, as opposed to a linear and rotational acceleration analysis.

Therefore, the current literature suggests a robust solution for path planning when applied to a large-scale, fixed-wing high-performance aircraft with online

path re-planning capabilities has not been demonstrated. This gap in knowledge means that both autonomy frameworks [12, 13] fail to be satisfied when considering highly autonomous systems. In addition, the lack of a robust solution means that any of the applied solutions would result in a low trust factor, especially due to the lack of path re-planning during dynamic mission scenarios with manned counterparts. However, developments in deep learning enable high-dimensional datasets to be represented as continuous non-linear functions. This use of NNs makes deep learning an admirable solution for enabling online path replanning, by training a NN using optimised path solutions.

2.6 Research Objectives

During Chapter 1, the following research aim was established:

Develop an online path planning algorithm enabling continuously flyable paths to be developed for a fixed-wing autonomous fighter jet during cooperative flight.

Having conducted an in-depth literature review into path finding and path smoothing algorithms, and how artificial intelligence can be incorporated into the path planning framework, the following objectives have been identified:

1. Develop a high-fidelity computational model which is representative of an industrial scale fast jet. This model acts as a testing framework for the presented approach and must incorporate the following.
 - A manual and autonomous flight control system, ensuring adequate dynamic performance and closed-loop stability throughout the operational envelope.
 - A guidance control system capable of supplying input demands to the autonomous flight control system based on interpreted path data.
 - The entire model must be programmed in a computationally efficient manner allowing all required calculations to be computed in real-time with high frequency.

2. Develop a path planning algorithm with control point optimisation capable of producing continuously-flyable and dynamically achievable paths between two points in space. This path planning algorithm must incorporate the following constraints.
 - Dynamic and kinematic analysis to ensure paths are continuously-flyable, achievable and manoeuvrable throughout the operational flight envelope.
 - Incorporate an energy analysis to ensure that paths resulting in an increase of altitude can be achieved for any state dynamics identified throughout the operational flight envelope.
3. Integrate deep learning allowing the path planning optimisation algorithm to train a deep neural network, allowing paths to be re-planned online and in real-time during cooperative flight simulations.

Chapter 3

Computational Flight Physics Model

This chapter introduces the mathematical modelling of the flight physics model. A derivation is performed to determine the transformation matrices used to translate the aircraft model from one axis system to another, before moving onto the derivation of the equations of motion. In Appendix A, a more detailed derivation of the equation of motions is demonstrated, with a velocity and acceleration analysis performed. The chapter concludes by defining the aerodynamic, engine and control surface models as described by Nguyen et al. [16], before defining the operational flight envelope.

3.1 Euler Angles

Euler angles are a set of angles used to describe an object's rotation in one frame relative to another. To describe the orientation of a body-axis system denoted as Ox_b, y_b, z_b with respect to an inertial frame Ox_i, y_i, z_i , one must perform three consecutive rotations of ψ , θ and ϕ to rotate the axes Ox_b, y_b, z_b to align with axes Ox_i, y_i, z_i [91].

Consider an aircraft with a body-axis defined by a right-handed orthogonal axis system as shown in Figure 3.1. Under this configuration a positive pitch (θ) occurs as the aircraft rotates clockwise about the y -axis, causing the nose of the aircraft to elevate. A positive roll (ϕ) occurs as the aircraft rotates clockwise about the x -axis, moving the starboard wing down. Finally, a positive yaw (ψ) occurs as the aircraft rotates clockwise about the z -axis, causing the nose of the aircraft to rotate towards the east. To obtain each of the three Euler angles, rotation occurs about each individual axis as follow [91]:

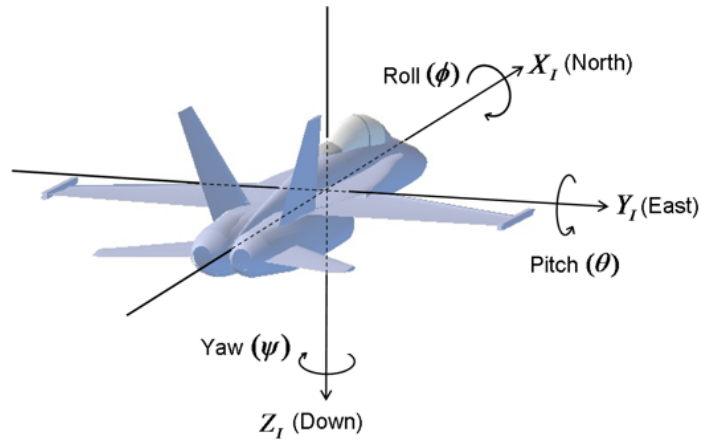


FIGURE 3.1 body-axis right-hand orthogonal axis system rotations [91]

1. A rotation of ψ about the Oz_1 axis translates Ox_1 to Ox'_1 and Oy_1 to Oy'_1 . This rotation occurs such that Ox'_1 is located in the plane Ox_2, z_2 and Oy'_1 is located in the plane Oy_2, z_2 (Figure 3.2a)
2. A rotation of θ about the Oy'_1 axis translates Ox'_1 to Ox''_1 and Oz'_1 to Oz''_1 . This rotation occurs such that Ox''_1 coincides with the Ox_2 axis and Oz''_1 is located in the plane Oy_2, z_2 (Figure 3.2b)
3. A rotation of ϕ about the Ox''_1 axis translates Oy''_1 to Oy'''_1 which is coincident with Oy_2 and translates Oz''_1 to Oz'''_1 which is coincident with Oz_2 (Figure 3.2c).

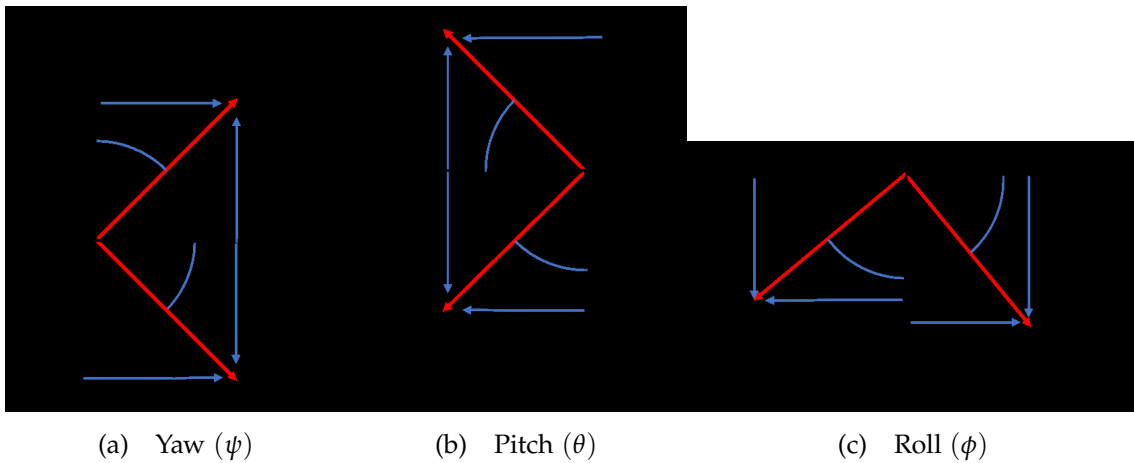


FIGURE 3.2 Euler angle rotations about each axis

Summating the angular components for yaw in Figure 3.2a

$$\begin{aligned}x'_1 &= x_1 \cos(\psi) - y_1 \sin(\psi) \\y'_1 &= x_1 \sin(\psi) + y_1 \cos(\psi) \\z'_1 &= z_1\end{aligned}\tag{3.1}$$

Summating the angular components for pitch in Figure 3.2b

$$\begin{aligned}x''_1 &= x'_1 \cos(\theta) + z'_1 \sin(\theta) \\y''_1 &= y'_1 \\z''_1 &= -x'_1 \sin(\theta) + z'_1 \cos(\theta)\end{aligned}\tag{3.2}$$

Summating the angular components for roll in Figure 3.2c

$$\begin{aligned}x'''_1 &= x''_1 \\y'''_1 &= y''_1 \cos(\phi) - z''_1 \sin(\phi) \\z'''_1 &= y''_1 \sin(\phi) + z''_1 \cos(\phi)\end{aligned}\tag{3.3}$$

Writing Equations 3.1, 3.2 and 3.3 in matrix form

$$\begin{bmatrix} x'_1 \\ y'_1 \\ z'_1 \end{bmatrix} = \begin{bmatrix} \cos \psi & -\sin \psi & 0 \\ \sin \psi & \cos \psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \\ z_1 \end{bmatrix} = C \rightarrow \text{Yaw}\tag{3.4}$$

$$\begin{bmatrix} x''_1 \\ y''_1 \\ z''_1 \end{bmatrix} = \begin{bmatrix} \cos \theta & 0 & \sin \theta \\ 0 & 1 & 0 \\ -\sin \theta & 0 & \cos \theta \end{bmatrix} \begin{bmatrix} x'_1 \\ y'_1 \\ z'_1 \end{bmatrix} = B \rightarrow \text{Pitch}\tag{3.5}$$

$$\begin{bmatrix} x'''_1 \\ y'''_1 \\ z'''_1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \phi & -\sin \phi \\ 0 & \sin \phi & \cos \phi \end{bmatrix} \begin{bmatrix} x''_1 \\ y''_1 \\ z''_1 \end{bmatrix} = A \rightarrow \text{Roll}\tag{3.6}$$

Applying matrix multiplication in the form of $C \times B \times A = D$ translate the vector from the inertial frame into the body-axis frame.

$$\begin{bmatrix} x'''_1 \\ y'''_1 \\ z'''_1 \end{bmatrix} = \begin{bmatrix} \cos \psi & -\sin \psi & 0 \\ \sin \psi & \cos \psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos \theta & 0 & \sin \theta \\ 0 & 1 & 0 \\ -\sin \theta & 0 & \cos \theta \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \phi & -\sin \phi \\ 0 & \sin \phi & \cos \phi \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \\ z_1 \end{bmatrix}\tag{3.7}$$

Therefore the direction cosine matrix D , is as follows

$$D = \begin{bmatrix} \cos \psi \cos \theta & -\sin \psi \cos \phi & \sin \phi \sin \psi \\ & +\cos \psi \sin \phi \sin \theta & +\cos \phi \cos \psi \sin \theta \\ \sin \psi \cos \theta & \cos \psi \cos \phi & -\sin \phi \cos \psi \\ & +\sin \psi \sin \theta \sin \phi & +\cos \phi \sin \theta \sin \psi \\ -\sin \theta & \cos \theta \sin \phi & \cos \theta \cos \phi \end{bmatrix} \quad (3.8)$$

The direction cosine matrix (Equation 3.8) is used to translate the aircraft from an inertial frame coordinate system (Ox_i, y_i, z_i) to a body-axis coordinate system (Ox_b, y_b, z_b) . To translate the aircraft from the body-axis frame (Ox_b, y_b, z_b) into the inertial frame coordinate system (Ox_i, y_i, z_i) the inverse of the direction cosine matrix, D^{-1} is required.

$$\begin{bmatrix} x_{n+1} \\ y_{n+1} \\ z_{n+1} \end{bmatrix} = D \begin{bmatrix} x_n \\ y_n \\ z_n \end{bmatrix} \rightarrow \begin{bmatrix} x_n \\ y_n \\ z_n \end{bmatrix} = D^{-1} \begin{bmatrix} x_{n+1} \\ y_{n+1} \\ z_{n+1} \end{bmatrix} \quad (3.9)$$

Where

$$D^{-1} = \begin{bmatrix} \cos \theta \cos \psi & \sin \psi \cos \theta & -\sin \theta \\ -\sin \psi \cos \phi & \cos \phi \cos \psi & \cos \theta \sin \phi \\ +\cos \psi \sin \theta \sin \phi & +\sin \phi \sin \theta \sin \psi & \\ \sin \phi \sin \psi & -\sin \phi \cos \psi & \cos \theta \cos \phi \\ +\sin \phi \cos \phi \cos \psi & +\cos \phi \sin \theta \sin \psi & \end{bmatrix} \quad (3.10)$$

3.2 Angular Velocity Transformation

During a kinematic analysis, knowledge of the rate of change of Euler angles is a desirable quantity for many flight dynamic problems. However, since the Euler angles are an angular representation of an object's orientation in one axis system when observed by another, measuring this change is not possible. However, changes to the Euler angles, when observed in the body-axis, occur due to the

angular velocity of the aircraft, meaning that changes to the Euler angles during translation from the body-axis to the inertial frame, are equal to the body-axis angular velocities of the aircraft.

Consider an aircraft with a body-axis coordinate system Ox_b, y_b, z_b being observed by an inertial frame Ox_i, y_i, z_i . It has been shown that transforming Ox_b, y_b, z_b to Ox_i, y_i, z_i is performed using the directional cosine matrix (Equation 3.8), or by performing three individual rotations using Equations 3.4-3.6.

During the first Euler rotation (using Equation 3.4), the aircraft is transformed from a position vector Ox_b, y_b, z_b to Ox'_b, y'_b, z'_b by an angular rotation, ψ , about the Oz_b axis. To transform the vector from Ox'_b, y'_b, z'_b to Ox_i, y_i, z_i , the corresponding transformation matrix is the product of Equation 3.5 and Equation 3.6.

$$\dot{\psi}_b = A(-1)B(-1) \begin{bmatrix} 0 \\ 0 \\ \dot{\psi}_i \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & \sin\phi \\ 0 & -\sin\phi & \cos\phi \end{bmatrix} \begin{bmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ \dot{\psi}_i \end{bmatrix} \quad (3.11)$$

During the second Euler rotation (using Equation 3.5), the aircraft is transformed from a position vector Ox'_b, y'_b, z'_b to Ox''_b, y''_b, z''_b by an angular rotation, θ about the Oy'_b axis. To transform the vector from Ox''_b, y''_b, z''_b to Ox_i, y_i, z_i , rotation occurs using Equation 3.6 about the Oy'_i axis.

$$\dot{\theta}_b = A(-1) \begin{bmatrix} 0 \\ \dot{\theta}_i \\ 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & \sin\phi \\ 0 & -\sin\phi & \cos\phi \end{bmatrix} \begin{bmatrix} 0 \\ \dot{\theta}_i \\ 0 \end{bmatrix} \quad (3.12)$$

Finally, the angular velocity vector $\dot{\phi}$ is aligned along the Ox_b axis such that,

$$\dot{\phi}_b = \begin{bmatrix} \dot{\phi}_i \\ 0 \\ 0 \end{bmatrix} \quad (3.13)$$

The summation of Equations 3.11, 3.12 and 3.13 form the inertial axis angular velocities as vectors $\dot{\psi}$, $\dot{\theta}$ and $\dot{\phi}$, which during small angle perturbations, can be

assumed equal to the body-axis angular velocities, described as p , q and r .

$$\begin{bmatrix} p \\ q \\ r \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \phi & \sin \phi \\ 0 & -\sin \phi & \cos \phi \end{bmatrix} \begin{bmatrix} \cos \theta & 0 & -\sin \theta \\ 0 & 1 & 0 \\ \sin \theta & 0 & \cos \theta \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ \dot{\psi}_i \end{bmatrix} + \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \phi & \sin \phi \\ 0 & -\sin \phi & \cos \phi \end{bmatrix} \begin{bmatrix} 0 \\ \dot{\theta}_i \\ 0 \end{bmatrix} + \begin{bmatrix} \dot{\phi}_i \\ 0 \\ 0 \end{bmatrix} \quad (3.14)$$

Simplifying Equation 3.14,

$$\begin{bmatrix} p \\ q \\ r \end{bmatrix} = \begin{bmatrix} 1 & 0 & -\sin \theta \\ 0 & \cos \phi & \sin \phi \cos \theta \\ 0 & -\sin \phi & \cos \phi \cos \theta \end{bmatrix} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \quad (3.15)$$

From Equation 3.15 when the Euler angles for roll (ϕ) and pitch (θ) approach zero, the Euler angle rates of change and the angular velocity components are equal. To determine the rates of change of the Euler angles when ϕ and θ are not equal to zero, the inverse of the matrix must be derived from the determinant and the matrix adjugate. Let,

$$\begin{bmatrix} 1 & 0 & -\sin \theta \\ 0 & \cos \phi & \sin \phi \cos \theta \\ 0 & -\sin \phi & \cos \phi \cos \theta \end{bmatrix} = \lambda \quad (3.16)$$

where

$$\begin{bmatrix} p \\ q \\ r \end{bmatrix} = \lambda \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \therefore \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \lambda^{-1} \begin{bmatrix} p \\ q \\ r \end{bmatrix} = \left(\frac{1}{\Delta \lambda} \right) \text{adj}(\lambda) \begin{bmatrix} p \\ q \\ r \end{bmatrix} \quad (3.17)$$

and the determinant of λ , ($\Delta \lambda$), is

$$\begin{aligned} \Delta \lambda &= (\cos \phi \cos \phi \cos \theta) - (-\sin \phi \sin \phi \cos \theta) \\ &= (\cos^2 \phi \cos \theta) + (\sin^2 \phi \cos \theta) \\ &= \cos \theta \end{aligned} \quad (3.18)$$

The adjugate of λ , ($\text{adj } \lambda$), is determined using the following procedure. The matrix of minors takes each individual cell within the λ matrix as a function of

the determinant of the adjacent rows and columns.

$$\begin{aligned}
 & \left[\begin{array}{cc} \begin{bmatrix} \cos \phi & \sin \phi \cos \theta \\ -\sin \phi & \cos \theta \cos \phi \end{bmatrix} & \begin{bmatrix} 0 & \sin \phi \cos \theta \\ 0 & \cos \theta \cos \phi \end{bmatrix} & \begin{bmatrix} 0 & \cos \phi \\ 0 & \sin \phi \end{bmatrix} \\ \begin{bmatrix} 0 & \sin \theta \\ -\sin \phi & \cos \theta \cos \phi \end{bmatrix} & \begin{bmatrix} 1 & -\sin \theta \\ 0 & \cos \theta \cos \phi \end{bmatrix} & \begin{bmatrix} 1 & 0 \\ 0 & -\sin \phi \end{bmatrix} \\ \begin{bmatrix} 0 & -\sin \theta \\ \cos \phi & \sin \phi \cos \theta \end{bmatrix} & \begin{bmatrix} 1 & -\sin \theta \\ 0 & \sin \phi \cos \theta \end{bmatrix} & \begin{bmatrix} 1 & 0 \\ 0 & \cos \phi \end{bmatrix} \end{array} \right] = \\
 & \begin{bmatrix} \cos^2 \phi \cos \theta + \sin^2 \phi \cos \theta & 0 & 0 \\ -\sin \phi \sin \theta & \cos \theta \cos \phi & -\sin \phi \\ \sin \theta \cos \phi & \sin \phi \cos \theta & \cos \phi \end{bmatrix} = \begin{bmatrix} \cos \theta & 0 & 0 \\ -\sin \phi \sin \theta & \cos \theta \cos \phi & -\sin \phi \\ \sin \theta \cos \phi & \sin \phi \cos \theta & \cos \phi \end{bmatrix} \\
 & \hspace{15em} (3.19)
 \end{aligned}$$

Equation 3.19 is then multiplied by a matrix of cofactors which assigns a mathematical sign to each cell

$$\begin{bmatrix} \cos \theta & 0 & 0 \\ -\sin \phi \sin \theta & \cos \theta \cos \phi & -\sin \phi \\ \sin \theta \cos \phi & \sin \phi \cos \theta & \cos \phi \end{bmatrix} \begin{bmatrix} + & - & + \\ - & + & - \\ + & - & + \end{bmatrix} = \begin{bmatrix} \cos \theta & 0 & 0 \\ \sin \phi \sin \theta & \cos \theta \cos \phi & \sin \phi \\ \sin \theta \cos \phi & -\sin \phi \cos \theta & \cos \phi \end{bmatrix} \quad (3.20)$$

From Equation 3.20 each of the rows are translated about the left hand pivot to become the columns of the adjugate matrix

$$\text{adj}(\lambda) = \begin{bmatrix} \cos \theta & \sin \phi \sin \theta & \sin \theta \cos \phi \\ 0 & \cos \theta \cos \phi & -\sin \phi \cos \theta \\ 0 & \sin \phi & \cos \phi \end{bmatrix} \quad (3.21)$$

Therefore from Equation 3.17

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \lambda^{-1} \begin{bmatrix} p \\ q \\ r \end{bmatrix} = \frac{1}{\cos \theta} \begin{bmatrix} \cos \theta & \sin \phi \sin \theta & \sin \theta \cos \phi \\ 0 & \cos \theta \cos \phi & -\sin \phi \cos \theta \\ 0 & \sin \phi & \cos \phi \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix} \quad (3.22)$$

Finally

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 1 & \tan \theta \sin \phi & \tan \theta \cos \phi \\ 0 & \cos \phi & -\sin \phi \\ 0 & \sec \theta \sin \phi & \sec \theta \cos \phi \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix} \quad (3.23)$$

Having derived the rate of change of Euler angles in-terms of velocity components, the kinematics for the aircraft have been determined, in-terms of angular rotations.

$$\begin{aligned} \dot{\phi} &= p + \tan \theta (q \sin \phi + r \cos \phi) \\ \dot{\theta} &= q \cos \phi - r \sin \phi \\ \dot{\psi} &= \sec \theta (q \sin \phi + r \cos \phi) \end{aligned} \quad (3.24)$$

3.3 Equations of Motion

When deriving the equations of motion, assumptions were made that the aircraft is a rigid body with constant mass and inertia. Under these assumptions, six equations of motion have been derived governing the aircraft's body-axis translational and rotational accelerations, when observed with respect to a non-rotating fixed-axis system. According to Blakelock [92], assuming a non-rotating fixed-axis system does not result in a significant impact to the validity of the equations of motion, since many on-board sensors such as gyroscopes and accelerometers do not detect the effects of the Earth's angular velocity [92]. However, to validate this claim, a full velocity and acceleration analysis was performed (Appendix A.2.1) to identify the magnitude of the Coriolis acceleration. The analysis confirmed that when considering a rotating Earth-fixed axis system, the Coriolis acceleration was small due to low altitude flight and a low airspeed flight envelope. The analysis enabled the translational equation of motion to be described in terms of a single acceleration vector using Newton's second law of motion, as shown in Equation 3.25.

$$F = m(\vec{a}_{b,i}) \quad (3.25)$$

3.3.1 Translational Motion

Consider the axis system depicted in Figure 3.3 showing an aircraft whose body-axis Ox_b, y_b, z_b lies coincident with the vehicles centre of gravity. The body-axes are observed with respect to a fixed Earth non-rotating inertial frame, defined as Ox_i, y_i, z_i .

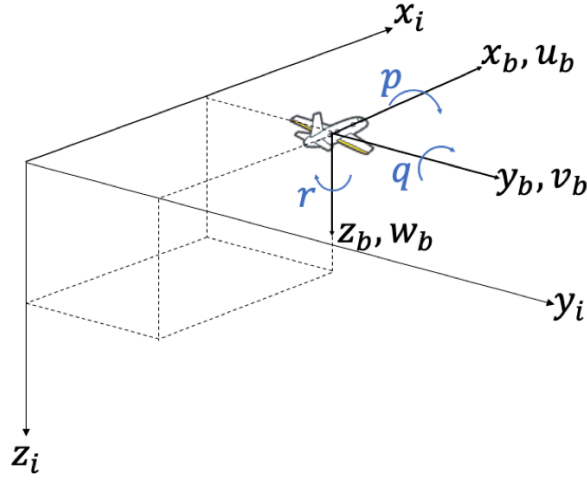


FIGURE 3.3 Translational motion schematic between inertial and body-axes

Since Newton's second law is only valid when the acceleration is being observed from an inertial frame, the translational acceleration in the body-axis $(\vec{a}_0)_i$, can be observed from the inertial frame using the moving axis theorem [91]. Therefore,

$$F = m(\vec{a}_0)_i = m \left(\frac{d\vec{V}_0}{dt} \right)_i = m \left(\frac{d\vec{V}_0}{dt} \right)_b + \vec{\omega}_{i,b}^b \times \vec{V}_0 \quad (3.26)$$

By expressing the three individual components from Equation 3.26 as cartesian unit vectors $\hat{i}$, $\hat{j}$ and $\hat{k}$

$$\vec{\omega}_{i,b}^b = p\hat{i} + q\hat{j} + r\hat{k} \quad (3.27)$$

$$\vec{V}_0 = u\hat{i} + v\hat{j} + w\hat{k} \quad (3.28)$$

$$\left(\frac{d\vec{V}_0}{dt} \right)_b = \dot{u}\hat{i} + \dot{v}\hat{j} + \dot{w}\hat{k} \quad (3.29)$$

The vector cross product can be used on terms $\vec{\omega}_{i,b}^b$ and $\vec{V}_0$

$$\vec{\omega}_{i,b}^b \times \vec{V}_0 = (qw - rv)\hat{i} - (pw - ru)\hat{j} + (pv - qu)\hat{k} \quad (3.30)$$

Combining Equations 3.29 and 3.30 into the form of Equation 3.26,

$$F = (\dot{u} + qw - rv)\hat{i} + (\dot{v} + ru - pw)\hat{j} + (\dot{w} + pv - qu)\hat{k} \quad (3.31)$$

Equation 3.31 is expressed as the total force acting along each of the axes x_b , y_b and z_b .

$$F_{x,i} = (\dot{u} + qw - rv) \quad (3.32)$$

$$F_{y,i} = (\dot{v} + ru - pw) \quad (3.33)$$

$$F_{z,i} = (\dot{w} + pv - qu) \quad (3.34)$$

3.3.2 Rotational Motion

When considering rotational motion, it is important to remember that angular momentum does not have a definitive value, meaning that to determine the angular momentum of an object or particle requires an observational point known as the origin. For radial motion, this observation point is often referred to as the radius of curvature, but in reality is just describing a perpendicular position vector between itself and the objects angular velocity ($\vec{\omega}_{i,b}^i$). When considering a particle of mass δm , the translational momentum can be described as

$$\delta h = (\vec{\omega}_{i,b}^i \times \vec{r}_0) \delta m \quad (3.35)$$

acting at a distance perpendicular to the observation point. Hence,

$$\frac{dH}{dt} = (\vec{r}_b \times \vec{\omega}_{i,b}^i \times \vec{r}_b) \delta m \quad (3.36)$$

Using the triple vector cross product rule, the vectors $\vec{r}_b \times \vec{\omega}_{i,b}^i \times \vec{r}_b$ can be resolved. Where $\vec{\omega}_{i,b}^i$ has unit vector components of (p, q, r) and $\vec{r}_b$ has unit vector components of (x, y, z) . Therefore,

$$\begin{aligned} H &= \vec{\omega}_b(\vec{r}_b \cdot \vec{r}_b) - \vec{r}_b(\vec{\omega}_b \cdot \vec{r}_b) \\ H &= (p\hat{i} + q\hat{j} + r\hat{k})(x^2\hat{i} + y^2\hat{j} + z^2\hat{k}) - (x\hat{i} + y\hat{j} + z\hat{k})(px\hat{i} + qy\hat{j} + rz\hat{k}) \end{aligned} \quad (3.37)$$

Expanding and applying into matrix form,

$$H = \begin{bmatrix} p(x^2 + y^2 + z^2) - (px^2 + qxy + rzx) \\ q(x^2 + y^2 + z^2) - (pxy + qy^2 + rzy) \\ r(x^2 + y^2 + z^2) - (pxz + qxy + rz^2) \end{bmatrix} = \begin{bmatrix} p(y^2 + z^2) - (qxy + rzx) \\ q(x^2 + z^2) - (pxy + rzy) \\ r(x^2 + y^2) - (pxz + qxy) \end{bmatrix} \quad (3.38)$$

Multiplying both sides by the particle mass, δ_m and integrating over the entire aircraft,

$$H = \begin{bmatrix} p \int (y^2 + z^2) dm - q \int xy dm + r \int zx dm \\ q \int (x^2 + z^2) dm - p \int xy dm + r \int zy dm \\ r \int (x^2 + y^2) dm - p \int xz dm + q \int xy dm \end{bmatrix} \quad (3.39)$$

The integrals in Equation 3.39 are defined as the moments and products of inertia.

$$\begin{aligned} I_{xx} &= \int (y^2 + z^2) dm & I_{yy} &= \int (x^2 + z^2) dm & I_{zz} &= \int (x^2 + y^2) dm \\ I_{xy} &= \int xy dm & I_{xz} &= \int xz dm & I_{yz} &= \int yz dm \\ I_{yx} &= \int yx dm & I_{zx} &= \int zx dm & I_{zy} &= \int zy dm \end{aligned}$$

where, $I_{xy} = I_{yx}$, $I_{xz} = I_{zx}$ and $I_{yz} = I_{zy}$. In practice the moments and products of inertia would change as the aircraft uses fuel or releases payloads. However, it has been assumed that the aircraft has a constant mass, implying that the moments and products of inertia do not change during flight. Substituting the moments and products of inertia into Equation 3.39,

$$\begin{bmatrix} H_x \\ H_y \\ H_z \end{bmatrix} = \begin{bmatrix} pJ_{xx} - qJ_{xy} - rJ_{xz} \\ qJ_{yy} - pJ_{xy} - rJ_{zy} \\ rJ_{zz} - pJ_{xz} - qJ_{yz} \end{bmatrix} \quad (3.40)$$

Since the body-axis system is coincident with the centre of gravity and the x -axis is located along the aircraft's line of symmetry, i.e., $J_{xy} = J_{yx} = 0$ and $J_{yz} = J_{zy} = 0$. Therefore, Equation 3.40 becomes

$$\begin{aligned} H_x &= pJ_{xx} - rJ_{xz} \\ H_y &= qJ_{yy} \\ H_z &= rJ_{zz} - pJ_{xz} \end{aligned} \quad (3.41)$$

Having established a set of equations governing the angular momentum vector, the rate of change of angular momentum is determined through vector differentiation. To perform vector differentiation, angular momentum is written as a function of its unit vectors, where $1_H = (\hat{i}, \hat{j}, \hat{k})$ [92]. Therefore,

$$M = \frac{dH}{dt} = \frac{d}{dt}(1_H \cdot H) \quad (3.42)$$

Differentiating using the chain rule ($\frac{d}{dt} = \frac{du}{dt}v + \frac{dv}{dt}u$),

$$\frac{dH}{dt} = \frac{d1_H}{dt}H + \frac{dH}{dt}1_H \quad (3.43)$$

Where $\frac{d\mathbf{1}_H}{dt}H$ is the rate of change of momentum due to the angular momentum.

$$\frac{d\mathbf{1}_H}{dt}H = \vec{w}_b \times \vec{H} \quad (3.44)$$

Combining Equations 3.43 and 3.44

$$\frac{dH}{dt} = \frac{dH}{dt}\mathbf{1}_H + \vec{w}_b \times \vec{H} \quad (3.45)$$

Using the vector cross product rule, resolve $\vec{w}_b \times \vec{H}$, where $\vec{w}_b = (p, q, r)$ and $\vec{H} = (H_x, H_y, H_z)$

$$\vec{w}_b \times \vec{H} = (qH_z - rH_y)\hat{i} - (pH_z - rH_x)\hat{j} + (pH_y - qH_x)\hat{k} \quad (3.46)$$

Using the the dot notation,

$$\frac{dH}{dt}\mathbf{1}_H = \frac{d}{dt}H = \begin{bmatrix} \frac{d}{dt}H_x \\ \frac{d}{dt}H_y \\ \frac{d}{dt}H_z \end{bmatrix} = \begin{bmatrix} \dot{H}_x \\ \dot{H}_y \\ \dot{H}_z \end{bmatrix} \quad (3.47)$$

Therefore from Equation 3.41,

$$\begin{aligned} \dot{H}_x &= \dot{p}J_{xx} - \dot{r}J_{xz} \\ \dot{H}_y &= \dot{q}J_{yy} \\ \dot{H}_z &= \dot{r}J_{zz} - \dot{p}J_{xz} \end{aligned} \quad (3.48)$$

Combining Equations 3.46 and 3.48,

$$\Sigma M = \frac{dH}{dt} = \begin{bmatrix} \dot{H}_x + (qH_z - rH_y) \\ \dot{H}_y - (pH_z - rH_x) \\ \dot{H}_z + (pH_y - qH_x) \end{bmatrix} = \begin{bmatrix} \dot{p}J_{xx} - \dot{r}J_{xz} + q(rJ_{zz} - pJ_{xz}) - r(qJ_{yy}) \\ \dot{q}J_{yy} - (prJ_{zz} - p^2J_{xz} - rpJ_{xx} + r^2J_{xz}) \\ \dot{r}J_{zz} - \dot{p}J_{xz} + (pqJ_{yy} - qpJ_{xx} + qrJ_{xz}) \end{bmatrix} \quad (3.49)$$

The vector for the summation of moments has components of $\bar{L}$, M and N . So expanding Equation 3.49 into the individual components

$$\begin{aligned} \bar{L} &= \dot{p}J_{xx} - \dot{r}J_{xz} + qrJ_{zz} - pqJ_{xz} - qrJ_{yy} \\ &= \dot{p}J_{xx} - \dot{r}J_{xz} + qr(J_{zz} - J_{yy}) - pqJ_{xz} \end{aligned} \quad (3.50)$$

$$\begin{aligned} M &= \dot{q}J_{yy} - prJ_{xx} + p^2J_{xz} + prJ_{xx} - r^2J_{xz} \\ &= \dot{q}J_{yy} - J_{xz}(p^2 - r^2) - pr(J_{zz} + J_{xx}) \end{aligned} \quad (3.51)$$

$$\begin{aligned} N &= \dot{r}J_{zz} - \dot{p}J_{xz} + pqJ_{yy} - pqJ_{xx} + qrJ_{xz} \\ &= \dot{r}J_{zz} - \dot{p}J_{xz} + pq(J_{yy} - J_{xx}) + qrJ_{xz} \end{aligned} \quad (3.52)$$

3.3.3 Forces and Moments

The three translational equations of motion (Equations 3.32 - 3.34) describe the total force acting along the x_b , y_b and z_b axes. According to Cook [15], the force acting on the aircraft is categorised as being aerodynamic, gravitational, power generation and atmospheric disturbances.

Since the equations of motion have been derived in the body-axis as opposed to wind or stability axes, gravitational forces are modelled parallel to the z_b axis ($[0, 0, mg]$) when both $\theta = 0^\circ$ and $\phi = 0^\circ$. Using the inverse of the direction cosine matrix (Equation 3.10) that translate the aircraft from the inertial axis system into the body-axis system,

$$\begin{bmatrix} F_{g,x} \\ F_{g,y} \\ F_{g,z} \end{bmatrix} = D^{-1} \begin{bmatrix} 0 \\ 0 \\ mg \end{bmatrix} = mg \begin{bmatrix} -\sin\theta \\ \sin\phi \cos\theta \\ \cos\theta \cos\phi \end{bmatrix} \quad (3.53)$$

Thrust was modelling by assuming the force occurs parallel along the x -axis and without thrust vectoring.

$$\begin{bmatrix} F_{T,x} \\ F_{T,y} \\ F_{T,z} \end{bmatrix} = \begin{bmatrix} T \\ 0 \\ 0 \end{bmatrix} \quad (3.54)$$

The aerodynamic forces are modelled as total aerodynamic force coefficients, derived from [16].

$$\begin{bmatrix} F_{A,x} \\ F_{A,y} \\ F_{A,z} \end{bmatrix} = \bar{q}S \begin{bmatrix} C_{x,t} \\ C_{y,t} \\ C_{z,t} \end{bmatrix} \quad (3.55)$$

Combining the gravitational, thrust and aerodynamic forces into Equations 3.32, 3.33 and 3.34 and resolving for $\dot{u}$, $\dot{v}$ and $\dot{w}$

$$\dot{u} = rv - qw - g \sin\theta + \frac{1}{m}(\bar{q}SC_{x,t} + T) \quad (3.56)$$

$$\dot{v} = pw - ru + g \sin\phi \cos\theta + \frac{1}{m}(\bar{q}SC_{y,t}) \quad (3.57)$$

$$\dot{w} = qu - pv + g \cos\theta \cos\phi + \frac{1}{m}(\bar{q}SC_{z,t}) \quad (3.58)$$

The three rotational equations of motion (Equation 3.50 - 3.52) describe the total moment acting about the x_b , y_b and z_b axes. Similarly to the translational

force equations, the total moments are defined by gravitational, aerodynamic and propulsive components.

The moments induced by gravity act at a distance, l_g , from the centre of gravity. However, since the body-axis system has been assumed to be coincident with the centre of gravity, the distance l_g is zero, meaning that there is no gravitational moment induced onto the aircraft.

$$\begin{bmatrix} L_g \\ M_g \\ N_g \end{bmatrix} = \begin{bmatrix} F_{g,x} l_g \\ F_{g,y} l_g \\ F_{g,z} l_g \end{bmatrix} = l_g g \begin{bmatrix} -\sin\theta \\ \sin\phi \cos\theta \\ \cos\theta \cos\phi \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \quad (3.59)$$

It was assumed whilst collating the translational forces that thrust produced by the engine occurred solely along the x -axis. This means that no force is produced along either the y or z axes and, therefore, moments induced through thrust generation are also null. Additionally, although a force is produced along the x -axis, it has been assumed that this force acts coincident and perfectly parallel to the x -axis, and therefore the applied moment is also null. Hence,

$$\begin{bmatrix} L_T \\ M_T \\ N_T \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \quad (3.60)$$

However, according to Nguyen et al. [16], gyroscopic effects by the engine due to angular momentum are applied as a constant rate of $216.9 \text{ kgm}^2/\text{s}$. This constant angular momentum is applied to the pitch and yaw axes due to their applied distance from the centre of gravity, and are scaled based on the angular velocity. Therefore,

$$\begin{bmatrix} L_T \\ M_T \\ N_T \end{bmatrix} = \begin{bmatrix} 0 \\ H_e r \\ H_e q \end{bmatrix} \quad (3.61)$$

The moment induced due to aerodynamic forces is modelled using the total aerodynamic coefficients induced by roll, pitch and yaw, as described by Nguyen et al. [16].

$$\begin{bmatrix} L_{a,x} \\ M_{a,y} \\ N_{a,z} \end{bmatrix} = \bar{q} S \begin{bmatrix} bC_{l,t} \\ \bar{c}C_{m,t} \\ bC_{n,t} \end{bmatrix} \quad (3.62)$$

where $\bar{q}$ is the free-stream dynamic pressure, S is the wing reference area, b is the wing-span and $\bar{c}$ is the wing mean geometric chord [18].

Compiling the moments into Equations 3.50, 3.51 and 3.52 and resolving for $\dot{p}$, $\dot{q}$ and $\dot{r}$

$$\dot{p} = \frac{I_y - I_z}{I_x}qr + \frac{I_{xz}}{I_x}(\dot{r} + pq) + \frac{\bar{q}Sb}{I_x}C_{l,t} \quad (3.63)$$

$$\dot{q} = \frac{I_z - I_x}{I_y}pr + \frac{I_{xz}}{I_y}(r^2 - p^2) + \frac{\bar{q}S\bar{c}}{I_y}C_{m,t} - h_e r \quad (3.64)$$

$$\dot{r} = \frac{I_x - I_y}{I_z}pq + \frac{I_{xz}}{I_z}(\dot{p} - qr) + \frac{\bar{q}Sb}{I_z}C_{n,t} + h_e q \quad (3.65)$$

Note that both the roll and yaw axes are dynamically coupled, meaning that both $\dot{p}$ and $\dot{r}$ appear in both Equation 3.63 and Equation 3.65. These internal acceleration components are solved by obtaining a solution to a set of simultaneous equations. Therefore, Equations 3.63 and 3.65 become,

$$\dot{p} = \frac{1}{I_x I_z - I_{xz}^2} \left\{ qr(I_z(I_y - I_z) - I_{xz}^2) + \right. \\ \left. pq(I_{xz}(I_z + I_x - I_y)) + I_z \bar{L} + I_{xz} \bar{N} + I_{xz} I_z h_e q \right\} \quad (3.66)$$

$$\dot{r} = \frac{1}{I_x I_z - I_{xz}^2} \left\{ [I_{xz}(I_y - I_x - I_z)]qr + \right. \\ \left. [I_{xz}^2 + I_x^2 - I_x I_y]pq + I_{xz} \bar{L} + I_x \bar{N} + I_x I_z h_e q \right\} \quad (3.67)$$

Writing the moments of inertia into separate I and λ variables

$$\begin{aligned} I_1 &= I_z(I_y - I_z) - I_{xz}^2 & I_2 &= I_{xz}(I_z + I_x - I_y) & I_3 &= I_z \\ I_4 &= I_{xz} & I_5 &= I_{xz} I_z & I_6 &= (I_z - I_x)/I_y \\ I_7 &= I_{xz}/I_y & I_8 &= 1/I_y & I_9 &= I_{xz}(I_y - I_x - I_z) \\ I_{10} &= I_{xy}^2 + I_x^2 - I_x I_y & I_{11} &= I_{xz} & I_{12} &= I_x \\ I_{13} &= I_x I_z & \lambda &= \frac{1}{I_x I_z - I_{xz}^2} \end{aligned} \quad (3.68)$$

Therefore, Equations 3.66, 3.64 and 3.67 are written as,

$$\begin{aligned} \dot{p} &= qrI_1\lambda + pqI_2\lambda + \bar{L}I_3\lambda + NI_4\lambda + h_e q I_5\lambda \\ \dot{q} &= prI_6 + I_7(r^2 - p^2) + I_8M - h_e r \\ \dot{r} &= qrI_9\lambda + pqI_{10}\lambda + \bar{L}I_{11}\lambda + NI_{12}\lambda + h_e q I_{13}\lambda \end{aligned} \quad (3.69)$$

3.4 Navigation Equations

The navigation equations are a set of equations which translate the orientation of the aircraft into a north, east and altitude coordinate system. For many flight dynamic problems, the Earth can be considered to be a flat. This is an appropriate approximation given certain flight condition parameters. For example, if the airspeed is low and simulation time is low, the curvature of the Earth is not substantial enough to cause navigational issues. Furthermore if the flight altitude is low, compared to the radius of curvature of the Earth, the curvature does not cause a significant change to the navigational position of the aircraft. As a result the linearised F-16 model published by Stevens et al. [18] uses a set of flat Earth navigation equations to translate the aircrafts position from the inertial frame of reference to the navigation frame of reference. These flat Earth navigational equations utilise the Euler transformation matrix (Equation 3.8) to give,

$$\begin{bmatrix} \dot{p}_n \\ \dot{p}_e \\ \dot{h} \end{bmatrix} = \begin{bmatrix} u & v & w \\ u & v & w \\ u & v & w \end{bmatrix} \cdot \begin{bmatrix} \cos \theta \cos \psi & \cos \theta \sin \psi & -\sin \theta \\ \sin \phi \sin \theta \cos \psi & \sin \phi \sin \theta \sin \psi & \sin \phi \cos \theta \\ -\cos \phi \sin \psi & +\cos \phi \cos \psi & \\ \cos \phi \sin \theta \cos \psi & \cos \phi \sin \theta \sin \psi & \cos \phi \cos \theta \\ +\sin \phi \sin \psi & -\sin \phi \cos \psi & \end{bmatrix} \quad (3.70)$$

Therefore

$$\dot{p}_n = u(\cos \theta \cos \psi) + v(\sin \phi \sin \theta \cos \psi - \cos \phi \sin \psi) + \quad (3.71)$$

$$w(\cos \phi \sin \theta \cos \psi + \sin \phi \sin \psi)$$

$$\dot{p}_e = u(\cos \theta \sin \psi) + v(\sin \phi \sin \theta \sin \psi + \cos \phi \cos \psi) + \quad (3.72)$$

$$w(\cos \phi \sin \theta \sin \psi - \sin \phi \cos \psi)$$

$$\dot{h} = u(-\sin \theta) + v(\sin \phi \cos \theta) + w(\cos \phi \cos \theta) \quad (3.73)$$

Since a right-handed orthogonal axis system depicts a positive change in altitude as downwards, Equation 3.73 is multiplied by negative one to invert the direction, resulting in a more conventional positive change to the altitude.

$$\dot{h} = u(\sin \theta) - v(\sin \phi \cos \theta) - w(\cos \phi \cos \theta) \quad (3.74)$$

3.5 Quaternions

Although Euler angles were introduced in Section 3.1, use of Euler angles was limited to the derivation of the equations of motions, forces and moments, navigation equations, and aircraft kinematics. When solving the equations of motions and translating the aircraft model's dynamics from a body-axis coordinate system to an inertial frame axis system, quaternions [93] were used. Quaternions are used in many flight dynamics problems, and are described as four-dimensional vectors used to rotate vectors in three-dimensional space [94], as they enable a *"unique representation of attitude which does not suffer from the singularity and wraparound problems of the Euler equations"* [18].

3.6 Aerodynamic Model

Aerodynamic data for the F-16 was published by Nguyen et al. [16] from subsonic wind tunnel test data on a scaled model of the F-16 and arranged into 50 multi-dimensional lookup tables. Each lookup table is arranged based on either angle of attack, side-slip angle, aerodynamic surface deflection or a combination of all three. The data covers angles of attack ranging from -20° to $+90^\circ$ and sideslip angles ranging from -30° to $+30^\circ$. The range of motion covered by the aileron, rudder and elevator deflections are dependant on applied control surface limits, whilst the leading-edge flap and speed brake values are normalised against their maximum deflection angles and then used to scale internal aerodynamic data components.

Shown in Equations 3.75 - 3.80 are the total aerodynamic force coefficients, denoted as $C_{x,t}$, $C_{y,t}$, $C_{z,t}$, $C_{l,t}$, $C_{m,t}$ and $C_{n,t}$. These coefficients are applied directly into the six equations of motion and are formed based on numerous internal aerodynamic variables, where the value of each variable is obtained by performing a lookup of the corresponding dataset. For example, one of the components forming $C_{x,t}$, denoted as C_x , is a three-dimensional dataset requiring the angle of attack α , sideslip angle β and elevator deflection angle δ_h to determine the value.

x-axis force coefficient:

$$C_{x,t} = C_x(\alpha, \beta, \delta_h) + \Delta C_{x,lef} \left(1 - \frac{\delta_{lef}}{25}\right) + \Delta C_{x,sb}(\alpha) \left(\frac{\delta_{sb}}{60}\right) + \frac{\bar{c}q}{2V} \left[C_{x_q}(\alpha) + \Delta C_{x_q,lef}(\alpha) \left(1 - \frac{\delta_{lef}}{25}\right) \right] \quad (3.75)$$

where

$$\Delta C_{x,lef} = C_{x,lef}(\alpha, \beta) - C_x(\alpha, \beta, \delta_h = 0^\circ)$$

y-axis force coefficient:

$$C_{y,t} = C_y(\alpha, \beta) + \Delta C_{y,lef} \left(1 - \frac{\delta_{lef}}{25}\right) + \left[\Delta C_{y,\delta_a=20} + \Delta C_{y,\delta_a=20,lef} \left(1 - \frac{\delta_{lef}}{25}\right) \right] \left(\frac{\delta_a}{20}\right) + \Delta C_{y,\delta_r=30} \left(\frac{\delta_r}{30}\right) + \frac{b}{2V} \left\{ \left[C_{y_{r,lef}}(\alpha) \left(1 - \frac{\delta_{lef}}{25}\right) \right] r + \left[C_{y_p}(\alpha) \Delta C_{y_{p,lef}}(\alpha) \left(1 - \frac{\delta_{lef}}{25}\right) \right] p \right\} \quad (3.76)$$

where

$$\Delta C_{y,lef} = C_{y,lef}(\alpha, \beta) - C_y(\alpha, \beta)$$

$$\Delta C_{y,\delta_a=20} = C_{y,\delta_a=20} - C_y(\alpha, \beta)$$

$$\Delta C_{y,\delta_a=20,lef} = C_{y,\delta_a=20,lef} - C_{y,lef}(\alpha, \beta) - C_{y,\delta_a=20}(\alpha, \beta) - C_y(\alpha, \beta)$$

$$\Delta C_{y,\delta_r=30} = C_{y,\delta_r=30} - C_y(\alpha, \beta)$$

z-axis force coefficient:

$$C_{z,t} = C_z(\alpha, \beta, \delta_h) + \Delta C_{z,lef} \left(1 - \frac{\delta_{lef}}{25}\right) + \Delta C_{z,sb}(\alpha) \left(\frac{\delta_{sb}}{60}\right) + \frac{\bar{c}q}{2V} \left[C_{z_q}(\alpha) + \Delta C_{z_q,lef}(\alpha) \left(1 - \frac{\delta_{lef}}{25}\right) \right] \quad (3.77)$$

where

$$\Delta C_{z,lef} = C_{z,lef}(\alpha, \beta) - C_z(\alpha, \beta, \delta_h = 0^\circ)$$

x-axis moment coefficient:

$$\begin{aligned}
 C_{l,t} = & C_l(\alpha, \beta, \delta_h) + \Delta C_{l,lef} \left(1 - \frac{\delta_{lef}}{25}\right) \\
 & + \left[\Delta C_{l,\delta_a=20} + \Delta C_{l,\delta_a=20,lef} \left(1 - \frac{\delta_{lef}}{25}\right) \right] \left(\frac{\delta_a}{20}\right) \\
 & + \Delta C_{l,\delta_r=30} \left(\frac{\delta_r}{30}\right) + \frac{b}{2V} \left\{ \left[C_{l,r,lef}(\alpha) \left(1 - \frac{\delta_{lef}}{25}\right) \right] r \right. \\
 & \left. + \left[C_{l_p}(\alpha) \Delta C_{l_p,lef}(\alpha) \left(1 - \frac{\delta_{lef}}{25}\right) \right] p \right\} + \Delta C_{l_b}(\alpha) \beta \quad (3.78)
 \end{aligned}$$

where

$$\begin{aligned}
 \Delta C_{l,lef} &= C_{l,lef}(\alpha, \beta) - C_l(\alpha, \beta) \\
 \Delta C_{l,\delta_a=20^\circ} &= C_{l,\delta_a=20^\circ} - C_l(\alpha, \beta) \\
 \Delta C_{l,\delta_a=20^\circ,lef} &= C_{l,\delta_a=20^\circ,lef} - C_{l,lef}(\alpha, \beta) - C_{l,\delta_a=20^\circ}(\alpha, \beta) - C_l(\alpha, \beta) \\
 \Delta C_{l,\delta_r=30^\circ} &= C_{l,\delta_r=30^\circ} - C_l(\alpha, \beta)
 \end{aligned}$$

y-axis moment coefficient:

$$\begin{aligned}
 C_{m,t} = & C_m(\alpha, \beta, \delta_h) \eta_{\delta_h}(\delta_h) + C_{z,t}(x_{cg,ref} - x_{cg}) + \Delta C_{m,lef} \left(1 - \frac{\delta_{lef}}{25}\right) \\
 & + \Delta C_{m,sb}(\alpha) \left(\frac{\delta_{sb}}{60}\right) + \frac{\bar{c}q}{2V} \left[C_{m_q}(\alpha) + \Delta C_{m_q,lef}(\alpha) \left(1 - \frac{\delta_{lef}}{25}\right) \right] \\
 & + \Delta C_m(\alpha) + \Delta C_{m,ds}(\alpha, \delta_h) \quad (3.79)
 \end{aligned}$$

where

$$\Delta C_{m,lef} = C_{m,lef}(\alpha, \beta) - C_m(\alpha, \beta, \delta_h = 0)$$

z-axis moment coefficient:

$$\begin{aligned}
 C_{n,t} = & C_n(\alpha, \beta, \delta_h) + \Delta C_{n,lef} \left(1 - \frac{\delta_{lef}}{25}\right) \\
 & + \left[\Delta C_{n,\delta_a=20} + \Delta C_{n,\delta_a=20,lef} \left(1 - \frac{\delta_{lef}}{25}\right) \right] \left(\frac{\delta_a}{20}\right) \\
 & + \Delta C_{n,\delta_r=30} \left(\frac{\delta_r}{30}\right) + \frac{b}{2V} \left\{ \left[C_{n,r,lef}(\alpha) \left(1 - \frac{\delta_{lef}}{25}\right) \right] r \right. \\
 & \left. + \left[C_{n_p}(\alpha) \Delta C_{n_p,lef}(\alpha) \left(1 - \frac{\delta_{lef}}{25}\right) \right] p \right\} \quad (3.80)
 \end{aligned}$$

where

$$\Delta C_{n,lef} = C_{n,lef}(\alpha, \beta) - C_n(\alpha, \beta)$$

$$\Delta C_{n,\delta_a=20^\circ} = C_{n,\delta_a=20^\circ} - C_n(\alpha, \beta)$$

$$\Delta C_{n,\delta_a=20^\circ,lef} = C_{n,\delta_a=20^\circ,lef} - C_{n,lef}(\alpha, \beta) - C_{n,\delta_a=20^\circ}(\alpha, \beta) - C_n(\alpha, \beta)$$

$$\Delta C_{n,\delta_r=30^\circ} = C_{n,\delta_r=30^\circ} - C_n(\alpha, \beta)$$

3.7 Engine Model

The aircraft engine has been modelled with thrust data obtained from Nguyen et al. [16], and with first-order dynamics. However, the first-order time constant is variable and dependant on the difference between the actual power and the power demand, as depicted in Figure 3.4.

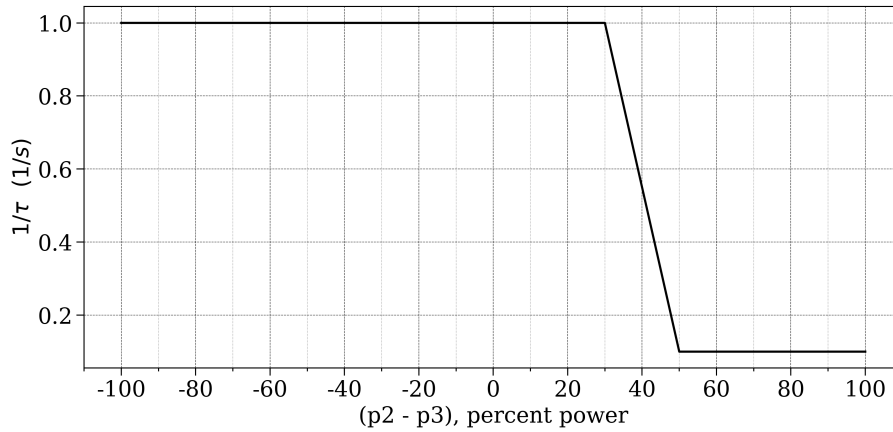


FIGURE 3.4 Variation of inverse of thrust time constant with incremental power command. Adapted from Nguyen et al. [16]

The thrust data is formatted as a three-dimensional lookup table as shown in Table A.1 located in Appendix A.3. The output thrust in Newtons is tabulated against altitude, Mach number and throttle position. The altitude ranged from 0 to 15.24 km, with Mach number from 0.2 to 1.0. The throttle position was described as a percentage of travel, and had three core values: (i) minimum power; (ii) military power; and (iii) maximum power. Maximum power incorporated the afterburner and is only engaged when the throttle percentage of travel exceeded 75%.

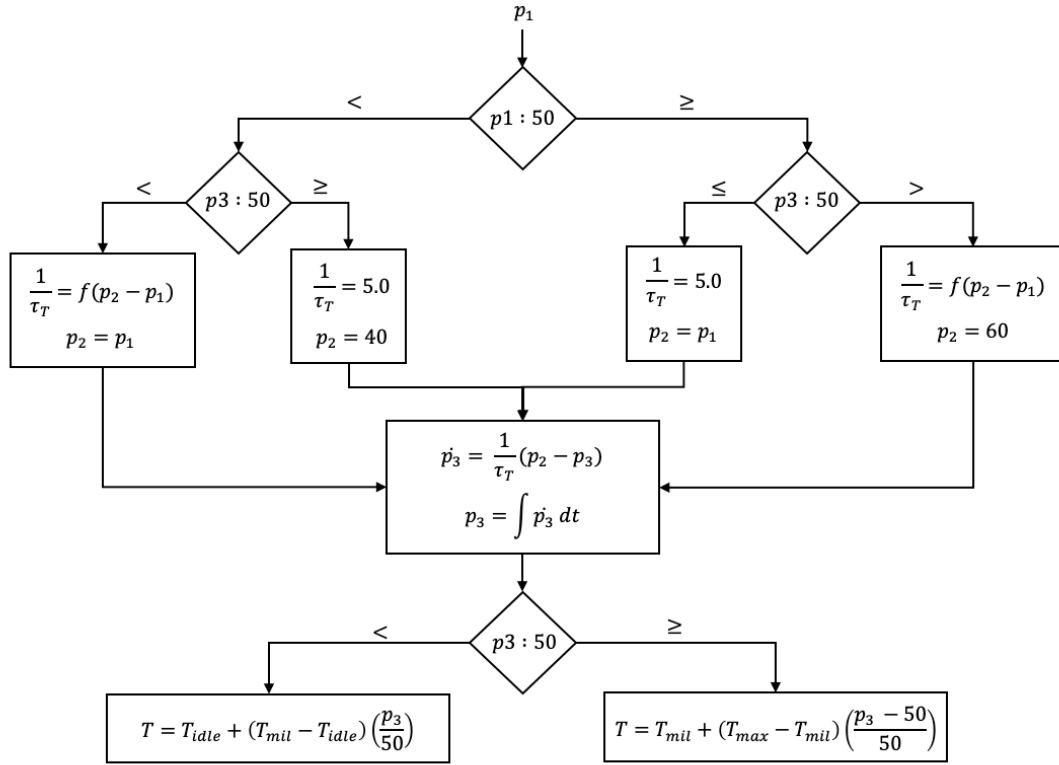


FIGURE 3.5 First order engine model tree, adapted from Nguyen et al. [16]

Although the output from Table A.1 corresponds to the a specific thrust value at a given set of state dynamics, this does not correspond to an instantaneous output from the engine model. Shown in Figure 3.5 is an engine model logic tree schematic used for determining the output thrust from the engine at any specific time, based on the difference between the power demand and the actual power. This engine logic tree schematic required the calculation of four variables, before numerical integration is used to determine the output:

- p_1 is the engine power command based on throttle position as a percentage of the maximum power available for that set of state dynamics;
- p_2 is the engine power command to the engine as a percentage of the maximum power available for that set of state dynamics;
- $\dot{p}_3$ is the rate of change of the current power over time. This value can be integrated numerically to determine the power output from the engine at that time interval;

- p_3 is the current engine power as a percentage of the maximum power available for that set off state dynamics.

3.8 Aircraft Model Parameters

The translational and rotational equations of motion derived in Section 3.3 assumed constant mass, rigid body dynamics. This means that the mass of the aircraft is fixed and the centre of the gravity does not change during simulation. In addition, the products and moments of inertia described in Equations 3.68 and applied to the rotational equations of motion in Equation 3.69 are also assumed to be constant, with the values displayed in Table 3.1 according to Nguyen et al. [16].

TABLE 3.1 Aircraft model weight, mass, and moments of inertia used in simulation [16]

Weight	91,188 N	Mass	9,295.4 kg
I_x	12,875 kg-m ²	I_z	12,875 kg-m ²
I_{xz}	12,875 kg-m ²	I_y	12,875 kg-m ²

The wing dimensions and reference location for the centre-of-gravity used throughout the simulation, can be found in Table 3.2. The reference distance for the C_g occurs at 35% of mean aerodynamic chord ($\bar{c}$).

TABLE 3.2 Wing dimensions and centre of gravity location using in simulation [16]

Wing Span (b)	9.44 m
Wing Area (S)	27.87 m ²
Mean Aerodynamic Chord ($\bar{c}$)	3.45 m
Reference centre of gravity (c_g)	0.35 $\bar{c}$

3.9 Atmospheric Model

The dataset used to populate the atmospheric model was obtained from the International Organisation for Standardisation (ISO 2533:1975), for Standard Atmosphere conditions. The dataset contains atmospheric characteristics up to

80 km, which are stored offline and loaded into the model as an object-based variable. Similar to the aerodynamic model, linear interpolation is performed online, allowing the density, temperature and static pressure to be determined for intermediate state dynamics. Knowledge of these parameters are required for the equations of motion, calculation of the aerodynamic forces and moments and for thrust generation by the engine model.

Although the F-16 has a ceiling height of 15 km, operating at the upper limits of the flight envelope were deemed to be out scope of this research, since this research focuses on routine cooperative flight between manned and unmanned aircraft. Therefore, operational envelope was defined to have an altitude range of 10 km starting from sea-level.

3.10 Control Surfaces

The aerodynamics discussed in Section 3.6 have parameter coefficients representing each control surface on the aircraft. These control surfaces include the leading-edge flaps, ailerons, elevators, rudders and speed brakes. The three primary control surfaces: ailerons, rudders and elevators have been modelled as first-order systems with a constant time constant, with the leading edge flap deflection scheduled against the angle of attack, in accordance with Nguyen et al. [16].

$$\delta_{lef} = 1.38 \frac{2S + 7.25}{S + 7.25} \alpha - 9.05 \frac{\bar{q}}{p_s} + 1.45 \quad (3.81)$$

Where p_s is the static pressure and $\bar{q}$ the dynamic pressure. Writing Equation 3.81 in state space form:

$$\dot{x}_1 = \alpha - 7.25x_1 \quad (3.82)$$

$$\delta_{lef} = 2.76\dot{x}_1 + 10.005x_1 - \frac{\bar{q}}{p_s} + 1.45 \quad (3.83)$$

where x_1 is the state deflection variable and $\dot{x}$ is the first derivative. The actuator for the leading-edge flap has been modelled with first-order dynamics and a constant time lag of 0.136 seconds with a rate limit of 25 degrees per second and with deflection limits ranging from 0° to 25° [16].

Unlike the leading-edge flap, the aileron, elevation and rudder are not scheduled against a set of system dynamic states. The actuator system for each of the

control surfaces has been modelled with first-order dynamics and with a time constant shown in Table 3.3. All four of the applied control surfaces were solved numerically using Runge-Kutta fourth-order integration. The derivatives of each surface deflection were stored every time-step and used in determining the deflection angle of the subsequent time-step.

TABLE 3.3 Control surface first-order dynamic [16]

Control Surface	Time Constant (s)	Rate Limit (deg/s)	Deflection Limit (deg)
Elevator	0.0495	60	± 25.0
Aileron	0.0495	80	± 21.5
Rudder	0.0495	120	± 30.0
Leading-Edge Flaps	0.1360	25	0 – 25

3.11 Flight and Operational Envelope

The flight envelope is a diagrammatic representation of the operational capabilities of the aircraft. Generally, the flight envelope represents the capabilities of the aircraft in terms of altitude and Mach number.

Using the coefficient of lift, the flight envelope for the aircraft can be estimated analytically. Under the assumption that for fixed-wing aircraft operating in equilibrium flight conditions, the weight of the aircraft is equal to the amount of lift required to maintain altitude. Since the F-16 aircraft has a weight of 91.188 kN [16], the lift force required to maintain altitude can be determined using the following equation:

$$L = 91188 \text{ N} = \frac{1}{2} \rho V^2 C_L S \quad (3.84)$$

Where ρ is the density of air in kg/m^3 for a given altitude, V is the free-stream velocity of the aircraft and S is the reference area. Given a reference area of 27.87 m^2 and a density of air at sea-level of 1.225 kg/m^3 , the stall speed for the aircraft was found to be approximately 54 m/s, since the maximum coefficient

of lift is approximately 1.8 [16].

$$V = \sqrt{\frac{91188 \text{ N}}{0.5 \times 1.225 \frac{\text{kg}}{\text{m}^3} \times 1.8 \times 27.87 \text{ m}^2}} = 54.5 \text{ m/s} \quad (3.85)$$

The scope of this research is aimed towards development of an online path planning algorithm for determining continuously flyable and dynamically achievable paths during fully autonomous flight. A reasonable assumption can be made that the flight control system should prevent the aircraft from operating at the lower end of the flight envelope near the stall region, due to dynamic instability concerns. Additionally, due to limitations with the aerodynamic dataset published by Nguyen et al. [16], the aerodynamic data is only valid during subsonic flight. Therefore, by evaluating the dynamics of the vehicle analytically over the entire altitude range and airspeed range, the operational flight envelope in Figure 3.6 was determined.

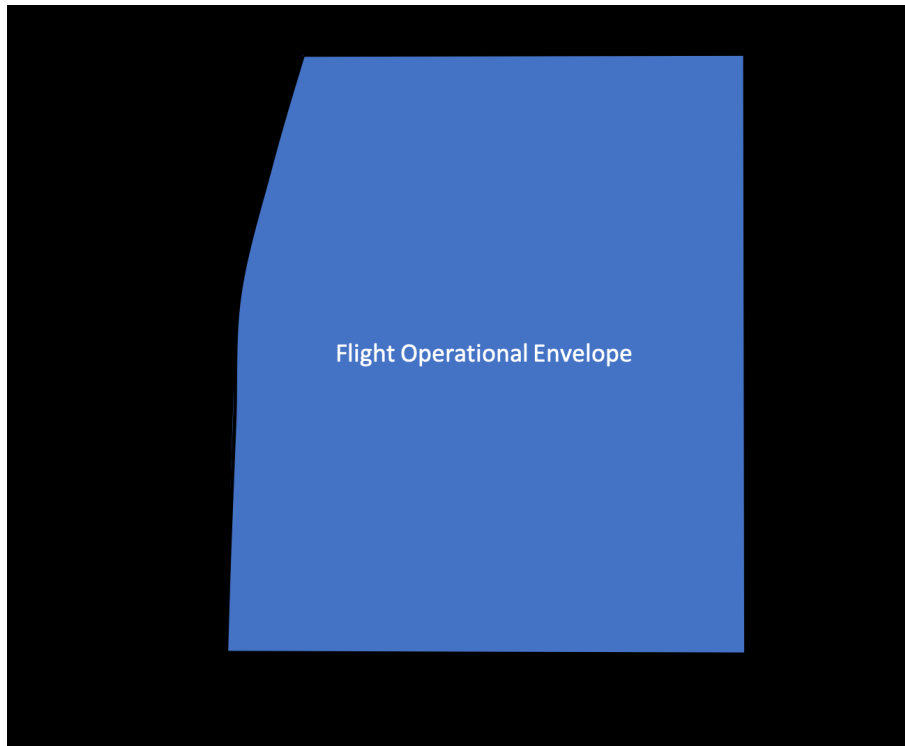


FIGURE 3.6 Analytically derived flight operational envelope

3.12 Summary

Throughout Chapter 3, the equations of motion for the flight physics model have been derived, including the forces and moments and axis system translations. In addition, the aerodynamic model, engine model and control surface models for the aircraft have been described, allowing the flight operational envelope to be established. During Chapter 4, the flight control system for the flight physics model is introduced, allowing rotational and translation motion to be controlled.

Chapter 4

Flight Control System

Having defined the mathematics of the flight physics model in Chapter 3, Chapter 4 introduces the fundamental control system which governs all translational and rotational motion of the flight physics model. This includes an overview of control algorithms, before performing an investigation into controller tuning methodologies using optimisation algorithms. A novel gain-scheduling technique is developed [19], enabling gain surfaces to be represented as normalised hypersurfaces, which are then applied to each of the angular rate controllers, ensuring that both the performance and stability criterion are satisfied regardless of the flight operational conditions.

4.1 Controller Architectures

4.1.1 Background

In the 1930s, Taylor Instrument Companies introduced a redesigned pneumatic controller [95]. This controller differed over previous designs by implementing a proportional and reset control action. This addition eventually evolved into a controller which provided a hyper-reset action to each of the controllers internal states, providing a control action proportional to the integral of the error signal, hence proportional-integral-derivative (PID) controller [95]. This controller architecture rapidly became the most widely used industrial control algorithm, due to its simple structure, convenience and implementation ease [96], and is comprised of three internal controller gains: proportional (k_p), integral (k_i), and derivative (k_d). It is these internal controller gains which determine the control-

lers output signal, $u(t)$, based upon Equation 4.1, where the error signal, $\epsilon(t)$, is measured as the difference between the reference signal and controller output signal, using closed-loop feedback.

$$u(t) = k_p \epsilon(t) + k_i \int_0^t \epsilon(t) + k_d \frac{d\epsilon(t)}{dt} \quad (4.1)$$

Proportional-integral-derivative control has been used to solve many flight control problems, primarily for longitudinal autopilot systems. Wahid and Hassan [97], Forushani et al. [98], Beygi et al. [99] and Kada and Ghazzawi [100] all used PID controllers as pitch axis autopilots for longitudinal aircraft models. These researchers applied control solely to the longitudinal axis for two reasons. Firstly, when mathematically modelling flight, it is accustom to model the aircraft with symmetrical lateral and directional dynamics, meaning that during wings-level flight, the net torque acting on the aircraft about these axes is zero. This negates the need to implement control surrounding lateral and navigational dynamics, particularly when the focus of the research surrounds PID controller tuning as opposed to 6-DoF flight control. Secondly, when considering 6-DoF non-linear dynamics, coupling is present about the lateral and navigational axes. Since PID control is a form of linear control where the output is homogenous based on the input reference signal and the internal controller parameters, when applying PID control around all three rotational axes, controller conflict can occur. For these reasons, when demonstrating tuning techniques, the authors have opted to apply control solely around the longitudinal axis, as this provides a robust solution whilst also neglecting the effects of dynamic coupling on the control architecture. When focusing on more complex aircraft control applications, a more complex control algorithm known as robust control theory, is more readily applicable for dynamic systems with cross-coupling [101].

Robust control theory, first proposed by Zames [102] to address both the performance and stability criterion of a control system, aims to provide better closed loop control when compared to PID controllers [103]. Many robust control theory techniques exist, such as adaptive control, H_2 and H_∞ , parametric estimation, Lyapunov, and Fuzzy Control [104].

H-infinity (H_∞) controllers utilise pure controller error towards synthesising the controller and achieving stabilisation with guaranteed performance. This

gives H_∞ techniques an advantage over PID as they are readily applicable to multivariate systems with dynamic variable cross-coupling [101]. However, not only is H_∞ a frequency domain technique [104], it requires a greater mathematical understanding to apply them successfully when compared to PID control, since they are embedded into the system plant, as opposed to being applied as a layer of control around a system. In addition, an exhaustive optimisation process is required to understand all of the systems uncertainties and output relationship.

When comparing PID and H_∞ controllers, a study conducted by Papakollu et al. [101] compared both PID and H_∞ controllers on a 2-DoF non-linear quarter car suspension system. Although the authors concluded that both controllers did not offer significant benefits when compared to a passive suspension system, PID control displayed better performance characteristics over H_∞ . This improvement included reduced suspension deflection, reduced controller effort, and a reduction to the transient period of the time response. Similar characteristics were also confirmed by Kaur and Ohri [105], with significant reductions to the rise-time when using PID control. However, despite these characteristics, PID control reduced system stability by inducing unwarranted oscillatory motion. As a result, Kaur and Ohri [105] concluded that the settling time of the system was quicker for the H_∞ controller due to increased system stability and reduced overshoot.

During Chapter 1, it was discussed how the scope of this research focuses on developing dynamically feasible paths for a fixed-wing large-scale military fast jet, with online path re-planning during cooperative flight missions. To achieve this aim, the path planning solution is demonstrated on a dynamic model representative of an industrial fast jet, as described in Chapter 3. Although selecting an optimal method of control will impact the performance of the dynamic model to manoeuvre autonomously along paths, identifying the optimal solution is outside the scope of this research, provided the applied methodology ensures adequate dynamic performance and closed-loop stability throughout all stages of flight. Based on this information and the fact that PID control is a simple and robust method of control to implement for time domain systems, PID control was selected as the primary control algorithm when governing all translational

and rotational motion of the aircraft model.

Although it has been discussed how PID controllers are easy to implement due to their simple structure, for PID to be an effective control architecture, knowledge of the optimal controller parameters are required. If PID control is applied with improper controller parameters, this could lead to cyclic and slow recovery, poor robustness and plant instability [106]. Obtaining the optimal internal controller gains requires tuning or optimisation to be performed. In addition, for dynamical systems where the performance of the systems varies based on the state dynamics, the controller gains for the PID controller must be optimised for various state dynamics throughout the operating domain and scheduled using a process known as gain scheduling.

4.1.2 Gain Scheduling

Gain scheduling (GS) is a form of open-loop adaptive control which has been applied to aircraft autopilots since the end of the Second World War [107]. The use of GS enables a family of linear controllers to be applied to non-linear or parameter dependant system plants [108], by defining local optimal control laws based on system state dynamics. For example, if a PID controller is applied to adjust the elevator deflection angle to maintain altitude on a longitudinal aircraft model, the internal controller parameters require tuning. Tuning of a PID controller ensures that both performance and stability criteria are satisfied. However, the efficacy of this altitude hold controller, is dependant on the effectiveness of the elevator control surface, which varies based on altitude and Mach number [107]. As the aircraft either increases in altitude, airspeed or both, the effectiveness of the elevator varies and eventually the applied control law no longer satisfies the performance and stability criteria. However, being able to identify regions within the operational flight envelope based on these two system state dynamics enables a set of trim points to be selected. This trim point set are the locations within the flight envelope where the current control law is no longer satisfactory, and controller tuning is required to establish a new control law which satisfies the performance and stability criteria.

Identifying the location and spacing between each trim point is a quantity estimated by the designer of the system, with the intention of ensuring closed-loop

system stability throughout the entire operating domain [19]. If the spacing of each trim point is too large, as the plant dynamics transition away from the trim point, controller performance can degrade [109]. However, if the system states transition too rapidly between trim points, this can cause unnecessary internal controller fluctuations, which is undesirable and can induce instability. This was the reasoning why Richardson et al. [110] states that scheduling the control law against slowly-varying dynamic states such as altitude and airspeed, is advantageous, as this prevents the system states from transitioning rapidly between control laws. These two factors create a fundamental issue with GS, since if the location and spacing is too large, system stability and performance can degrade at intermediate state dynamics. On the other hand, if the spacing is too small or the system states are quickly varying, this can create rapid fluctuations between control laws and potentially induce instability.

To overcome the plant stability limitations of GS whilst operating at intermediate state dynamics, McConley et al. [111] developed an alternative GS procedure. This procedure applied a quadratic robust control Lyapunov function (RCLF) at trim points, before computing the stability region. This enabled the authors to systematically determine the location of each trim point based on the defined stability regions. However, whilst this methodology systematically determined the location of the trim points, control law switching still occurred at stability region intersections, creating internal parameter discontinuities.

To eliminate controller discontinuities, Stilwell and Rugh [112] linearly interpolated the control law at intermediate state dynamics. Similarly, Choi et al. [113] developed a polynomial model to replace the nearest neighbour and interpolation algorithms for intermediate state dynamics. The authors confirm that use of a polynomial regression reduces computational costs associated with bi-linear interpolation and controller degradation associated with the nearest neighbour algorithm.

Hamilton et al. [19] developed a GS technique which created a hypersurface of control laws, allowing each law to be appropriately applied based on system states and controller input reference signals. This methodology was demonstrated to not only reduce controller effort, but also improved the transient and steady-state error time response, for roll and pitch rate controllers applied to a

non-linear 6-DoF aircraft model representative of a fast jet. To reduce discontinuities, the author normalised the primary control laws against the tuned reference amplitude, with scaling performed internally to the controller, allowing continuous interpolation of the control law with no additional computational costs.

4.2 Design Specification

It is known that the F-16 is dynamically unstable during subsonic flight [114]. To stabilise the aircraft model and achieve controllable flight, a flight control system (FCS) is implemented using series of controllers that govern all radial motion, as well as the airspeed of the aircraft. These controllers should be robust, easy to implement, and output a deflection angle for the control surface models and a throttle position for the engine model discussed in Chapter 3. In addition, the FCS must guarantee closed-loop stability and adequate dynamic performance throughout the operational flight envelope.

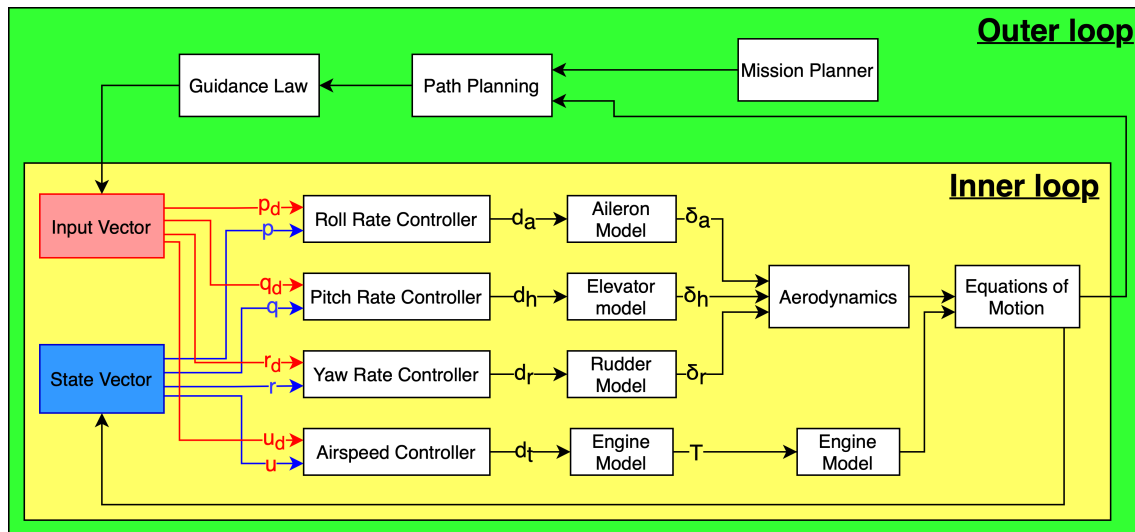


FIGURE 4.1

Shown in Figure 4.1 is the architecture design inner and outer loop of the FCS. The outer loop is comprised of an abstracted mission planner, path planning algorithm and guidance law. The guidance control system (introduced in Chapter 5) receives path planning information from the path planning algorithm (introduced in Chapters 6 and 7) and outputs a set of dynamic demands through the input vector. These input demands are applied as reference signals to the

roll rate controller, pitch rate controller, yaw rate controller, and airspeed controller. The numerical solutions to the equations of motion (discussed in Chapter 3) feedback into a state vector, where the dynamics for roll rate, pitch rate, yaw rate, and airspeed are applied as negative feedback to the corresponding controller in the form an error signal. Each controller outputs a deflection angle or throttle position setting, which is interpreted by the corresponding control module, adjusting the dynamics of the aircraft model and enabling the model to navigate along any dynamically feasible and continuously flyable path.

During Section 4.1.1, a brief review and introduction into various control algorithms was discussed. This review of control methodology concluded that for the purpose of this research, use of PID controllers allow a simple yet effective flight control system to be implemented for the purpose of demonstrating the path planning solution. However, improper PID controller parameters could lead to cyclic and slow recovery, poor robustness and plant instability [106], therefore requiring PID controller tuning. Tuning of the internal parameters of the PID controller allows the optimal control law to be established, ensuring the performance and stability criterion remain satisfied.

However, use of PID controllers have limitations that need to be overcome, particularly due to the integral and derivative components. When the output of a PID controller is saturated, the accumulated error over time expands. When applied to the integral component, the output of the controller remains saturated by the integral component regardless of the controller's position error. This is known as integral wind-up. To alleviate integral wind-up, two mechanisms can be implemented. First, an integral error reset can be initiated when controller saturation occurs. This approach simply resets the integral error and prevents integral wind-up from driving the controller's output. Second, the integral component can be neglected for large position errors when controller saturation occurs for prolonged periods, creating a proportional-derivative (PD) controller. In addition, the derivative term amplifies high frequency sensor noise. Although this can be alleviated by applying a derivative filter creating a PIDF controller [115], it was opted to utilise velocity state feedback as the derivative signal as opposed to computing the derivative of the controller's position error. This created a proportional-integral-velocity (PIV) controller, as opposed to PID controller.

In addition, it was discussed in Section 4.1.2 that a scheduling technique is required to ensure closed-loop stability throughout the operating environment. The application of GS ensures adaptability of the control laws based on the system states, ensuring that throughout the operational flight envelope, the performance and stability criterion remain satisfied.

4.3 Optimisation of PID/PIV Controllers

4.3.1 Heuristic Methods

Early heuristic tuning techniques for PID controllers involved methodologies such as Ziegler-Nichols [116] and Cohen-Coon [117]. However, these methodologies propose limited studies on non-linear systems, since both methodologies require the system plant to sustain an oscillatory response [118]. For systems which are non-homogenous, oscillatory motion from the system plant can be damped by external forces, such as aerodynamics, making heuristic tuning methodologies obsolete.

When attempting to perform controller tuning using Ziegler-Nichols, the operator focuses on increasing the proportional gain (k_p) until a constant amplitude oscillatory response is obtained. Upon achieving this behaviour, the proportional gain is quantified as being the systems, $k_{p,max}$, with numerous rules applied to estimate the integral and derivative components [116]. Although tuning PID controllers using Ziegler-Nichols has been applied to aircraft control, the solution is often regarded as inadequate, due to an aircraft ever-changing operating environment.

Deepa and Sudha [119] compared Ziegler-Nichols to Cohen-Coon for PID tuning of a controller applied to a longitudinal aircraft model. However, based on the dynamics of the aircraft, it can be assumed that the authors either linearised the dynamics of the aircraft or held certain internal dynamic states at constant values. By holding certain dynamic states at constant values, damping of the longitudinal dynamics can be alleviated, allowing methods such as Ziegler-Nichols to be applicable for PID controller tuning. However, this is an impractical solution for larger-scale applications. For example, many highly-maneuvrable fast

jets such as the F-16 aircraft are longitudinally unstable at subsonic airspeeds [114]. Therefore, increasing the proportional gain without having already established some level of control, will only increase the susceptibility of the aircraft to instability.

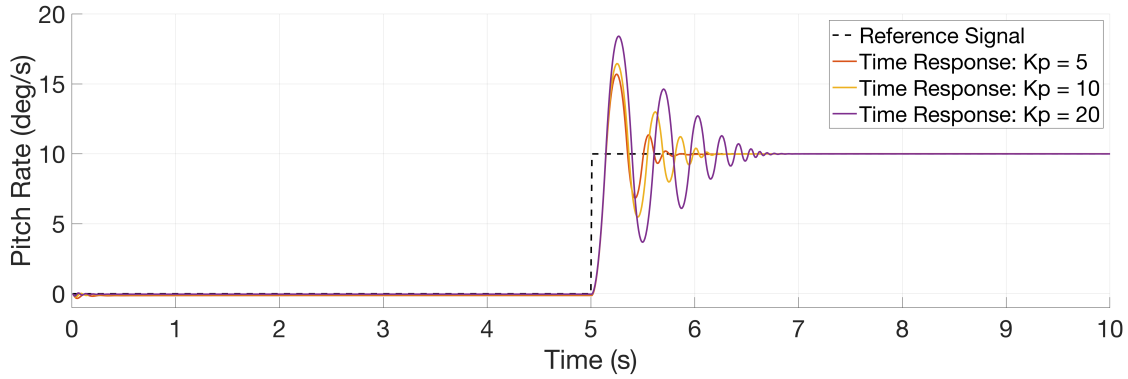


FIGURE 4.2 Pitch rate response for a 10 deg/s demand conducted at various proportional gain values whilst the integral and derivative gains are set at zero

In addition, holding specific dynamic states at constant values to establish a set of estimated tuning parameters does not provide any benefit to the system. As the moment factors such as the aerodynamics are re-established, the state dynamics of the aircraft will begin to fluctuate once again, and the tuned PID controller will be sub-optimal. Figure 4.2 shows the pitch rate time-response of the flight physics model described in Chapter 3 when a reference signal of amplitude 10 deg/s is applied to the pitch rate controller. Despite both the derivative and integral gains being null for the purposes of Ziegler-Nichols tuning, there is no k_p value to maintain a constant amplitude oscillatory response due to aerodynamic damping.

This concludes that the findings of Deepa and Sudha [119] are not transparent due to the aircraft dynamics not being adequately discussed. Furthermore, when tuning a PID controller using one of the discussed techniques, the performance of the controller is only guaranteed when the aircraft is operating within close proximity to those specific state dynamics. To guarantee closed-loop stability whilst also maintaining optimal performance throughout the flight envelope, the controller gains require optimisation at multiple locations, known as trim points, with the controller gains then scheduled based on dynamic state feedback. Obtaining a solution throughout the operational envelope of the flight

physics model using a manual heuristic tuning method is far too exhaustive and an inefficient use of time. Therefore, the focus shifted towards meta-heuristic optimisation algorithms

4.3.2 Meta-heuristic Algorithms

Many modern meta-heuristic algorithms mimic specific behaviours in an attempt to solve complex problems more efficiently [120], whilst also incorporating a stochastic element. Such behaviours mimic the movements of biological entities, such as ants, birds or bats. Regardless of which algorithm is used, the algorithm attempts to minimise or maximise an applied fitness function. The fitness function describes the topology of the search space and defines the dynamic characteristics of the problem.

Selecting a meta-heuristic optimisation algorithm for solving a specific problem requires careful consideration of the fitness function and corresponding surface topology. Many researchers have attempted to develop baseline performance characteristics between different meta-heuristic algorithms. Pareek et al. [121] performed a study comparing various heuristic and meta-heuristic algorithms applied to the tuning of PID controllers. Although the background theory was loosely reported, optimisation was performed on a fourth-order transfer function representative of a ball-and-hoop system, using bacteria foraging optimisation (BFO), particle swarm optimisation (PSO) and a genetic algorithm (GA). The authors concluded that the PID controller optimised using BFO resulted in less overshoot whilst the GA produced a system with the least rise-time and least settling times, whilst also achieving faster convergence. Similar research was conducted by Amaral et al. [122], where a GA was analysed against Ziegler-Nichols. Although optimisation was performed on a low-order transfer function that did not represent any industrial application, the authors identified the advantages of the GA, with reductions to the rise-time, settling time and overshoot. Similarly, Mirzal et al. [123] performed an in-depth analysis into a PID controller tuning using a GA against an undefined iterative solution and Ziegler-Nichols. The authors concluded that the GA achieved an overshoot reduction of up to 70%, whilst a moderately sized improvement to the rise-time and settling time was also achieved.

Arora et al. [124] demonstrate a modified bat algorithm (BA), which utilises an estimated gain set obtained from Ziegler-Nichols to not only reduce the search space, but also provide a faster rate of convergence. This modified BA was compared against other meta-heuristic approaches, such as PSO, harmony search (HS) and ant colony optimisation (ACO). Omar and Saida [125] used a BA and PSO for optimisation of a PID controller for an automatic voltage regulator (AVR). The authors claim the BA is able to outperform PSO in both controller effectiveness and efficiency to obtain optimal solutions. Furthermore, the BA was able to improve the dynamics of the AVR system during high load in uncertain conditions.

Based on the reviewed literature, it is evident that meta-heuristic optimisation algorithms provide suitable solutions for high-dimensional problems involving dynamically-complex systems. In addition, although numerous meta-heuristic optimisation algorithms exist capable of tuning PID controllers, the BA and GA have been identified as popular solutions, with many literature sources indicating clear advantages to both methods. To validate the literature, as well as to understand the benefits of each selected algorithm, an in-depth study was conducted on both meta-heuristic optimisation algorithms. This study aimed to determine which algorithm displayed the best results in terms of convergence accuracy and time to obtain a solution, which could then be applied to tuning a set of PID controller for a non-linear system

Bat Algorithm

The bat algorithm (BA), first proposed by Yang [126] is a meta-heuristic algorithm that simulates the migration of micro-bats hunting for prey. The principles behind the BA are similar to that of PSO, by which a virtual population moves through the search space using a set of kinematics. Since micro-bats typically hunt for prey during night or in dark environments, such as caves, micro-bats navigate the environment using a type of sonar called echolocation. By implementing echolocation, which includes a pulse rate emission for each bat as well as a loudness value, it also adds an element of harmony search optimisation, helping to increase search space exploration. To simulate the migration of virtual micro-bats using echolocation, the following algorithmic process is

idealised.

1. Each i^{th} bat within a population is initialised with a series of echolocation dynamics corresponding to a pulse rate emission (r_i) and loudness (A_i). The pulse rate emission is an indication of the frequency at which each bat performs echolocation. Each bat is initialised with a small pulse rate emission and large loudness value. As the bats migrate through the search space and begin to locate food sources, the loudness decreases whilst the pulse rate emission increases using the following relationships,

$$A_i^{t+1} = \alpha A_i^t \quad r_i^{t+1} = r_i^0 [1 - \exp(-\gamma t)]$$

where α is the loudness degradation coefficient, γ is the pulse rate exponential decay constant and r_i^0 is the initial pulse rate emission when time was equal to zero.

2. For every movement the i^{th} bat flies randomly throughout the search space based on its frequency (F_i) and velocity (V_i). The velocity of the bat is then used to update the its new temporary location within the search space. The frequency is given by,

$$F_i = F_{min} + (F_{max} - F_{min})\beta$$

where β is a random vector between zero and one ($\beta \in [0, 1]$), drawn from a uniform distribution. The values F_{min} and F_{max} represent the minimum and maximum frequency coefficients, which are set as internal algorithm parameters. Adjusting F_{min} and F_{max} changes the range of frequency values, allowing the bats to become more explorative and perform larger movements. However, if the frequency range is too large, the bats may never converge onto a solution. The velocity at a given time is updated based on the previous velocity and the relative distance to the current best solution of the bat population. By using a relative distance between the location of the current bat and the best bat, this attempts to navigate each bat towards the current best known solution, similar to PSO.

$$V_i^t = V_i^{t-1} + (x_i^t - x_{best})F_i$$

Finally, the bat can be given a new temporary location, x_i^t

$$x_i^t = x_i^{t-1} + V_i^t$$

3. Since each bat moves in the direction of the best bats locations, to add diversity and increase exploration, each bat can be subjected to a random walk. This random walk only occurs when the pulse rate emission of the i^{th} bat's echolocation is greater than a random value generated from a normal distribution. The random walk is applied using the following,

$$x_i^t = x_i^{t-1} + \eta A^t$$

where η is a random vector between negative one and positive one drawn from a normal distribution ($\eta \in [-1, 1]$). The value A^t represents the average loudness of all the bats within the population at this given time interval.

4. The new location of the bat is stored as a temporary location, with the actual position of the bat only being updated if the fitness of the temporary location is better than the fitness of the previous position. This process is similar to that of PSO as the bats move through the space based on knowledge of their best location and the population's best location.
5. If the bat successfully moves to a new location, the loudness (A_i) decreases, whilst the pulse rate emission (r_i) increases based on the relationships as explained in the first point.

Configuration of the internal parameters for any optimisation algorithm is an important aspect that needs to be carefully considered based on the topology and size of the search space. By increasing the size of the population, the stochastic search element of the optimiser is increased, leading to a higher probability that one of the bats will be initialised closer to the global solution. To understand how increasing the size of the population influences the convergence rate of the BA, a study was conducted. For the purpose of this study the algorithm was initialised with an arbitrary set of internal parameters, denoted as the default settings throughout the remainder of this discussion. The default settings for the BA were created based on the algorithm settings explained in

[126], and are as follows: F_{min} set with a value of zero with F_{max} equal to one, giving a total frequency range of one unit. Yang [126] specifies that the magnitude of the bat dynamics and echolocation parameters are dependant on the search space domain as well as the convergence characteristics, with each bat assigned a loudness value randomly generated between zero and one. Therefore, for the purpose of this study, the coefficients for echolocation were applied with mid-point values of 0.5, for both the loudness decay, α , and pulse rate emission coefficient, γ . For reasons which will become apparent later in the discussion, each bat within the population was allowed to move 50 times within the search space.

TABLE 4.1 Bat algorithm optimisation of test functions with default settings (Population Size 20)

FUNCTION	CONVERGENCE ACCURACY (%)									
	1	2	3	4	5	6	7	8	9	10
Ackley	96.9	96.8	97.2	96.9	97.2	97.1	97.5	96.8	96.7	97.3
Beale	47.4	50.1	50.1	48.5	46.9	49.2	49.8	46.2	50.6	50.5
Cross-In-Tray	99.9	99.9	99.9	99.8	99.8	100	99.9	99.9	100	100
Egg Crate	97.6	96.5	98.4	97.2	96.8	96.9	97.8	96.8	96.5	97.6
Goldstein-Price	97.2	96.8	97.3	97.1	98.2	98.1	97.6	97.4	97.6	97.2
Holder-Table	67.3	66.2	68.9	67.5	66.0	71.2	66.9	68.9	67.6	67.7
McCormick	97.6	96.7	97.2	97.0	97.5	97.6	96.5	98.1	96.5	96.8
Levi	80.2	78.4	78.2	81.4	77.9	78.3	78.9	77.3	77.6	78.8

Using the default settings indicated above, optimisation was performed with an initial population size of 20 bats for all eight of the optimisation test functions in Appendix B. A total of 10 evaluations were performed, where each evaluation comprised of 1000 randomly produced optimisation runs. By performing this many optimisations, the performance of the algorithm could be accurately depicted, with the performance evaluated as a function of the accuracy to converge onto the global solution, as well as the total runtime to obtain a solution. Convergence onto the global solution was satisfied if any member of the bat population achieved a fitness value within three decimal places of the

globally defined solution. The convergence accuracy can be seen in Table 4.1.

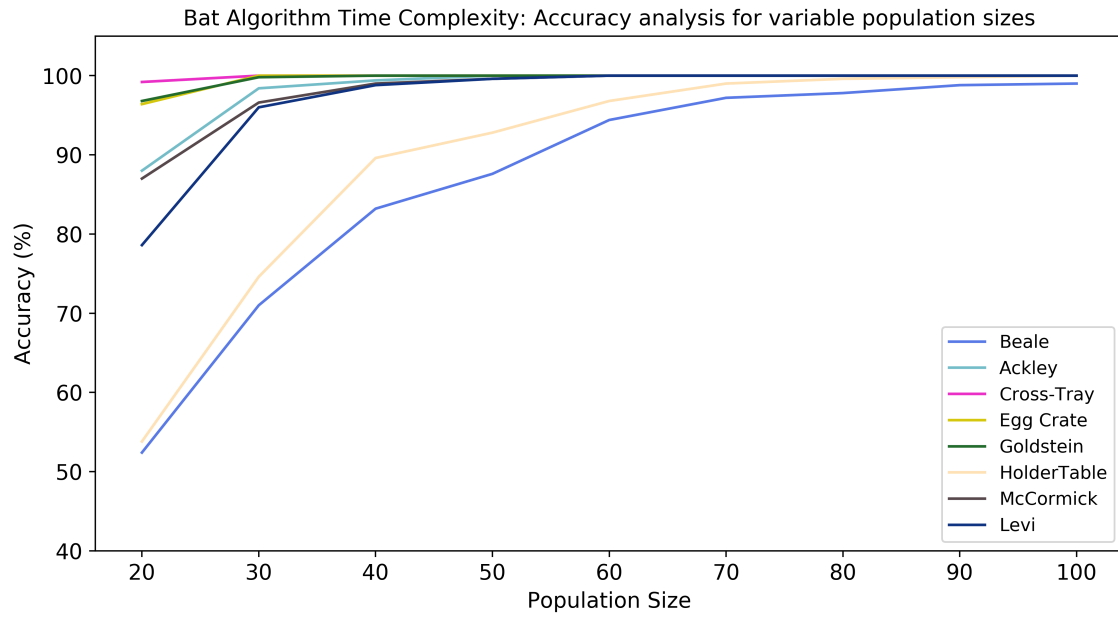
It can be seen from the results in Table 4.1, that for a population size of 20 bats with a fixed number of movements, the BA performed quite well across several of the optimisation test functions. Functions such as the Ackley, Egg Crate and Cross-In-Tray all contain very complex surface topologies with multiple local minima. Despite this, the BA attained a very high rate of convergence towards the global solution, and very rarely become entrapped by local minima. However, for functions that have a very shallow gradient topology with very few local minima, such as the Beale function, the BA performed inadequately, with a convergence accuracy around 50%. Having performed an initial analysis for a population size of 20, the experiment was repeated, but this time for a population size of 50, with the convergence accuracy shown in Table 4.2.

TABLE 4.2 Bat algorithm optimisation of test functions with default settings (Population Size 50)

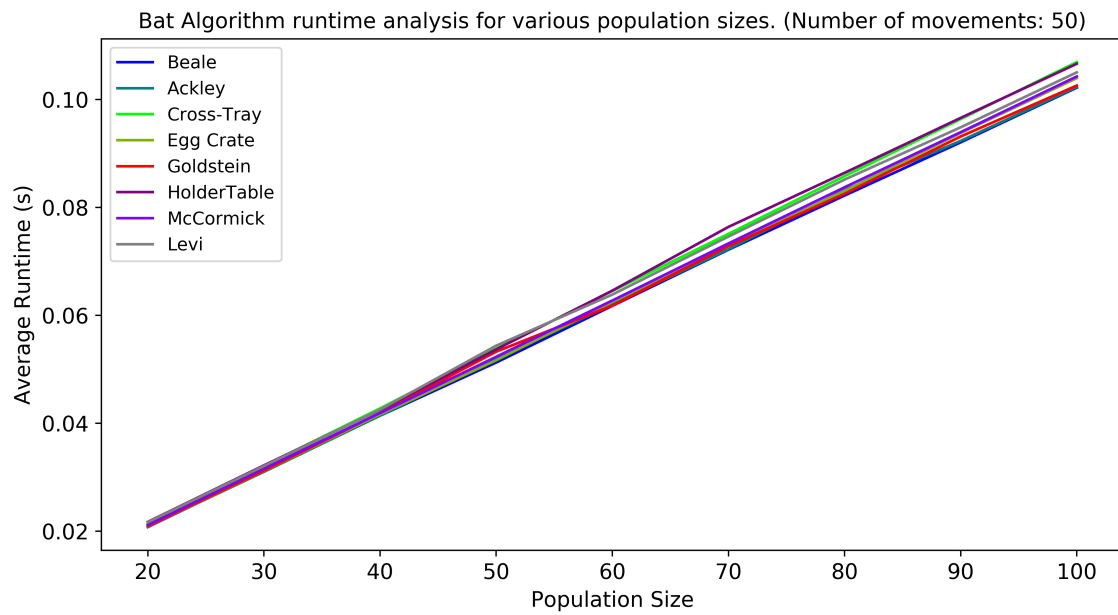
FUNCTION	CONVERGENCE ACCURACY (%)									
	1	2	3	4	5	6	7	8	9	10
Ackley	99.9	100	100	100	100	100	100	100	100	100
Beale	88.1	88.9	88.7	90.9	87.6	88.9	87.6	88.6	89.1	88.8
Cross-In-Tray	100	100	100	100	100	100	100	100	100	100
Egg Crate	100	99.9	100	100	100	100	100	100	100	100
Goldstein-Price	100	100	100	100	100	99.9	100	100	100	100
Holder-Table	95.3	97.4	97.1	96.6	95.9	96.5	96.9	96.5	96.8	97.5
McCormick	100	100	100	100	100	100	100	100	100	100
Levi	99.9	99.5	99.8	99.9	99.9	99.8	99.7	99.7	99.8	99.7

By increasing the size of the population from 20 to 50, the accuracy of the optimiser to converge onto the global solution increased for all tested optimisation functions. Whilst this was not a surprise as the initial stochastic search of the algorithm was increased, the severity of the increase was quite substantial, particularly for functions where the BA previously showed low performance, such as the Beale and Holder-Table functions, which saw an increase of 40% and 30%. Functions where the BA performed exceptionally with a population size of 20,

experienced either 100% or near 100% convergence, proving the capabilities of the BA to achieve a global solution for single-objective optimisation problems.



(a) Convergence accuracy



(b) Average runtime

FIGURE 4.3 Bat algorithm performance analysis for variable population sizes analysed with a fixed number of 50 movements per bat.

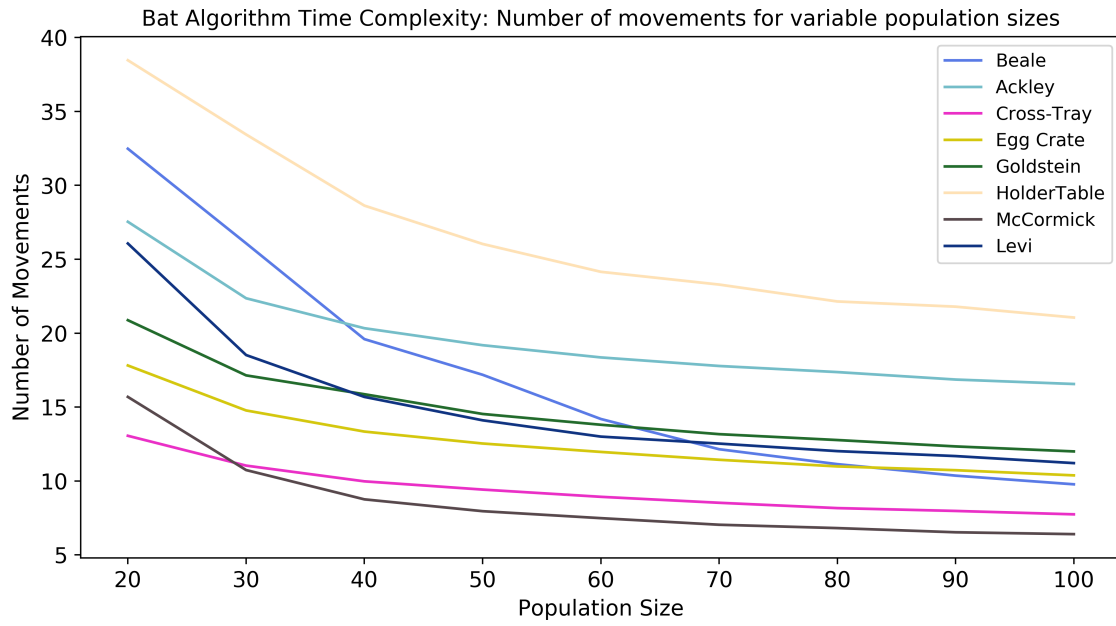
Although the BA was able to produce a convergence accuracy towards 100% for the majority of the functions tested, even at low population sizes, Beale was

the only optimisation test function where it was unable to attain this level of accuracy even when increasing the population size to 100 (Figure 4.3a). Whilst the BA was able to converge onto a global solution for the Beale function with an accuracy of 97%, it is proof that simply increasing the size of the population does not necessarily guarantee that the optimiser will always converge onto the global solution.

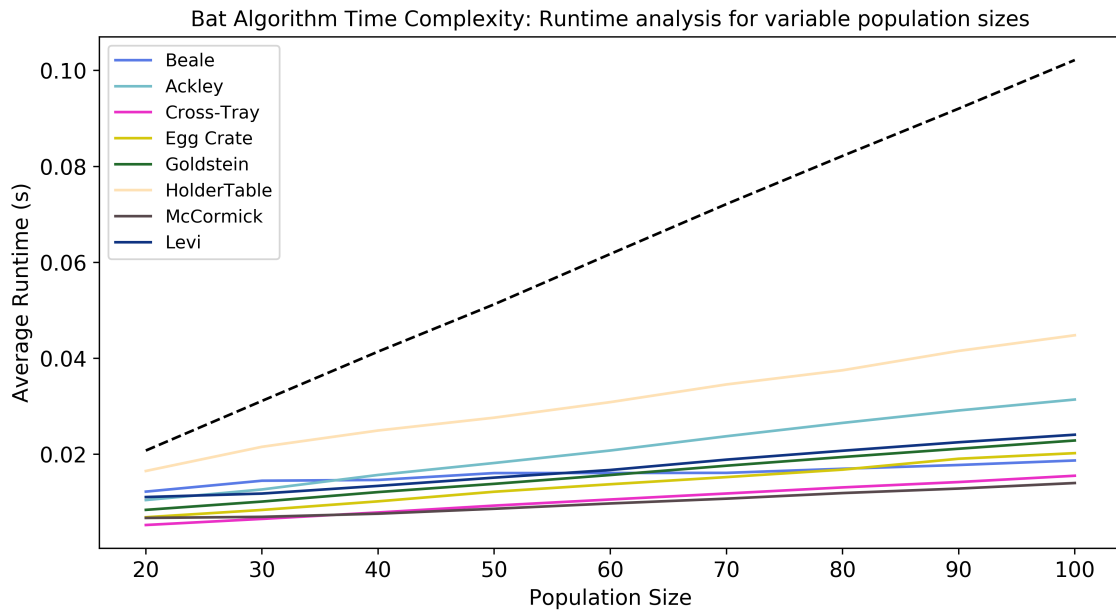
Furthermore, it can be seen from Figure 4.3b that increasing the size of the population for the BA results in a linear time complexity ($O(n)$) when performing the same optimisation over the same number of movements. By increasing the size of the population from 20 to 100, the average runtime increased by over 400%. Although functions where the BA had originally achieved a low converging performance, such as the Beale and Holder-Table, did manage to utilise the larger population size with an convergence accuracy increase of up to 40%, functions where the BA originally achieved a high rate of convergence could not achieve similar increases. For example, when obtaining the global solution for functions such as the Ackley, Cross-In-Tray, Egg Crate and McCormick, the BA was only able to increase convergence accuracy by a few percent despite a runtime increase of over 400%.

During the initial analysis, the number of movements performed by each bat was held at a fixed value of 50, regardless of whether or not the solution had converged. This allowed the time complexity of the BA to be established, which was linearly dependent on the size of the population. However, it can be expected that by increasing the size of the population the initial stochastic search element of the algorithm is more likely to be located closer to the global solution. This means that as the size of the population increases, the required number of movements to converge onto the global solution should reduce. To test this, a second study was conducted. During this study, rather than having each optimisation run conclude once the number of movements had expired, the simulation ended once one of the bats reached the optimisation functions global solution within three decimal places, or if the diversity of the population declined such that the standard deviation was below 0.001. Upon completing the optimisation, the size of the population was then incrementally increased.

Based on the results in Figure 4.4a it is evident that increasing the size of the



(a) Average number of movements



(b) Average runtimes. Dashed black line indicates the average runtime for all optimisation functions from Figure 4.3b

FIGURE 4.4 Bat algorithm number of movements and runtime analyses for various population sizes conducted on all optimisation test functions.

population did decrease the required number of movements to converge onto the global solution for all tested functions. However, the relationship between the size of the population and the required number of movements is not linear.

As the accuracy of the optimiser tended towards 100%, further reductions to the number of movements begins to stagnate, and any additional increase to the size of the population only increased the runtime. This can be seen in Figure 4.4b where the Levi function displays a constant time complexity when the population size is less than 50. However, as the population approaches a size of 50 and the accuracy of the solution tends towards 100% (Figure 4.3a), the time complexity of the Levi function is no longer constant and increases linearly. Whilst it is expected that this behaviour would be present for the additional functions, since they converged and flattened very quickly at low population sizes to an accuracy of 100%, it was not possible to confirm this characteristic with the available data. However, the Beale function does not experience curve flattening within the analysed population sizes as the solution does not achieve 100% accuracy. As a result, the time complexity of the BA when analysing the Beale function remains constant with a big-O notation of $O(1)$.

Based on the computational studies conducted into the BA, it is evident that increasing the size of the population does increase the rate of convergence and provides a more robust solution, with linear time complexity. However, by enabling the solution to converge and satisfy an end criterion, the time complexity of the optimiser was reduced but remained linear with a shallower gradient for most optimisation functions. It has also been proven that the BA has very good diversity and has shown a high degree of convergence towards a global solution over a wide range of contrasting optimisation functions.

Genetic Algorithm

Genetic Algorithms (GA), first proposed by John Holland in the 1960s, are a method of solving optimisation problems using a stochastic search inspired by Darwinian evolutionary principles associated with natural genetics. Genetic Algorithms are typically comprised of three operator functions: selection, crossover and mutation, all of which are chosen by analogy with the natural world [127]. Genetic algorithms can be implemented computationally with the following algorithmic process:

1. Configure the initial population(s), using one of two methods:

- (a) An individual can be introduced into the search space. This individual can be formulated based on an educated guess, assumptions, or pre-existing knowledge of the search space [127]. The individual is then subjected to mitosis cell division, with small-scale random mutation being applied to each genome forming a total population.
 - (b) The entire population can be randomly generated throughout the search space. This is a more common approach when pre-existing knowledge of the search space does not exist.
2. Determine the fitness of each individual within the population and perform selection. At this stage, the selection process determines which individuals within the population will move onto the next generation and have the potential to reproduce. The selection process is the most important aspect of a GA as this is the driving force behind how the algorithm evolves over time towards converging onto the global solution.
3. The surviving population are subjected to a secondary selection by which they are chosen to enter the reproduction pool. This process can be similar to the previous selection or an entirely different selection method. Individuals who are chosen to enter the reproduction pool are responsible for the next generation offspring.
4. Crossover is performed by pairing two individuals from the selection pool to create a series of offspring. To increase the diversity of the offspring, each gene can have a random mutation applied. This mutation has a probabilistic chance of occurring and has a set mutation range, limiting the random search of the GA.
5. This process is repeated until the population has either converged onto a single solution, the diversity within the population dies out, or the generation number exceeds the maximum number of generations.

Upon initialising the population and evaluating the fitness, selection is the first step to every subsequent generation of a GA and is aimed at exploiting the best characteristics of individuals within a population. These characteristics

should guide the GA to converge to an optimal solution over multiple generations [128]. One of the most important aspects of the selection process is finding the balance between having a high or low selection pressure.

Selection pressure is the severity at which the fittest individuals are favoured and selected to move into the reproduction pool or onto the next generation. It is selection pressure that drives the GA to converge onto an optimal or a sub-optimal solution [129]. If the selection pressure is too high, then the selection process will favour fitter individuals. Whilst this does increase the convergence rate of the GA onto a single solution, the diversity of the population can degrade quickly, causing evolution to stagnate and become dependent on mutation. This can cause the GA to converge onto a sub-optimal solution due to the lack of exploration and population diversity. On the other hand, if the selection pressure is too low then the GA may never converge or take an unnecessarily long time to converge on to a solution [129]. Numerous selection methods exist that aim to diversify the population during selection and keep evolutionary growth competitive throughout multiple generations, such as roulette wheel selection, rank selection and tournament selection.

Roulette wheel selection (RWS) is one of the most frequently used selection strategies, which is modelled around a casino roulette wheel. A traditional casino roulette wheel is comprised of 37 individual sections all of which have a unique numerical identifier from 0 to 36. The player can opt to place their token(s) on individual numerical sections, or they can opt to place their token(s) on a much more defined region, increasing the overall probability of being selected: such as red/black and odd/even. The exception to this rule is that the numerical value zero does not correspond to any of the larger play areas defined above.

When theoretically modelling the roulette wheel, the wheel is partitioned into individual sections all of which correspond to different levels of money. Furthermore, the roulette ball is replaced with a pointer which ultimately decides the location of the winning area. Whilst one cannot foresee which section the pointer will point to, the probability of that sector being chosen is proportional to the central angle of the sector [130]. For the application of a GA, the sections on the roulette wheel represent each individual within the population with the

central angle size determined by the proportion of the individual's fitness when compared to the best fitness within the population. Therefore, the probability of an individual P_i being selected can be determined as follows [131]:

$$P_i = \frac{F_i}{\sum_{j=1}^{Pop\ Size} F_j} \quad (4.2)$$

Rank selection (RS) aims to prevent premature convergence associated with RWS [132]. Firstly, the population is sorted from best to worst based on each individual's fitness. Secondly, each individual is assigned a specific rank based on their location within the sorted population. The fittest individual is assigned rank N , whilst the least fit individual is assigned a rank of one [128]. Finally, selection is performed proportional to an assignment function [133], where the assignment function for RS is most commonly applied as a linear function (LRS), however exponential assignment functions (ERS) also exist. The probability of selecting an individual P_i based on LRS can be expressed as the following:

$$P_i = \frac{rank(i)}{n \times (n - 1)} \quad (4.3)$$

whilst the probability of selecting an individual P_i based on ERS can be expressed as follows: [128]:

$$P_i = 1.0 \times \exp\left(\frac{-range(i)}{c}\right) \quad (4.4)$$

where

$$c = \frac{(n \times 2 \times (n - 1))}{(6 \times (n - 1) + n)} \quad (4.5)$$

Tournament selection (TS) is the process of choosing a group of individuals randomly from the population and analysing the group, with the fittest individuals within a group selected for further genetic processing [133]. The individuals within a group that are not selected, are either returned to the general population where they have a chance to enter a new group and compete in a new tournament to be selected, or can be eliminated from the selection process. This process is repeated a set amount of times, or until the selection pool has been filled. Tournaments are often held as binary competitions (BTS) where pairs of individuals face off against one another. However, binary competitions increase the time complexity as TS is proportional to the number of competitions being

held for each generation, giving TS a time complexity of $O(n)$ [133]. Therefore, by expanding the number of individuals entered into each competition, the time complexity can be reduced as the total number of competitions, n , is also reduced. However, although increasing the number of individuals entered into each competition can reduce the time complexity, it can also decrease diversity. When considering larger tournaments, it is important to not have groups that are large relative to the overall population. For example, if a group of 20 is entered into a single tournament round and the total population is made up of 100 individuals, there is a high probability that the fittest individual will be matched against individuals with similar fitness rankings. If individuals not selected during a tournament are not returned to the general population, this process could eliminate high ranking individuals from being selected and resulting in weak selection and slow evolution.

Comparative studies have often been the focus of research when it comes to selecting the best selection method for use in a GA. Goldberg and Deb [133] performed a comparative study into multiple selection schemes including a linear RS and BTS. While the authors concluded that both RS and BTS had almost identical performance characteristics, BTS would be the preferred method due to a reduction in time complexity. Zhong et al. [130] performed a similar comparative study between RWS and TS, and concluded that despite the size of the population being relatively small, the computational efficiency of TS gave a clear advantage over RWS when applied to seven experimental optimisation functions.

To understand how different selection methods impact the diversity of the population over multiple generations, a basic study was performed. This study was conducted on two different selection methods (TS and RWS) identified through the review of the literature. A benchmark selection method was also tested, denoted as the best selector (BS). The BS method is a simple selector method which ranks the population in ascending order based on the fitness value. The fittest individuals are then selected to continue into the next generation and are also selected to go onto the reproduction pool. It is expected that this method will have the highest selection pressure since the fittest individuals are guaranteed to be selected. This should result in a lack of long-term

population diversity with the GA becoming a random search algorithm where exploration of the search space is based solely on the random mutation applied to the offspring. Finally, a hybrid selection (HS) method was also introduced. This custom selector method is a combination of TS and RWS. During traditional TS, the tournament is populated randomly, and the winner chosen based solely on their fitness value when compared to the rest of the tournament population. This means that provided the fittest individuals are selected at some point to compete in a tournament, they are guaranteed to be selected, giving TS a high selection pressure. To reduce the selection pressure and increase the diversity of the population, the winner of the tournament for HS is selected based on a probability through performing a miniature RWS. Whilst this method does give fitter individuals a higher probability of being selected, it does not guarantee that they will be selected. Use of HS will increase the exploration of the population throughout the search space and keep the population diverse for more generations.

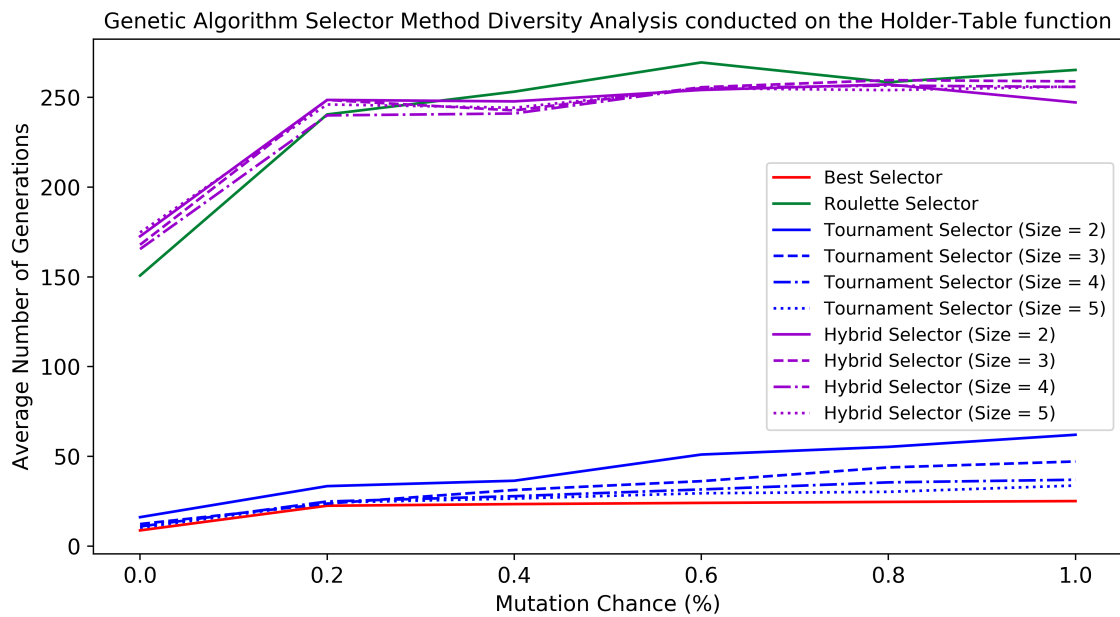


FIGURE 4.5 Genetic Algorithm selector method population diversity study conducted on the Holder-Table optimisation function.

For the diversity study, the GA was given a population size of 100 and a reproduction pool size of 50, meaning that every generation 25 couples were subjected to crossover. During crossover, the GA was configured to have a muta-

tion chance of 0% which was then iteratively increased throughout the study up to 100%. The mutation range was held constant throughout the study with a value of 0.1, limiting the effect of random search. By iteratively increasing the chance of mutation with each evaluation, the stochastic search aspect of the GA increases, prolonging the diversity for more generations. To measure the diversity of the population the standard deviation was determined at each generation. If the standard deviation of the population was less than 0.01, the simulation would end due the population having converged onto a specific region of search space. Each selection method was evaluated 1000 times, establishing a mean value corresponding to the average number of generations. Both TS and HS were both evaluated for various tournament sizes, ranging from two to five. The results from the selector methods diversity study can be seen in Figure 4.5.

Based on the results, it is evident that the BS and TS methods have a very high selection pressure, favouring fitter individuals. The BS method was able to retain population diversity for an average of only nine generations when the mutation chance was set at 0%, which increased to 19 generations as the chance of mutation increased towards 100%. This demonstrates how a high selection pressure method causes the population to become susceptible to premature convergence, due to the lack of exploration and diversification within the population.

Despite the literature indicating that TS is the preferred method for selection, the high selection pressure associated with TS means diversity within the population becomes stagnant very quickly. Additionally, due to a flaw within the TS process, increasing the size of each tournament has a proportional effect on the selection pressure. For example, the selection process for TS randomly populates each tournament, with the fittest individual winning the tournament and the losers returning to the population. This process is repeated until the selection pool has been filled. Consider a GA using the same internal parameters as used during the diversity study. At each generation a total of 100 individuals are analysed using TS to fill a reproduction pool size of 50. Selecting the population for the first tournament of any generation gives each individual a 2% chance of being selected. As the tournaments progress, the unselected population declines and the probability of being selected for the subsequent tournament increases. Given that the fittest individual is also chosen to win the

tournament, the chances of the fittest individuals being selected each generation is relatively high for this configuration. Furthermore, as the size of each tournament increases, the chance of being selected for any specific tournament also increases, further increasing the selection pressure. This can be seen in Figure 4.5 where the population diversity declines for TS as the size of the tournament increases. This means that TS heavily favours the fittest individuals from each generation and the GA can quickly become a random search algorithm which is dependent on the mutation range.

Unlike TS, HS does not guarantee that the fittest individuals will win a tournament, but does increase the probability of such an occurrence. By implementing a RWS into determining the winner of each tournament, the diversity of the population remained high for hundreds of generations regardless of the mutation chance. When factoring no chance of mutation, RWS and HS were still able to provide a diverse population for over 150 generations, with this number increasing to over 240 generations as the chance of mutation approached 100%.

One interesting characteristic is how the chance of mutation directly impacts the overall diversity of the population throughout numerous generations. For BS and TS, both of which heavily favour the fittest individuals, the relationship between mutation chance and the number of generations is linear. However, since both HS and RWS do not guarantee the fittest individuals will progress, the relationship between the number of generations and mutation chance is less predictable. This indicates that RWS is not as heavily impacted by the application of random mutation, as diversity is prolonged through the probabilistic selection approach, and not due to a randomly applied mutation.

From the literature, it was discussed that the ideal selector method needs to find the right balance between a high and low selection pressure. If the selection pressure is too low, the evolution of the population is prolonged for too many generations and convergence can be slow. However, if the selection pressure is too high then the GA will favour fitter individuals and premature convergence can cause the GA to be susceptible to sub-optimal solutions. The diversity study conducted has proven that the RWS and HS have a low selection pressure creating very high diversity characteristics, which could produce slow evolution and stagnated convergence towards the global solution. This corroborates the liter-

ature which indicates that TS is computationally more efficient than RWS due to a higher selection pressure whilst the accuracy to attain the global solution is similar for both selector methods.

To further understand the impacts of selection pressure on the performance of the GA a similar study to that conducted on the BA was performed. This study enabled a direct comparison to be established between the GA and the BA based on the required number of generations to converge onto a solution, the accuracy of the convergence onto the global solution and the average runtime for the selector method. Throughout the performance study, the GA was configured with a set of internal parameters that were held constant. Similarly to the BA, the GA was initialised with a population size consisting of 50 individuals whilst the reproduction pool size was set at 20 individuals for each generation. During reproduction, each set of parents identified through the selection process produced a total of four offspring.

To increase the exploration of the search space and prolong population diversity, each offspring had a 100% chance of mutation being applied. The range at which mutation could be applied was the only internal parameter of the GA not held constant between optimisation functions. Since the size of the search space varied between optimisation test functions and the size of the population was not adjusted accordingly to ensure a constant population density, allowing the initial mutation range to be scaled based on the size of the search space should produce a more robust optimiser. This aids in alleviating any discrepancies associated with exploration and population density due to the size of the search space.

At the end of each generation the mutation range decayed 5%, reducing the range of mutation applied to each offspring over time, favouring long-term convergence. Furthermore, since this performance analysis was conducted with constant internal parameters, this facilitated in establishing which selector method provided a robust GA capable of solving many complex optimisation problems without requiring internal parameter changes. Successfully converging onto the global solution was acknowledged if the best individual within the population achieved an absolute difference between itself and the target global solution less than 1×10^{-3} . The optimiser classified a fail-to-converge if this difference was

not achieved within 500 generations or the standard deviation of the population dropped below 1×10^{-3} , signalling that the diversity within the population had died out. Each selector method was evaluated 500 times for each optimisation test function, establishing a convergence accuracy, average number of generations and runtime analysis.

TABLE 4.3 Convergence Accuracy (%) for Genetic Algorithm Selector Methods

Functions	BS	RWS	TS				HS			
	-	-	2	3	4	5	2	3	4	5
Ackley	56.20	98.90	89.40	70.60	68.70	67.70	99.40	90.20	82.42	79.40
Beale	90.80	95.20	96.80	92.60	87.80	90.20	98.00	94.80	92.40	93.40
Cross-In-Tray	100.00	99.20	100.00	100.00	100.00	100.00	88.40	81.40	71.60	78.00
Egg Crate	99.20	100.00	99.40	99.40	98.40	99.00	99.60	100.00	99.80	100.00
Goldstein-Price	99.80	96.00	100.00	100.00	100.00	100.00	88.80	88.00	89.20	88.00
Holder-Table	100.00	73.40	100.00	100.00	100.00	100.00	15.00	24.60	30.40	29.40
McCormick	100.00	100.00	100.00	100.00	100.00	100.00	80.20	90.60	93.20	93.40
Levi	100.00	100.00	100.00	99.60	100.00	99.80	99.60	100.00	100.00	99.60

When analysing the convergence accuracy results in Table 4.3 it is evident that the difference in selection pressure across the various selector methods has advantages and disadvantages based on the surface topology of the optimiser functions. It can be seen that both BS and TS, which were identified to have a high selection pressure struggled with global convergence for the Ackley function. This high selection pressure associated with the BS resulted in premature convergence onto a local solution for the Ackley function approximately 50% of the time. Whilst TS did achieve a 89.40% accuracy for a tournament size of two, the performance degraded as the size of the tournament increased. It was known from the diversity study that increasing the size of the tournament increased the selection pressure, resulting in an optimiser more susceptible to convergence onto sub-optimal solutions. However, low selection pressure methods such as the RWS and HS managed to achieve an accuracy of almost 100% for the Ackley function. This solidifies the literature that low selection pressure is less susceptible to local minima as it maintains a high level of exploration as opposed to focusing on one solution. Similarly to TS, HS appears to amplify its selection pressure as the size of the tournament increases, causing the GA to become more susceptible to convergence onto sub-optimal solutions.

One of the most interesting characteristics from this analysis occurs for op-

timisation test functions that have more than one global solution. Functions such as the Cross-In-Tray and Holder-Table consist of four global solutions. Since the population is initialised randomly throughout the search space, there is an increased probability that at least one individual will be located close to one of the global solutions, and therefore exploration is not as highly valued. For high

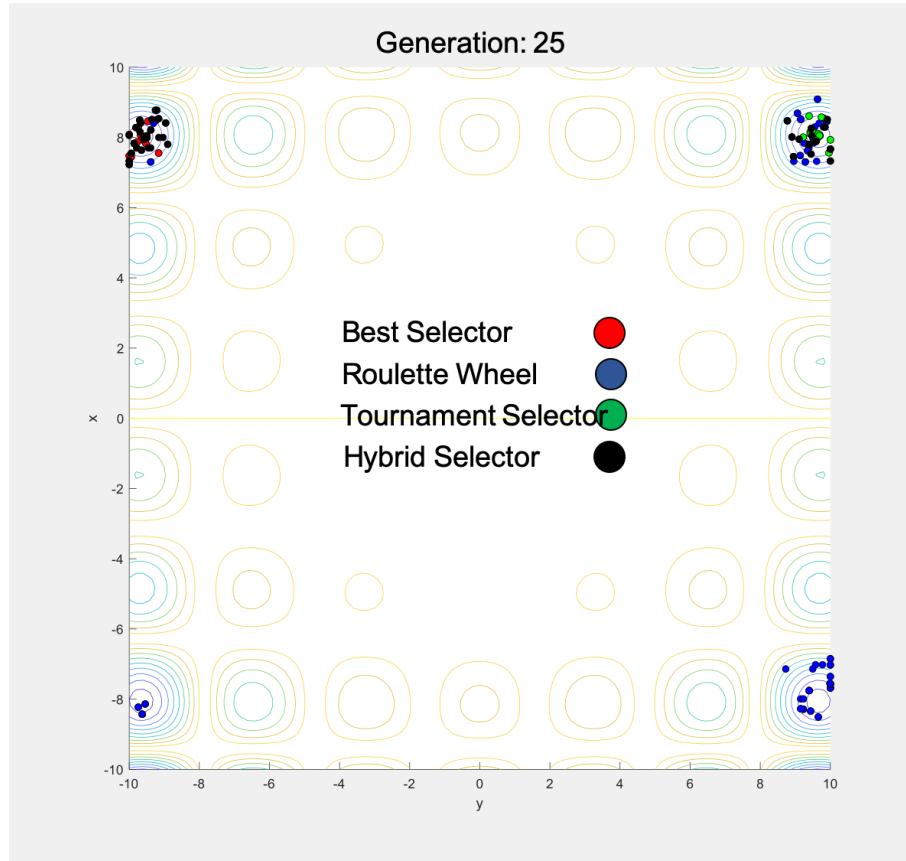


FIGURE 4.6 Population exploration for all four selector methods conducted on the Holder-Table function after 25 generations.

selection pressure methods, this enabled very quick convergence with a high level of accuracy, since the selection method prioritises fitter individuals. This allowed the BS and TS methods to achieve 100% accuracy for these optimisation test functions by exploiting the initial stochastic characteristic. Since low selection pressure methods do not favour the fittest individuals, RWS and HS were unable to exploit this search space characteristic. As a result, RWS and HS favoured long-term exploration and ended up partitioning their populations between multiple global solutions, creating a fragmented search. This can be seen in Figure 4.6 where all four selection methods were initialised with the

identical populations. After 25 generations the BS and TS methods are both exploring a single solution with the direct aim of convergence. Conversely, the HS method has divided its population between exploring three of the global solutions, whilst and the RWS is exploring all four global solutions. This fragmented population results in slow evolution for RWS and HS as the selector method is prioritising exploration of the entire search space rather than converging onto a single solution.

TABLE 4.4 Average number of Generations to converge for GA Selector Method Performance Analysis

Functions	BS	RWS	TS				HS			
	-	-	2	3	4	5	2	3	4	5
Ackley	150.88	148.93	149.23	151.78	151.70	151.97	151.49	150.28	150.99	150.95
Beale	78.91	78.85	72.63	76.31	82.90	79.65	71.27	78.39	87.02	83.98
Cross-Tray	58.31	58.53	54.35	56.56	57.81	58.12	88.51	106.92	136.81	117.95
Egg Crate	98.44	94.94	95.36	98.45	98.58	98.66	91.35	93.72	97.70	97.02
Goldstein-Price	104.77	109.50	100.60	103.54	104.19	103.69	122.79	126.05	124.03	126.05
Holder-Table	111.61	158.09	106.97	110.85	111.17	110.28	286.78	267.92	255.98	255.02
McCormick	57.03	53.42	54.54	57.69	57.85	57.59	103.34	81.09	75.98	74.46
Levi	100.51	100.44	98.33	101.85	101.85	101.60	106.74	97.78	99.64	102.21

Despite both methods exploring a correct region of the search space and not falling susceptible to local minima, since the population is divided amongst many global solutions, convergence was unable to occur within the applied tolerances. This meant that RWS was only able to achieve an accuracy of 73% for the Holder-Table function, with 15-24% for HS. The slow evolution, fragmented search and stagnated convergence can be seen when analysing the average number of generations during the study in Table 4.4. The HS method averaged 250-280 generations for the Holder-Table function whilst the RWS averaged 158 generations, much lower due to faster convergence. When comparing this to the BS and TS methods which averaged 100-110 generations, it is clear how stagnated the evolution was for these low selection pressure methods. However, this substantial increase in the number of generations can be attributed to a flaw within the stop criteria for the GA. The optimiser would indicate failed convergence once the standard deviation had dropped to less than 1×10^{-3} and the difference between the global solution and fittest individual was greater than 1×10^{-3} . Since the HS and RWS methods were exploring more than one global

solution, the standard deviation remained very high despite a low rate of convergence, causing a prolonged number of generations.

TABLE 4.5 Average runtime to converge for GA Selector Method Performance Analysis

Functions	BS	RWS	TS				HS			
	-	-	2	3	4	5	2	3	4	5
Ackley	0.1884	0.2641	0.3356	0.3447	0.3404	0.3445	0.3301	0.3324	0.3452	0.3392
Beale	0.1094	0.1399	0.1625	0.1728	0.1822	0.1690	0.1659	0.2177	0.1977	0.1829
Cross-In-Tray	0.0837	0.1069	0.1243	0.1345	0.1304	0.1315	0.2115	0.3038	0.3168	0.2663
Egg Crate	0.1394	0.1647	0.2149	0.2250	0.2214	0.2197	0.2163	0.2637	0.2157	0.2157
Goldstein-Price	0.1454	0.1960	0.2268	0.2333	0.2322	0.2328	0.2965	0.2770	0.2812	0.2770
Holder-Table	0.1586	0.2473	0.2452	0.2516	0.2552	0.2534	0.6456	0.6076	0.5800	0.5510
McCormick	0.0804	0.1017	0.1241	0.1314	0.1307	0.1323	0.2316	0.1803	0.1681	0.1600
Levi	0.1393	0.1808	0.2249	0.2304	0.2297	0.2323	0.2271	0.2715	0.2193	0.2153

When comparing the performance of the GA to the BA it is clear that for this set of optimisation test functions the BA shows clear advantages. The BA produced a more robust solution when using the same population size as the GA, with all optimisation test functions converging with an accuracy of 88% or higher. The runtime analysis for the GA (Table 4.5) indicates that the BS method has the lowest average runtime for all optimisation test functions. This was an expected result given the algorithmic process for the BS consisted only of a sorting algorithm which has a linear time complexity. Additionally, since the BS method also had the highest selection pressure, the number of generations was lower than other methods, reducing the average runtime.

One contrasting result is that the RWS appears to have a quicker runtime than the TS, contradicting some of the literature. However, this could be attributed to an artefact of using some default toolboxes within the development environment, creating additional overheads for TS. Regardless, when comparing all of the GA selector methods to the BA it is evident that none of the selector methods can compete with the efficiency of the BA. The slowest runtime achieved for the BA occurred for the Holder-Table function, with a runtime of approximately 0.03 seconds. Despite this being the slowest runtime, the BA was still able to converge onto the global solution in 62% less time than the GA was able to achieve for its fastest function (McCormick), and over 90% quicker than the GA's slowest converging function (Holder-Table).

4.3.3 Summary

Based on the study performed on the BA and GA, it is evident that both algorithms are capable of acquiring solutions to the eight tested optimisation test functions, with clear advantages in favour of the BA. Firstly, based on the tested environments, the BA is able to produce a more robust solution across the various optimisation test functions and across numerous contrasting search space topologies. Secondly, the BA is able to converge onto the global solution between 60-90% faster than the GA across the tested functions. This analysis concluded that, for optimisation of PID controllers, the BA was selected as the most appropriate solution due to its ability to perform highly across different search space topologies as well as fast convergence towards the global solution.

4.4 Angular Rate Controllers

During the development of the computational flight physics model, the equations of motion were derived to represent a six-degree-of-freedom (6-DoF) nonlinear dynamic model, corresponding to three translational accelerations and three angular accelerations.

When applying control to govern the rotational motion of a fixed-wing aircraft, many researchers control the angular position about each axis, through adjustments made to the deflection angle of the corresponding control surfaces. This is deemed to be an effective method of control for many fixed-wing flight applications, particularly applications with either constant-altitude flight conditions or conditions where the aircraft does not perform a full 360° rotation about the roll or pitch axes. However, when considering a highly manoeuvrable fast jet that is capable of completing a full 360 degree rotation about the roll and pitch axes, controlling the angular rotation in this manner is inefficient. This is due to angular wrapping as a full rotation is being completed, and introduces unnecessary controller action reset complications. Therefore, controlling the angular velocity as opposed to the angular position, posed numerous advantages.

Figure 4.7 shows a block diagram depicting the applied angular rate control system that governs the angular velocities of the aircraft. This block diagram includes a highlighted GS block consisting of nine outputs, each representing

the three individual control gains (k_p , k_i , k_d) applied to each of the angular rate controllers. In Section 4.4.1, the optimisation algorithm and optimisation process used to determine the internal controller gains for the three angular rate controllers is discussed. This also introduces how the location of the optimised trim points were identified, allowing a novel gain-scheduling technique to be applied, ensuring closed-loop stability and optimal dynamic performance throughout the flight envelope.

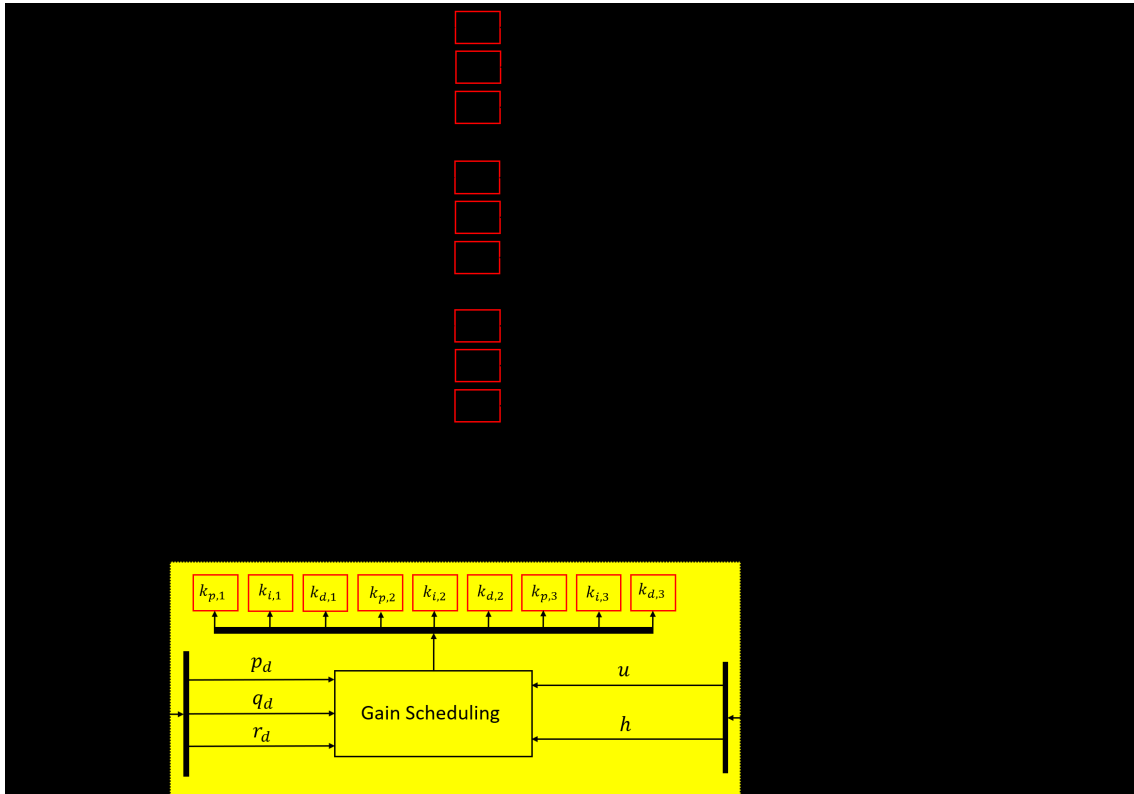


FIGURE 4.7 Block diagram for the angular rate controllers [19]

4.4.1 Optimisation Algorithm

In Section 4.3, various meta-heuristic optimisation techniques were discussed in the form of a literature review. This review determined that, for multi-objective problems, many algorithms exist capable of solving such problems. Therefore, an in-depth study was conducted in Section 4.3.2 to understand the advantages and disadvantages of the BA and GA. Whilst the results from the analysis indicated that the BA outperformed the GA in terms of algorithm robustness and

total runtime, it was important to understand the surface topology before making a decision.

To understand the topology of the search space, a brief exhaustive search was conducted. This exhaustive approach acts as a low-resolution, heuristic brute-force search, where combinations of controller gains are iteratively evaluated over what was envisaged to be the entire search space. The size of the search (Equation 4.6) expanded from a minimum gain value of 0 towards a maximum gain of 100 for the proportional gain. This value was selected as this would result in severe controller saturation, signalling the boundaries of the search space. In hindsight, the proportional component should be evaluated over a reduced range to occupy a more representative solution.

$$k_p \in [0, 100] \quad k_i \in [0, 10] \quad k_d \in [0, 10] \quad (4.6)$$

Since the flight physics model is dynamically symmetric about the x - z and y - z axes, providing no disturbances are applied, the model is located in equilibrium flight about the lateral and directional axes. This characteristic allowed the pitch rate controller to be adjusted without inducing any lateral or directional moments. The fitness function used to describe the performance of the controller was a direct metric between the reference signal and the pitch rate dynamics of the aircraft. The topology was evaluated for the pitch rate controller between gain limits described in Equation 4.6, with k_p analysed at interval sizes of two units, and k_i and k_d at interval sizes of 0.25 units.

When the computational model is initialised, since the trim conditions have not been analytically solved for each flight level, there exists a small period of dynamic initialisation where the longitudinal dynamics of the vehicle can be erratic and unpredictable. Although the longitudinal dynamics are eventually damped by the aerodynamics of the vehicle, this required consideration. To prevent dynamic initialisation from influencing the performance of the aircraft and its controllers, the reference signal was applied as a step input administered two seconds into the simulation and with an amplitude of 20 deg/s. Since the applied fitness function was a direct measure between the reference signal and the pitch rate dynamics, the dynamic initialisation period will induce a large error which cannot be fully accounted for by the controller gains. To prevent this large error from becoming an overwhelming factor in the overall fitness,

the cost function applied was the weighted-sum-squared-error (WSSE) fitness function.

The WSSE fitness function calculates the sum-squared error between the reference signal and the time response, and scales discretised portions of the time-response. This creates a fitness function that emphasises a specific region of the time response, such as transient or steady-state error, and can even neglect specific regions or artefacts within the response. By using the WSSE fitness function as a metric for the brute-force analysis, the time response was partitioned into discrete sections with each section assigned an individual weighting based on varying levels of importance. This enabled the first two seconds of the simulation to be disregarded from the cost function of the topology study. For the entirety of the topology study, the aircraft was initialised at an altitude of 2 km and with an airspeed of 200 m/s. Since the aircraft model did not have an airspeed controller during this analysis to maintain airspeed during a pitch up manoeuvre, upon application of the step input to the controller, the throttle demand to the engine was set at 100% to prevent loss of airspeed.

Since the fitness function is a measure of the performance of the controller at specific gain values, the overall dimensionality of a PID controller and fitness function is four. To visualise the topology of the search space as a three-dimensional surface, three surface topologies were developed, as depicted in Figure 4.8. For each of the three surfaces, a single gain value was held constant, as described in the corresponding plots caption. This enabled the x and y axes to represent the remaining two controller gains over their entire operating range, with the z axis corresponding to the fitness.

The exhaustive search conducted was purely to visualise the topology of the search space and not to locate adequate controller gains or specific regions of interest, and was the reason for using large intervals to reduce the number of computations. Although this produced a coarse representation of the search space topology, it is evident that the topology is complex, consisting of a valley with high gradient slopes consisting of numerous regions of local minima. The non-smooth topology can also be attributed to the low frequency at which numerical integration of the equations of motion was performed. With a time-step of 0.01 seconds for each simulation, errors accumulated despite use of RK4 in-

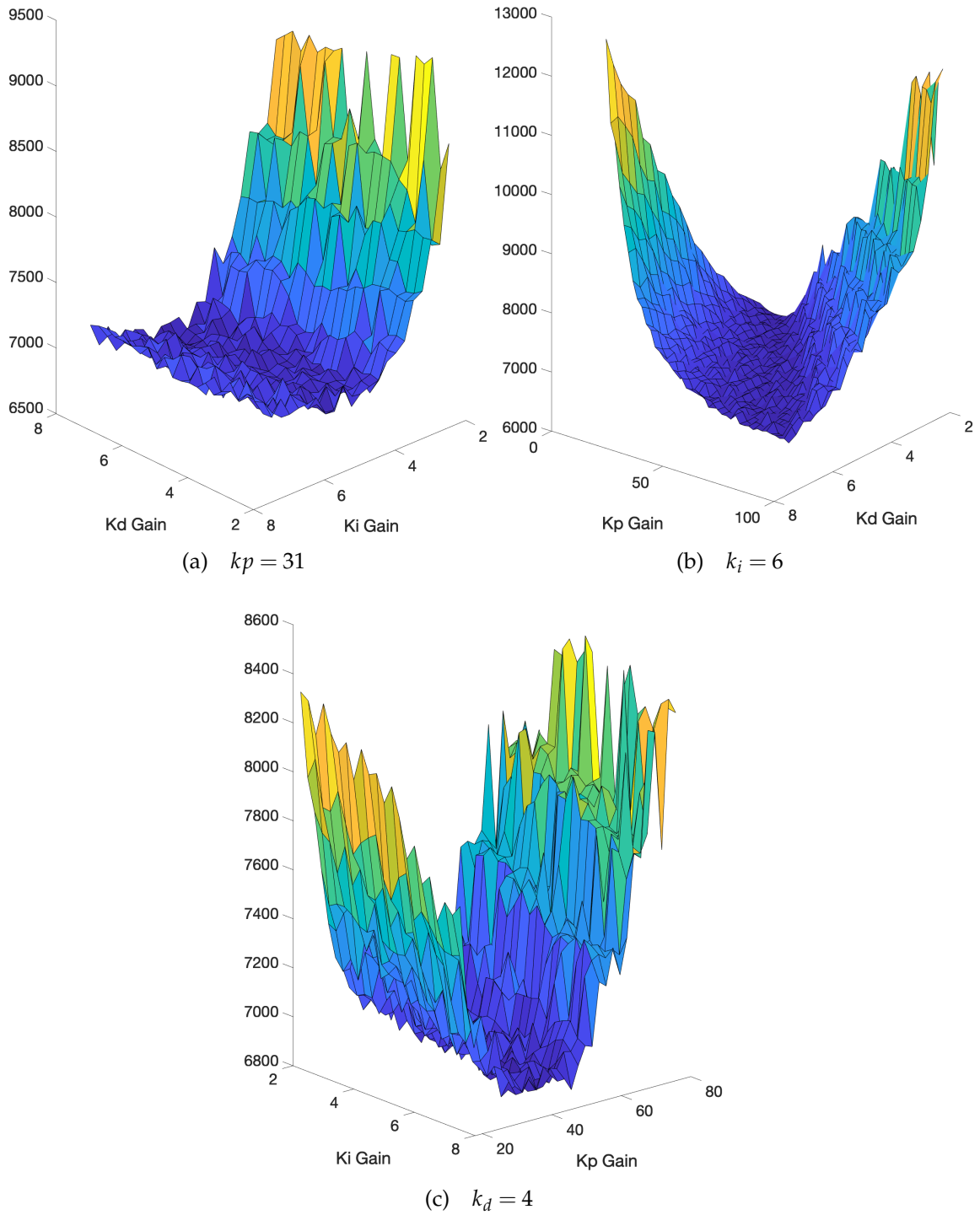


FIGURE 4.8 Three-dimensional topology profile for the search space of the angular rate controllers. Each subplot shows a singular controller value which has been held constant whilst the other two controllers indicate the fitness over the entire range

tegration, compromising accuracy and introducing noise into the cost function in order to establish a tractable solution. This validates the requirement of a meta-heuristic optimisation algorithm which focuses on solving the global optimisation problem whilst avoiding the entrapment of local minima, and thus, validates use of the BA.

4.4.2 High Precision Gain Surface

Since the F-16 is longitudinally unstable at subsonic airspeeds, prior to optimisation being performed for specific reference signal demands, optimisation was performed for all three angular rate controllers to establish a gain surface capable of maintaining a 0 deg/s angular velocity with a high level of accuracy and near-zero steady-state error.

To achieve these high-precision gain surfaces, the aircraft was initialised close to trim conditions and with a small angular velocity applied about a single axis. For example, when optimising the high-precision gain surface for the pitch rate controller, the model was initialised with an initial pitch rate of 1 deg/s. By inducing a small angular velocity, the model was disturbed allowing optimisation to determine an optimal set of gains that would return the dynamics to 0 deg/s, and with a near-zero steady-state error. Throughout optimisation of the high-precision gain surfaces, all three angular rate controllers had their fitnesses ranked using the sum-squared-error fitness function measured directly against the reference signal. Since this analysis optimised the controller gains for a disturbed aircraft, any discrepancies that occurred during the dynamic initialisation period, were directly measured as a position error.

Optimisation was performed over a range of airspeeds and altitudes, allowing the optimised controller gains to be scheduled based on these two dynamic states. Starting at the lower end of the flight envelope with an airspeed of 40 m/s and an altitude of 0 km, the BA was initialised with a population size of 30. The location of the bat corresponding to the proportional gain was initialised with a random value drawn from a uniform distribution between 0 to 100. Similarly, the integral and derivative gains were drawn from a uniform distribution but with a reduced range of 0 to 10. Echolocation was performed with a frequency range of 0 to 0.25 and a loudness decay coefficient of 0.90. Although each bat

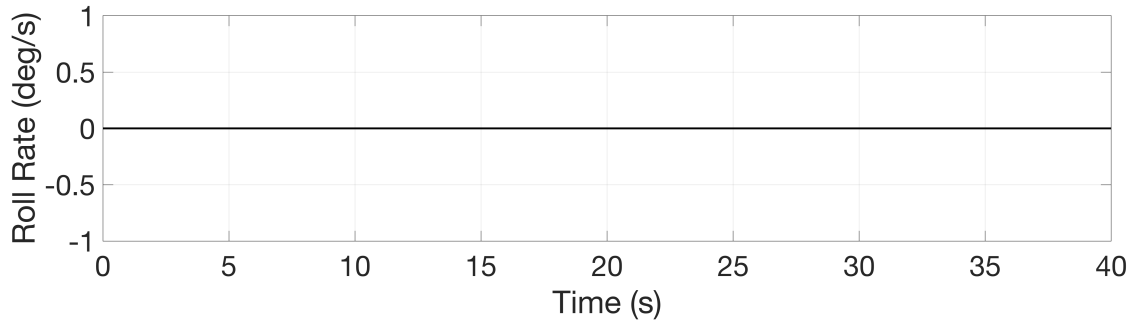
was given a total of 50 movements, optimisation would prematurely end if the standard deviation of the population fell beneath 0.001.

Upon converging onto a solution, the optimised gains were analysed at increasing airspeed values. When the time response resulted in an overshoot approaching 2-3%, or the steady-state error of the time response exceeded 0.05 deg/s, the airspeed value was noted, and this became the location of a neighbouring trim point. Optimisation of the newly identified trim point was not conducted on a randomly generated population, but rather a population produced based on the previous control law's optimised solution. Whilst the population size remained constant, utilising knowledge of the previous solution aided in the rate convergence towards the global solution. This process was repeated until the airspeed had exceeded 340 m/s, which was the limit for the available aerodynamic data.

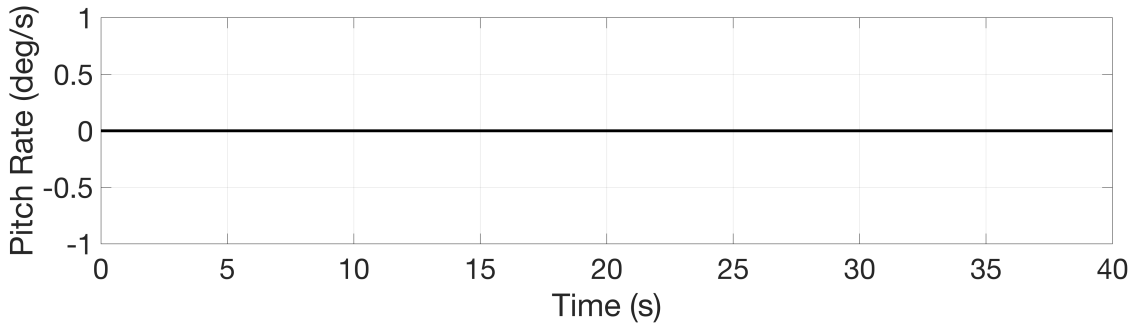
TABLE 4.6 Optimisation local control law boundaries for high-precision gain surface

Altitude (km)	Airspeed (m/s)								
	40	60	80	100	150	200	250	300	340
0	•	•	•	•	•	•	•	•	•
2	•								
6		•							
8			•						
10				•	•	•	•	•	•

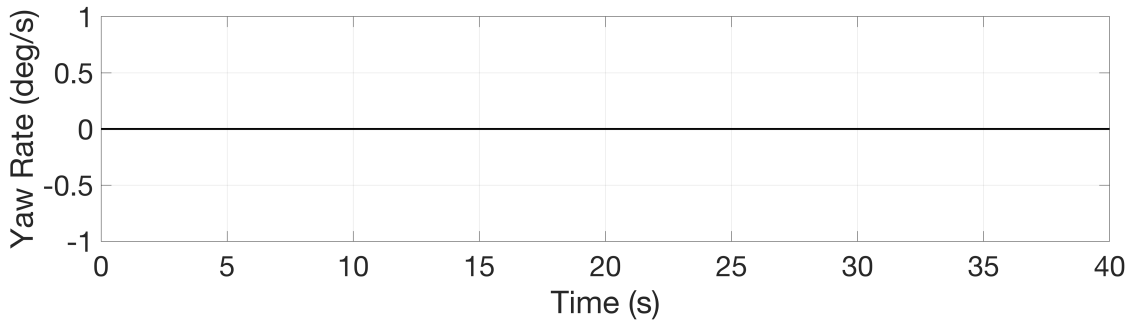
Upon completing optimisation for a single flight level over the entire operating range of airspeed values, the same analysis was conducted but this time for increasing levels of altitude. Throughout the optimisation process, it became apparent that the high-precision gain surface only required optimisation to be performed at two altitude levels for each identified airspeed trim point. The location of the trim point used during optimisation can be seen in Table 4.6, with the black dots representing a location where optimisation was performed. The same optimisation procedure was applied for optimisation of the roll rate and yaw rate controllers, with small angular velocity offsets applied during the dynamic initialisation period.



(a) Roll rate



(b) Pitch rate



(c) Yaw rate

FIGURE 4.9 Angular velocity analysis using zero-demand surface gains for roll, pitch and yaw rate controllers. Analysis conducted at an altitude of 4 km with an airspeed of 200 m/s

After successfully optimising all of the identified trim points for all three angular rate controllers, a simulation was performed at an altitude of 4 km and with an airspeed of 200 m/s. This simulation used a set of system states not located within the set of trim points. Therefore, linear interpolation was applied to estimate the internal controller gains at these intermediate state dynamics. The simulation was conducted for a total of 43 seconds, where the demanded roll rate, pitch rate and yaw rate was 0 deg/s. The resultant angular velocity time

response for the aircraft during the simulation can be seen in Figure 4.9, where all three angular velocities have a near-zero steady-state error. Additionally, the high-precision surface gains for all three controllers can be found in Appendix C.

4.4.3 Pitch Rate Optimisation

Having optimised a single gain surface for all three angular rate controllers enabling the computational model to maintain an angular velocity of 0 deg/s, optimisation was performed, creating the primary gain surface. The primary gain surface represents a gain surface corresponding to the internal controller gains at specific trim points during large rotational input demands. Optimisation of the primary gain surfaces was performed individually for all three controllers.

Throughout optimisation of the primary gain surfaces, the three controllers were assigned operating states as either active or passive. An active controller indicates that optimisation was currently being performed for that specific controller. The remaining two controllers remain engaged throughout the optimisation process, albeit in a passive state. This passive state simply implies that the reference signal demands an angular velocity of 0 deg/s using the previously optimised high-precision gain surface. The active controller has a reference signal where the amplitude represents the maximum angular velocity demand attainable given the current state dynamics of the aircraft. This maximum demand does not represent the maximum instantaneous angular velocity, but rather the maximum angular velocity that can be sustained for five seconds before being returned to 0 deg/s.

Use of the maximum sustainable angular velocity was selected over the maximum instantaneous angular velocity due to stability concerns surrounding the longitudinal axis. During a pitch up or pitch down manoeuvre the angle of attack is subjected to rapid changes, impacting the aerodynamics of the aircraft. Depending on the magnitude of the pitch rate demand, this can cause large changes to the longitudinal dynamics with aerodynamic stall being a primary concern. By limiting the magnitude of the reference signal to a sustainable demand, it can be assured that the dynamics of the aircraft do not result in an abrupt stall. Additionally, since the maximum sustainable velocity is only achieved

if the controller can return the angular velocity back to zero, it can be ensured that even if the aircraft does result in a loss of airspeed during the manoeuvre, the airspeed remained sufficient as to enable orientation recovery of the aircraft.

Primary Gain Surface

Since the aircraft is not dynamically symmetrical about the y -axis, the pitch rate dynamics differ based on whether the aircraft has a positive or negative pitching moment. This means that if the aircraft can sustain a certain pitch up demand, this does not necessarily indicate the same pitch down dynamics can be achieved, and vice versa. Furthermore, during optimisation the same set of controller gains applied for a positive pitch rate demand are not necessarily optimal when a negative pitch rate demand is applied. This meant two sets of controller gains were required for the pitch rate controller, one corresponding to positive pitch rate demand and one corresponding to negative pitch rate demands. Based on this information, the primary gain surface for the pitch rate controller was considered as two surfaces: a primary-positive and primary-negative surface. The resulted gain surface was then scheduled based on the state dynamics of the aircraft as well as the sign of the reference signal.

Prior to optimisation of the primary gain surfaces, the amplitude of the reference signal needed to be determined. It was previously discussed that the amplitude of the reference signal corresponded to the maximum angular velocity that was sustainable for five seconds before returning to 0 deg/s. Starting at the lower end of the operational envelope, the aircraft was initialised with an airspeed of 40 m/s at an altitude of 0 metres. Since no optimised controller gains currently existed allowing the maximum sustainable pitch rate to be determined, a set of manual gains were applied. Although these gains produce a sub-optimal time response, they were solely used to determine the maximum sustainable pitch rate given the state dynamics.

The methodology used to determine the maximum sustainable pitch rate was as follows.

1. Initialise the computational model with a demand of 0 deg/s with trim conditions achieved through the high precision gain surface

2. After 0.5 seconds a pitch rate demand was applied as a step input, lasting for five second before being removed.
3. Upon removal of the step input, the pitch rate dynamics were allowed to return to 0 deg/s before the simulation commenced.
4. If the demanded pitch rate was successfully achieved and sustained for five seconds before successfully returning to 0 deg/s, the applied pitch rate was deemed to be sustainable.

For each successful pitch rate demand, the process was repeated with a slightly larger demand until any of the conditions failed to be satisfied. Once the conditions failed to be satisfied, the maximum sustainable pitch rate was determined to be the last successful pitch rate.

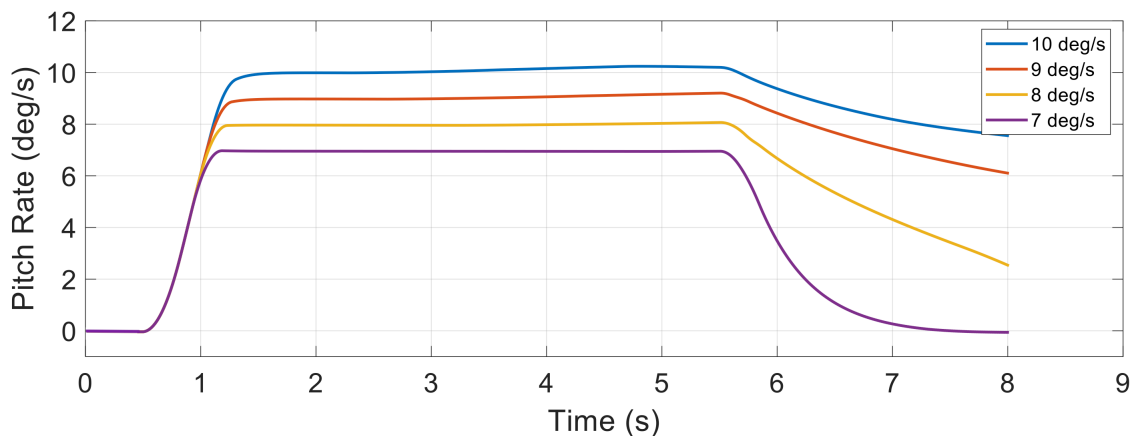


FIGURE 4.10 Sub-optimal manual pitch rate gain analysis to determine the maximum sustainable pitch rate for airspeed 40 m/s with an altitude of 0 metres

Figure 4.10 shows the process used to determine the maximum sustainable pitch rate demand for the first set of state dynamics (airspeed: 40 m/s, altitude: 0 metres). Although an instantaneous pitch rate of 10 deg/s is achievable, the pitch rate dynamics are unable to return to zero, indicating that the aerodynamic moment produced by the elevator is not sufficient to return the pitch rate to zero. This is likely due to a loss in airspeed suffered as a result of the manoeuvre. However, when a demand of 7 deg/s was applied, the dynamics of the aircraft were sufficient and enabled the pitch rate to return to 0 deg/s, meaning that this value was the maximum sustainable pitch rate.

Although the maximum sustainable pitch rate was determined based on the controllers ability to return the dynamics to zero, the primary gain surface was never intended to perform this entire manoeuvre alone. Given the aircraft model is representative of a high performance aircraft, the primary gain surface was only intended to engage during large input demands, with a more appropriate gain surface engaging during removal of input demand. For this reason optimisation of the primary gain surface was performed over a short 2.5 second simulation, with the step input used for optimisation of the pitch rate controllers primary surface shown in Figure 4.11. This reference signal allowed the primary gain surfaces to be optimised based on rise-time and steady-state error during large amplitude pitch rate reference signals.

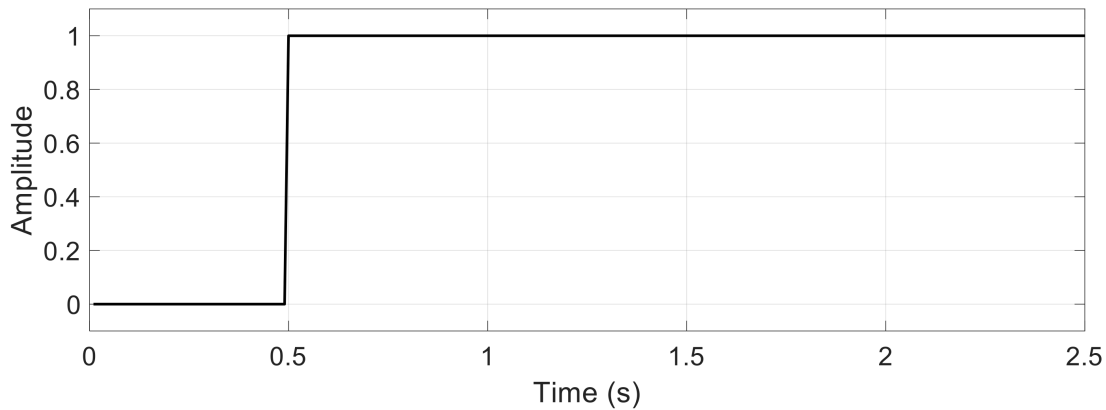


FIGURE 4.11 Primary positive gain surface reference signal used for pitch rate optimisation

Since the amplitude of the step input varies depending on the state dynamics, the amplitude of the step input in Figure 4.11 has been normalised between zero and one for clarity. The fitness of the time response was measured using the sum-squared-error fitness function, measured directly against the reference signal. Use of the sum-squared-error fitness function meant reducing the rise-time was the main objective of optimisation, giving the system a fast response.

Having determining the maximum sustainable pitch rate for the first set of state dynamics and outlined the optimisation manoeuvre, optimisation was performed for this trim point. Since no optimised gains had already been established for this primary surface, the bat population was initialised randomly throughout the search space, with each controller gain assigned the same spe-

cific range of values as used during optimisation of the high precision gain surface. Echolocation was applied with a frequency range of 0.5 and a loudness decay rate coefficient of 0.9. The pulse rate emission for each bat increased by an exponent factor γ of 0.5. Optimisation prematurely ended if the number of movements exceeded 50, or the standard deviation of the population dropped below 0.001.

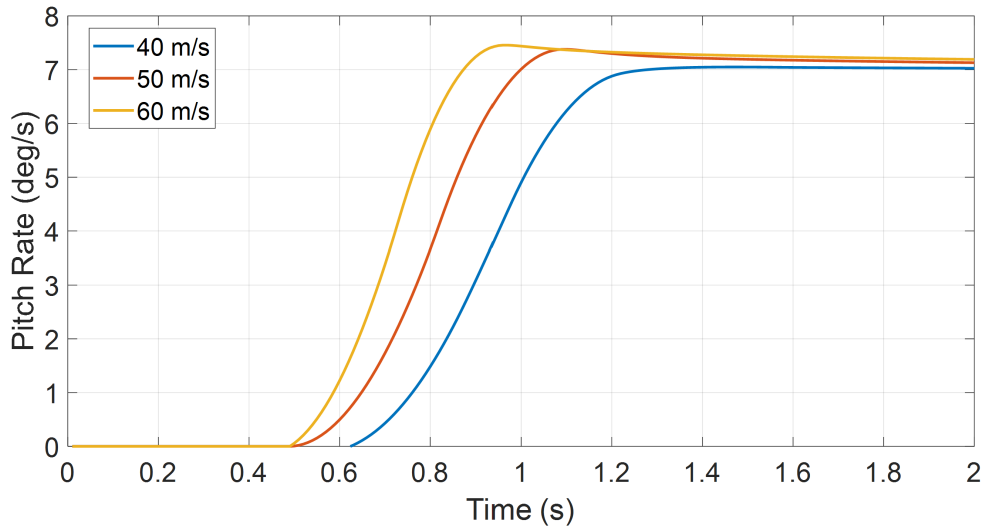


FIGURE 4.12 Optimised pitch rate response for an altitude of 0 metres with an airspeed of 40 m/s and a demand of 7 deg/s

Upon achieving the optimal solution for the first trim point, a similar analysis to that conducted in Section 4.4.2 was performed to determine neighbouring trim points. By performing the same manoeuvre but at incrementally larger airspeeds, the aerodynamic forces acting on the elevator produce a larger moment, increasing the rotational capabilities of the aircraft. This decreases the rise-time and eventually results in overshoot as the controller gains are no longer able to damp the increased dynamic performance. When the response resulted in an overshoot of over 2-3% or the steady-state error exceeded 0.2 deg/s, the airspeed value was noted and rounded to the nearest 10 m/s. The resultant airspeed was located as the neighbouring trim point with optimisation required to locate an optimal set of controller gains.

An example of this methodology can be seen in Figure 4.12, where the optimised controller gains for the first trim point at 40 m/s have been applied to the same 7 deg/s input demand conducted at 50 m/s and 60 m/s. It can be seen

that attempting to perform the same manoeuvre at increased airspeeds has resulted in an overshoot during the transient period for both airspeeds. However, the overshoot and steady-state error of the response for 50 m/s are within the defined dynamic tolerances, whereas at 60 m/s the overshoot and steady-state error both exceed the defined tolerances. This meant that the location of the next trim point was identified to be 60 m/s at an altitude of 0 metres.

Before optimisation could be performed for the next trim point, the amplitude of the reference signal needed to be identified. Since a set of optimised gains exist for the previous trim point, these gains were used in determining the maximum sustainable pitch rate for the next trim point, using the same methodology discussed in Figure 4.10. Upon determining the maximum sustainable pitch rate for this trim point, optimisation was performed using the same manoeuvre depicted in Figure 4.11. However, rather than initialising the bat population randomly throughout the search space, the previous optimised solution was used as an estimated guess, with the bat population generated randomly surrounding this location.

TABLE 4.7 Maximum sustainable pitch rates (deg/s) for varying flight conditions throughout the operating envelope

Altitude (km)	Airspeed (m/s)								
	40	60	80	100	150	200	250	300	340
0	7	12	12	17	25	30	30	30	30
2	4	8	12	15	21	26	30	30	30
4		5	8	12	18	22	26	28	30
6		2	5	8	13	18	22	25	28
8			4	7	11	15	17	20	22
10				4	4	12	15	18	19

This entire process was repeated until the range of airspeeds forming the operational envelope had been identified. The same analysis was then conducted, but for changes to the altitude. The maximum sustainable positive pitch rates forming the primary surface were determined iteratively throughout the optimisation process and can be seen in Table 4.7 for all of the identified trim points.

A full overview of the optimised controller gains for the primary positive gain surface can be seen in Appendix C.1 along with the optimised time responses for each trim point.

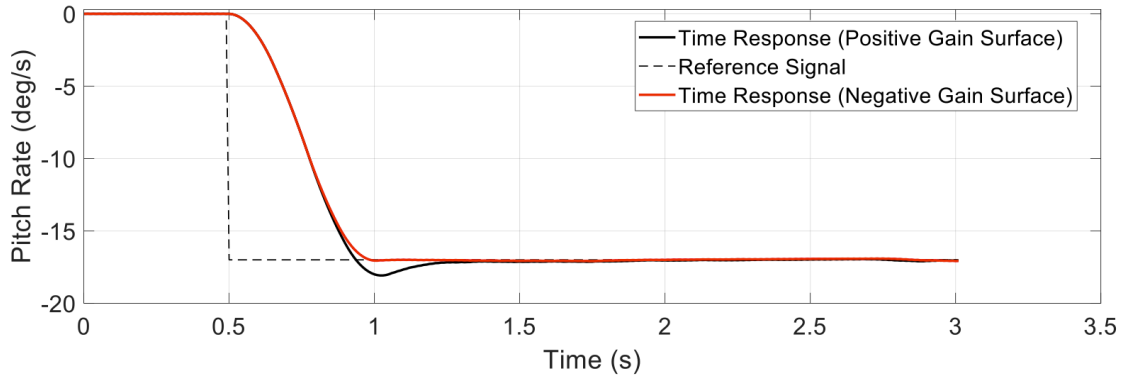


FIGURE 4.13 Time response comparison for negative pitch rate demand applied using positive pitch rate gains against optimised negative pitch rate gains

It was previously discussed that since the aircraft model does not have symmetry about the y -axis, different pitch rate dynamics exist for positive and negative demands. Upon optimising all of the identified trim points forming the primary-positive gain surface, an analysis was conducted to understand how these controller gains translated when applied to negative pitch rate demands. Figure 4.13 shows the optimised time response for a 17 deg/s input demand to the pitch rate controller corresponding to an airspeed of 100 m/s with an altitude of 0 metres. When using the same optimised controller gains but demanding a -17 deg/s pitch rate, the dynamics result in a small overshoot of approximately 5-6%. Whilst this is not necessarily an indication that the response will cause any stability issues for the aircraft, the time-response is sub-optimal. Therefore, optimisation was performed using the same procedure as the primary gain surface for positive rate demands, but for negative pitch rate demands.

Starting at the first trim point located at the lower end of the operational envelope, the bat population was randomly generated throughout the same operational space as the positive gain surface. Optimisation was then performed with the same algorithm settings and the same fitness function. The optimised solution was then used as an initial estimation when producing the population for the neighbouring trim points, and the entire process repeated until all trim points had been optimised. The same trim points identified during optimisa-

tion of the primary-positive gain surface was also used during optimisation of the primary-negative surface. A comparison between using the primary-positive gain surface and primary-negative gain surface can be seen in Figure 4.13, where the primary-negative gain surface has a reduced overshoot and reduced steady-state error.

Neutral Surface

Having optimised both primary surfaces for the pitch rate controller, optimisation of the neutral surface was conducted. The neutral surface is denoted as being the gain surface used to return the pitch rate dynamics to 0 deg/s, when a large position error is present. The neutral surface differs from the high precision gain surface, which is only intended for use when the position error is very small to maintain a near-zero steady-state error.

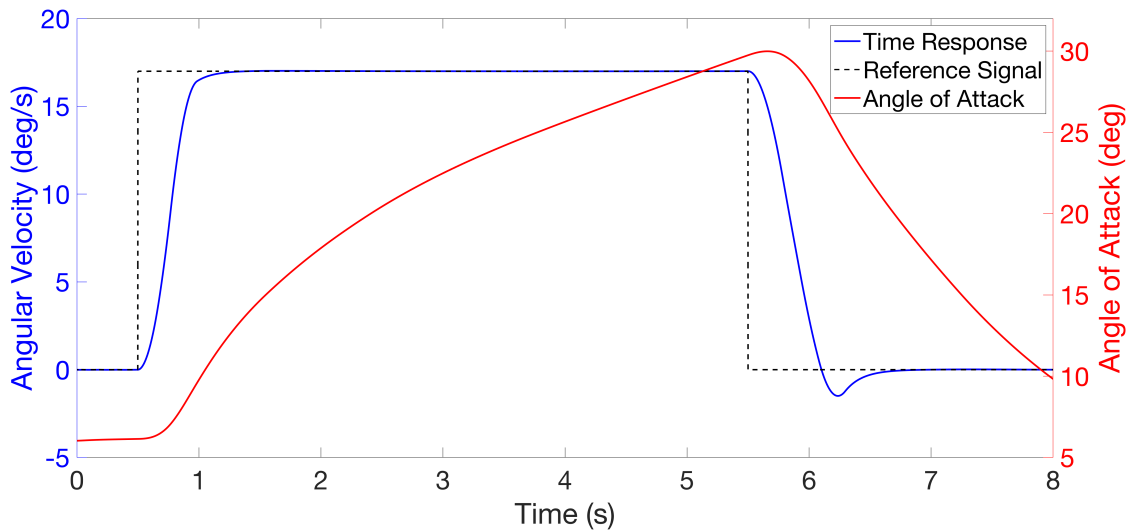


FIGURE 4.14 Primary positive gain surface resulting in overshoot upon removal of the input demand. Airspeed 100 m/s, Altitude: 0 metres

Consider the same flight conditions as seen in Figure 4.13. The controller gains were optimised for an input demand of 17 deg/s, at an airspeed of 100 m/s and an altitude of 0 metres. Shown in Figure 4.14 is the same manoeuvre conducted with the same flight conditions, however, the optimised controller gains corresponding to the primary gain surface have also been used to bring the angular velocity from 17 deg/s back to 0 deg/s. Since the controller gains were optimised for the application of a step input as opposed to the removal of

the step input, the controller gains produce a sub-optimal time response with a large overshoot. Although the dynamics do settle with quite a low steady-state error, this relatively large overshoot of almost three degrees could induce unwanted dynamic disturbances. This overshoot was eliminated by optimising an additional gain surface that engages during the removal of a step input.

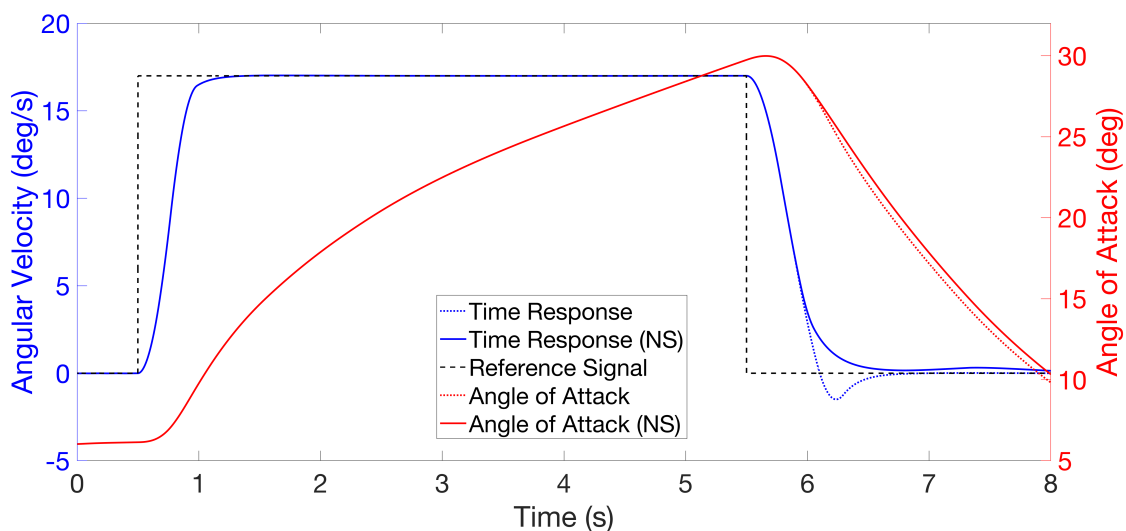


FIGURE 4.15 Optimised time response from Figure 4.14 with comparison to the implementation of an optimised and scheduled neutral surface

Optimisation of the neutral surface was conducted by altering the initial conditions of the aircraft during the initialisation period of the simulation. This process was similar to that performed during optimisation of the high-precision gain surface in Section 4.4.2, but involved a much larger angular velocity offset. Whilst the aircraft was still initialised close to trim conditions, the pitch rate was initialised with an angular velocity of 30 deg/s. This value was chosen since 30 deg/s was identified as the maximum sustainable angular velocity during optimisation of the primary gain surface. Throughout optimisation of the neutral surface, the reference signal applied to the controller demanded a pitch rate of 0 deg/s. The configuration of the bat algorithm and optimisation process remained the same when optimising the neutral gain surface as it was for the primary gain surfaces, with the only exception being the initial condition and amplitude of the reference signal. Optimisation was also performed for the same trim points identified during optimisation of the high-precision gain surface in Section 4.4.2.

By optimising a neutral surface for the pitch rate controller and scheduling the gain surface against the amplitude of the reference signal, the controller switches to a more suitable gain surface during low demand or zero demand reference signals. This can be seen in Figure 4.15 where the same manoeuvre has been performed to that seen in Figure 4.14. During the second phase of the manoeuvre where the reference signal demands a pitch rate of 0 deg/s, the pitch rate of the aircraft no longer results in an overshoot, as the gains switch to a more appropriate gain surface.

Hypersurface Gain Scheduling

During optimisation of the neutral surface, it was apparent that the process prioritised the transient portion of the time response. Since optimisation was performed using the sum-squared-error (SSE) fitness function measured between the pitch rate of the aircraft and the reference signal, the large position error at the start of the simulation drove the controller to reduce the error as fast as possible, with little regard towards the steady-state error. Although this improved the responsiveness of the controller to return the pitch rate dynamics towards 0 deg/s with little-to-no overshoot, the steady-state error of the system was hindered. In addition, provided the steady-state error was less than 1 deg/s, the SSE fitness function would diminish the implications due to its squared component. These factors negated the effect of the steady-state error, since its

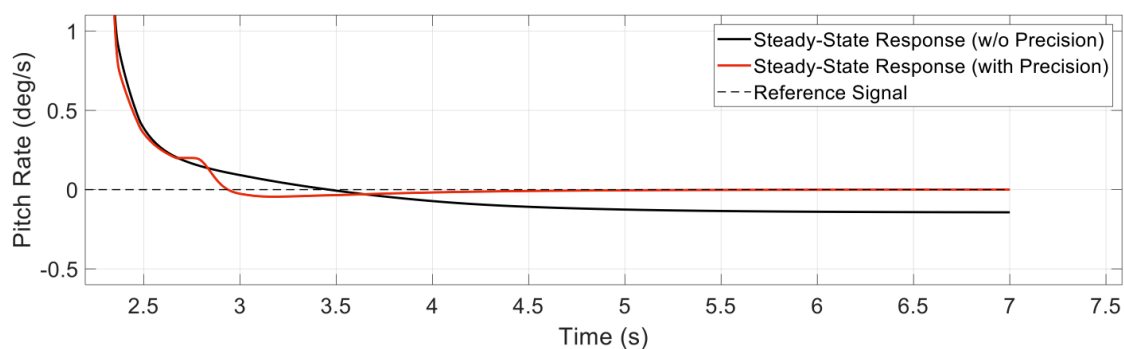


FIGURE 4.16 Steady-state response improvements using the high-precision gain surface. Simulation conducted with an airspeed of 200 m/s at an altitude of 0 metres. The input demand was a 30 deg/s pitch rate prior to being removed

weighting was reduced and overwhelmed by the large initial error. However, in Section 4.4.2 a high-precision gain surface was optimised with the sole aim of reducing the steady-state error to near zero during low position error scenarios.

By introducing the high-precision gain surface and scheduling the controller gains against the position error, this surface is used eliminate the steady-state error occurring due to the neutral surface. This has been demonstrated in Figure 4.16, where the controller is attempting to reduce the pitch rate to 0 deg/s. Although it can be seen that introducing the high-precision gain surface to engage when the position error is less than 0.1 deg/s does result in abrupt changes to the dynamics, the dynamics are quickly damped and the steady-state error significantly reduced towards 0 deg/s.

When operating a higher airspeeds (≥ 200 m/s), the time-step of the simulation becomes more important. Although numerical integration is performed using fourth-order Runge-Kutta, if the time-step is large (≥ 0.01 s) the veracity of the simulation is affected, reducing the performance of the controller, particularly during the steady-state phase of the time-response. This occurs because large changes are being calculated within the controller whilst adjustments are not being made with enough fidelity due to the size of the time-step reducing the frequency of computations. This causes the aircraft to oscillate around the demanded pitch rate as opposed to holding the demanded rate constant. An example of this can be seen in Figure 4.17 when operating with an airspeed of 300 m/s. The graph displays the steady-state portion of the time-response where the simulation was conducted at three different time-steps. When operating with a time-step of 0.01 seconds, the time-response is noisy and the dynamics indicate a lot of micro-adjustments are being made to the elevator as the error within the controller fluctuates positively and negatively. Although the oscillations are not growing over time, this behaviour is undesirable as performing unnecessary adjustments increases energy usage.

Although decreasing the size of the time-step from 0.01 seconds to 0.005 seconds does reduce the amplitude of the oscillations, with a time-step of 0.001 seconds alleviating this behaviour entirely (Figure 4.17), this is not a computationally efficient or viable solution, particularly during real-time simulations. During real-time simulations the size of the time-step is not controllable and

determined based on the clock speed of the CPU. Since the time-step is not a controlled variable during these simulations, the magnitude of the time-step could be large depending on the processing capabilities.

However, by allowing the gain surface to change from primary to high-precision based on the magnitude of the position error, the issue regarding micro-adjustments due to the size of the time-step is negated. This is demonstrated in Figure 4.17, where the steady-state error of the time-response whilst using the high precision gain surface is able to maintain near-constant dynamics despite the applied time-step being 0.01 seconds. Additionally, the magnitude of the steady-state error has also been reduced when compared to use of the primary-positive gain surface during this region of the response, despite using a time-step 10 times larger.

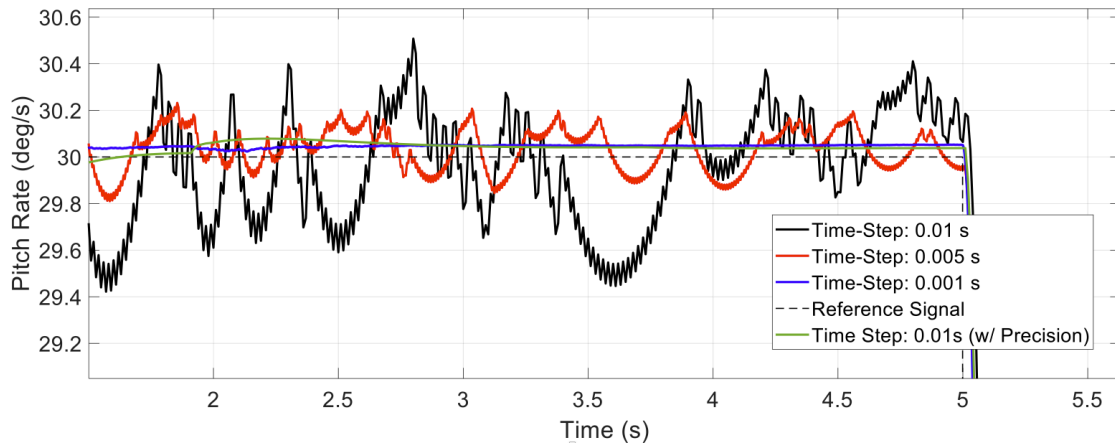


FIGURE 4.17 Steady-state response for input demand of 30 deg/s with an airspeed of 300 m/s at an altitude of 0 metres. Simulation has been conducted at three different time-steps to highlight time-sensitive issues within the controller

Currently, the primary, neutral and high-precision gain surfaces engage based on the amplitude of the reference signal as well as the magnitude of the position error. However, since both primary surfaces were optimised at various trim points using the maximum sustainable pitch rate demands, whenever either primary surface is engaged the controller gains correspond to the largest input value. This means that when an input demand of 10 deg/s is applied to the pitch rate controller, the corresponding controller effort is the same as a demand of 30 deg/s. During real-world applications, this is an inefficient use of energy and can be alleviated by normalising the optimised gain surface against

the input demand used during optimisation, as demonstrated by Hamilton et al. [19]. The controller gains are then linearly scaled based on the amplitude of the reference signal during a manoeuvre.

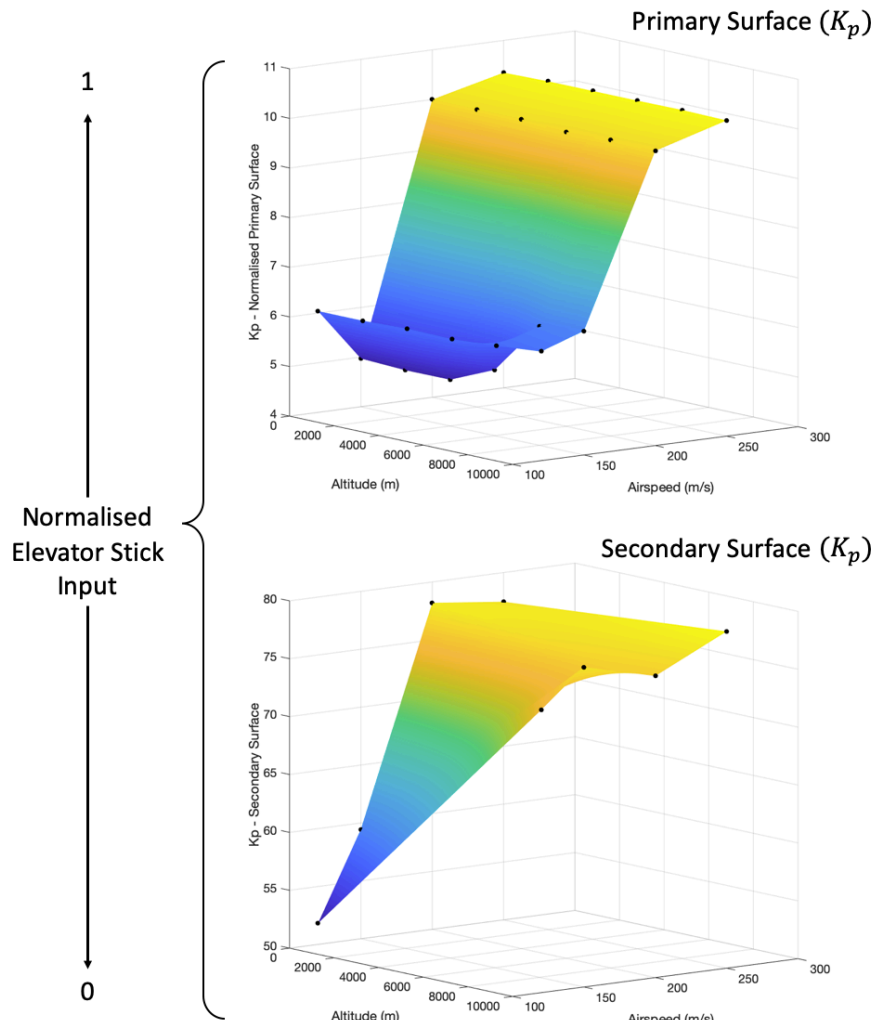
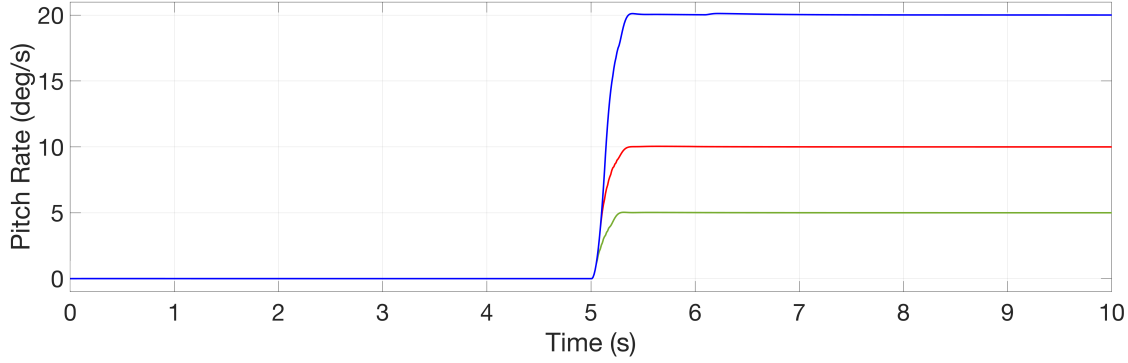


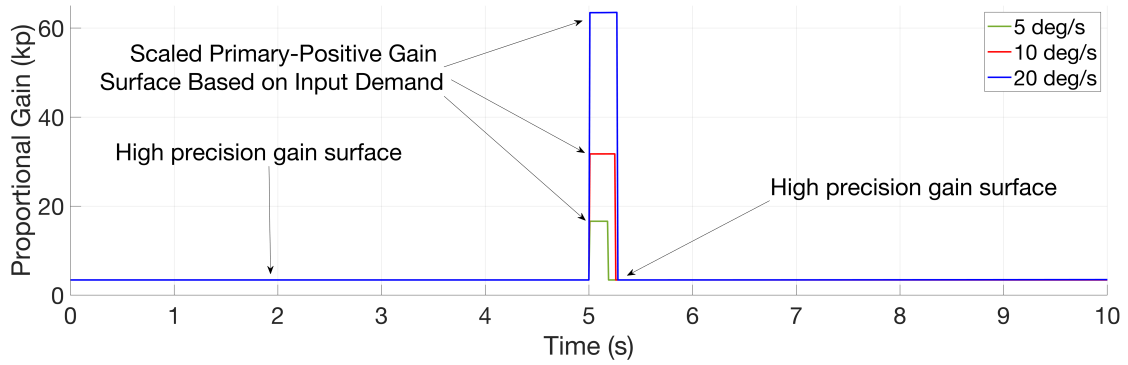
FIGURE 4.18 Normalised primary-positive surface engaging with the secondary surface (neutral surface) over a normalised elevator input demand [19]

Since, this process reduces the magnitude of the corresponding controller gains and scales them based on the input demand, this created performance issues during low amplitude inputs. As the input demand approached 0 deg/s, the magnitude of the controller gains were not sufficiently scaled, resulting in inadequate dynamic performance. This was alleviated by applying the neutral surface as a lower limit, preventing the primary surfaces from engaging below this threshold. A visual demonstrated of this can be seen in Figure 4.18, where

the primary gain surface and secondary gain surface are scheduled based on a normalised elevator input demand, where the elevator input corresponds to a pitch rate demand.



(a) Time response



(b) Proportional gain component

FIGURE 4.19 Demonstration of scaling the primary-positive surface based on the input demand

When considering the implications this hypersurface scheduling technique has on controller performance and the controller gains, it is evident benefits are experienced. Figure 4.19a shows three pitch rate time responses for input demands of 5 deg/s, 10 deg/s and 20 deg/s. These simulations were all conducted with an airspeed of 200 m/s at an altitude of 0 metres. It can be seen that during the first five seconds of the simulation an input demand of 0 deg/s is applied, with the high-precision gain surface engaged during this period. As soon as the input demand is applied to the controller, the primary-positive gain surface engages, where the normalised gain surface is scaled based on the magnitude of the input demand. This can be seen in Figure 4.19b, where the proportional gain for all three responses has been linearly scaled, with the gain

corresponding to a demand of 20 deg/s being scaled twice that as the demand for 10 deg/s and four times larger than a demand of 5 deg/s. However, as the pitch rate approaches the target demand, resulting in a low position error, the high-precision gain surface engages, maintaining a near-zero steady-state error, whilst also seeing large reductions to the proportional gain.

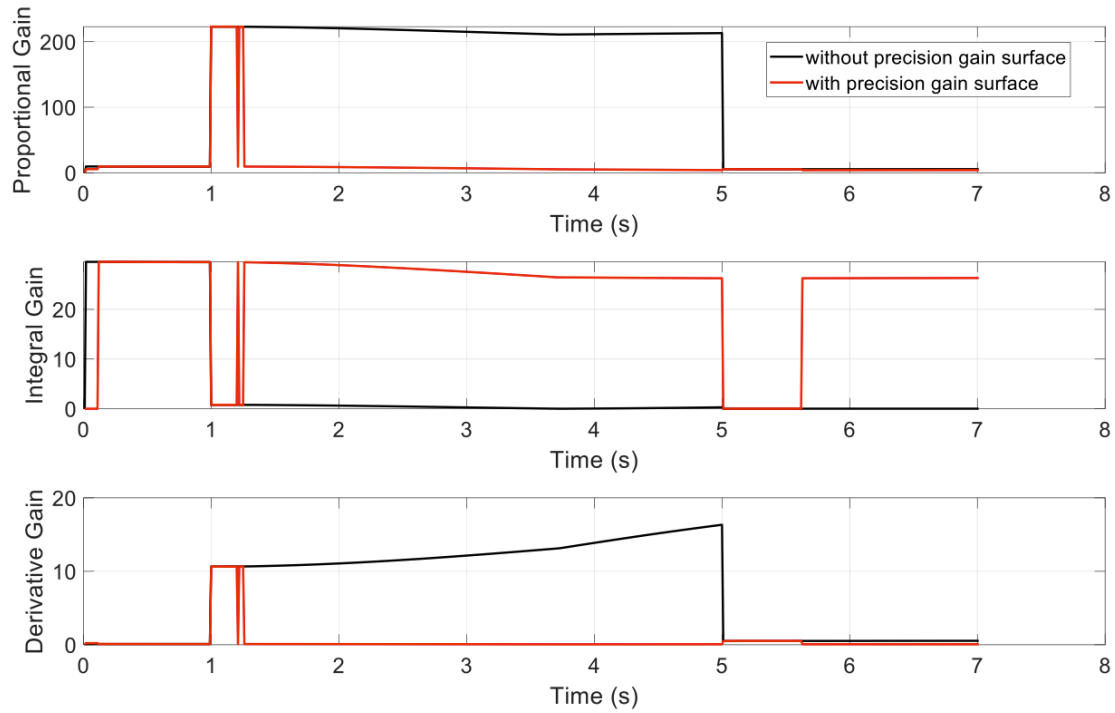


FIGURE 4.20 Controller gain comparison between the use of the precision gain surface applied during the manoeuvre conducted in Figure 4.15

Although it is evident that quite a substantial reduction has been made to the proportional gain, this usually occurs as a trade-off between the proportional gain and integral gain. Since the high-precision gain surface favours a reduction to the steady-state error, the integral component is generally larger than the other gain surfaces, whilst the proportional and derivative gains are lower. This has been demonstrated in Figure 4.20, where the proportional, integral and derivative components have been displayed for the manoeuvre conducted in Figure 4.15. For comparison, this manoeuvre was conducted with and without the high-precision gain surface.

Although the manoeuvre conducted with the high-precision gain surface does result in quite a large increase the integral component, this increase is

accompanied by a substantially larger decrease in the proportional component, and a small reduction to the derivative component, reducing the accumulative controller effort. In addition, this manoeuvre demonstrates how the primary, neutral and high-precision gain surfaces engage and disengage during a full manoeuvre, based on the magnitude of the reference signal and magnitude of the position error.

Algorithm 1: Algorithmic process for determining which controller gain surface is active for the pitch rate controller

```

 $\epsilon$  = current rate – demanded rate // Compute position error
// compute each gain individually
for controller gain  $\in [k_p, k_i, k_d]$  do
    if  $\text{abs}(\epsilon) > \mathcal{A}$  then
        //  $\epsilon$  is large, so use primary or neutral gain surface
        if  $\text{abs}(\text{demanded rate}) \geq \mathcal{T}$  then
            if  $\text{sign}(\text{demanded rate})$  is -1 and primary-negative > neutral then
                // gain uses primary-negative surface  $\times \text{abs}(\text{demanded rate})$ 
            else if  $\text{sign}(\text{demanded rate})$  is 1 and primary-positive > neutral then
                // use primary-positive surface gains  $\times \text{abs}(\text{demanded rate})$ 
            else
                // gain uses neutral surface
        hp = false // set high-precision void variable
    else
        //  $\epsilon$  is small, so use high-precision gain surface
        if not hp then
             $\epsilon_{\text{sum}} = 0$  // reset integral error
            hp = !hp // switch hp state
        // gain uses high-precision gain surface

```

Although the demonstrations above depicts the switching of gain surfaces between the primary-positive and high-precision gain surface, the entire switching procedure encompasses all four gain surfaces, with a surface selected based on the magnitude of the position error relative to an activation function, $\mathcal{A}$, as described by Algorithm 1. In addition, it can be seen in Algorithm 1, that two checks are performed that ensure an appropriate gain surface is selected for low magnitude inputs. These checks are performed by first comparing a thresholded

value ($\mathcal{T}$) against the demanded rate, before then comparing the corresponding gains obtained from the primary and neutral surfaces. This is a requirement to ensure that during normalisation of the primary gain surfaces, the scaled quantities are sufficient to ensure adequate dynamic performance during low input demanded rates. In addition, when the high-precision gain surface engages, it was previously mentioned that this surface elevates the integral component, reducing the steady-state error. This requires the accumulated integral error to be reset every time the surface engages. This is performed by applying a boolean variable (hp) which resets the error when true. This boolean triggers a reset every time the high-precision gain surface engages.

4.4.4 Roll Rate Controller

Optimisation of the roll rate controller was conducted using a similar process as the pitch rate controller in Section 4.4.3. Optimisation was split into three gain surfaces: high-precision, primary and neutral gain surfaces. Since the high precision gain surface for the roll rate controller was optimised in Section 4.4.2, the primary and neutral surfaces are discussed throughout Section 4.4.4.

Primary Gain Surface

Optimisation of the primary gain surface for the roll rate controller focused on achieving a quick and responsive control system, with little-to-no overshoot. Additionally, since the aircraft has symmetrical dynamics about the roll axis, it was assumed that the aircraft could sustain the same positive and negative roll demands and performance. This meant that optimisation of the primary surface only needed to be performed once, as opposed to the development of a primary-positive and primary-negative gain surfaces for the pitch rate controller. To optimise the primary gain surface for the roll rate controller, the same reference signal used during optimisation of the pitch rate controller in Figure 4.11 was used. However, before this could be performed, the amplitude of the reference signal needed to be determined.

For the pitch rate controller, it was important that the reference signal did not cause large and instantaneous changes to the longitudinal dynamics as this could cause the aircraft to prematurely stall and induce instability. Although the

roll rate controller does not exhibit abrupt and instantaneous changes to occur to the longitudinal dynamics, dynamic coupling is present between the roll and yaw axes. This means that as the aircraft performs a roll, a yawing moment is induced. This yawing moment either lifts the nose up or down due to sideslip. Since the yaw rate controller remained passively engaged during optimisation of the roll rate controller, as a yawing moment is induced, the yaw rate controller attempts to counteract and maintain a yaw rate of 0 deg/s, creating a dynamic altercation between the two controllers. This dynamic altercation can be seen in Figure 4.21, where the roll rate time response is displayed for a demand of -120 deg/s whilst travelling with an airspeed of 100 m/s at an altitude of 0 metres. As the dynamics of the vehicle approached a roll rate of -100 deg/s, the yaw rate controller attempted to counteract the induced yaw rate, resulting in a noticeable deceleration to the roll rate.

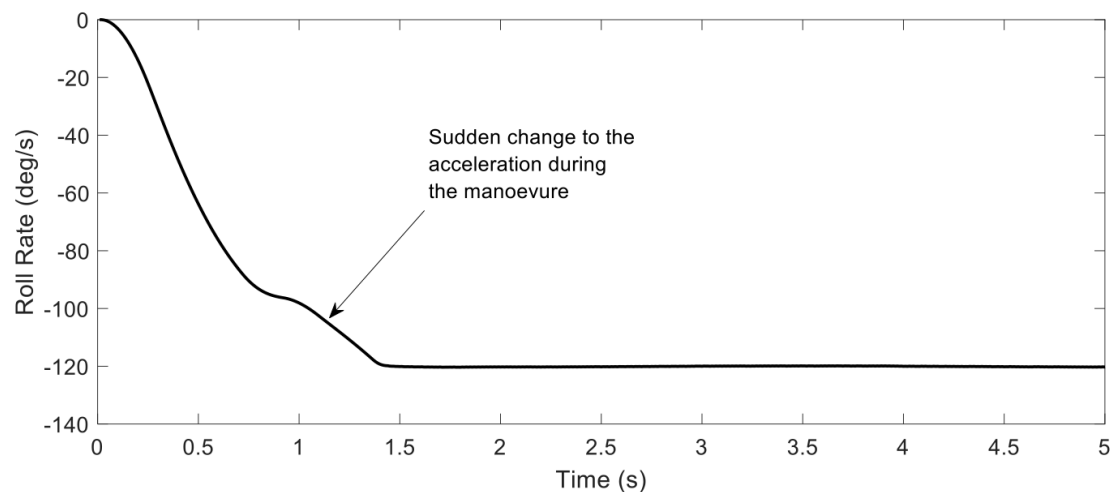


FIGURE 4.21 Roll rate time response for a demanded roll rate of -120 degrees-per-second. The non-linear roll/yaw coupling induces a deceleration to occur as the yaw rate controller attempts to damp the induced yaw rate.

Although the roll rate controller was able to overcome the yaw rate during the simulation shown in Figure 4.21 with very little dynamic implications, this was not always the case, particularly at lower airspeeds. When operating at lower airspeeds, the aircraft required a larger pitch angle to achieve equilibrium flight. This increase to the pitch angle elevated the susceptibility and severity of sideslip due to roll-yaw coupling. It was therefore determined that when oper-

ating at lower airspeeds, the characteristics of the aircraft should focus towards maintaining stability as opposed to maximising performance by achieving the highest possible roll rate.

TABLE 4.8 Roll rate primary surface maximum input demands in deg/s

Altitude (m)	Airspeed (m/s)								
	40	60	80	100	150	200	250	300	340
0	2	30	60	100	220	320	360	360	360
2,000	2	25	60	80	220	320	360	360	360
4,000		15	50	70	220	300	360	360	360
6,000		3	35	60	200	300	360	360	360
8,000			20	60	200	300	360	360	360
1,0000				40	180	280	360	360	360

This meant that for airspeeds of 100 m/s or below, the demanded roll rate used for optimisation would not be the maximum sustainable roll rate, but the maximum roll rate which would not induce a large coupling altercation between the roll and yaw rate controllers. For example, based on the time response in Figure 4.21, although the dynamics of the aircraft enable a roll rate of 120 deg/s, trying to achieve this induced a large coupling moment. Therefore, a more conservative roll rate of 100 deg/s was used during optimisation of the primary gain surface for this trim point. However, for trim points located at an airspeed greater than 100 m/s, the amplitude of the reference signal was determined as the maximum sustainable roll rate using the same methodology as the pitch rate controller. These roll rate demands were iteratively determined during the optimisation process, and can be found in Table 4.8.

Optimisation of the primary gain surface for the roll rate controller was performed using an identical procedure to that used during optimisation of the pitch rate controller in Section 4.4.3. The BA consisted of the same internal parameters with the fitness of the time response measured against the reference signal using a WSSE fitness function. Use of the WSSE fitness function enabled the fitness to neglect the dynamic initialisation period of the simulation, where the controller gains applied to the roll rate controller were determined using the

high-precision gain surface. In addition, the WSSE allowed the transient portion of the time-response to be appropriately scaled during high position error scenarios, preventing large errors from overwhelming the overall fitness.

Similarly to the pitch rate controller, since optimisation of the primary gain surface focused on achieving a demanded roll rate when a large demand was applied, the gains were sub-optimal during the removal of the input demand. This can be seen in Figure 4.22, where a roll rate of 200 deg/s was demanded at an airspeed of 150 m/s with an altitude of 0 metres. Upon removal of the step input, the optimised gains from the primary gain surface are unable to adequately control the aircraft, with a damped oscillatory overshoot of around 20-25%. During optimisation of the pitch rate controller, this sub-optimal transient response was remedied by introducing a neutral gain surface, optimised primarily to reduce the dynamics back to 0 deg/s upon removal of an input demand.

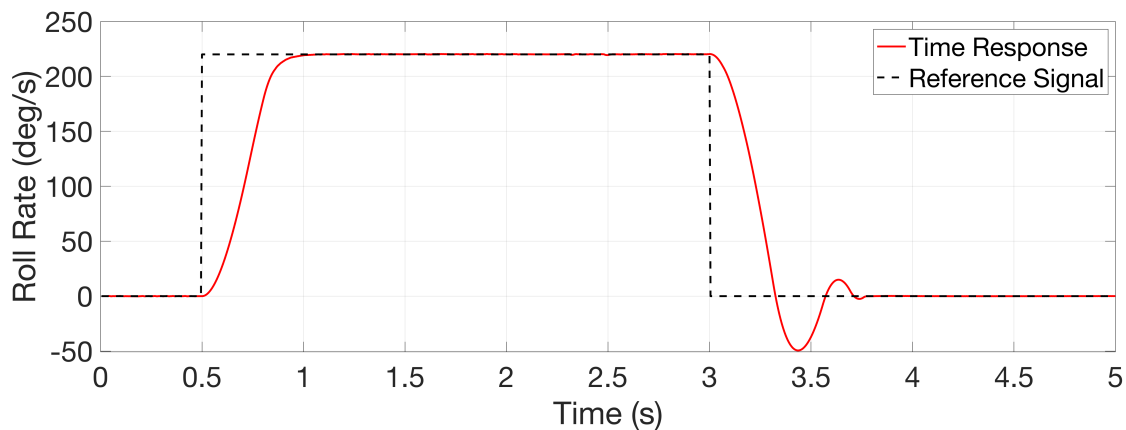


FIGURE 4.22 Time response for 220 degree-per-second demand for airspeed 150 m/s and altitude 0 metres. Simulation was conducted using only the primary optimised surface.

Neutral Surface

The neutral gain surface for the roll rate controller was optimised using an identical process to the pitch rate controller. The aircraft was initialised at trim conditions, albeit with a large roll rate. The magnitude of the roll rate corresponded to the roll rate demands used during optimisation of the primary gain

surface from Table 4.8. Optimisation was performed for the same trim points identified during optimisation of the high precision gain surface in Section 4.4.2.

When attempting to use the sum-squared-error (SSE) fitness function to measure the difference between the roll rate and the reference signal, numerous issues occurred. When compared to the pitch rate controller, the roll rate demands are substantially larger for all trim points. This meant that squaring the error between the roll rate and reference signal was extremely overwhelming. For example, consider the aircraft with an airspeed of 300 m/s and an altitude of 0 metres. The maximum roll rate demand from Table 4.8 indicates an optimised roll rate demand of 360 deg/s. When initialising the aircraft during optimisation of the neutral surface with this large roll rate, the error during the first time-step of the simulation using the sum-squared error fitness function is 129,600. This large initial error causes numerous problems during optimisation. Firstly, the fitness function attempted to reduce the error as quickly as possible creating aggressive dynamics. These aggressive dynamics allowed a large overshoot, provided the magnitude of the overshoot outweighed the corresponding fitness associated with a less aggressive and gradual reduction of the position error. Secondly, the large initial error maintained an overwhelming influence on the overall fitness, with the steady-state error of the system almost entirely neglected due to its low magnitude error. It was therefore determined that a weighted-sum-squared error (WSSE) fitness function was more appropriate metric, with the time response considered as three individual sections: (i) initialisation, (ii) overshoot, and (iii) steady-state error.

Each section of the time responses was analysed individually based on performance, with a reduced weighting applied to the initial transient period. The transient period was then investigated for overshoot, and if during the transient period an overshoot of more than 2% occurred, a harsh penalty factor was imposed onto the response. Finally, to try and reduce the steady-state error, the final 50 to 100 time-steps were analysed for their mean value. If the mean value exceeded 0.2 deg/s, a harsh penalty factor was imposed onto the fitness.

Use of the WSSE fitness function allowed optimisation of the neutral surface to create a responsive and fast system with little-to-no overshoot during the removal of a roll rate input demand. Figure 4.23 shows a comparison between

the manoeuvre conducted in Figure 4.22 using just the primary gain surface, then re-performed using the both the primary and neutral gain surfaces, where each surface was scheduled based on the magnitude of the reference signal.

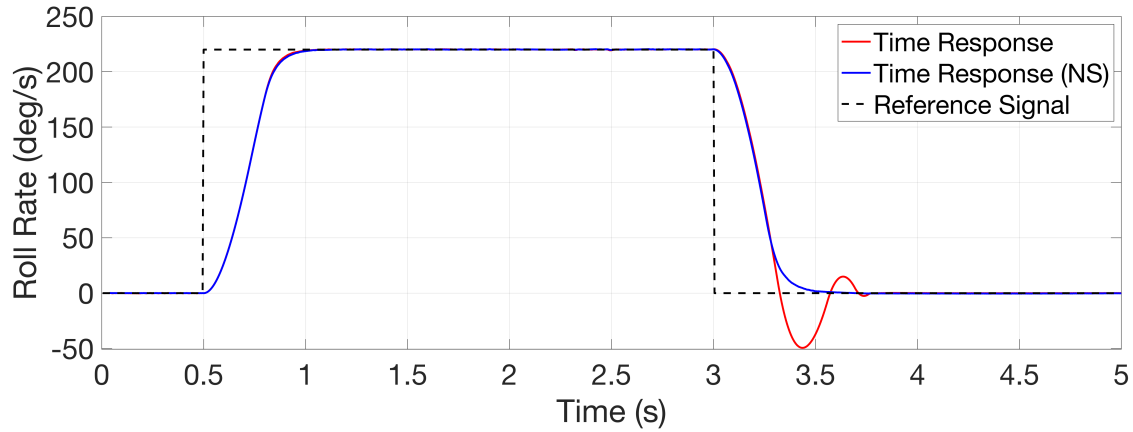


FIGURE 4.23 Comparison between the roll rate time response seen in Figure 4.22 against the time response using the neutral surface.

Hypersurface Gain Scheduling

The full implementation of the roll rate controller also utilises the hypersurface gain scheduling methodology implemented onto the pitch rate controller. This methodology sees the primary gain surface controller gains normalised against roll rate demands used during optimisation in Table 4.8. The primary gain surface and neutral gain surface are scheduled based on the magnitude of the reference signal, with the neutral gain surface acting as a lower limit threshold for low-to-medium amplitude input demands.

In addition, similarly to the pitch rate controller, the neutral surface was unable to adequately account for the steady-state error, despite use of the WSSE fitness functions. However, reducing the steady-state error towards zero was not a necessity of the neutral surface since the high performance gain surface can be used to reduce the steady-state error. However, the neutral surface needed to reduce the position error such that the high precision gain surface was able to engage to successfully reduce the position error to 0 deg/s. By introducing the high precision gain surface during low position error situations, the steady-state error of the time response was reduced towards 0 deg/s, as demonstrated in Figure 4.24.

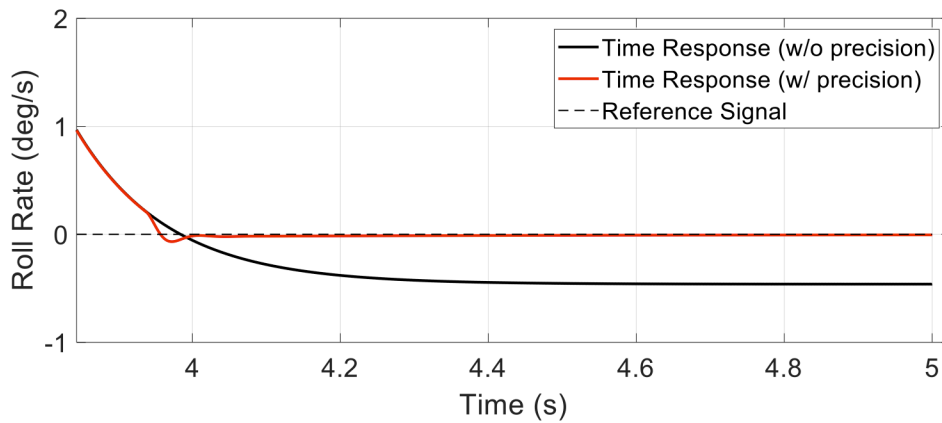


FIGURE 4.24 Steady-State error time comparison between roll rate controller using primary/secondary gain surfaces and a roll rate controller using primary/secondary with precision gain surfaces

4.4.5 Yaw Rate

During optimisation of the roll rate controller in Section 4.4.4, it was explained how due to dynamic coupling, an altercation between the roll rate and yaw rate controller occurs. Although the roll rate controller was able to overcome the moment induced by the yaw rate controller since the moment of inertia about the x -axis is much smaller than the z -axis, the effects were amplified at lower airspeeds. For this reason, it was determined that for optimisation of the roll rate controller, a reduced roll rate demand was used during these low-air-speed scenarios.

When trying to optimise the yaw rate controller, due to a high moment of inertia the rotational motion was overwhelmed by the roll and pitch rate controllers. Figure 4.25 shows a simulation where after two seconds a demand of -10 deg/s was applied to the yaw rate controller. Since the roll and pitch rate controllers remained passively engaged demanding a rate of 0 deg/s, the roll rate controller immediately applied an opposing roll rate to counteract roll induced through yaw. As the yaw rate began to approach the demanded -10 deg/s, the roll rate controller becomes more aggressive with its demands, overwhelming the yaw rate controller. The power of the roll rate controller ensures that the yaw rate is never able to achieve a large enough demand that induces a noticeable roll. The yaw rate controller eventually settles into a damped oscillatory

response around the -3 deg/s.

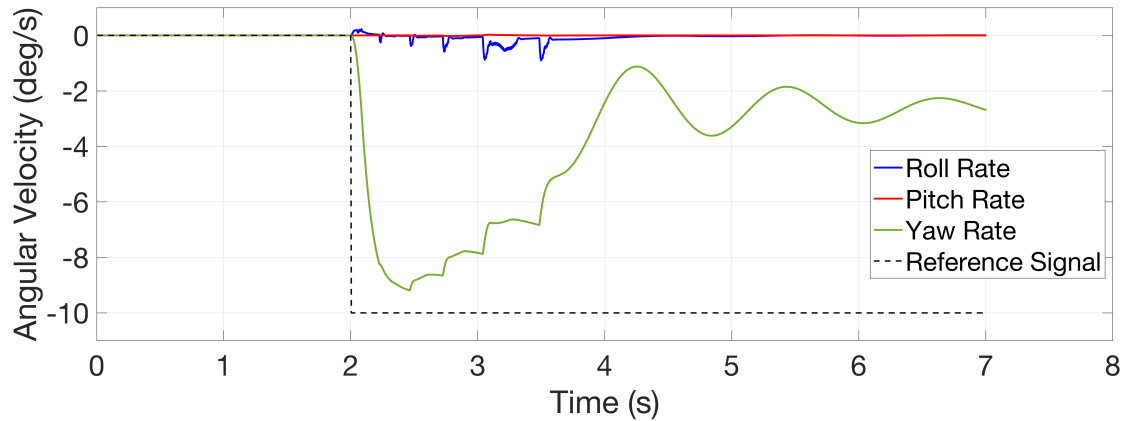


FIGURE 4.25 Angular velocity controllers producing counteracting aerodynamic moments due to passive engagement

This made optimisation very difficult for the yaw rate controller since any input demand was negated by the passively engaged roll and pitch rate controllers. The pitch rate controller is essential for maintaining longitudinally stable flight, so disengaging it was not an option since the aircraft would quickly become unstable and fall out of the sky.

Since the aircraft model is representative of a fighter-type aircraft, it can be assumed that general flight is performed using a bank-to-turn manoeuvre. During this type of manoeuvre, the rudder is used primarily for reducing sideslip through minor corrective inputs. Additional uses for the rudder would include minor azimuth corrections during wings-level flight and landing. For these reasons, developing a yaw rate controller capable of sustaining high yaw rates was not only not feasible, but not required. It was therefore determined that the yaw rate controller would act as a corrective controller, where small magnitude inputs are applied when necessary to prevent sideslip.

For these reasons optimisation of the yaw rate controller was performed slightly differently to that of the roll and pitch rate controllers. A high precision gain surface already exists, which accounts for small deviations occurring to the yaw rate controller, with an emphasis on maintaining a near-zero steady-state error. This surface was accompanied by a primary gain surface which would engage solely during larger position error scenarios.

Optimisation of the primary gain surface was performed by determining the

TABLE 4.9 Maximum instantaneous yaw rate demands (deg/s) used during optimisation of the primary surface for the yaw rate controller

Altitude (m)	Airspeed (m/s)								
	40	60	80	100	150	200	250	300	340
0	5	10	10	10	10	11	12	12	12
2000	3	10	10	10	10	11	12	12	12
4000		10	10	10	10	11	12	12	12
6000		10	10	10	10	11	12	12	12
8000			10	10	10	10	12	12	12
10000				10	10	10	12	12	12

maximum instantaneous yaw rate that could be achieved before aggravating the roll and pitch rate controllers. The maximum instantaneous yaw rate was selected for optimisation since the demand would not be sustained for a prolonged period, and merely acting as instant corrections. The maximum instantaneous yaw rate was determined for the same optimisation trim points used during optimisation of the primary gain surfaces for the pitch and yaw rate controllers, with the demands shown in Table 4.9. These values were determined iteratively throughout the optimisation process.

Since the yaw rate controller acts as an impulse response, the reference signal applied lasted for one second before being removed. Unlike the roll and pitch rate controllers, the primary gain surface was responsible for the application and removal of the step input. Therefore, the fitness function used was the sum-squared-error measured between the yaw rate and reference signal. Optimisation was performed over a four second simulation, with the input demand applied after one second. The roll and pitch rate controllers remained passively active throughout the optimisation process, with a demand of 0 deg/s.

Since the yaw rate controller was only optimised to engage for short pulse inputs, simulating a minor correctional manoeuvre, the controller effort was not heavily considered. This meant that the primary gain surface was not normalised and scaled based on the input demand, as performed for the roll and pitch rate controllers. In addition, the performance of the yaw rate controller primary

surface was not as heavily scrutinised as the roll and pitch rate controllers, since this surface was never optimised for maintaining stability. Maintaining stability about the directional axis was performed by ensuring the lateral and longitudinal axes remained stable. This meant that the primary gain surface was only used for short pulse inputs, with the high precision gain surface used to maintain the steady-state error.

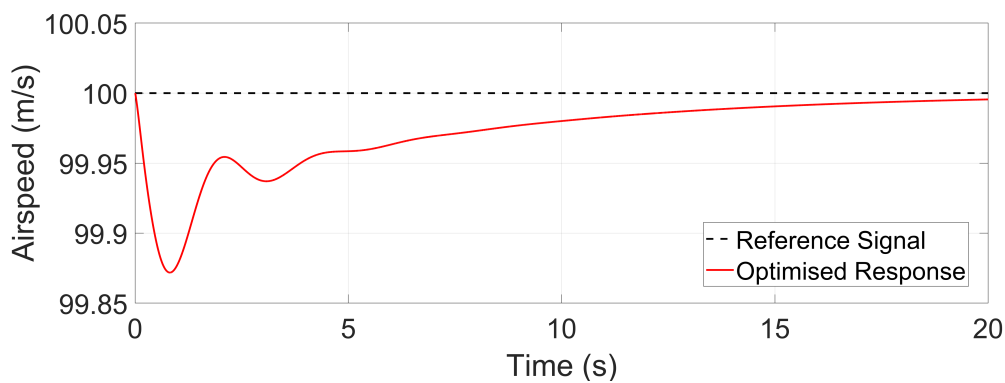
4.5 Airspeed Controller

The airspeed controller controls the airspeed of the aircraft using three modes: (i) increase the airspeed, (ii) decrease the airspeed, and (iii) hold the airspeed constant. The controller outputs a reference signal interpreted by the engine model as a stick input command. Similarly to the three angular rate controllers, the airspeed controller is comprised of PID architecture, with optimisation performed using the bat algorithm to tune the controller gains.

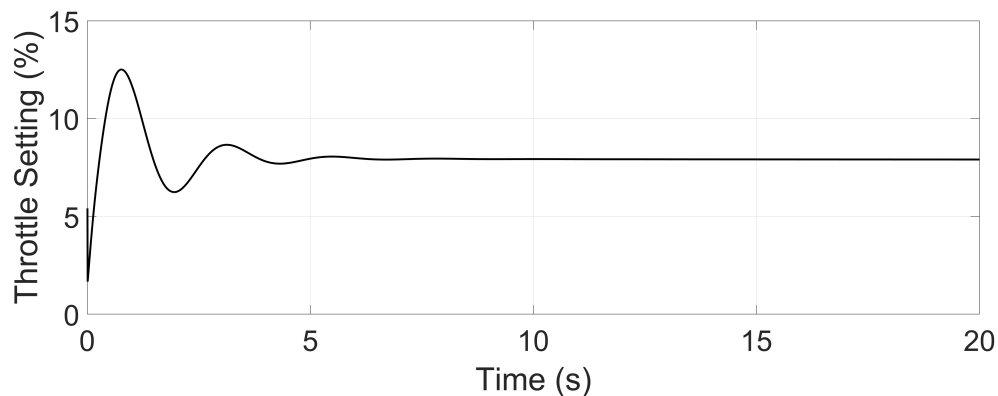
During optimisation of the pitch and roll rate controllers, the reference signal applied to the controller had an amplitude corresponding to the maximum sustainable rate demand, with optimisation performed at various airspeed and altitude values, known as trim points. Since optimisation of the airspeed controller results in a change to the airspeed, it was not possible to schedule the controller gains against this dynamic state. In addition, since the engine model has a variable first-order time lag up to one second, any output from the airspeed controller results in a lagged response from the engine model. This gave the airspeed controller slow dynamic performance when compared to the three pitch controllers optimised in Section 4.4, and was something that needed to be considered during the optimisation process.

Similarly to the optimisation process for the the roll rate controllers, the airspeed controller was first optimised for a gain surface that would maintain the dynamics at a constant value. Starting with an altitude of 0 metres, an initial airspeed of 100 m/s was selected, since this was the lowest airspeed value that could be achieved throughout the entire operating altitude range. The fitness was measured as the difference between the actual airspeed and reference signal using the sum-squared-error (SSE) metric.

The optimised time response with the corresponding stick input value can be seen Figure 4.26. Despite the slowly-varying engine dynamics, the optimised controller gains have been able to maintain the desired airspeed with a very low steady-state error (Figure 4.26a). When analysing the throttle response (Figure 4.26b), the output from the airspeed controller is maintaining a near-constant throttle setting throughout the duration of the simulation. This demonstrates a desired response as the controller is exhibiting very few low amplitude changing micro-adjustments over the duration of the simulation.



(a) Time response

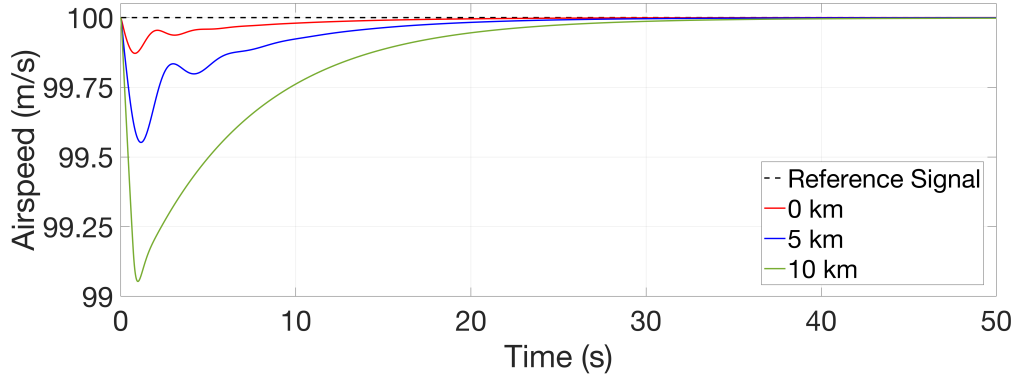


(b) Throttle response

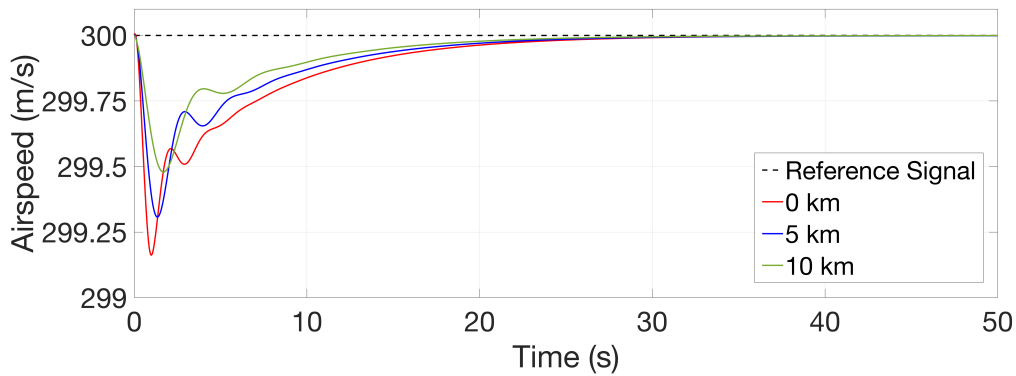
FIGURE 4.26 Optimised responses for maintaining an airspeed of 100 m/s with optimisation performed at 0 metres altitude.

When analysing these optimised controller gains at higher altitudes for the same airspeed demand (Figure 4.27a), it is evident that dynamic performance has been maintained, despite changes to the altitude. Although it appears that at 10 km there is a large drop in the airspeed, this is a function of the trim conditions and dynamic initialisation of the simulation as opposed to the con-

troller gains. Given this drop in airspeed was less than 1 m/s and the controller gains were able to reduce the error to a near-zero steady-state error, this was not considered a problem, with the optimised controller gains yielding optimal performance across all the entire range of altitude values.



(a) Airspeed value of 100 m/s maintained over various altitudes



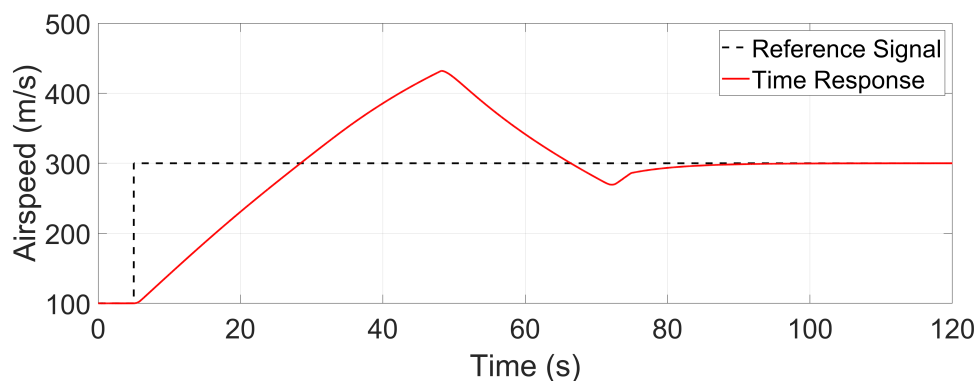
(b) Airspeed value of 300 m/s maintained over various altitudes

FIGURE 4.27 Optimised 100 m/s airspeed gains used various airspeeds and altitude levels.

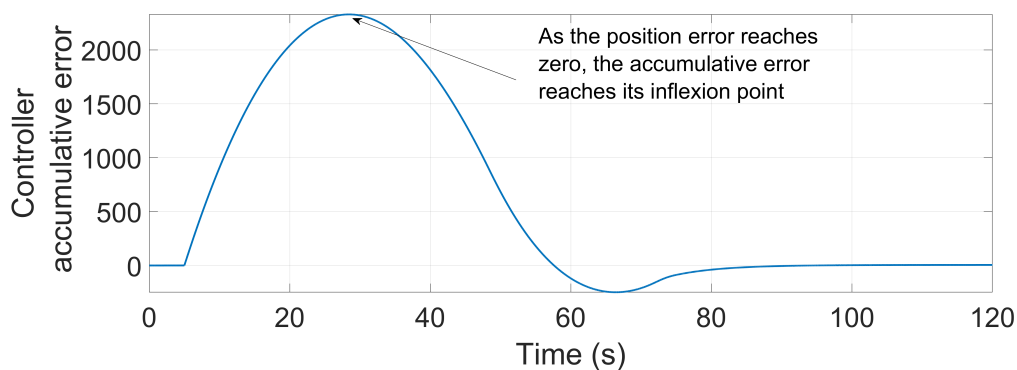
This same characteristic was also seen when applying the same optimised controller gains not only at higher altitudes, but also at higher airspeed values (Figure 4.27b). The airspeed controller is sufficiently able to control and maintain the airspeed of the aircraft despite operating at an airspeed demand for which optimisation was not performed and at various altitude levels throughout the flight envelope for which optimisation was also not performed. This analysis concludes that due to the slowly varying dynamics of the airspeed controller, dynamic performance deviations due to altitude and airspeed do not negatively impact the performance of the controller whilst maintaining a fixed airspeed

value.

However, when applying the optimised gains for a simulation requiring a change to the airspeed to be conducted, the slowly varying dynamics of the controller starts to become a problem. As the input demand changes, a large position error drives the output of the controller to increase the throttle demand and accelerate the aircraft. Due to the slowly varying dynamics associated with airspeed, this transition occurs too slowly, resulting in a large controller error accumulating over time. As the aircraft approaches the newly demanded airspeed value and the position error approaches zero, the proportional component of the controller decreases, which should reduce the output of the controller. However, this does not happen, since the accumulated error within the controller amplifies the integral component, saturating the output of the controller. The affects of controller saturation due to large accumulated error can be seen in Figure 4.28a.



(a) Airspeed overshoot due to controller error saturation



(b) Accumulated controller error over time

FIGURE 4.28 Airspeed controller performance during airspeed changing manoeuvre

The time response shown in Figure 4.28a corresponds to a simulation performed where the aircraft was initialised at 100 m/s with a step response demanding 300 m/s applied five seconds into the simulation. Approximately 30 seconds into the simulation, the airspeed of the aircraft approaches the demanded 300 m/s and the position error within the controller approaches zero. At this instance in time, a large error has accumulated in the controller (Figure 4.28b), amplifying the integral component and forcing the aircraft to continue accelerating. As the aircraft accelerates past the demanded airspeed the position error changes sign, resulting in a decrease to the controllers accumulated error, as represented by the inflexion on Figure 4.28b. From this instance onwards, as the accumulated error begins to decrease, it remains large enough to continue saturating the controller output. As the aircraft continues to accelerate past 400 m/s, the position error and proportional components of the controller start to overcome the accumulated error resulting in a reduced throttle output from the airspeed controller, and the aircraft begins to decelerate.

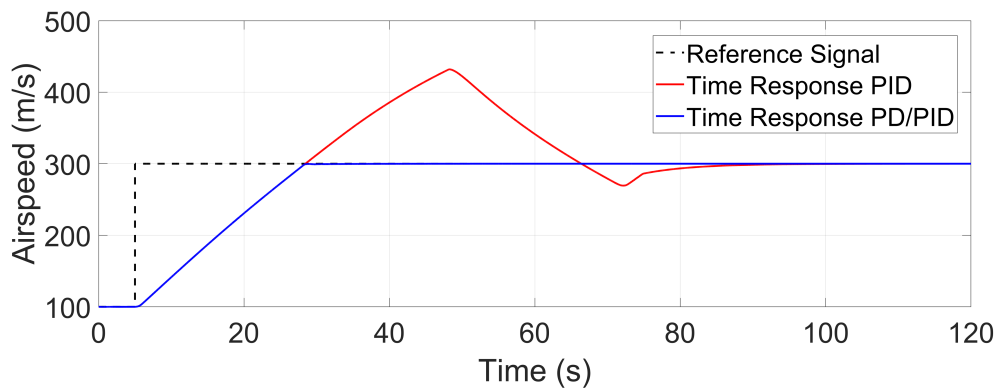


FIGURE 4.29 Time response for PID controller when compared to PD/PID controller

This unwanted performance occurs due to the single set of optimised controller gains being relatively large. Since optimisation was performed for an airspeed-holding procedure where the position error was relatively small, the controller gains needed to be large to allow for a sufficient output. On the other hand, during these airspeed changing manoeuvres, the position error and accumulated errors are very large, meaning that the large controller gains are only amplified further, saturating the controller. Rather than re-optimize the controller gains for these manoeuvres and schedule the gains based on the position

error, the controller components themselves were scheduled. By scheduling the integral component based on the magnitude of the position error, this prevented large errors from accumulating within the controller, thus reducing integral saturation from driving the output.

This transformed the PID controller to a PD/PID controller, as the controller functioned as a PD controller during large position errors, and then engaged the integral component to reduce the steady-state error during small position errors. The advantages of this newly scheduled controller architecture are demonstrated in Figure 4.29, where the airspeed controller is able to accelerate the aircraft from 100 m/s to 300 m/s with little-to-no overshoot, and then maintain 300 m/s for 90 seconds with a near null steady-state error. For clarity, the previous time-response from Figure 4.28a using only PID control has displayed for comparison purposes.

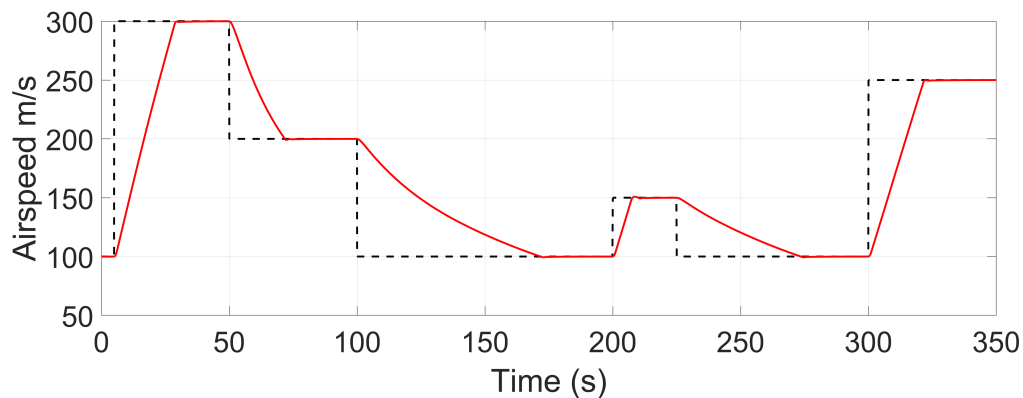


FIGURE 4.30 Demonstration of PD/PID airspeed controller during accelerations, decelerations and hold

This controller component scheduling technique also works for decreases to airspeed. However, these manoeuvres require longer to perform since the aircraft decreases its airspeed by reducing the throttle towards zeros and allowing drag to slowly decelerate the aircraft. A demonstration can be seen in Figure 4.30 for the airspeed controller increasing and decreasing the airspeed throughout the operational airspeed ranges. This simulation demonstrates how scheduling the internal states of the controller enabled a single set of controller gains to be applied for various different types of airspeed changing manoeuvres, without having to perform re-optimisation of the controller gains and gain scheduling based on dynamic state feedback.

4.6 Summary

Throughout Chapter 4, rotational and translational motion of the flight physics model was controlled by implementing a series of angular rate controllers and an airspeed controller. Each of the controller architectures is PID, with optimisation performed using a BA to obtain the internal controller gains. The BA was selected for optimisation after a review of literature followed by an in-depth study into various meta-heuristic algorithms. This in-depth study concluded that the BA was able to converge faster than a GA whilst maintaining the same global solution convergence rate for eight optimisation test functions.

The optimisation process for each of the angular rate controllers was discussed individually. Each of the angular rate controllers are comprised of multiple gain surfaces, allowing positive input demands, negative input demands, zero or low input demands, and precision demands. Each gain surface is scheduled using a novel gain scheduling approach, forming controller gain hypersurfaces. This scheduling technique enables the flight physics model to maintain adequate dynamic performance and closed-loop stability throughout the operating domain. In addition, the positive gain surfaces for both the roll and pitch rate controllers are normalised against the optimised demand, allowing appropriate scaling to be performed within the controller [19]. Not only did this demonstrate improvements to dynamic performance, it also reduced the necessary controller effort by allowing the controller gains to scale more appropriately based on the amplitude of the reference signal.

Although it was mentioned in Section 4.1.1 that fast-switching control laws can induce instability, the switching action of the applied controllers does not occur during large accelerations of transient time response, and has been demonstrated to not induce dynamic instabilities. In addition, each optimised gain surface which creates a hypersurface is optimised and switched based on a specific set of rules, ensuring the appropriate surface engages based on the system states and reference signal.

Finally, unlike the angular rate controllers, the airspeed controller did not require a hypersurface gain scheduling due to its slowly varying state dynamics. However, these slowly varying state dynamics plagued the internal controller

states by introducing integral wind-up, and saturating the output of the controller during low position errors. This was alleviated by scheduling the internal controller components against the controller position error, allowing the controller to function as a PD controller during high position error scenarios, and a PID controller during low position error scenarios. This was demonstrated to allow the flight physics model to increase, decrease and maintain airspeed, throughout the operating domain with a high level of accuracy, despite only having a single set of optimised controller gains.

Having applied various autopilots and control systems to the flight physics model in Chapter 4, the path planning algorithm is discussed in Chapter 5.

Chapter 5

Path Planning Algorithm and Guidance

This chapter introduces the path planning problem, allowing various order Bézier curves to be investigated. The optimal order Bézier curve is determined based on solution robustness, measured by ensuring paths of all position/waypoint configurations are curvature and orientation constrained. The chapter concludes by introducing the guidance control system. The guidance control system interprets path information and outputs corresponding reference signals to the angular rate controllers, allowing the aircraft to autonomously follow any dynamically achievable path.

5.1 Path Planning Algorithm

It was clear, based on the literature reviewed in Chapter 2, that when producing paths for a fixed-wing aircraft that are dynamically achievable, continuously flyable and computationally efficient, Bézier curves stand out as a viable solution. However, before applying the mathematics allowing a solution to the Bernstein polynomial to be computed, the path planning problem needed to be formulated.

5.1.1 Problem Formulation

To navigate from one location in space to another, the aircraft must follow a flight path. The flight path must be constrained to the dynamic capabilities of

the aircraft, ensuring that any proposed path is dynamically achievable by the aircraft. In addition to the dynamic achievability constraint, since the aircraft does not have a hover mechanic and cannot instantaneously rotate its velocity vector, the orientation of the path must account for the aircraft's forward air-speed. This will ensure that whenever a path is produced online during flight, the path orientation is constrained to the aircraft's current direction of flight, with the curvature of the path adequately compensating for the required change in direction.

To formulate the path planning solution, the following assumptions have been made:

1. The aircraft is located at some position in the inertial frame (p_1) and intends to navigate to a single location in space (p_2). This inertial target is denoted as the waypoint target (Figure 5.1).
2. The aircraft and the waypoint target are defined in individual three-dimensional body-axis coordinate systems, with respect to the inertial frame.
3. The difference between the aircraft's flight path angles and the orientation of the body-axis with respect to the inertial frame are negligible, i.e. $\alpha = \theta_b$ and $\beta = \psi_b$.
4. A path finding algorithm already exists within the mission planning framework. This path finding algorithm is responsible for determining a collision-free path between the aircraft and the waypoint target. Since the aircraft is operating at mid-to-high altitude levels, it has been assumed that obstacles do not exist within the inertial frame, therefore the aircraft can path directly to the waypoint target without requiring intermediate waypoints.

One of the assumptions discussed above, was that the difference between the flight path angles and orientation of the aircraft, was negligible. This reduced the dimensionality of the path planning problem, thus reducing its complexity and the number of variables which have to be accounted for within the solution. Although this could impact the validity of the solution, the solution is applied to the full 6-DoF aircraft model for validation. In addition, the guidance law (introduced at the end of Chapter 5) performs navigation using a pure pursuit

guidance law, by which controller demands are applied to the inner loop control system. This alleviates any discrepancies between the flight path angles and orientation of the aircraft model, by LOS alignment.

Based on these assumptions, for an aircraft located in an inertial frame described by cartesian coordinates, there exists two sets of geometric parameters that are of interest: the location of the aircraft (x_i, y_i, z_i) and the orientation of the aircraft's body-axes $(\phi_i, \theta_i, \psi_i)$, both with respect to the inertial frame.

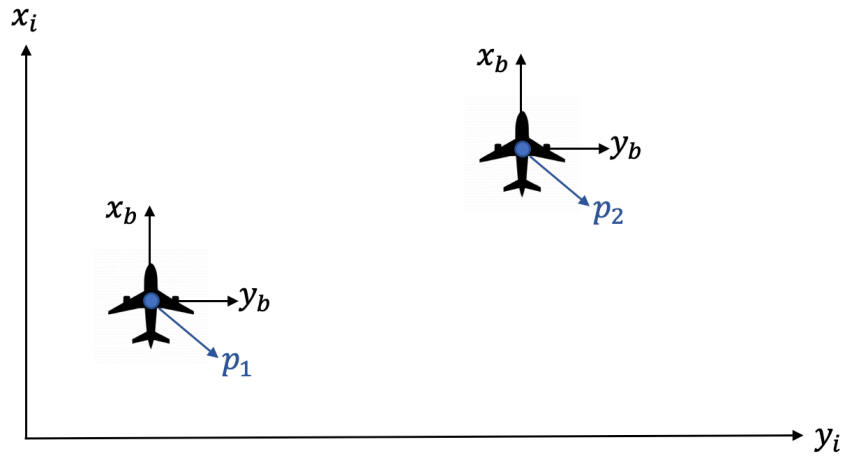


FIGURE 5.1 Path planning problem formulation in two-dimensions

Consider the scenario depicted in Figure 5.1. The aircraft is located at some arbitrary position, p_1 , within an inertial frame, and has a body-axis coordinate system where the origin is coincident with the centre of gravity. One of the assumptions made is that the difference between the flight path angle and the body-axis orientation was negligible, meaning the motion vector is described solely by the orientation of the body-axis vectors. Considering the problem solely in the x - y plane, the dimensionality of the aircraft's orientation vector is reduced, allowing the orientation to be described solely by the aircraft's azimuth angle (ψ). This allows the orientation of the aircraft and the direction of the motion vector to be described solely with respect to the heading angle, which, for the example shown in Figure 5.1, is 0° .

An inertial frame waypoint target has been specified in the inertial frame represented as p_2 . The waypoint target has also been described as a set of three-dimensional cartesian coordinates with a set of body-axis vector orientations. The waypoint represents the aircraft's desired position and orientation at some

future location in time. Under this formulation, as the aircraft navigates towards the waypoint, upon arriving at point p_2 , the aircraft should maintain the same orientated motion vector as indicated by the waypoint body-axis vectors.

Based on the literature review conducted in Section 2.2, the design of the aircraft and its dynamic capabilities is something that needs to be carefully considered when path planning. For example, since the computational model is based on a fixed-wing aircraft, and does not have mechanical capabilities to hover or actively reverse the velocity or motion vector, the flight path of the aircraft is restricted by its manoeuvrability. Considering the configuration in Figure 5.1, although the aircraft is initially located at p_1 , it is not a static object. This means that the representation at point p_1 was taken at some instance in time. An assumption was made that the motion vector of the aircraft is solely represented by the orientation of the x_b axis, i.e. there are no sideslip forces. Therefore, the dynamics of the aircraft at point p_1 describe a motion vector travelling parallel to the x_i axis when observed solely in the x_i - y_i plane. This adds the first constraint to path planning, since the initial direction of the path must adhere to the original dynamics and flight path of the aircraft at that specific instance in time. The same constraint can be applied when approaching the waypoint target (p_2). By specifying a distinct flight path angle at location p_2 , as the aircraft approaches the waypoint, the direction of the path needs to correctly align the aircraft such that the orientation is correct as the aircraft passes directly through p_2 . By applying these two dynamic constraints, it can be assured that the direction of travel along the path is correctly orientated for the initial and final conditions. Throughout the remainder of the discussion, these conditions have been denoted as the orientation constraint.

To satisfy the orientation constraint, the aircraft and waypoint are considered as individual unit vectors, described as a three-dimensional right-handed orthogonal body-axis system. The position and orientation of the unit vectors is observed with respect to the inertial frame and are described by Equation 5.1. Both the aircraft and waypoint target are described individually, by their own

unit vectors.

$$x_b = \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} \quad y_b = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} \quad z_b = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} \quad (5.1)$$

Each unit vector x_b , y_b and z_b , represent a body-axis coordinate system, with rotation performed using a quaternion transformation matrix, Λ , allowing the orientation of the three vectors to be determined with respect to the inertial frame. The quaternion transformation vector, Λ , is described based on the corresponding Euler orientation angles ϕ , θ and ψ . Therefore, $\Lambda = f(\phi, \theta, \psi)$.

The un-rotated body-axis unit vectors in Equation 5.1 are rotated based on the aircraft's orientation at point p_1 and desired orientation at waypoint p_2 , both with respect to inertial frame.

$$x_{b,rot} = \Lambda \times x_b \quad y_{b,rot} = \Lambda \times y_b \quad z_{b,rot} = \Lambda \times z_b \quad (5.2)$$

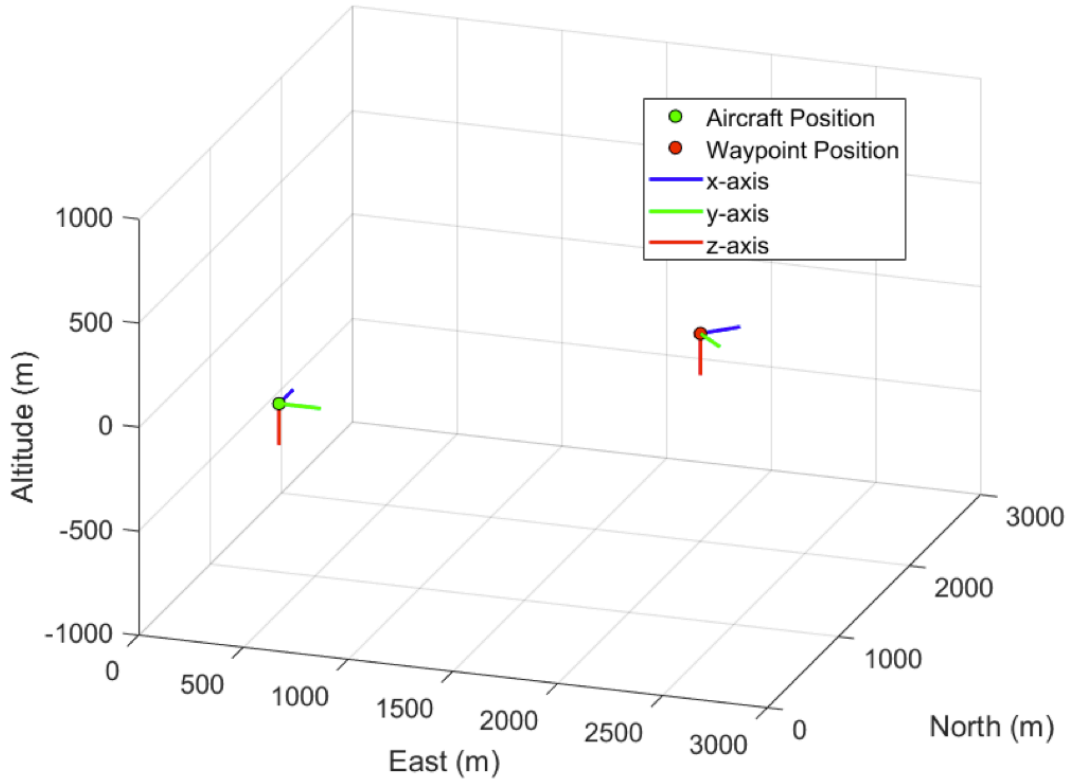


FIGURE 5.2 Demonstration of the aircraft and waypoints unit vectors described in the inertial frame

The vectors in Equation 5.2 represent a rotated body-axis system when observed from the inertial frame, but not yet located in the inertial frame. These unit vectors are translated back into the inertial frame by offsetting them against the aircraft's inertial frame cartesian coordinates at p_1 and the waypoint's inertial frame cartesian coordinates p_2 .

$$\begin{aligned} x_{i,rot} &= x_{b,rot} + [x_i \ y_i \ z_i] \\ y_{i,rot} &= y_{b,rot} + [x_i \ y_i \ z_i] \\ z_{i,rot} &= z_{b,rot} + [x_i \ y_i \ z_i] \end{aligned} \quad (5.3)$$

These unit vectors mathematically describe the location and orientation of both the aircraft and the waypoint in the inertial frame, allowing visualisation of the unit vectors purely for clarity when constructing the solution.

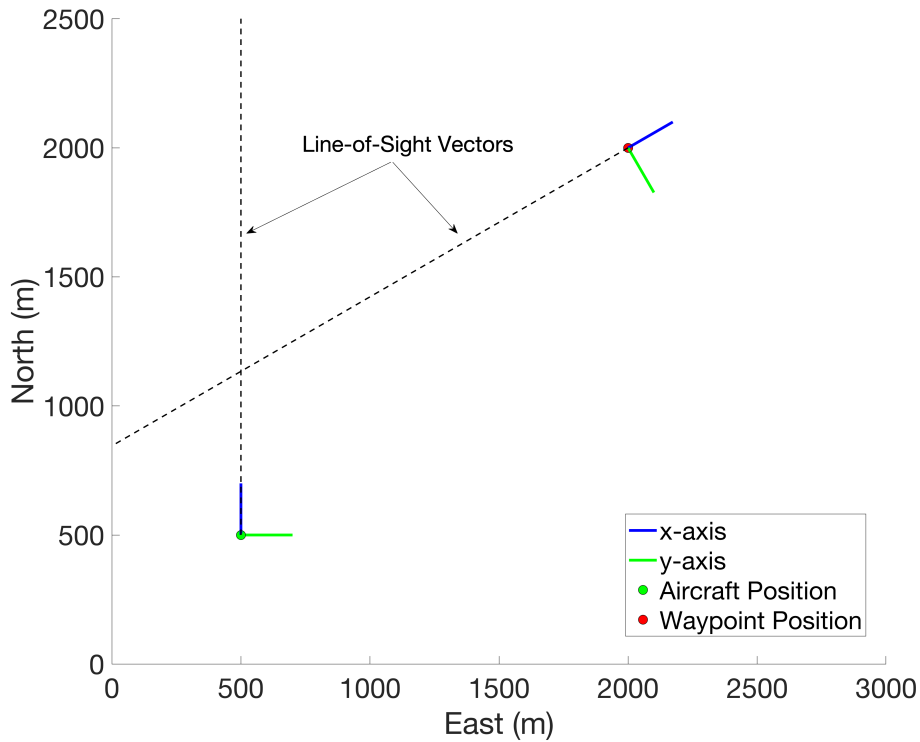


FIGURE 5.3 Line-of-sight heading vectors used to satisfy the path planning orientation constraint for both position and waypoint unit vectors

Figure 5.2 shows the unit vectors in the inertial frame corresponding to the aircraft being located at (500 m, 500 m, 0 m) with an orientation of $(0^\circ, 0^\circ, 0^\circ)$. The waypoint target unit vectors are located in the inertial frame (at 2000 m,

2000 m, 0 m) with an orientation of $(0^\circ, 0^\circ, 60^\circ)$. When viewing the inertial frame representation in Figure 5.2 solely in the x - y plane, the orientation of both the aircraft and waypoint can be represented by two line-of-sight vectors, as depicted in Figure 5.3.

Representing both the aircraft and waypoint in this manner allows not only visualisation of the inertial frame positions, but also allows placement of the internal control points to be easily applied. Placement of the internal control points not only allows the orientation constraint to be adequately satisfied, but also allows manipulation of the curvature profile for the Bézier curve. However, as the number of internal control points increases, so too does the order of the Bernstein polynomial. As the order of the Bernstein polynomial increases, each internal control point has less influence over the resultant curvature. Despite this, as the number of internal control points increases, the overall fidelity of the Bernstein polynomial can be manipulated further, provided multiple control points are placed similarly together, allowing adequate and even manipulation of the curvature profile. However, as the number of internal control points increases, the computational complexity required to optimise the location of each internal control point increases. Therefore, it is advantageous to try to formulate a path planning solution surrounding the lowest order Bernstein polynomial that creates satisfactory solutions for all identified flight scenarios, starting with a second-order Bézier curve.

5.1.2 Bézier Mathematics

Given a set of control points, $p \in [p_0, p_n]$, a Bézier curve is formed by solving Equation 5.4 [134].

$$C(t) = \sum_{i=0}^n P_i B_{i,n}(t) \quad (5.4)$$

where t is a dimensionless parameter $\in [0, 1]$, and $B_{i,n}$ represents the solution to the Bernstein polynomial. Where,

$$B_{i,n} = \binom{n}{k} t^k (1-t)^{n-1} \quad (5.5)$$

The coefficients of the Bernstein polynomial, represented by n and k , are obtained through binomial expansion. During binomial expansion, the order of

the polynomial is indicated by the number of coefficients, which also defines the order of the Bézier curve with the relationship $n-1$, where n is the number of control points. Shown in Equation 5.6 is the coefficients which comprise a zeroth order up to fifth order binomial expansion.

$$\begin{aligned}
 (a+x)^0 &= (1) & (a+x)^1 &= (1, 1) \\
 (a+x)^2 &= (1, 2, 1) & (a+x)^3 &= (1, 3, 3, 1) \\
 (a+x)^4 &= (1, 4, 6, 4, 1) & (a+x)^5 &= (1, 5, 10, 10, 5, 1)
 \end{aligned} \tag{5.6}$$

5.1.3 Second-Order Bézier Curves

When considering a second-order Bézier curve, there exists three control points, with only one internal control point. The location of the two external control points act as the start and finish location for the Bézier curve, meaning they coincide with the position of the aircraft and waypoint target. Since a second-order Bézier curve only has a single internal control point, determining its location is ultimately the determining factor in the resultant Bézier curvature.

It was previously discussed that an orientation constraint must be satisfied for path planning of a fixed-wing aircraft, as this ensures the direction of the path at p_1 and p_2 (Figure 5.1) coincides with the initial and final path planning orientation demands. It was also discussed how both the aircraft and waypoint are described as a set of body-axis unit vectors, where the x -vector indicates the heading vector or flight path whilst assuming no sideslip. By extending the heading vectors of both the aircraft and waypoint, a set of line-of-sight vectors is produced, as depicted in Figure 5.3. Placing the internal control point along these two heading vectors satisfies the orientation constraint, as both the initial and final flight path adequately aligns with the initial and final conditions.

However, since a second-order Bézier curve only has one internal control point, to satisfy both the initial and final orientation constraint, the control point must be placed at the intersection of both heading vectors. This can be seen in Figure 5.4, where the location of the internal control point occurs at the intersection of both heading vectors, resulting in a path that satisfies both the initial and final orientation constraint. The direction of travel at the start of the path occurs directly ahead of the aircraft's x -vector, allowing the curvature of the path to

gradually rotate the aircraft's velocity vector before aligning perfectly along the x -vector of the waypoint target.

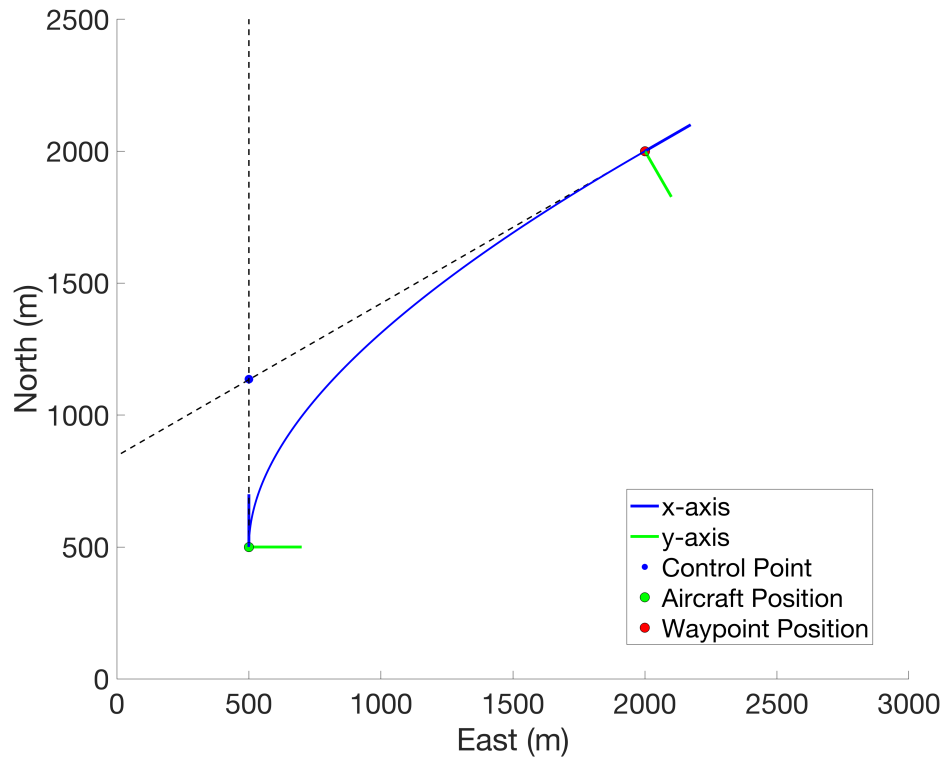


FIGURE 5.4 Second-order Bézier curve with intersecting control point placed at the intersection of the heading vectors

Although placing the internal control point at the intersection of both heading vectors does satisfy the orientation constraint, there is no consideration for the dynamic feasibility of the path since the location of the internal control point cannot be adjusted without dissatisfying the orientation constraint. In addition, the orientation of the aircraft and waypoint depicted in Figure 5.4 is such that an intersection of both heading vectors is possible. This is not true for all flight scenarios, meaning that if an intersection between both vectors is not possible, the location of the internal control point can only occur along one of the heading vectors, failing to satisfy the orientation constraint. However, this can be remedied by implementing an interim waypoint at some location between the position of the aircraft and the waypoint target, provided the orientation of the waypoint allows its own heading vectors to coincide with the heading vectors of both the aircraft and waypoint target, as demonstrated in Figure 5.5

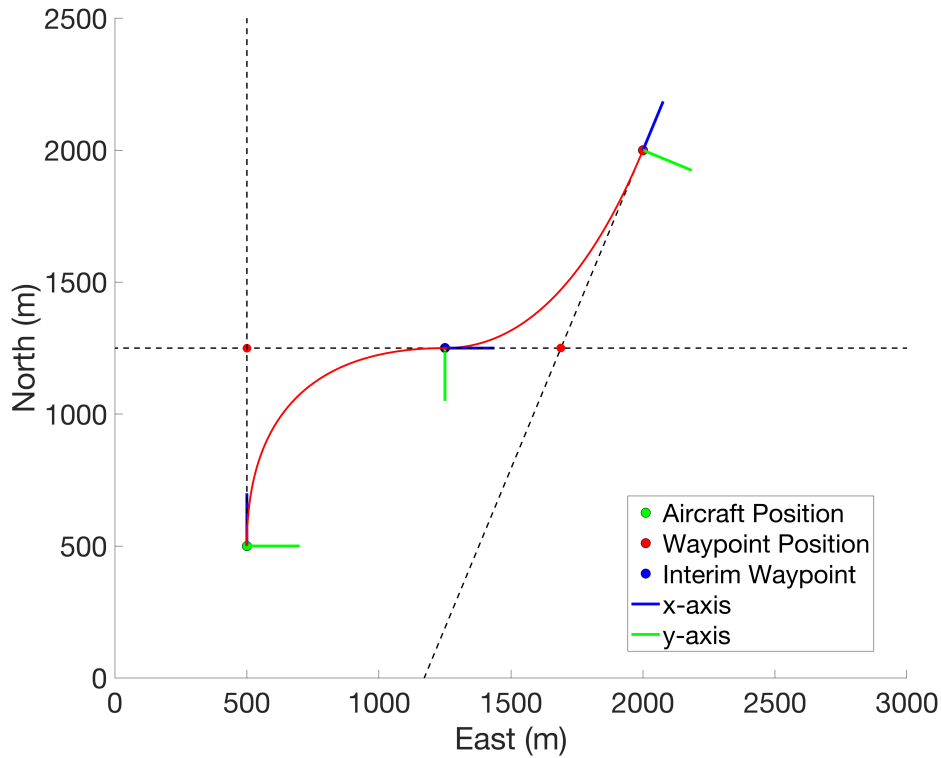


FIGURE 5.5 Second-order Bézier curve with intersecting control point placed at the intersection of the heading vectors

By introducing an interim waypoint with an azimuth of 90 degrees, both the positive and negative heading vectors for the interim waypoint create intersections with the single heading vectors of the position and waypoint target. Additionally, since the location and orientation of the interim waypoint is not fixed, the curvature profile of the final path can better account for the dynamics of the vehicle by adjusting the location and orientation of the interim waypoint, whilst still satisfying the orientation constraint. However, as more interim waypoints are introduced, the number of Bézier curves that require computing increases, as each individual Bézier curve is stitched together to form the final path. Not only does this increase the computational complexity to obtain a solution, it introduces discontinuity at each joining section, which was one of the reasons Dubins paths was neglected as a solution.

Currently, computing Bézier curves has concentrated solely on paths in the x - y plane with both the aircraft and waypoint target located at the same altitude. This enabled the heading vectors to intersect in the x - y plane, and where

an intersection is not possible introduce an interim waypoint, allowing the orientation constraint to be satisfied. When considering altitude changes between the position and waypoint target, whilst an intersect may be visually present in the x - y plane, it is not present when considering all three axes. This causes issues for the generation of a second-order Bézier curve because although the orientation constraint is satisfied in the x - y plane, it remains unsatisfied in the x - z and y - z planes. A demonstration of this can be seen in Figure 5.6.

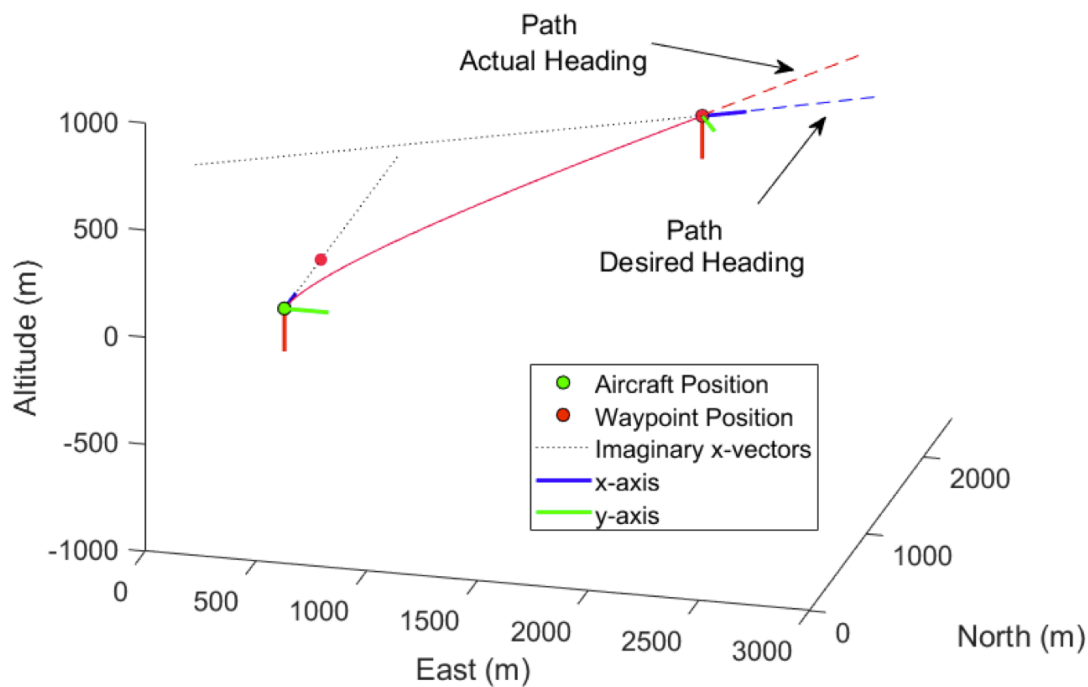


FIGURE 5.6 Second-Order three-dimensional Bézier curve with unsatisfied orientation constraints

Although the location of the internal control point occurs along the x - y intercept of both heading vectors, due to the altitude change and a pitch angle of 0° , both heading vectors are parallel, meaning an intersection is not possible in the x - z and y - z plane. For the purposes of illustration, the control point has been placed at the location of the x - y intercept but along the aircraft's heading vector as opposed to that of the waypoint target. Although this creates the desired flight path at the start of the path, there exists an angular offset towards the end of the path as the control point does not sufficiently align the path along to the waypoint's heading vector.

This angular offset occurs as a pitch offset as opposed to an azimuth offset

and is illustrated in Figure 5.6 by extending the desired path heading and actual path headings in three-dimensional space. Although the magnitude of the angular offset is reduced by placing the internal control point closer to the midpoint on the z -axis between the position and waypoint, in doing so an equal offset occurs at the start and end of the path as opposed to solely at the end of the path. Despite this, the path remains unsatisfactory when considering the orientation constraint.

Whilst this problem could be solved by introducing one or more interim waypoints that are orientated with pitch and yaw angles allowing heading vector intersections to be produced in all planes, the complexity of the resulting solution makes the problem inefficient. Depending on the altitude change and the various position/waypoint locations and orientations, the interim waypoint needs to be adequately positioned in the x - y plane and along the z -axis with a specific pitch and azimuth angle. Knowledge of this exact location is required to ensure that the interim waypoint x -vector successfully creates an x , y , z intersect with both the position and waypoint heading vectors. Whilst this could be solved as an optimisation problem, the introduction of the interim waypoint creates a more computationally complex problem than necessary. A simpler solution is to place a control point along both the position and waypoint heading vectors, creating a third-order Bézier curve.

5.1.4 Third-Order Bézier Curve

As discussed in Section 5.1.3, the application of second-order Bezier curves for two- and three-dimensional path planning is not without issue. The workarounds associated with these issues increase the overall complexity of the solution, and in some cases require the computation of more than one Bézier curve. However, many of the issues associated with the use of a second-order Bézier curve are eradicated by implementing an additional internal control point. This allows for two internal control points to manipulate the curvature profile of the path, whilst also ensuring the orientation constraints remain satisfied. The same unit vector analysis as described in Section 5.1.3 is applied to determine the heading vectors of the aircraft and the waypoint target, for control point placement and demonstration purposes.

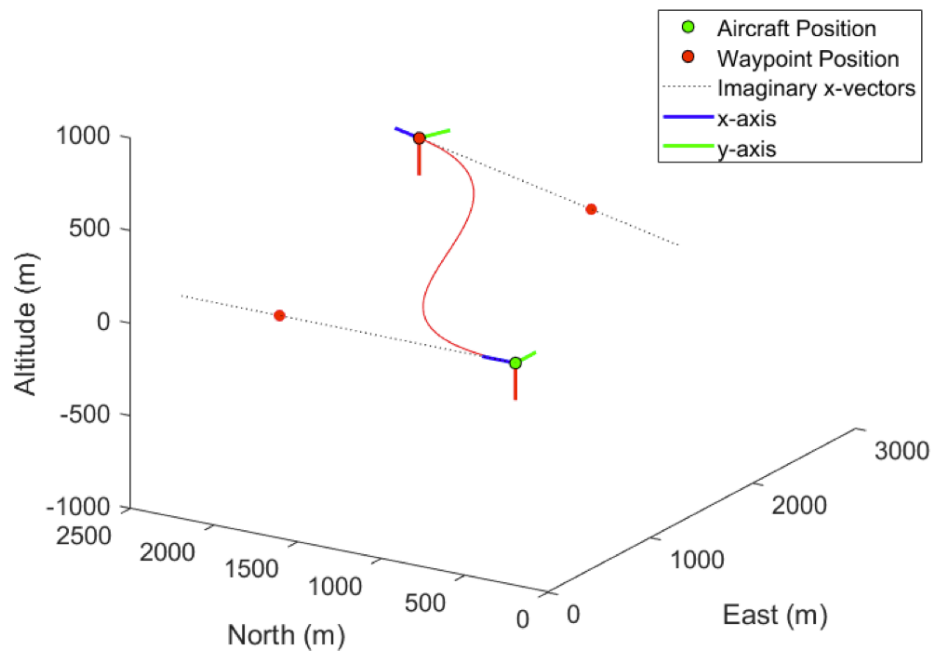
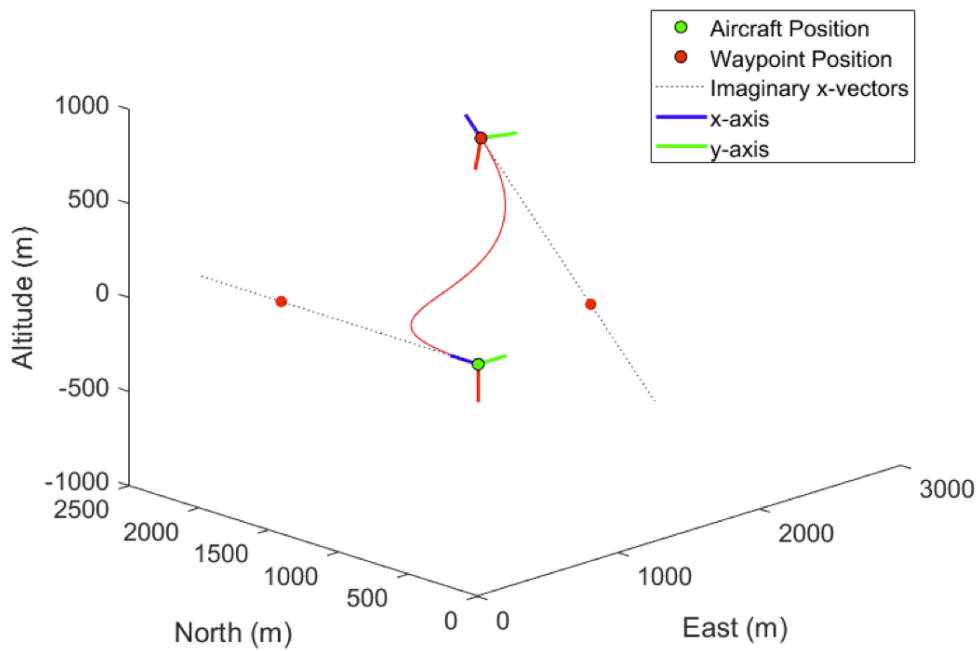
(a) Pitch angle 0° (b) Pitch angle 20°

FIGURE 5.7 Third-order Bézier curve in three-dimensions with variable pitch angle

Consider the two-dimensional path planning scenario depicted in Figure 5.5. For a second-order Bézier, this posed issues since the heading vectors did not

result in an x - y intersect occurring. Producing a path that satisfies both aircraft and waypoint target orientation constraints using a third-order Bézier curve is simple, since a control point can be placed along each of the heading vectors, eliminating the need for an intersection or interim waypoint. Additionally, use of a third-order Bézier curve also extends easily from two-dimensions to three-dimensions as seen in Figure 5.7(a). Since a three-dimensional rotation is applied to the position and waypoint unit vectors, the orientation of the heading vector is always correctly aligned, meaning that describing the location of the control point along the heading vector ensures the orientation constraint remains satisfied. This can be seen in Figure 5.7a where the orientation constraint remains satisfied despite the waypoint target having a pitch angle of 20° .

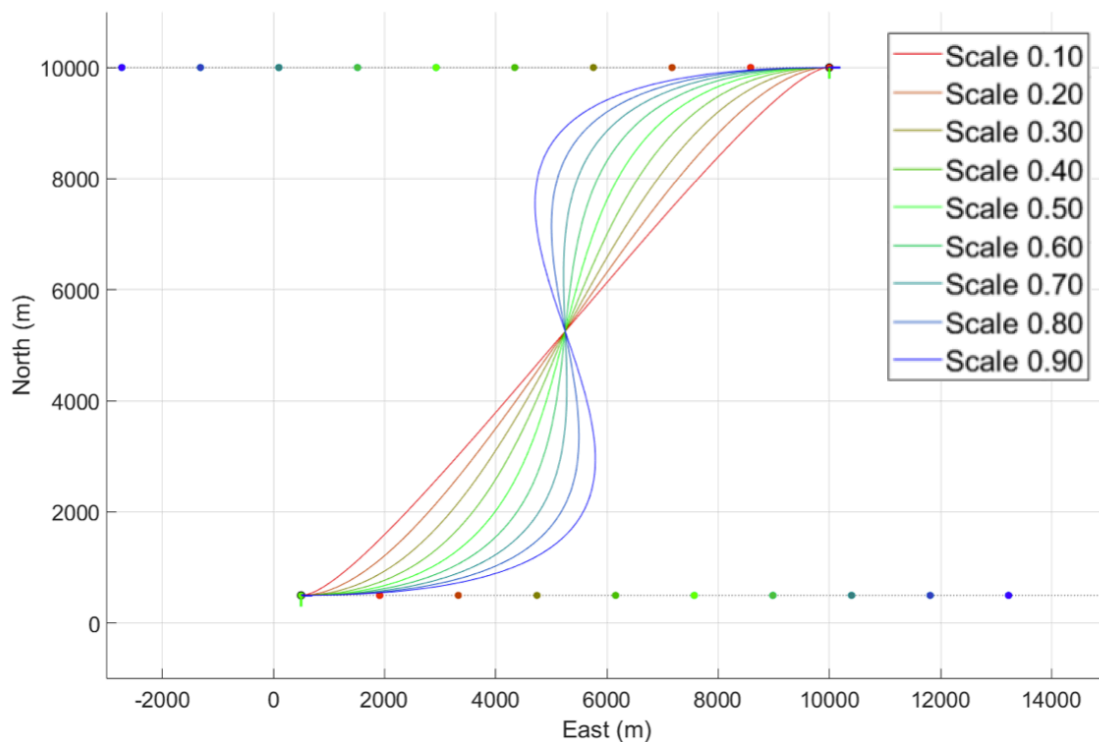


FIGURE 5.8 Symmetrical control point scaling for second-order Bézier curves using the globally defined variable.

For the two scenarios depicted in Figure 5.7, the locations of the internal control point are placed at some arbitrary distance along each heading vector. This distance is an important factor as it allows the curvature profile of the Bézier to be manipulated, altering the dynamics of the path. Rather than describing the distance along the heading vector as a coordinate relative to the unit vectors

origin, these quantities are described as dimensionless coefficients that are scaled against a globally defined variable.

The use of a globally defined variable ensures repeatability and consistency, whilst also ensuring that paths that have a small point-to-point euclidean distance result in small scaling functions, whereas large point-to-point euclidean distance paths have larger scaling functions. This enables predictions to be performed, which was a vital aspect of the applied intelligence.

The global variable was defined based on the operational envelope for the computational model. It was discussed in Section 3.11 that the computational model has an altitude ceiling of 10 km. When considering a two-dimensional plane with edges 10 km by 10 km, the corner-to-corner distance defined the global variable with a magnitude of 14.142 km. The effects of increasing the scaling functions can be seen in Figure 5.8, where a series of third-order Bézier curves have been produced for symmetrical levels of scaling.

As the scaling function approaches zero the Bézier curves begin to straighten out, reducing the curvature profile. If the scaling function becomes zero, this would replicate a first-order Bézier curve or a straight line. However, expanding the control points towards one, pulls the Bézier curves in alternative directions, increasing the curvature profile whilst also satisfying the orientation constraint. This demonstration indicates how a Bézier curve can be manipulated by simply adjusting a single scaling factor applied to a third-order Bernstein polynomial. The level of manipulation can also be increased by allowing each of the control points to scale independently of one another, although not seen in this demonstration.

Having solved numerous issues associated with the orientation constraints by implementing an additional internal control point, the dynamics of the Bézier were considered. The dynamics of the Bézier focused on deriving the curvature profiles and predicting whether or not a path would be feasible based on the required dynamics.

$$\kappa_{xy} = \frac{\dot{x}\ddot{y} - \dot{y}\ddot{x}}{(\dot{x}^2 + \dot{y}^2)^{3/2}} \quad \kappa_{xz} = \frac{\dot{x}\ddot{z} - \dot{z}\ddot{x}}{(\dot{x}^2 + \dot{z}^2)^{3/2}} \quad \kappa_{yz} = \frac{\dot{y}\ddot{z} - \dot{z}\ddot{y}}{(\dot{y}^2 + \dot{z}^2)^{3/2}} \quad (5.7)$$

Initially, the curvature of the Bézier is analysed individually for each two-dimensional plane, using Equation 5.7. The terms $\dot{x}$, $\dot{y}$ and $\dot{z}$ are the first derivatives of the path and $\ddot{x}$, $\ddot{y}$ and $\ddot{z}$ are the second derivatives. Both the first

and second derivatives are discretised quantities, where the subscript terms x , y , and z represent the change in the path along that specific axis. This enabled an iterative computation of the curvature profile to be established at various points along the path, using Equation 5.7. However, during Chapter 7, the curvature profile is analysed in three-dimensions as opposed to two.

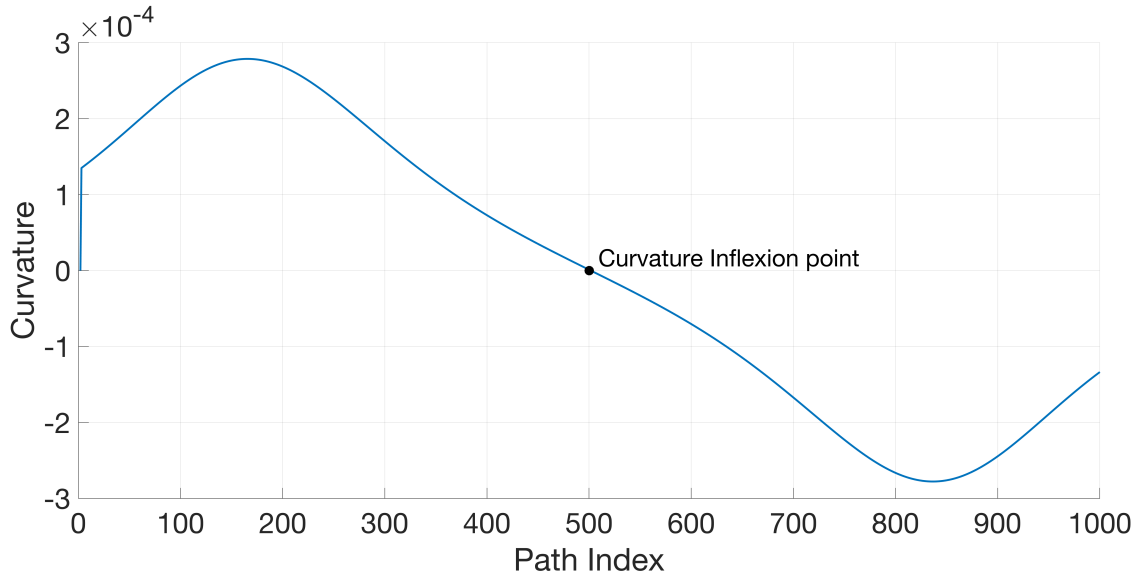


FIGURE 5.9 Curvature profile in the x - y plane for Bézier path in Figure 5.8 with a scaling factor of 0.5.

Consider the path in Figure 5.8 for scaling functions of 0.50. The length of the path is relatively large with a length of almost 15 km. Due to the path length, the curvature profile remains small. Additionally, since there is no change in altitude over the course of the path, the curvature profile in the x - z plane (k_{xz}) and y - z plane (k_{yz}) are zero. Therefore, the curvature is only analysed in the x - y plane, as seen in Figure 5.9.

The curvature profile of a path is a dimensionless quantity and is of little dynamic importance. A more appropriate dynamic quantity is the radius of curvature, which is calculated as the reciprocal of the curvature profile. Hence,

$$r_{xy} = \frac{1}{\kappa_{xy}} \quad r_{xz} = \frac{1}{\kappa_{xz}} \quad r_{yz} = \frac{1}{\kappa_{yz}} \quad (5.8)$$

It is important to note that since the path is symmetrical, as the midpoint is approached, the profile straightens and the curvature approaches zero. This results in a radius of curvature which tends towards infinity, as seen in Figure

5.10. As the radius of curvature approaches infinity, this indicates that the path has reached its inflexion point, and begins to transition from a left turn to a right turn, or vice-versa.

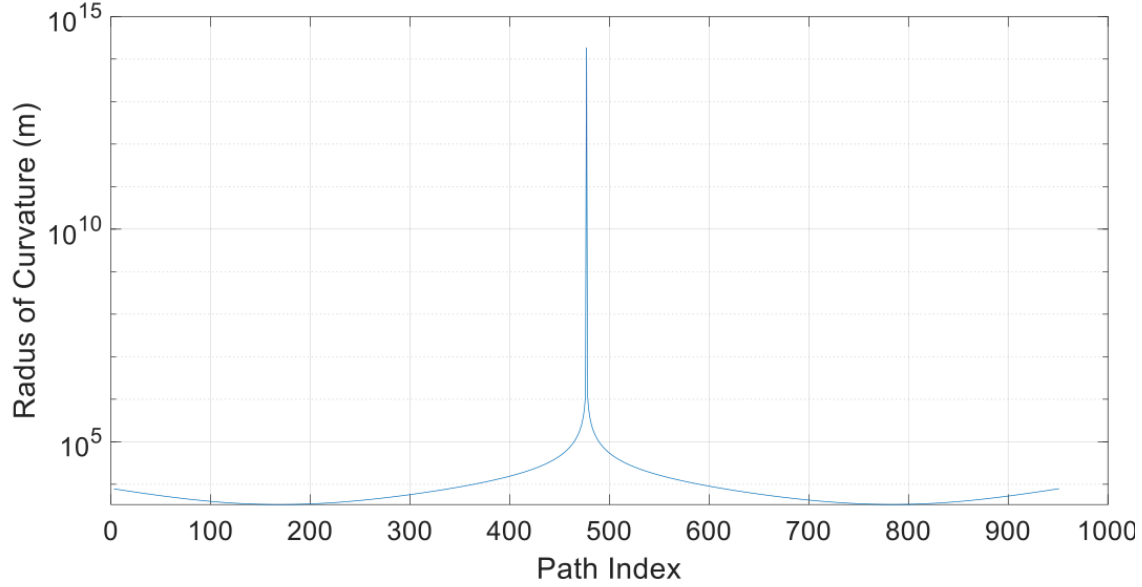


FIGURE 5.10 Radius of curvature profile in the x - y plane for Bézier path in Figure 5.8 with a scaling factor of 0.5.

The radius of curvature is an important property since it is a requirement when determining the acceleration of the path (a), acceleration due to gravity (a_g), bank angle (ϕ) and turn rate (ω), as per Equation 5.9, where V represents the velocity vector and g represents gravitational acceleration (9.81 m/s). These dynamic parameters are the first step in determining the dynamic feasibility of a path for a given set of state dynamics.

$$a = \frac{V^2}{r_{xy}} \quad a_g = \frac{a}{g} \quad \phi = \text{atan}\left(\frac{V^2}{gr_{xy}}\right) \quad \omega = \frac{V}{r_{xy}} \quad (5.9)$$

Due to the length of the path being so large, the minimum radius of curvature exceeds 3 km, meaning that a maximum turn rate of 4.29 deg/s is required when travelling at an airspeed of 250 m/s. This indicates that the dynamics of the path are achievable, since, in two-dimensions the turn rate is lower than the maximum turn rate attainable by the aircraft. This analysis was one of the best-case scenarios since the magnitude of the initial and final points was so large whilst the heading change was zero.

Consider one of the worst case scenarios. The aircraft is located in the inertial frame at a given location (500 m, 500 m, 0 m) and with an orientation $(0^\circ, 0^\circ, 0^\circ)$. The waypoint is located in the inertial frame (2000 m, 2000 m, 0 m) and with an orientation $(0^\circ, 0^\circ, 270^\circ)$. This configuration results in a large heading change demand over a relatively short distance. When applying a third-order Bézier curve with two scaling functions, both of size 0.10, the resultant path contains a tight cusp, as seen in Figure 5.11.

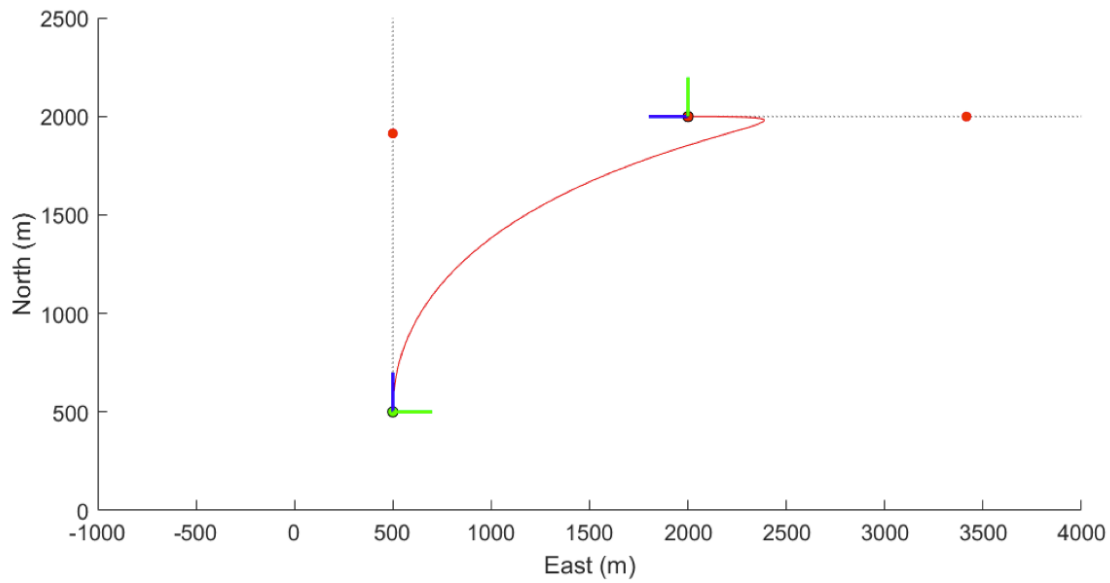


FIGURE 5.11 Second-order Bézier curve for large heading change resulting in a large cusp

At the location of the cusp, the curvature of the path approaches one resulting in a radius of curvature as low as 3.3 metres. This low radius of curvature produces a very visible and dynamically challenging cusp, with a required turn rate exceeding 3400 deg/s and 1200g at 250 m/s, making the path dynamically unachievable. To reduce the cusp, additional control points can be implemented, producing a higher-order polynomial.

5.1.5 Higher-Order Bézier Curves

The previous analysis concluded that the use of a third-order Bernstein polynomial did successfully enable the orientation constraint to be satisfied by placing two vector constrained control points. However, the use of a third-order

polynomial did not allow for enough controllability when developing dynamically feasible trajectories under the applied worst-case scenarios. For high heading changes at small distances, the low-fidelity control over the positions of the two internal control points did not allow for adequate manipulation of the curvature, resulting in the presence of a high curvature cusp. By introducing one or more internal control points, the Bernstein polynomial becomes fourth-order or higher.

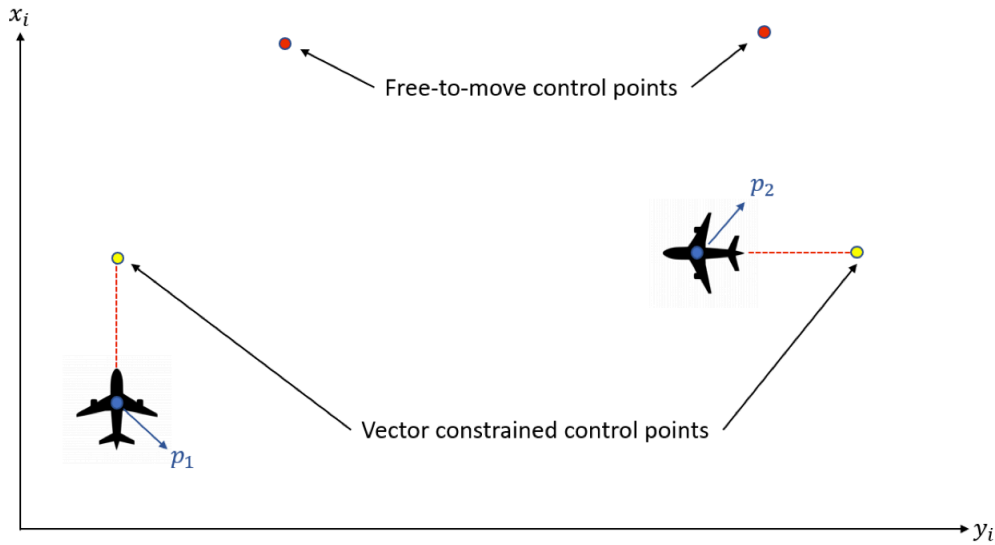


FIGURE 5.12 Higher-Order Bézier curve control point schematic

Unlike the previous two control points that are constrained along an orientated vector, the additional control points are free to move throughout the inertial reference frame using the same globally defined variable as the previous analysis. A schematic of this can be seen in Figure 5.12. Since these free-to-move control points can be placed anywhere within the inertial frame, the location of the control points requires an additional three scaling functions per point, each corresponding to a location along the x , y and z axes with respect to the inertial frame.

Consider the same worst-case scenario as depicted in Figure 5.11. Due to the large heading change demand and a relatively low point-to-point distance, solving the problem with a third-order polynomial induced a large internal cusp when using scaling factors of 0.10 and 0.10. The cusp was unable to be reduced by simply increasing the size of the scaling factors towards one. However, by

introducing additional control points, where the control points are free to move throughout the inertial axis, as depicted in Figure 5.12, the profile of the path can be manipulated to completely remove the third-order cusp.

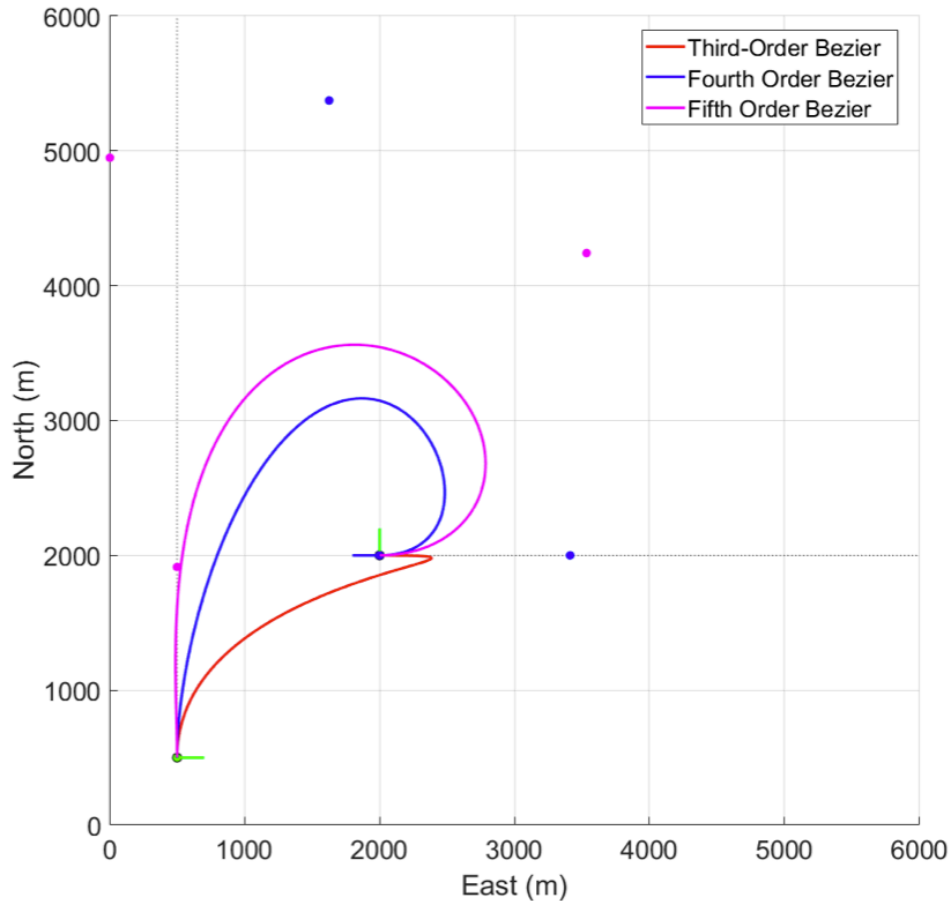


FIGURE 5.13 High heading change at low magnitude distance comparison for third, fourth and fifth-order Bernstein polynomials

By implementing a fourth- and fifth-order polynomial, the resultant Bézier curve is able to loop around the back of the waypoint before aligning its heading with the waypoint. Whilst the trajectories shown in Figure 5.13 may be sub-optimal due to manual placement of the additional control points, the form factor of the curve is somewhat desirable, since the orientation constraint has been satisfied and the path has a low overall curvature profile. One noticeable issue associated with the fourth-order polynomial occurs in the second phase of the response.

Whilst the response remains free from cusps, it is evident that the radial turn is quite tight, resulting in a low radius of curvature and increasing the

required turn rate dynamics. This results in a required turn rate approaching 30 deg/s for the fourth-order path if the velocity of the vehicle is 200 m/s, approaching the upper limits of the aircraft's potential. When comparing this to the fifth-order path which has a larger, more relaxed radial turn, the necessary turn rate has decreased to less than 18 deg/s, well within the dynamic limits of the aircraft. However, the location of the internal control points was manually selected and does not necessarily represent the optimal solution. Therefore, finding the location of the internal control points can be considered to be an optimisation problem.

To solve and determine the optimal location of the internal control points, the path planning problem was broken into two complex studies: constant-altitude path planning and variable altitude path planning, both of which are discussed individually in Chapters 6 and 7.

5.2 Guidance Control System

To enable the aircraft model to autonomously follow paths, a guidance control system was created. The guidance control system receives path information from the path planner and outputs three reference signals. These three reference signals are applied as inputs for the roll, pitch and yaw rate controllers, as depicted in Figure 5.14.

Conventional fighter-type aircraft manoeuvre using bank-to-turn (BTT). This type of manoeuvre sees the aircraft perform a roll in the proposed direction of travel, before pitching up towards the target. Whilst the aircraft could also roll in the opposing direction of travel and pitch down towards the target, this is undesirable during manned flight for two reasons. Firstly, pitching down induces negative g-force, forcing blood to rush towards the pilots head and potentially inducing redout. Secondly, due to geometric asymmetry about the x - z plane, the aircraft is able to sustain greater pitch up dynamics, allowing for faster and more agile manoeuvres.

Since this research focuses on autonomous flight, it is a fair assumption that future unmanned aerial systems will not incorporate a cockpit, allowing for more symmetrical aircraft designs with near identical pitch up and down per-

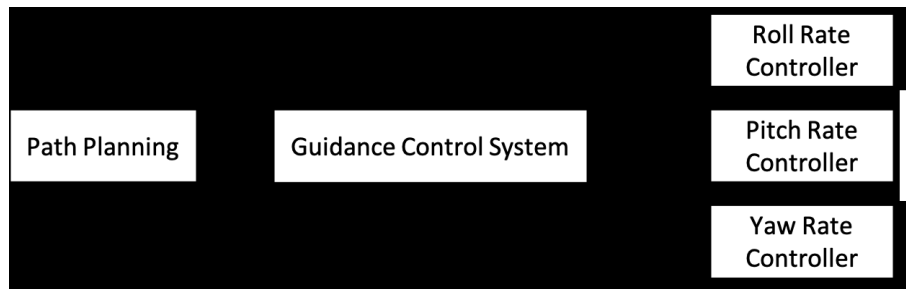


FIGURE 5.14 Guidance block diagram interaction with the angular rate controller

formance. However, since these designs are purely conceptual and the aircraft model developed in this thesis is representative of a current manned aircraft, these factors were not considered during the development of the guidance control system, with the aircraft favouring a pitch up BTT, except when executing small pitch correctional manoeuvres.

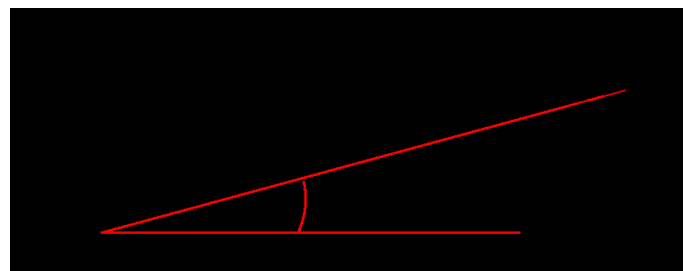


FIGURE 5.15 Pure pursuit line-of-sight illustration

Pure pursuit guidance is one of the simplest guidance laws. Pure pursuit orientates the aircraft along the line-of-sight (LOS) to the reference point, as shown in Figure 5.15. Although this illustrates pure pursuit guidance with the LOS demonstrated for one aircraft intercepting another aircraft, this scenario is also valid when following paths, since paths contain information corresponding to coordinate locations in the inertial frame. By selecting a single coordinate as a reference point, the LOS between between the aircraft and reference points is determined using trigonometry. The LOS angle is used to update the pitch rate dynamics, as this directly correlates to translational position error between the aircraft and reference point.

To enable BTT autopilot outputs from the guidance controller, paths are translated from the inertial frame into the aircraft body-axis frame. This translation is performed using the aircraft's body-axis orientations with respect to

the inertial frame. By performing this translation, the coordinates forming the path are described as relative coordinates to the body-axis, with the orientation of the aircraft neutralised. This action is performed every time-step, creating a snapshot of the aircraft and path at an instance in time, allowing the required autopilot commands to be determined.

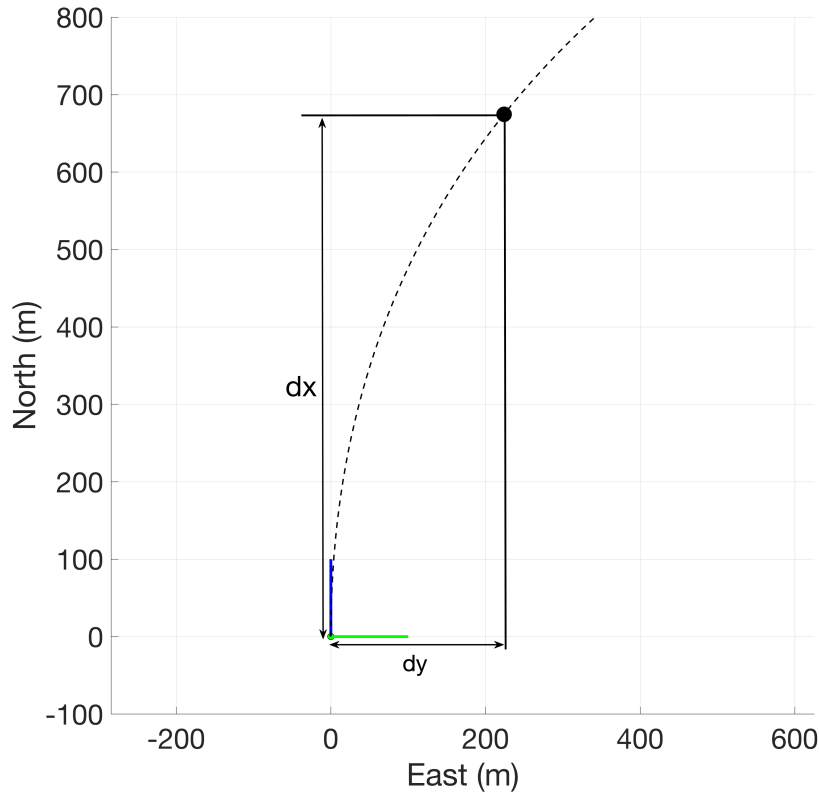


FIGURE 5.16 Path planning guidance initial scenario in the xy plane

Consider the scenario depicted in Figure 5.16. The aircraft is located in the inertial frame with a position (0 m, 0 m, 0 m) and with an orientation (0° , 0° , 0°). Since the aircraft is located at the origin of the inertial frame, and with nominal pose, the path illustrated by the dashed line can already be described in the body-axis system. The reference point as identified by the black marker, was selected arbitrarily for demonstration purposes and allows the distance dx along the north axis, and distance dy along the east axis to be determined. A third measurement dz is also computed along the altitude axis, although not present in the figure. These distances dx , dy and dz are used to compute the

required ϕ , θ and ψ angles, allowing the aircraft to align onto the LOS:

$$\phi = -\arctan\left(\frac{dy}{dz}\right) \quad \theta = -\arctan\left(\frac{dz}{dx}\right) \quad \psi = -\arctan\left(\frac{dy}{dx}\right) \quad (5.10)$$

Since the orientation angles in Equation 5.10 represent relative offsets between the LOS and the orientation of the aircraft along the path with respect to the body-axis, they are applied directly to the angular rate controllers as input demands. If either of the three angles is null, this indicates that along that specific axis, the aircraft is correctly orientated along the LOS, with the angular rate demand holding the angular velocity.

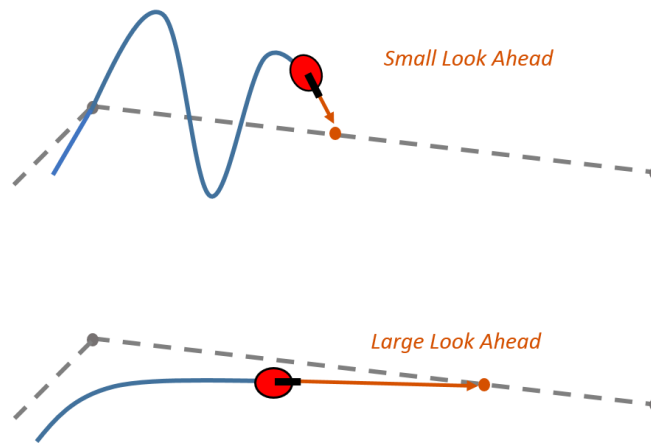


FIGURE 5.17 Small and large look ahead illustrations for pure pursuit guidance [135]

Since multiple coordinates exist forming the profile of the path, to select a coordinate as a reference point, a variable known as the *look ahead* is introduced. The look ahead value is a variable that needs to be carefully considered. For example, if the look ahead distance is small relative to the airspeed and dynamics, the aircraft will perform abrupt changes to the dynamics in an attempt to intersect the reference point. This can result in unwarranted oscillatory motion as the aircraft spirals back and forth around the path as opposed to following it directly. On the other hand, if the look ahead value is too large, the aircraft will perform a smoother transition between reference points, however, the path taken by the aircraft may not accurately resemble the proposed path. Examples of small and large look ahead values can be seen in Figure 5.17.

To determine an adequate look ahead, the aircraft was initialised at 300 m/s and given a series of paths to follow. These paths were simple three-dimensional

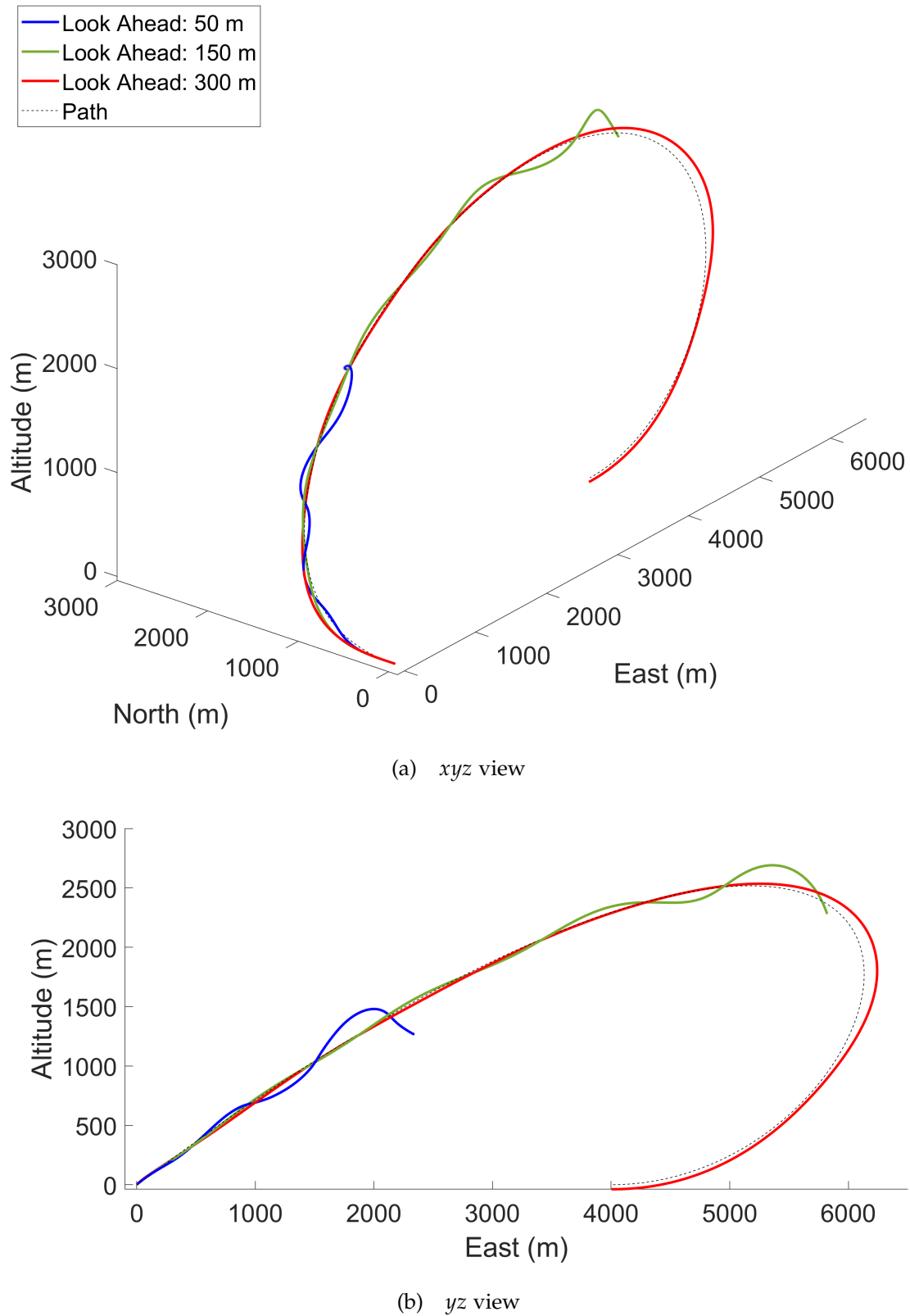


FIGURE 5.18 Look ahead distance comparison for flight simulation conducted at 300 m/s.

paths with an altitude changing component. For each analysis the simulation was completed with various look ahead values. Conditions were imposed that would prematurely end the simulation if the aircraft deviated more than 300 m away from any point along the path. This metric does not influence the look ahead value, since it is observed from any point along the path. The look ahead value can exceed this metric provided the aircraft remains within the defined path tolerances.

It can be seen from Figure 5.18 that when using look ahead distances of 50 m and 150 m, the aircraft exhibits similar position changes to that of small look ahead distances from Figure 5.17. During these simulations, the amplitude of the oscillatory motion expands over time, indicating unstable dynamics. Although both simulations prematurely ended due to the path distance conditions, it can be expected that should the simulation have been allowed to continue, the dynamics of the aircraft would have become unstable.

Although not demonstrated in Figure 5.18, when using a look ahead distance of 200 m, the aircraft was successfully able to complete the path with a similar flight path when using a look ahead distance of 300 m. However, it was opted to use a distance of 300 m as the look ahead distance, since a size of 200 m brought the look ahead distance within close proximity to a failed solution of 150 m. In addition, since there was very little deviation between the two, using the larger look ahead distance does not negatively impact guidance, and was demonstrated to produce a robust solution for numerous path configurations.

5.3 Summary

Throughout Chapter 5, the path planning formulation was created. This formulation determined that to satisfy the orientation constraints as well as the dynamic constraints, a fourth- or fifth-order Bézier curve is required. Additionally, it was demonstrated how deriving the curvature profile for a path allowed dynamics such as the g-force and turn rate to be established. These quantities have known state dynamic limits for the flight physics model, meaning that paths can be analytically determined as to whether or not they are dynamically feasible. Finally, a pure pursuit guidance law was implemented into the

autonomous flight control system. This guidance law enables the aircraft model to autonomously fly paths using a bank-to-turn manoeuvre.

During Chapter 6, a fitness function is developed allowing control point optimisation to be performed for constant-altitude flight conditions. This control point optimisation algorithm is used to train a deep neural network, allowing online path re-planning to be performed using control point classification.

Chapter 6

Constant-Altitude Path Planning

Having defined the path planning algorithm and guidance control system in Chapter 5, path optimisation is performed in Chapter 6 for constant-altitude path planning. This incorporates the development and testing of an evolutionary algorithm for locating the optimal placement for the control points of a fourth-order Bézier curve. This evolutionary algorithm is used to train a classification deep neural network, allowing path re-planning to be performed online. The robustness of performance of the network is demonstrated during a static and dynamic waypoint analysis on the aircraft model.

6.1 Introduction

During manned flight, the pilot of the vehicle decides how best to navigate from one point in space to another, using human intuition, knowledge of the operating conditions, experience collected through training or experience from having dealt with similar situations in the past. Removal of the human element requires path planning to be performed using algorithmic techniques, where the algorithm needs to also consider similar factors as the human pilot to arrive at a particular output. For example, in certain scenarios, it may be advantageous to increase the potential energy of the aircraft by increasing the altitude. Alternatively, depending on the vehicles dynamics, flight conditions and proposed manoeuvre, the vehicle may have to reduce altitude and in the process increase kinetic energy to successfully complete a desired manoeuvre. These are factors that require careful consideration, which a human operator would understand through training and prior experience. To simplify the path planning problem

it was initially assumed that each path would be conducted with a constant-altitude. This eliminated the need to initially encompass an energy evaluation and reduced the complexity of the solution.

Since potential energy remained constant ($\Delta PE = 0$) during the constant-altitude analysis, paths were analysed dynamically using the same curvature analysis defined in Chapter 5, which focused on computing the radius of curvature to derive to dynamic quantities, such as the translational acceleration (a) of the path, required turn-rate (ω), and gravitational acceleration (a_g), all of which are analysed as a piecewise operation at discrete intervals along the path.

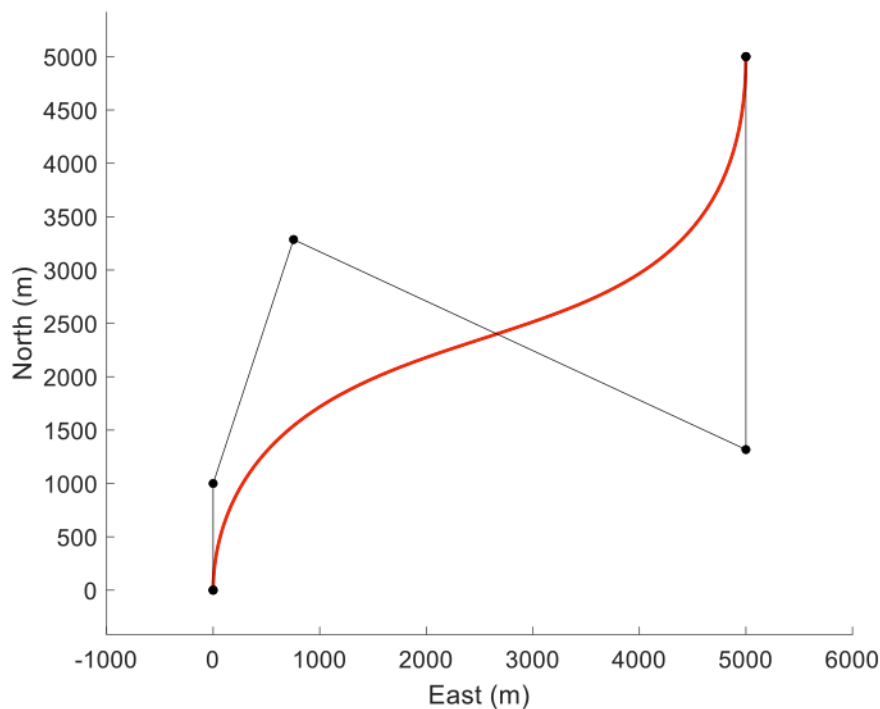


FIGURE 6.1 Control points optimisation. The red line indicates the optimal solution with the black line representing the control points and convex hull

Consider the following scenario as depicted in Figure 6.1. The aircraft is located with a position coordinate (0 km, 0 km, 0 km) in the inertial frame with pose of $(0^\circ, 0^\circ, 0^\circ)$. A waypoint target has been specified which is located in the inertial frame at (5 km, 5 km, 0 km) and has an pose of $(0^\circ, 0^\circ, 0^\circ)$. This gives the constant-altitude path planning problem a total of 12 variables required to define the position and pose of both the aircraft and the waypoint target. However, since changes to the inertial position are only considered along the x_i

(north) and y_i (east) axes for the both the initial position and waypoint position, and the pose is determined solely by the azimuth, this reduces to a total of six input variables required to obtain a solution. An additional input variable for airspeed could also be considered, but for reasons which will become apparent later in the chapter, changes to the airspeed were also neglected.

It is known from Section 5.1 that a fourth- or fifth-order Bézier curve is required to adequately satisfy the path orientation constraints as well as the dynamic and kinematic constraints. Regardless the order of the polynomial, it was established that to satisfy the orientation constraints, the first and final internal control points are vector constrained along the heading vectors of both the aircraft and waypoint target. This formulation leaves either one or two intermediate control points, which can be placed anywhere in the inertial frame to further manipulate the curvature.

The two vector constrained control points require a single scaling function to determine their placement, whereas each of the intermediate control points requires three scaling functions to describe their inertial axis location along the the x , y and z axes. However, for constant-altitude path planning it was assumed that the altitude remained constant throughout the duration of any path, meaning the z -axis scaling factor for each intermediate control point was set to zero. When considering a fourth-order polynomial, this gives a total of four scaling functions required to compute the solution to the Bernstein polynomial, with six scaling functions for a fifth-order polynomial. It is important to note that despite the z -axis scaling factor being set to zero for constant-altitude path planning, the solution to the Bernstein polynomial remains three-dimensional.

Locating the optimal placement of the internal control points is achieved using meta-heuristic optimisation, similar to optimisation of the PID controller gains in Chapter 4. However, before optimisation can be performed, a bespoke fitness function was developed, allowing paths to be quantitatively measured based on the required path dynamics.

6.2 Fitness Function

For operation of any dynamic system, abrupt and unwarranted changes to the dynamics result in a loss of kinetic energy either through induced friction, drag or both. An optimal solution produces a path with a free-flowing curvature profile, ensuring that the profile of the path is smooth and fluent. Achieving a path which is free-flowing, smooth and fluent ensures that paths contains minimal deviations to the acceleration profile, reducing the impact of dynamic loading on the aircraft.

To achieve this, a geometric analysis, similar to the analysis in Section 5.1.4, was conducted solely in the x - y plane. This geometric analysis uses the first and second derivatives of the path in both the x and y axes to compute the curvature at any point along the path, as seen in Equation 6.1.

$$\kappa = \frac{\dot{x}\ddot{y} - \dot{y}\ddot{x}}{(\dot{x}^2 + \dot{y}^2)^{(3/2)}} \quad (6.1)$$

Although knowledge of the curvature profile for a path is not a meaningful quantity in it's current state, taking the reciprocal of the curvature results in the radius of curvature at any point along the path. The radius of curvature is an important variable and is used to determine the acceleration profile and angular velocity profile of the curve based on a specific airspeed. Furthermore, dynamic factors such as the banking angle and aircraft load factor can also be derived from the radius of curvature.

For a path to be dynamically achievable and free-flowing, it is assumed that throughout the duration of the path, the aircraft should be able to manoeuvre along the path at a near-constant velocity. By achieving a near-constant velocity, this indicates that the path dynamics are achievable and not unnecessarily demanding given the current state dynamics. In addition, by considering a constant velocity over the entire path, the required turn rate is determined. Knowledge of the turn rate is essential in determining whether a path is dynamically feasible, since this quantity can be directly correlated to the known performance characteristics of the aircraft. Hence,

$$\omega = \frac{V}{r} \quad (6.2)$$

During optimisation of the pitch rate controller in Section 4.4.3, a maximum sustainable set of pitch rate dynamics was established at various levels of flight. Although the determined dynamics correlate to the maximum sustainable pitch rate that is achievable for a short period, sustaining these maximum pitch rates for prolonged periods is dynamically demanding and not always dynamically sustainable. Since the aircraft is a highly manoeuvrable fast jet which can sustain high instantaneous turn rates, these turn rates increase the angle of attack quite rapidly, whilst the velocity vector of the aircraft is unable to sustain high rotation. This increases the drag force acting on the aircraft and reduces the airspeed. Although comparing the required turn rate of a path against the maximum sustainable pitch rate does define dynamic feasibility, it is not an ideal measurement since the prolonged effect of the dynamics cannot be guaranteed. For this reason, a conservative maximum pitch rate of 20 deg/s was implemented onto the fitness function as a dynamical limit. If the geometric properties of the path indicate a turn rate that exceeded this limit, a penalty factor was imposed, which scaled proportionally to the maximum pitch rate of the path. This does not limit the aircraft to a maximum turn rate of ± 20 deg/s, but rather favours paths that do not exceed this limit if possible. This value was selected as during optimisation of the pitch rate controller in Section 4.4.3, it was determined that this angular velocity was achievable above 100 m/s for all altitude levels.

$$\omega_{penalty} = \begin{cases} 0 & \text{if } |\omega_{max} \leq 20^\circ/s| \\ 2\omega_{max} & \text{if } |\omega_{max} > 20^\circ/s| \end{cases} \quad (6.3)$$

To measure curvature continuity, the path should have an average angular velocity demand that closely resembles the aircraft maximum angular velocity along the path. For example, if the average angular velocity along a path is 5 deg/s with a maximum of 7.5 deg/s, the difference between the average and maximum is relatively small. This indicates that the curvature profile is continuous and large accelerations throughout the duration of the path are not experienced. However, if the average angular velocity is 5 deg/s with a maximum of 20 deg/s, although the path remains within the turn rate limit, this indicates that at some region along the path the radius of curvature has decreased and curvature continuity has not been achieved. In that event, a secondary penalty

function is applied that measures the absolute difference between the maximum and average turn rates. Therefore, Equation 6.3 becomes:

$$\omega_{penalty} = (|\omega_{max}| - |\omega_{avg}|) + \begin{cases} 0 & \text{if } |\omega_{max}| \leq 20^\circ/s \\ 2\omega_{max} & \text{if } |\omega_{max}| > 20^\circ/s \end{cases} \quad (6.4)$$

Further dynamic restrictions are applied by determining the gravitational acceleration induced onto the vehicle. Since the computational model is representative of the F-16 Fighting Falcon, which is a fast and highly manoeuvrable combat aircraft, it can be expected that structurally the aircraft has not been over-engineered as this would impose unnecessary weight. For this reason, a 9g acceleration limit was imposed onto the fitness function. Similarly to the turn rate penalty, a g-force penalty was imposed if the acceleration at any point along the path exceeded 9g. The penalty for exceeding the g-force limit was scaled such that the magnitude represented a similar penalty weighting for exceeding the maximum angular velocity.

$$g_{penalty} = \begin{cases} 0 & \text{if } |g_{max}| \leq 9 \\ 5g_{max} & \text{if } |g_{max}| > 9 \end{cases} \quad (6.5)$$

Currently, the fitness function produces paths where the difference between the average angular velocity and maximum angular velocity tends towards zero, provided the angular velocity and g-force limits are not exceeded. This condition is satisfied by extending the length of the path towards infinity as this increases the radius of curvature and reduces dynamic loading. To ensure that this does not occur, a trade-off between increasing the length of the path and decreasing dynamic loading was identified. A penalty weighting was applied that measured the ratio between the total overall length of the path against the euclidean distance between the aircraft and waypoint. This can be seen in Equation 6.6 where P_{Total} is the overall cumulative length of the path with P_i and P_f the initial and final points on the path.

$$P_{penalty} = \frac{5 \times P_{Total}}{(P_i^2 + P_f^2)^{1/2}} \quad (6.6)$$

Summing all of the penalty weightings allows a quantitative measure of any path, where a lower metric indicates a better path fitness. Therefore, the

overall fitness of a path is described as:

$$F = \omega_{penalty} + g_{penalty} + P_{penalty} \quad (6.7)$$

The applied fitness function does not allow paths to have a fitness of zero. Consider the following scenario. The aircraft is located at (0 m, 0 m, 0 m) with an orientation (0°, 0°, 0°), and the waypoint target is located at (3 km, 0 km, 0 km) with an orientation (0°, 0°, 0°). Since there is no change in orientation and the path only changes for coordinates in the x axis, this path can be solved as a straight-line trajectory. The curvature profile for a straight line is zero, which means the radius of curvature is infinite. This demands an angular velocity of 0 deg/s throughout the entire path, meaning that $\omega_{penalty}$ is 0 and therefore $g_{penalty}$ is 0. However, since $P_{penalty}$ is calculated as the ratio between the total path length and the magnitude between the two points, this ratio becomes 1:1. Since this ratio is then scaled by a factor of five (Equation 6.6), $P_{penalty}$ is five for a straight-line trajectory. This is the minimum fitness value attainable for this fitness function.

When considering the path profile generated in Figure 6.1, a breakdown of the individual path fitness weights are as follows:

- The cumulative length of the path is 7668.9 metres with a point-to-point magnitude of 7071.1 metres.
- When travelling with an airspeed of 250 m/s, the average turn rate for the path was 4.72 deg/s with a maximum turn rate of 7.41 deg/s. This is well within the dynamic capability of the vehicle, meaning that no additional penalties are applied based on the turn rate analysis.
- The maximum aircraft acceleration during the manoeuvre was 3.3g, which is within the aircraft's capabilities.

When applying these weightings to Equation 6.7, the cumulative fitness of the path is determined to be 8.1128.

$$F = \frac{5 \times 7668.9 \text{ m}}{7071.1 \text{ m}} + \left| \left(7.41 \frac{\text{deg}}{\text{s}} - 4.72 \frac{\text{deg}}{\text{s}} \right) \right| + 0 = 8.1128 \quad (6.8)$$

From the analysis of the path seen in Figure 6.1, the dynamics of the path are quite low. This is mainly due to the euclidean distance between the position and

waypoint being relatively large when compared to the airspeed. However, as the initial position and waypoint target are located closer together, the dynamics of the path begin to increase, and if they approach the imposed dynamic limits, the fitness function drives the optimiser to search further away, establishing a trade-off between increasing the dynamics of the path and increasing the length of the path.

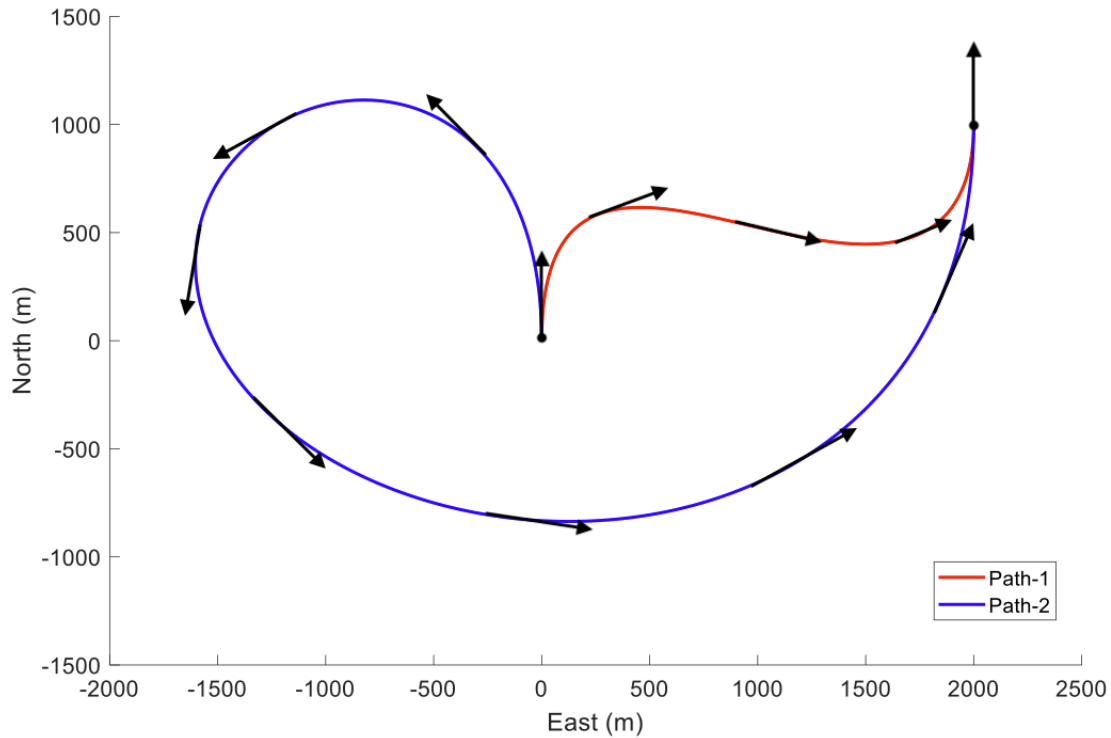


FIGURE 6.2 Path fitness comparison between a short and unfeasible path and a long and feasible path

This can be seen in Figure 6.2 where the euclidean distance between the initial position and waypoint target is only 2.2 km. Although the path labelled as *Path-1* follows a similar profile to the path shown in Figure 6.1, by pathing directly to the waypoint at an airspeed of 250 m/s the dynamic limits of the aircraft are exceeded, with a maximum angular velocity of 44 deg/s and a maximum acceleration of almost 20g. As a result, large penalty factors are imposed giving the path a fitness metric of 218.

Due to the poor fitness, the optimiser expands the location of the control points away from the origin, increasing the length of the path and relieving the flight dynamics. This begins a trade-off between how much the path length in-

creases relative to ensuring the dynamics are feasible. The optimiser eventually settles on the solution labelled *Path-2*. This path incorporates what is denoted as a dynamic deviation. A dynamic deviation occurs when the solution does not path directly from the position to the waypoint, but rather performs an intermediate phase, where the direction of travel moves away from the waypoint.

By initially moving away from the waypoint, the path increases the distance between itself and the waypoint target, allowing the dynamics of the path to be alleviated by performing a more relaxed manoeuvre. The resultant path is now within the dynamic limits of the aircraft with a maximum angular velocity less than 20 deg/s, but in doing so has increased the length of the path by three times when compared to *Path-1*.

It is also important to note that the path labelled as *Path-2* has two identical solutions that are mirror images of one another conducted about both axes due to symmetrical dynamics, meaning that during a dynamic deviation, two global solutions exist.

6.3 Control Point Optimisation

During optimisation of the angular rate and airspeed controller in Chapter 4, various optimisation algorithms were introduced. This included an overview into various literature sources as well as in-depth analysis performed on a series of low-dimensional optimisation test functions. This analysis concluded that the bat algorithm (BA) was able to converge more quickly than the genetic algorithm (GA) whilst achieving a near identical convergence accuracy. This conclusion led to the use of the BA for optimisation of the PID controllers applied to the aircraft's flight control system in Chapter 4, which governed all autopilot systems applied to the computational model.

During Section 5.1, it was explained that for constant-altitude path planning, four scaling functions are required to solve a fourth-order Bézier curve, with six required for a fifth-order. In both configurations, the first two scaling functions represent the distance the first and final control points are located along their respective heading vectors, whilst the additional scaling functions represent the location of the intermediate control points in three-dimensional space. When

performing control point optimisation for constant-altitude path planning, it was determined that the increased dimensionality impacted the rate of convergence for the BA, with solutions having poor repeatability. The BA struggled with exploration of the higher dimensional search space, resulting in premature convergence onto non-global solutions. This indicated that the BA was not a suitable optimisation algorithm to solve the path planning problem due to the increased dimensionality of the problem. For these reasons, it was determined that the GA was a more suitable optimisation algorithm for the purpose of path planning.

When deciding between a fourth- or fifth-order Bézier curve, it was logical to try to develop a solution surrounding a fourth-order polynomial due to the reduced computations required. Computing the solution to the Bernstein polynomial is an iterative process based on the number of polynomial coefficients. The number of coefficients increases proportionally with the order of the polynomial, which is determined by the number of control points. When considering path optimisation, the number of internal control points defines the dimensionality of the problem. For a GA each individual within the population is comprised of genes, where the number of genes is equal to the dimension of the problem. For a fourth-order polynomial, this creates individuals with four genes and six genes for fifth-order polynomials. The number of genes results in quite large computational overheads, particularly when performing crossover and mutation. Consider a four-dimension problem where two individuals within the population have been selected for reproduction. Individual number one is made up solely of zeros with individual number two made up solely of ones, as shown below.

$$\text{Individual 1} = [0, 0, 0, 0] \qquad \text{Individual 2} = [1, 1, 1, 1]$$

Whilst crossover can be performed in many ways, one popular approach is to create a binary matrix of all the possible gene combinations between the two individuals, as illustrated in Equation 6.9.

The number of offspring produced under this methodology is 2^n , where, n is the total number of genes. For a fourth-order polynomial, the number of offspring produced by each couple per generation is 16. When compared to a fifth-order polynomial, the total number of offspring combinations increases by a factor of four to 64. This creates large overheads associated with increasing the

dimensionality of the problem, since each couple selected for reproduction is capable of generating up to four times as many offspring. Although a threshold can be applied to limit the number of offspring produced per couple, this adds an additional level of randomness to the GA. For these reasons, the constant-altitude path planning optimiser focused on developing a solution surrounding a fourth-order Bézier curve.

$$\text{Offspring} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ \vdots & \vdots & \vdots & \vdots \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix} \quad (6.9)$$

Throughout constant-altitude path planning optimisation, the only constraint applied to the GA was associated with the two vector-constrained orientation scaling functions. These two functions are not free-to-move and must be positive values to ensure the path is correctly orientated. In addition, these two scaling functions cannot approach zero as this would indicate a straight-line trajectory, dissatisfying the orientation constraint. To prevent this, a minimum value was applied. This minimum value was based on the euclidean distance between the aircraft's position and the location of the waypoint target. By scaling the minimum value in this manner trajectories that had large euclidean distances produced a larger orientation scaling function, and vice versa.

When generating the initial population for the GA, it was found that performing cell mitosis increased the repeatability of the results and aided in convergence towards a global solution when compared to simply randomising the population. Cell mitosis was performed by initialising a single individual containing a default set of genome values. These default values were based on the calculated minimum value applied to the first and second scaling functions. This individual was duplicated 100 times, with each duplication containing a small mutation of ± 1 applied to each gene.

Throughout optimisation, individuals were selected for reproduction using a tournament selection process. Each tournament was filled randomly with 10 individuals, with the winner of the tournament selected as the individual with

the best fitness value. The winner entered the reproduction pool and was paired with the winner of the subsequent tournament, with the remaining tournament population returned to the general population. This process was repeated until the reproduction pool had been filled with a total of 50 individuals.

Reproduction, or crossover, was performed by producing a binary matrix of all possible gene combinations between each of the applied couples. Since each individual was comprised of four genes, each couple had the potential to create 16 offspring. Since couples were formed based on a randomly populated tournament selection process, there was a reasonable chance that one of the parents may have a poor fitness when compared to the population's fittest individual. Therefore, pairing of two individuals does not guarantee that the resultant offspring will generate fitter solutions. Hence, an exact replica of each parent was also introduced into the offspring before being applied with a small mutation. By performing a slight mutation of each parent, the population receives further diversification and increases the probability of convergence occurring towards a better solution in the next generation.

By implementing mutated clones of the parents into the offspring, the total number of offspring produced by each couple increased from 16 to 18 per generation. Mutation was applied to each of the offspring's genes with a probabilistic chance of 90%, scaled with an initial mutation range of 0.5. To aid in convergence, the mutation range decayed proportionally each generation by a coefficient of 0.8. However, to prevent the population's diversity from becoming too stagnant, a minimum mutation threshold of 5×10^{-3} was applied. After performing crossover, the population increased from 100 individuals to 550, as shown in Equation 6.10.

$$\begin{aligned} \text{Population Size} &= 100 + (\text{Offspring Size} \times \text{Reproduction Pool Size}) \\ &= 100 + \left(18 \times \frac{50}{2}\right) = 550 \end{aligned} \quad (6.10)$$

At the end of each generation, the same tournament selection process was applied to determine which individuals would be retained and proceed onto the next generation, reducing the population size from 550 back to 100. This concluded a single optimisation generation, which was repeated until 200 generations had expired, or until one of the following stop criteria were achieved:

(i) standard deviation of the population dropped below 5×10^{-3} ; (ii) best fitness failed to improve more than 1×10^{-2} for 10 consecutive generations.

Testing of the constant-altitude path planning optimisation algorithm was conducted by applying the GA to the computational model's path generation framework. At the start of each simulation, both the position and orientation of the aircraft and waypoint target was randomly generated. For the purpose of testing the optimiser, the location of the position and waypoint was limited to ± 10 km along both the inertial x (north) and y (east) axes. In addition, whilst testing the optimiser, the initial airspeed of the aircraft was held constant at 200 m/s, with the airspeed controller set to maintain this airspeed for the duration of the simulation.

The position and orientation vectors for the aircraft and waypoint target were applied as inputs to the GA, with optimisation performed to determine the optimal control point scaling functions prior to the simulation commencing. Upon completing the optimisation process, the path was generated and applied to the guidance and navigation flight control system. If the aircraft was successful in making it to the end of the path, the solution was deemed to be dynamically feasible. Additionally, upon obtaining the optimised solution, an estimated completion time was computed based on completing the path with a constant airspeed. If the aircraft was successful in reaching the end of the path without exceeding the estimated completion time by more than 10%, the solution was deemed to be optimal since this indicated that the optimised path was achievable with a near constant airspeed. After running 5,000 simulations, the optimiser achieved an accuracy of 100% when developing dynamically feasible paths and over 99.7% of solutions were deemed optimal by satisfying the completion time constraint.

Although the optimiser was successful in producing dynamically feasible paths for every tested scenario, on average the optimisation process took four seconds* to obtain a solution. This posed numerous issues when considering cooperative flight and online path re-planning applications. During cooperative flight missions, the waypoint target is dynamic and moving, meaning that periodic path re-planning is a requirement. However, path optimisation using the proposed optimiser is too computationally expensive, and prevents online path

*AMD Ryzen 2700X CPU clocked at 3.70 GHz running MATLAB R2018a

re-planning during real-time simulations.

During testing of the optimisation algorithm, it was discussed how the population was initialised using mitosis based around a default set of values. However, as optimisation was performed for more solutions, a database of observations and solutions was developed. This allowed the population to be initialised using previously optimised solutions, determined using a nearest neighbour algorithm based on historic observations. By initialising the population based on historic solutions, the initial stochastic behaviour of the algorithm became less effective, since the population was densely populated around a specific region of the search space as opposed to sparsely populated throughout the entire space. As a result, solution convergence occurred much more rapidly and the GA required smaller populations and reproduction pool sizes to converge onto the global solution.

This reduced the computational complexity with the optimiser able to converge on to a solution after approximately 0.5 seconds[†] as opposed to four seconds. Whilst this methodology has not necessarily enabled path planning and re-planning to be performed during real-time simulations, it can be expected that as more and more optimisations become available, the time required should continue to reduce, making online optimisation a potentially viable solution. However, one alternative method to performing online optimisation was to use the optimisation algorithm and solutions to train a deep neural network, allowing data classification to be performed online to obtain a solution to unknown queries.

6.4 Classification Neural Network

During the previous sections, a fitness function and path optimisation algorithm was developed, allowing optimal placement of the internal control points for a fourth-order Bézier curve during constant-altitude flight. However, due to the computational complexity of the optimisation algorithm, online path planning using this methodology was not feasible. Therefore, throughout Section 6.4 artificial intelligence is discussed and implemented in the form of a classification

[†]AMD Ryzen 2700X CPU clocked at 3.70 GHz running MATLAB R2018a

neural network. By training a neural network using optimised data, unknown observations can be classified as known solutions, allowing the network to predict solutions and allow paths to be re-planned online.

A classification neural network is a predictive neural network used for feature categorisation [136], where the quantity being modelled can be described as a binary variable located in a discretised space [137]. For reasons which will be apparent during Section 6.4, the operating environment for constant-altitude flight has a relatively small number of dimensions, enabling the environment to not only be discretised, but also tractably analysed at each discrete location. This enables accurate predictions to be performed online, where the waypoint target for flight is described by the network as a specific class with a corresponding set of control points obtained through optimisation. Because a classification network requires a discrete domain, a high level of accuracy can be obtained through supervised learning when compared to other learning methods [136]. However, the network can only perform classification against classes it was trained on, with training data requiring an exhaustive labelling system [138]. In addition, classification neural networks can be susceptible to overfitting, which could produce unrealistic categorisation [139]. Therefore, performing an overview of class interval sizing was important to obtaining realistic and appropriate classifications.

Training of the neural network was performed using supervised learning, meaning that a training dataset was obtained. This dataset was constructed by performing optimisation at discrete intervals along the inertial frame x_i (north) and y_i (east) axes, as well as for heading orientation angles. Since this discrete dataset was required to cover the entire operating domain, the interval sizes between each classification needed to be small enough that the classification adequately satisfied the dynamics, but not small enough that the optimisation process becomes exhaustive and unfeasible.

The aircraft model is governed by six non-linear equations of motion that represent body-axis translational and rotational accelerations. These body-axis accelerations are translated to an inertial axis system using a set of kinematic quaternion transformation matrices. Since the inertial axis system has virtually infinite dimensions along the x , y and z axes, it is impossible to perform op-

timisation and train a classification network that covers the entire inertial space. This led to the application of an intermediate reference frame (IRF) which was spatially bounded and allowed path planning optimisation to be exhaustively performed. Since this network focused on constant-altitude flight, the altitude level was not considered, meaning the spatially bounded IRF was considered as a two-dimensional plane located parallel to the x - y plane in the inertial axis system. This allowed optimisation to be performed exhaustively for all locations throughout the IRF, limiting paths to be planned within this frame. However, by considering the aircraft to be fixed at the centre of the IRF, this frame moved through the inertial frame with the same dynamics as the aircraft. This meant that flight could be considered throughout the entire limitless inertial frame provided paths were planned within the moving IRF, as depicted in Figure 6.3.

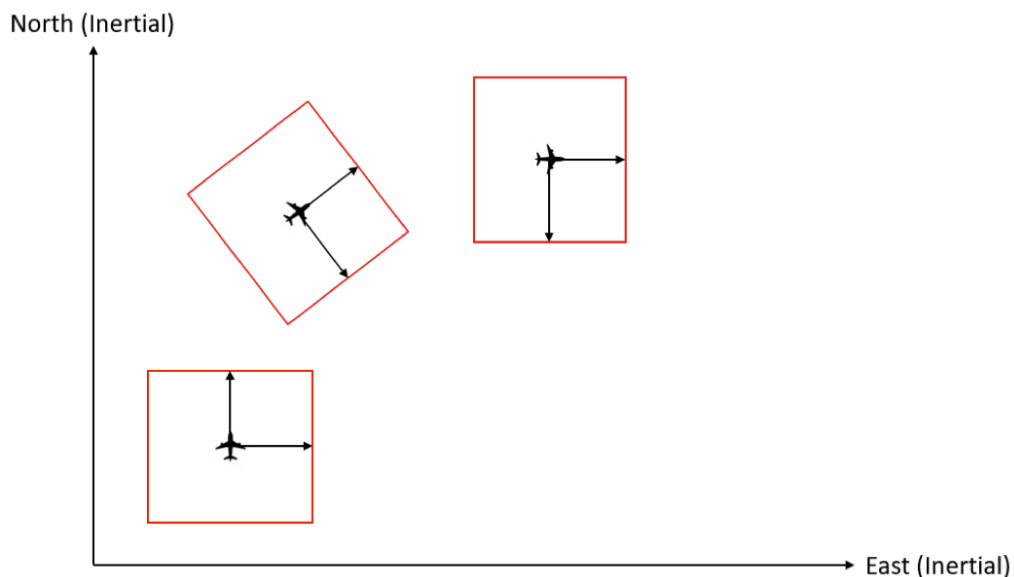


FIGURE 6.3 Path planning intermediate reference frame moving through the inertial axis system with the aircraft fixed at the centre

Since the IRF moves through the inertial frame with the same dynamics and orientation as the aircraft, the path planning problem allowed waypoints to be translated to this reference frame and described as relative coordinates to the origin-fixed aircraft. This reduced the number of observations required for optimisation, since the waypoint target was described as relative distances in the x and y axes as well as an absolute heading difference between the aircraft and waypoint, with respect to the IRF. In addition, the aircraft has symmetrical roll

dynamics. Given a bank-to-turn guidance system, any dynamically feasible path in one direction remains dynamically achievable when mirrored about the x axis. This is demonstrated in Figure 6.4. This halved the necessary optimised space, since solutions could be optimised in one half of the spatially bounded IRF and then mirrored to complete the reference space.

To allow exhaustive optimisation to be performed, the spatially bounded IRF was given dimensions of ± 10 km along both the x and y axes, giving a 400 km^2 path planning axis system.

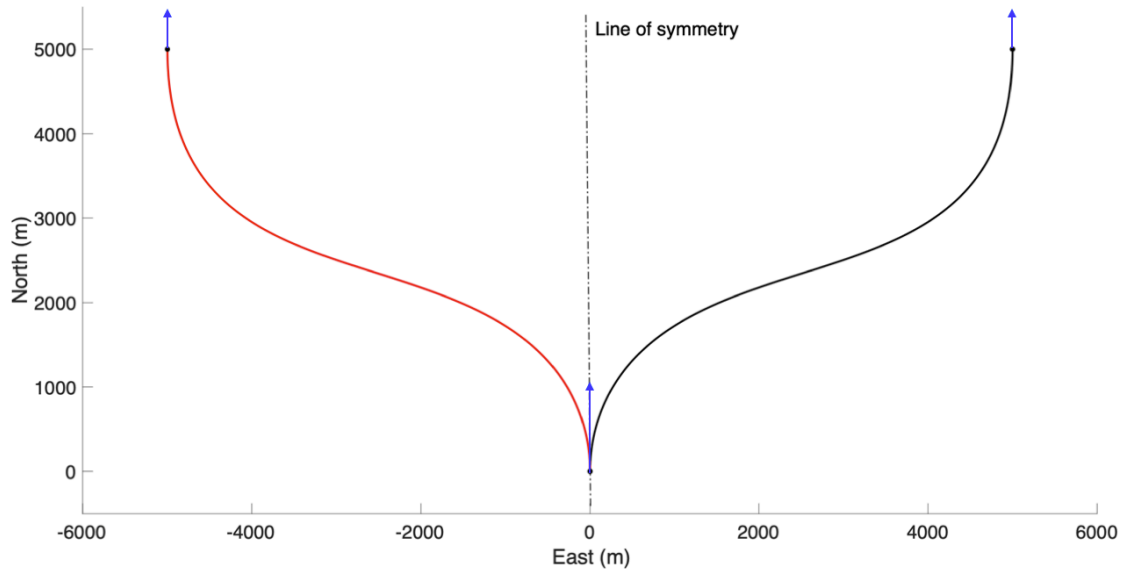


FIGURE 6.4 Constant altitude path plan mirroring

6.4.1 Preliminary Network

Prior to developing the structure of the classification network over the entirety of the IRF, a preliminary study was conducted. This study aimed to determine whether a classification network could be used as a solution to classify path planning data. Therefore, this study was initially conducted on a reduced area of IRF, where the x axis (north) was held constant at 2 km, with optimisation performed every 250 metres along the y axis (east), ranging from 0 to 5 km. In addition, for every interval along the east axis, optimisation was performed for variable waypoint headings every 10 degrees over a range of 0 to 360 degrees. This created a dataset consisting of optimised solutions corresponding to 720 discrete classifications.

Figure 6.5 shows four plots where each plot corresponds to a single path planning control point scaling function. The x axis represents changes to the east coordinate with heading angles described along the y axis. It is important to note that all four plots have different orientations to give the reader a better perspective of the three-dimensional topology for each of the four control point scaling functions.

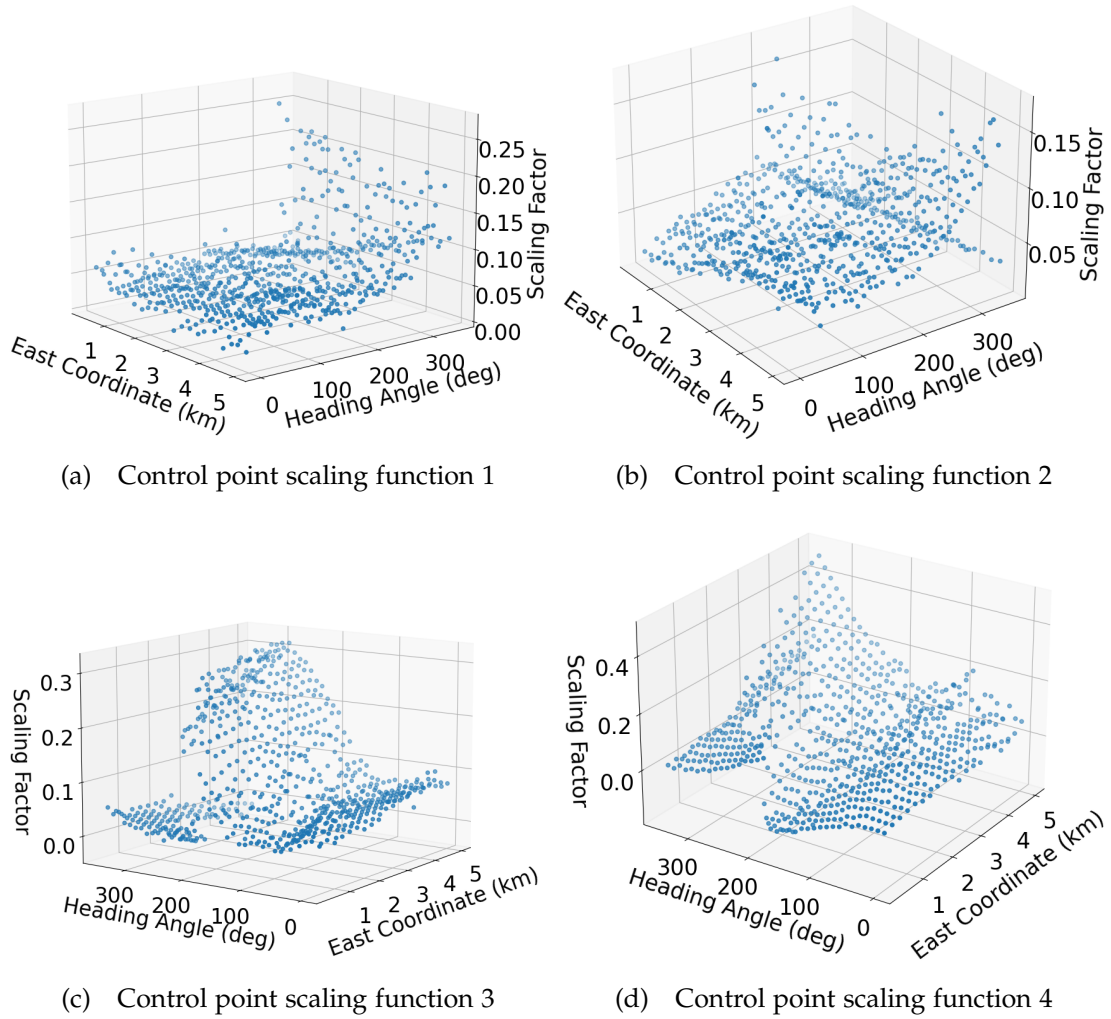


FIGURE 6.5 Control point scaling functions for a north coordinate of 2 km and varying east locations and heading values

Based on the control point scaling data, it is evident that as the heading angle increases the control point scaling functions also tend to increase. Additionally, as the east coordinate tends towards the origin, the control point scaling functions also increase. This occurs as the euclidean distance to the waypoint

decreases, causing an increase in the path dynamics. As a result, the optimiser increases the length of the path by incorporating a dynamic deviation, as discussed in Section 6.2.

The data presented in Figure 6.5 was used to train a classification neural network that contained one hidden layer and 100 fully connected neurons per layer. The network had three inputs, each corresponding to the number of observations, with the output from the network corresponding to all 720 classifications, giving 73,220 trainable parameters within the network. The activation function between each of the connecting layers was a rectified linear unit (ReLU), with softmax classification applied to the output layer. Softmax classification provides an activation value scaled between 0 and 1 to each classification, with the maximum value indicating the network classification for a specific observation. The loss function for the neural network was categorical cross-entropy using the Adam optimiser. After 6000 epochs the network was trained to classify all 720 classifications with an accuracy of 100%. This confirmed that a classification neural network could be trained to classify control point scaling functions for the purpose of path planning.

6.4.2 Observation Interval Size Analysis

During the development of the preliminary network, a dataset obtained through optimisation on a subset of the IRF was used to develop a relationship between the input observations and the magnitude of the four control point scaling factors. This relationship was used to train a classification network, which was able to classify solutions with 100% accuracy, when tested on the training dataset. Although this dataset only occupied a small portion of the IRF, due to intervals along the east axis being 250 metres and heading angle intervals of 10 degrees, a total of 720 classifications existed.

Whilst classifying solutions along the east axis in discrete intervals every 250 metres and the heading angles every 10 degrees might be an appropriate solution, it could be sub-optimal when considering the number of the classifications required. For example, if the size of the intervals between observations is small, the number of classifications increases. This results in a much larger training dataset, not only increasing the number of optimisations but also the size of

the network. On the other hand, if the size of the discrete intervals between classifications is too large, then waypoints located at intermediate observations could result in sub-optimal solutions, since the applied classification is too large. Therefore, an iterative study was performed to determine adequate interval size spacing for control point classification.

North-East Observations

To understand how varying the size of the observation intervals impacted the classification of solutions, an investigation was performed. This investigation initially focused on understanding the impact of varying the interval size for the north and east observations, and was conducted for observations between 0 and 10 km along both the north and east axes. For this initial study, the heading angle of the waypoint target was held constant at 0 degrees, whilst the size of the observation intervals along the north and east axes were set at 1 km, four times larger than the preliminary network. Additionally, optimisation was performed 15 times at each identified classification, allowing a mean solution to be obtained for each classification.

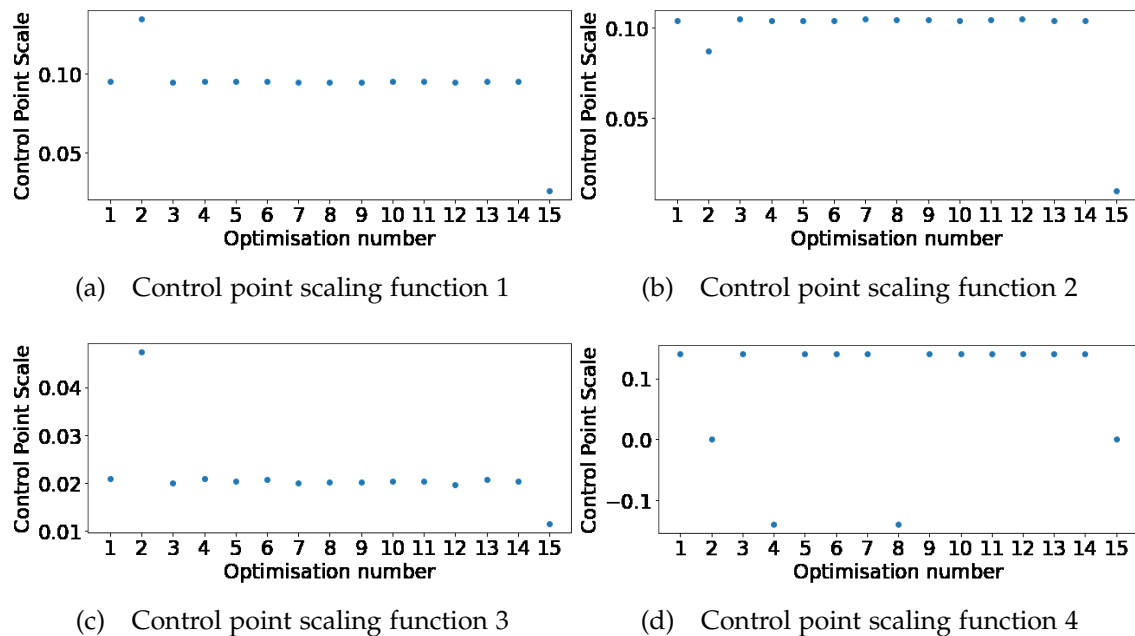


FIGURE 6.6 Control point scaling solutions for north observation of 1 km, east observation of 1 km and heading angle of 0 degrees.

Since optimisation was performed 15 times for each classification, outliers were identified and removed by performing a z-score analysis. A z-score analysis assigns a quantitative grade to each data point based on the mean and standard deviation of the entire dataset. An example of identifying outliers can be seen in Figure 6.6, where 15 optimised control point scaling solutions can be found for a north observation of 1 km, east observation of 1 km, and heading angle of 0 degrees.

Based on Figures 6.6a-c, it is visually apparent that optimised solutions 2 and 15 are outliers, since the control point scaling solutions do not follow the general trend of results. However, when analysing Figure 6.6d, it becomes evident that solutions 4 and 8 are also outliers. It was previously explained that as the euclidean distance to the waypoint decreases, the fitness function drives the optimiser to perform a dynamic deviation. Due to symmetrical dynamics, these dynamic deviations have two global solutions that are mirror images of one another. As the euclidean distance during this example is approximately 1.4 km, a dynamic deviation is expected. Since the fourth scaling function corresponds to the location of the intermediate control point along the y axis, this control point scaling function determines whether a left or right dynamic deviation turn occurs.

From the optimised results, it is evident that 11 of the solutions have positive scales (Figure 6.6d), which indicates a right hand dynamic deviation. Although solutions 4 and 8 have the same magnitude as the other 11 solutions, the sign is different, indicating that these solutions are a mirror image with a dynamic deviation occurring to the left. Since these paths are mirror images, the first three control point scaling solutions are identical for a left turn as they are for a right turn, which was the reason these outliers were not detected in the first three control point scaling solutions (Figure 6.6a-c).

Additionally, solutions 2 and 15 represent solutions where a path is created directly from the position to the waypoint with no dynamic deviation, as indicated by the fact that the magnitude of the fourth control point scaling function is 0. These solutions are most likely dynamically unachievable due to the low euclidean distance, and demonstrate that although the optimiser has good accuracy and repeatability, occasionally due to the stochastic nature of the algorithm,

solutions can fail. This was the reason for performing 15 optimisations as this allows outliers to be identified and removed without impacting the classification accuracy.

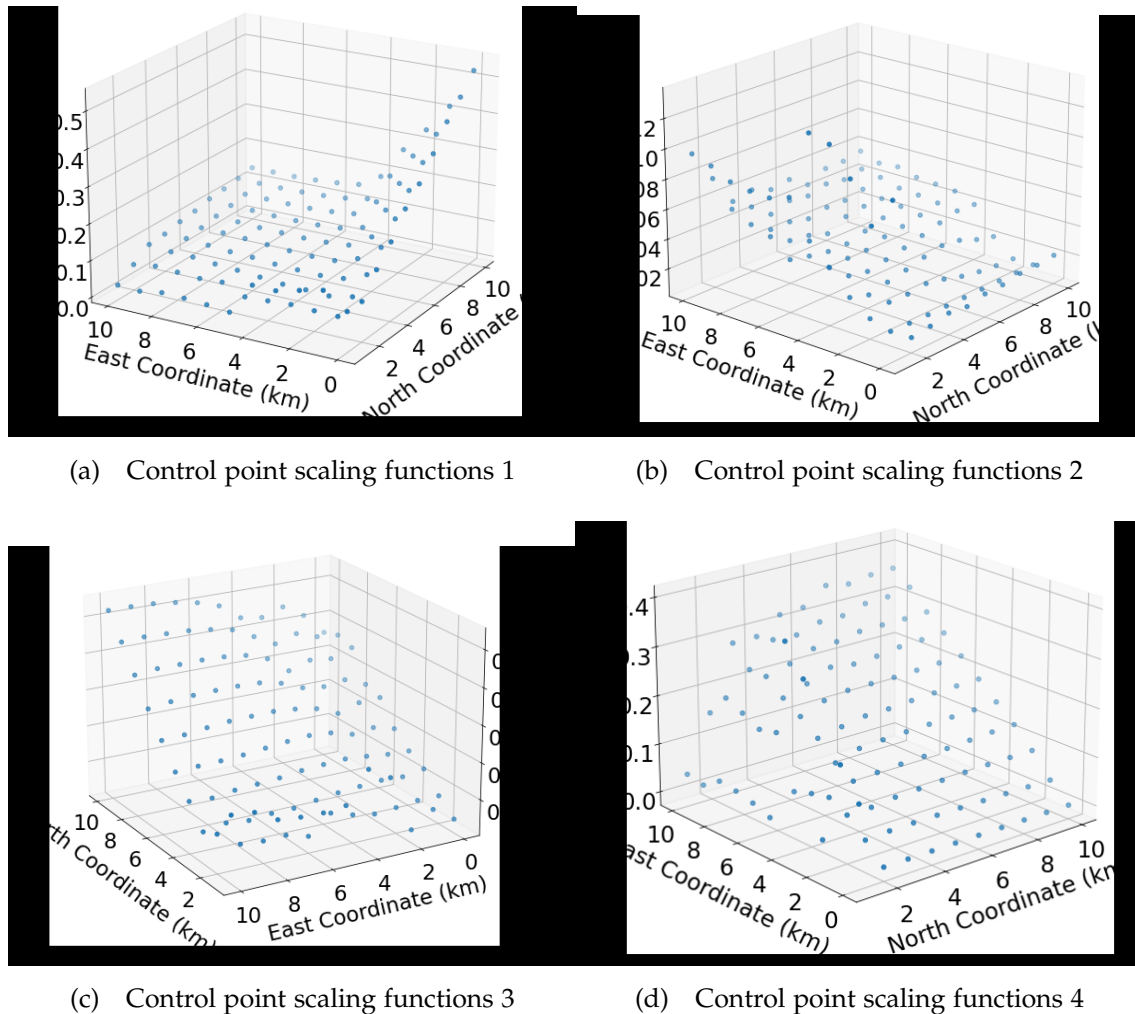


FIGURE 6.7 Optimised control point scaling solutions for the north-east interval study. Optimised was conducted for an airspeed of 150 m/s with the orientation of the waypoint target constant at 0 degrees.

Having performed optimisation for the north and east observations with a heading angle of 0 degrees, and performed a z-score analysis to remove outliers from within the optimised solutions, the four control point scaling functions can be seen in Figure 6.7. Similar to the preliminary network, a relationship is present between observations. As the east observation approaches the origin, the first scaling function expands away from the origin to initiate a dynamic deviation. This creates an almost linear relationship between the north and east

observations for all four control point scaling functions.

Due to the fact that the heading angle was kept constant during this analysis and the size of the intervals increased by a factor of four, the number of classifications decreased from 720 to 110 for the north-east study. This optimised dataset was used to train a classification network, which maintained the same architecture as the preliminary study. After approximately 5000 epochs, the network was able to classify the entire training dataset with an accuracy of 100% and a categorical cross-entropy loss of 0.05.

For this analysis it was important to understand how the classified and optimised solutions changed when the waypoint target was located at intermediate observations. Therefore a waypoint target was located with north observation of 3.7 km and an east observation of 8.2 km. Upon querying the classification network, the following classification was returned for each of the control point scaling solutions, rounded to two decimal places:

$$\begin{array}{ll} f_1 = 0.09 & f_2 = 0.36 \\ f_3 = 0.28 & f_4 = 0.10 \end{array}$$

To compare the classified solution, optimisation was performed for the same waypoint observation. The optimiser returned the following control point scales rounded to five decimal places:

$$\begin{array}{ll} f_1 = 0.08996 & f_2 = 0.35324 \\ f_3 = 0.27356 & f_4 = 0.10492 \end{array}$$

Despite the classification model classifying a solution based on relatively large discrete observation interval sizes of 1 km along both the north and east axes, the mean squared error between the classified and optimised solution was negligible, at 2.3×10^{-4} , when averaged over all four control point scaling solutions. Whilst this does produce a path with slight error when compared to the optimised solution, the measured fitness indicates the path produced based on a classified solution remains dynamically achievable and within the confines of the solutions produced by the optimiser. A visual comparison between the optimised and classification path can be seen in Figure 6.8.

Having established that a north and east observation interval size of 1 km retains accurate classification of intermediate waypoints, the same process was

applied for determining the observation intervals for the relative heading angle of the waypoint with respect to the intermediate reference frame.

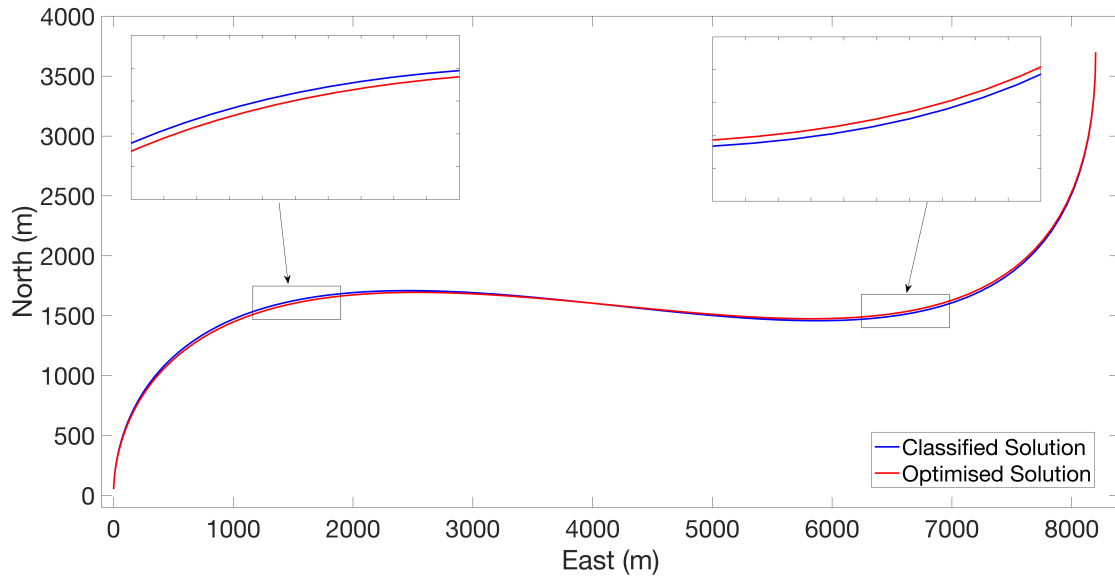


FIGURE 6.8 Intermediate waypoint classification path against the optimised path produced through use of the Genetic Algorithm

Heading Observations

During the development of the north-east observation intervals, optimisation was performed for north and east observations between 0 km and 10 km whilst the relative heading angle of the waypoint remained constant at 0 degrees. By allowing the heading angle to vary, the increased dimension has a profound impact on the number of classifications. For example, if the heading angle was to have intervals every degree between a range of 0 to 360, the number of classifications increases from 110 to 39,490. Additionally, since optimisation was performed for each discrete observational north and east location 15 times, a complete dataset under this configuration would require almost 600,000 optimisations. This is not a feasible number of optimisations to determine the adequate heading angle interval size.

When querying a classification network, the network analyses the input and determines which of the known classes best represents these input observations. For simple and low dimension problems, the classification process can be considered as a nearest neighbour algorithm, where classification is performed by

determining which class is closest to the input observation. As the dimensionality of the problem increases, the classification process becomes more complex. However, to analyse how changing the heading interval size impacts the resultant Bézier curve, the north and east observations were held constant, creating a one-dimensional problem and allowing artificially classified solutions to be analysed.

Consider a waypoint target with a north observation of 4 km, an east observation of 6 km and a heading angle observation of 120° . When performing path optimisation for these input parameters, the optimiser returns the following control point scaling solutions:

$$\begin{array}{ll} f_1 = 0.12803 & f_2 = 0.072111 \\ f_3 = 0.19613 & f_4 = 0.088718 \end{array}$$

Since this problem is currently defined as a one-dimensional problem where only the heading angle is allowed to vary, if a neural network was trained on a dataset optimised at heading intervals every 1 degree, the largest classification offset from this network would be ± 0.5 degree. For example, if this network was queried for a heading angle of 119.5 degrees, this would return a classification corresponding to observations of 120 degrees, and would return the four scaling solutions defined above. However, if the observation had a heading angle of 119.4 degrees, this would return the classification for the optimised solution at 119 degrees. In addition, if this same network was trained for heading intervals every 10 degrees, the classification offset increases to ± 5 degrees. This means that any query with a heading observation between 115 and 125 degrees, would return the classification solution for 120 degrees, and again, returns the scaling functions identified above. Understanding the difference between classifications and how this impacted the resultant path was the main focus on this study

Using this knowledge, the process of classification was artificially simulated by performing optimisation at various classification offsets based on observation intervals. For example, if the observational interval size was 20 degrees, then optimisation was performed at 110 degrees as this represents the largest classification offset for the network. The obtained scaling point solution was re-applied as the solution for an observation of 120 degrees, with the resultant paths displayed in Figure 6.9. The resultant path was then compared directly to

the optimised solution at 120 degrees, as indicated above. This process was performed for interval sizes of 1, 10, 20, 30 and 40 degrees, with the results shown in Table 6.1.

TABLE 6.1 Constant altitude heading size interval analysis

Interval Size (deg)	Classification Offset (deg)	Optimisations Requirement	Total Classifications	MSE (metres)	Path Fitness
1	0.5	359	39,490	7.40	6.1820
10	5.0	35	3,850	134.47	6.5046
20	10	18	1,980	237.96	6.8518
30	15	12	1,320	356.47	7.2364
40	20	9	990	502.28	7.6554

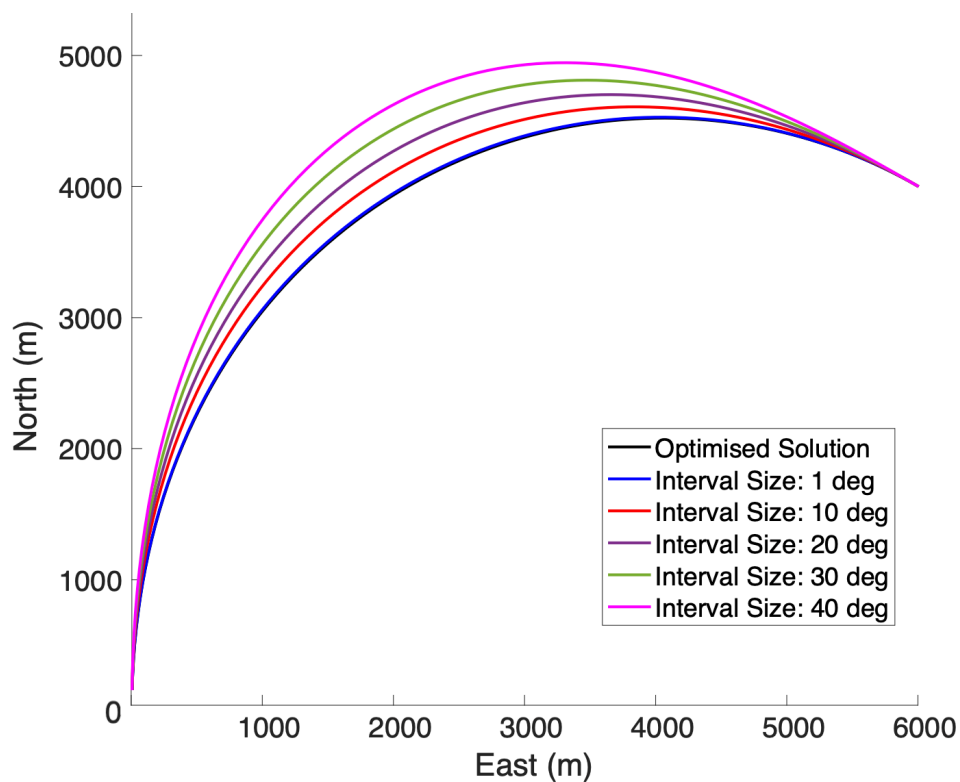


FIGURE 6.9 Comparison between the optimised solution and the output path based on different heading angle intervals.

Using the optimised solution for an observation of 120 degrees, the mean-squared-error (MSE) was determined against the paths produced from the arti-

ficially classified solutions. These MSE values represent the summed absolute difference between the two paths and are displayed in Table 6.1. Based on the analysis, the magnitude of the MSE correlates with the interval size, meaning that as interval size increases, the error against the optimal solution expectedly increases.

By performing optimisation with an offset of 0.5 degrees, a MSE of only 7.4 metres was experienced when compared to the optimal solution. This proves that for an interval size of 0.5 degrees, the error between the classified solution and optimal solution is negligible, with the resultant path near-identical to the optimised path in Figure 6.9.

Despite the low error, the number of optimisations required for this interval size is 359 heading observations for every single north/east observation. It has already been discussed how this number of optimisations is not feasible and too computationally expensive. However, by increasing the interval size from 1 to 30 degrees, the total number of classifications reduces by almost 97%, whilst the fitness of the path degrades by only 15% in the process. This degradation to the fitness of the path occurs since the overall length of the path increased by approximately 7%, or 460 metres. However, since the length of the path increased whilst the curvature profile remained visually similar, this increased the radius of curvature and reduced the path dynamics. This meant that although the artificially classified paths at high interval sizes are sub-optimal when compared to the optimised solution, they remain dynamically achievable, with the main factor associated with degrading the fitness being an increased path length. It was determined, therefore, that applying discrete heading angles with intervals of 30 degrees, enabled dynamically achievable paths to be classified, whilst reducing the necessary number of classifications by over 97%.

Airspeed Observation

Currently, only the north, east and heading angles of the waypoint target have been considered as observations for classification. During optimisation of the previous studies, the airspeed was held constant at 150 m/s. Depending on the size of the north and east observations, it can be expected that the dynamics of the path may not be achievable at higher airspeeds, since this increases

aerodynamic loading. Figure 6.10 shows four optimised solutions conducted at airspeeds of 100, 150, 200 and 250 m/s. Three of the four solutions are almost identical, with negligible differences. However, the path produced when optimised with an airspeed of 100 m/s was able to sustain tighter turns with a slightly smaller radius of curvature throughout the second part of the manoeuvre. This analysis was repeated for multiple north, east and heading observations with similar results.

Based on this analysis, it is evident that whilst changes to the airspeed do impact the dynamics of the path, they do not necessarily have as large an impact as other observations, at least for constant-altitude path planning. Additionally, performing optimisation at various airspeeds further increased the dimensionality of the problem. This increases the required number of optimisations to exhaustively account for all observations using classification. During the interval size study for the heading angle, it was determined that for an interval size of 30 degrees, there exists 1,320 classifications. If this configuration was to be conducted at four airspeed values, this increases to 5,280 classifications.

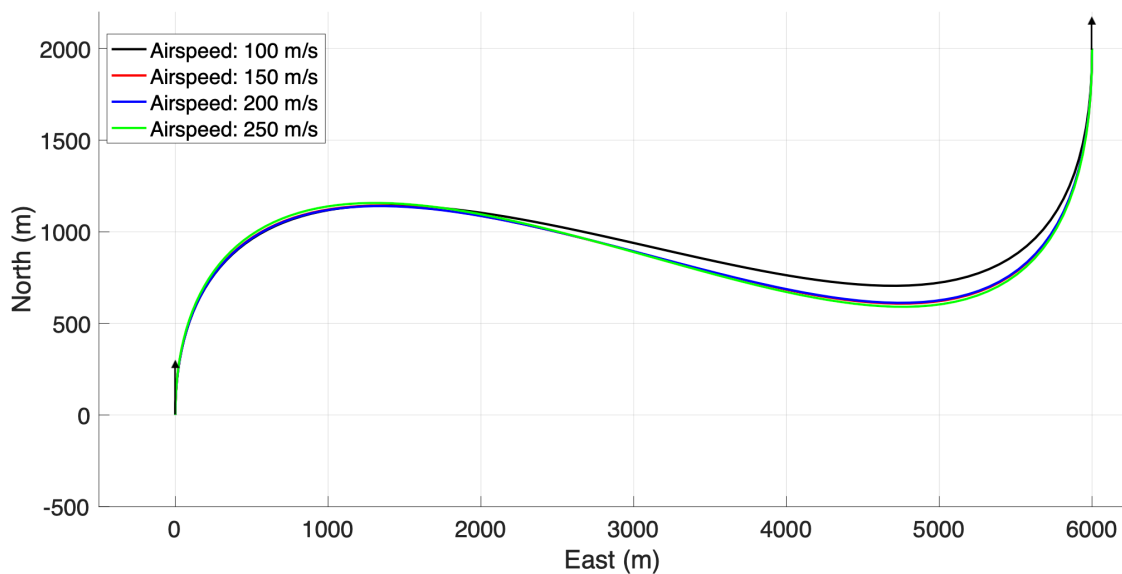


FIGURE 6.10 Path planning optimisation for a north coordinate of 2000 metres, east coordinate of 6000 metres and heading angle of 0 degrees for various airspeeds

Due to these factors, changes to the airspeed were neglected during optimisation for the constant-altitude path planning classification network as the in-

creased number of classifications did not provide any noticeable benefit to classified solutions. Additionally, for constant-altitude path planning, if a solution is optimised at a specific airspeed value, this solution should remain dynamically achievable at lower airspeeds, provided the dynamics of the aircraft allow this level of flight to be sustained. This meant that provided the optimised dataset was performed at an upper operational velocity value, the classified paths are guaranteed to be dynamically achievable at lower airspeeds. Therefore, optimisation of the final solution was performed for an airspeed of 300 m/s, which was the upper airspeed value for the scope of this research due to limitations within the aerodynamic data.

Final Solution

Having determined observational interval sizes of 1 km for the north and east observations and an interval size of 30 degrees for heading observations, optimisation was performed to create the entire classification dataset. To adequately allow the airspeed to be neglected whilst still ensuring paths remain dynamically feasible, optimisation was performed for an airspeed value of 300 m/s.

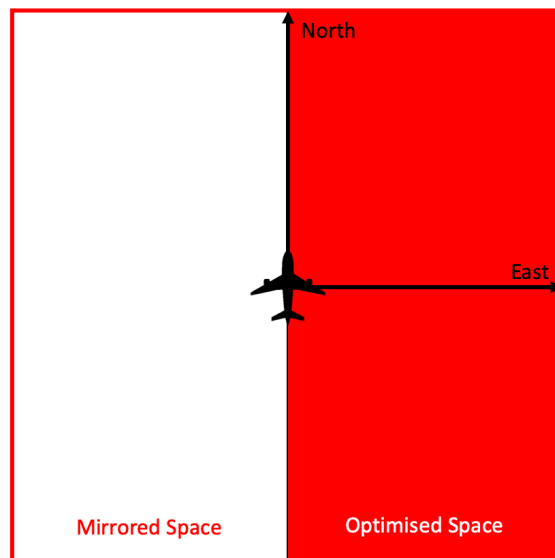


FIGURE 6.11 Path planning intermediate axis system formation

It was previously explained that for the purpose of constant-altitude path optimisation, an intermediate axis frame (IRF) is used to describe the location of the waypoint target as relative observations to the aircraft-fixed origin. Formulating

the path planning problem in this manner reduces the number of observations to three, whilst also allowing flight to be conducted through the entirety of the inertial axis. Additionally, since the aircraft has symmetrical dynamics about the lateral axis, optimisation was only required for relative waypoint observations located either in the left or right half of the IRF, with solutions mirrored to complete the analysis. An example of this can be seen in Figure 6.11. Based on the number of north, east and heading observations, a total of 2640 classifications were required to complete the analysis (Table 6.2).

TABLE 6.2 Constant-altitude path planning observations

Observation	Range	Intervals	Classifications
North	-10 km to 10 km	1 km	20
East	0 km to 10 km	1 km	11
Heading	0 deg to 360 deg	30 deg	12
Total Classifications			2640

Due to the high number of classifications, optimisation was performed 10 times per class as opposed to 15 times. Each of the 10 solutions per class was analysed using the same z-score metric as previous studies, with outliers removed from the dataset. The remaining solutions were used to establish a mean solution for each class, forming a training dataset comprised of 2640 discrete observation classifications. Since training of the classification network needed to be performed using solutions for all classifications, a validation dataset that was 10% the size of the training dataset was artificially created. This validation dataset randomly selected classifications and applied an offset to each observation. The magnitude of the offset was scaled such that the observations remained within the classification interval and acted as an intermediate observation for the purpose of validation.

The structure of the neural network was determined through a training trail-and-error analysis. This analysis would train a network for 10,000 epochs and based on the classification and validation accuracy, adjust the number of hidden layers and fully connected neurons. This analysis concluded that a network comprised of three hidden layers where the first layer contained 10 fully-connected

neurons allowed for classification accuracy of 100%. Additionally, an expansion coefficient of three was applied to each additional hidden layer, meaning that layers two and three had 30 and 90 fully-connected neurons. This gave a total of 359,370 trainable parameters within the neural network. Each neuron maintained the same ReLU activation function as the previous studies, with a softmax classification layer acting as the output activation. After approximately 1,000 epochs, the network was successfully able to classify solutions with 100% accuracy and with a categorical cross-entropy loss metric less than 0.02. The accuracy and loss metric during the training process can be seen in Figure 6.12.

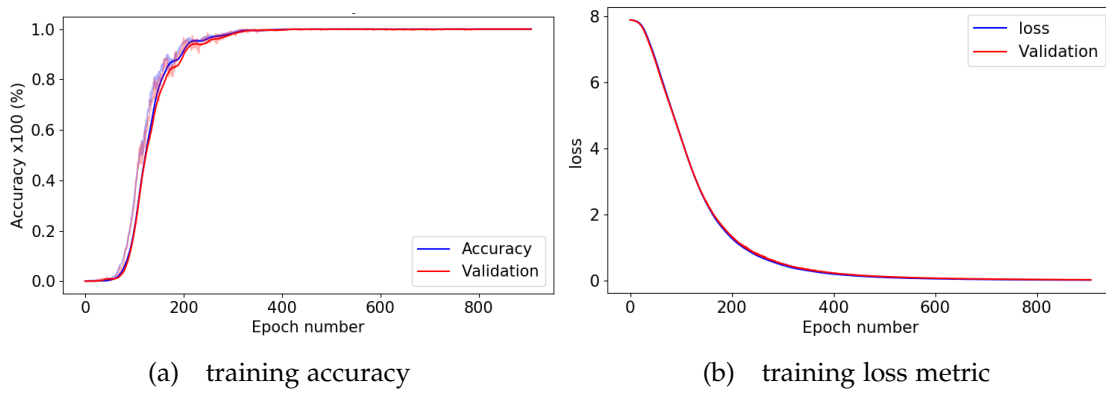


FIGURE 6.12 Neural network classification training accuracy and loss metrics

6.5 Classification Implementation

Since the classification model requires an IFR, this must be embedded within the path planning framework, allowing inertial frame target coordinates to be defined as relative coordinates. Prior to this, the path planning architecture must be defined.

It has already been discussed that the IRF has dimensions of 20 km along both the x and y axes. These axes move through the inertial frame with the same dynamics as the aircraft, with the origin of the reference frame coincident to the aircraft's centre of gravity. In addition, the orientation of the reference frame rotates solely based on the aircraft's azimuth, meaning reference frame rotation about the x and y axis is not performed. This ensures that the x - y plane of the IRF remains parallel to the x - y plane of the inertial axis system, and allows

the waypoint target location to be described in the IRF.

Figure 6.13 shows an inertial frame representation for the position of an aircraft located with a north coordinate of 5 km, east coordinate of 5 km and with an azimuth of 0 degrees. The inertial frame waypoint target is located at coordinates north 10 km and east 0.5 km, with an azimuth of -60 degrees when wrapped between $-\pi$ to $+\pi$. The location of the reference frame is coincident to the body-axis vectors of the initial position, and maintains the same orientation of the initial position vectors.

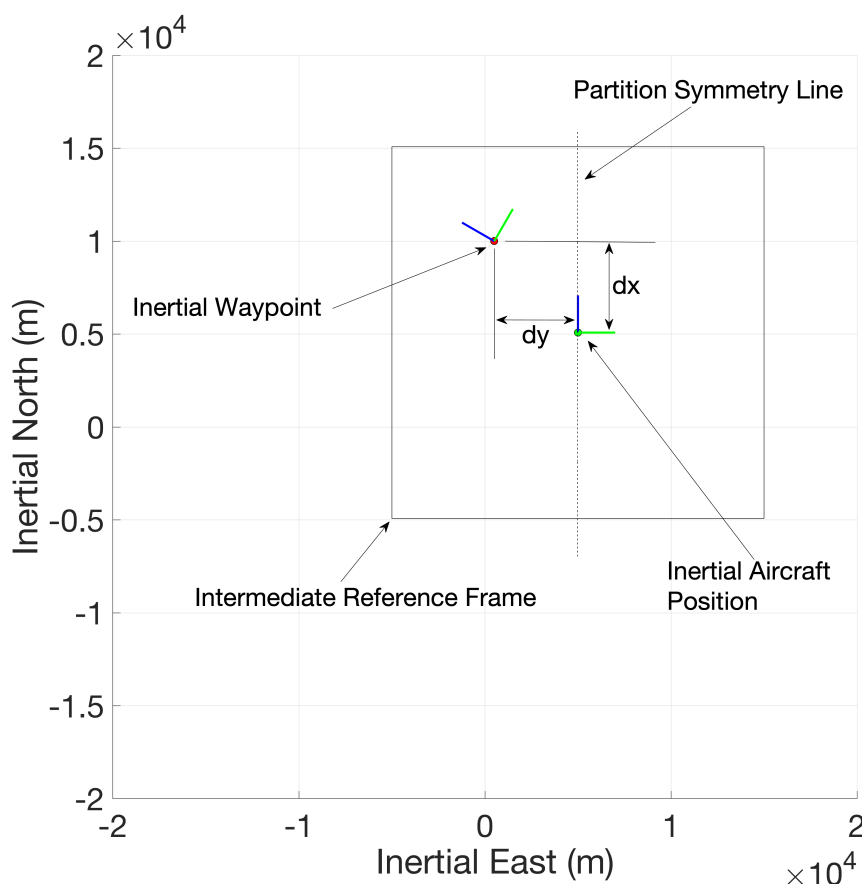


FIGURE 6.13 Inertial axis representation of the intermediate reference frame for constant-altitude path planning

It can be seen that the x and y axes of the IRF expand away from the origin by 10 km in both directions, creating a virtually bounded box. Although the waypoint target is described using inertial coordinates, the terms dx and dy allow its position to be described relative to the origin of the IRF. This illustration demonstrates how the IRF is described in the inertial axis system and how

the location of the waypoint target is translated into this frame and described a relative coordinates, allowing the appropriate observations to be applied as classifications for the neural network.

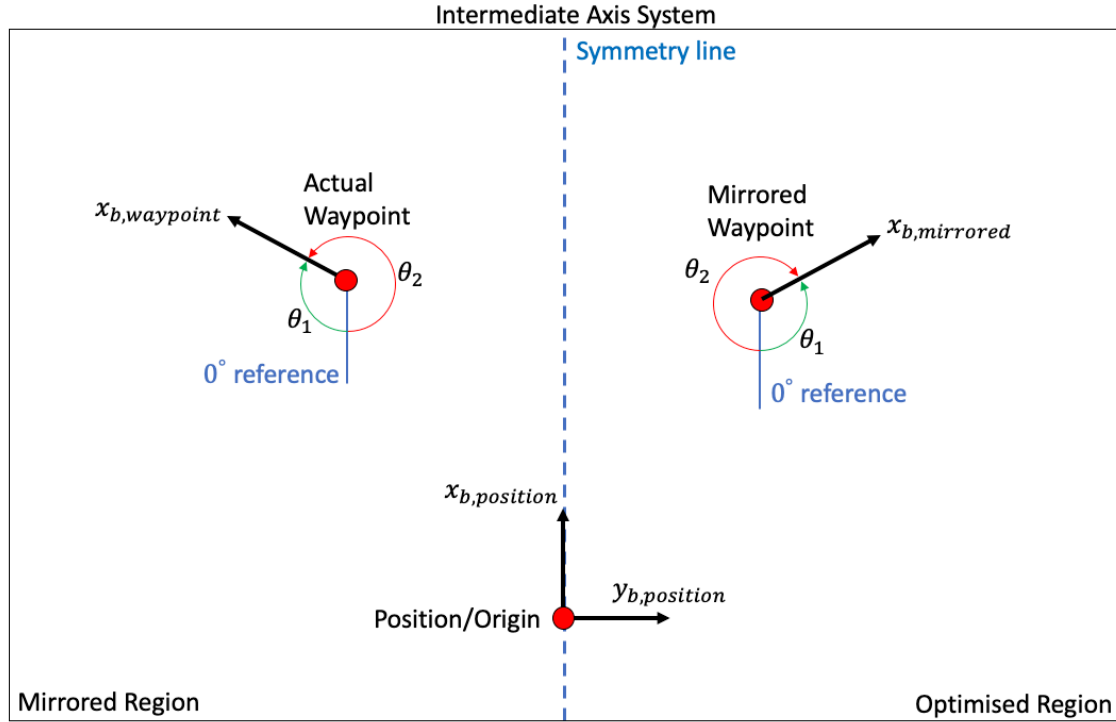


FIGURE 6.14 Classification model mirrored waypoint configuration two orientation angles

During optimisation of the training data for the classification network it was discussed how, due to dynamic symmetry, observational locations were only considered along the east observation from 0 km to +10 km, which corresponded to the right hand side of the IRF. This meant that if a waypoint target was located in the left-hand partition of the IRF, as depicted in Figure 6.13, the position and orientation of the waypoint requires translation to the right-hand side of the IRF, allowing classification to be performed. A demonstration of this process can be seen in Figure 6.14. This illustration shows the location of the aircraft at the origin of the IRF. The IRF is partitioned into two halves along the x axis, with the intended waypoint target located in the left hand side, known as the mirrored region. The azimuth of the waypoint target's x vector is described by two angles, θ_1 and θ_2 .

Knowledge of these two angles allows the waypoint to be mirrored from the

left hand side of the IRF to the right hand side. The corresponding north, east and heading observations in the right-hand side of the IRF allows classification to be performed and control point scaling solutions obtained for the optimal path. This path is produced in the right-hand side of the IRF, meaning that it has to be translated back into the left-hand side prior to be applied to the guidance control system.

Upon translating the optimised path back onto the correct side of the IRF, the path coordinates relate to location with respect to the IRF, which is not compatible with the guidance control system. Therefore, using a quaternion transformation matrix, the path is translated from the IRF into the inertial frame, allowing guidance to be administered.

6.6 Distance Controller

During cooperative flight missions, a distance controller was implemented allowing the wingman to adjust its separation distance relative to the lead aircraft. The block diagram for the distance controller can be seen in Figure 6.15, where it has been integrated into the computational model of the wingman which is embedded inside the inertial frame simulation environment along with the computational model for the lead aircraft.

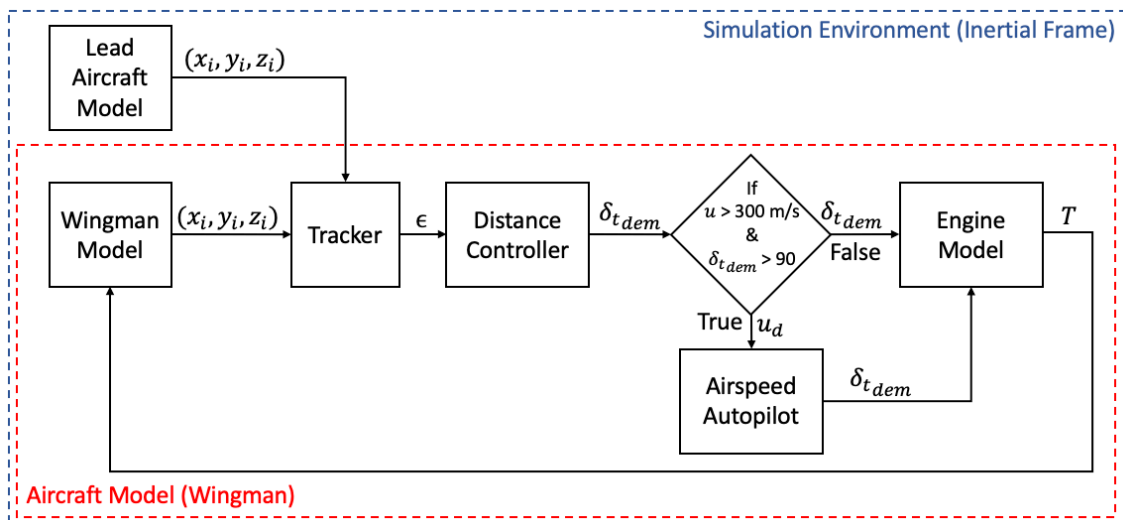


FIGURE 6.15 Distance controller block diagram schematic

This schematic shows how both the inertial position of the wingman and

lead aircraft are passed to a wingman tracker, which determines the Euclidean separation distance, ϵ . This separation distance is passed as an input to the distance controller, which utilises the same PID architecture as the controllers described in Chapter 4. The output from the distance controller, $\delta_{t_{dem}}$, describes the demanded throttle stick position as a percentage of the maximum throttle. Since the aerodynamics are only valid for subsonic flight conditions, the distance controller should not allow the aircraft to exceed this flight conditions. Therefore, as a precaution, a conditional statement checks the current airspeed of the aircraft and limits the output from the distance controller to prevent further acceleration past 300 m/s. This limit is applied by engaging the airspeed autopilot if the conditional statement is true. The airspeed autopilot is set with a demanded airspeed of 300 m/s, preventing the aircraft model from exceeding the aerodynamic limits. If the conditional statement is false, the output signal from the distance controller is applied directly to the engine model as a throttle stick demand.

Since the distance controller outputs a throttle stick demand or an airspeed autopilot input demand, both are which are interpreted by the engine model, this autopilot is controlling a slowly-varying state dynamic. Similar to the airspeed controller discussed in Chapter 4, the performance of the controller is not as severely impacted by changes to the operating conditions when compared to quickly-varying dynamics. For this reason, optimisation of the distance controller was performed for a single set of controller gains. Additionally, although the controller architecture is PID, the integral component only engages during low position errors, preventing integral wind-up and allowing the controller to function as a PD controller during large position errors.

Optimisation of the distance controller gains was performed in two stages using the same BA as used during optimisation of the angular rate controllers in Chapter 4. During cooperative flight, the lead aircraft communicates over TCP sockets to the wingman, this created large overheads during the optimisation process, since establishing communication required synchronicity between multiple subsystems. Therefore, the lead aircraft was simulated offline by positioning it in the inertial frame at (0 km, 0 km, 0 km) with an orientation of (0°, 0°, 0°) and an airspeed of 200 m/s. The lead aircraft was instructed to

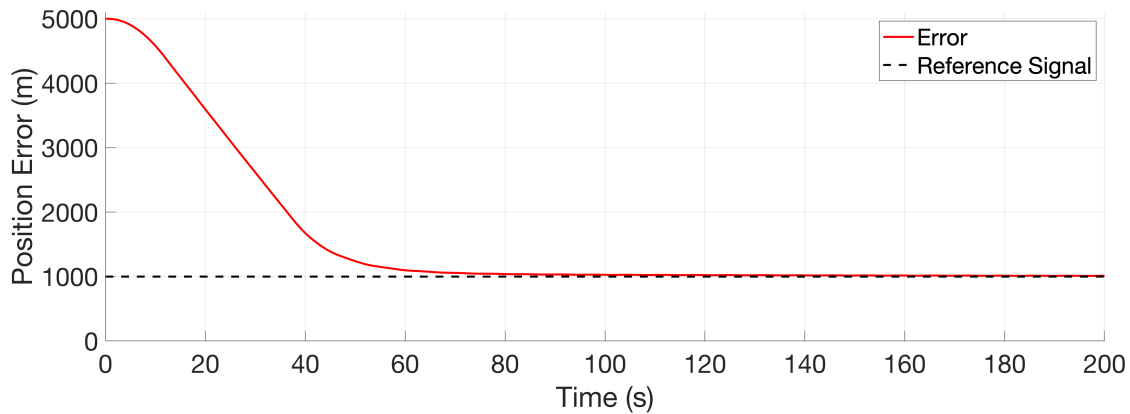
follow a 100 km straight line path for a total of 100 seconds.

By simulating the lead aircraft offline, this reduced the complexity of the optimisation process, since the offline generated data was loaded into the wingman's flight control system as opposed to simulating the lead aircraft during optimisation. This enabled the offline data to act as a cooperating aircraft for the purpose of optimisation. The wingman was initialised 5 km behind the lead aircraft, with the same orientation and with an airspeed of 200 m/s. Similarly, the wingman was given a 100 km straight line path to follow for a total of 100 seconds. At each simulation time-step, the offline data for the lead aircraft was applied to the tracker as seen in Figure 6.15, allowing the separation distance between the two aircraft to be established. For the duration of optimisation, the wingman was instructed to maintain a separation distance of 1 km from the lead aircraft, with the output from the distance controller adjusting the throttle demand to the engine model.

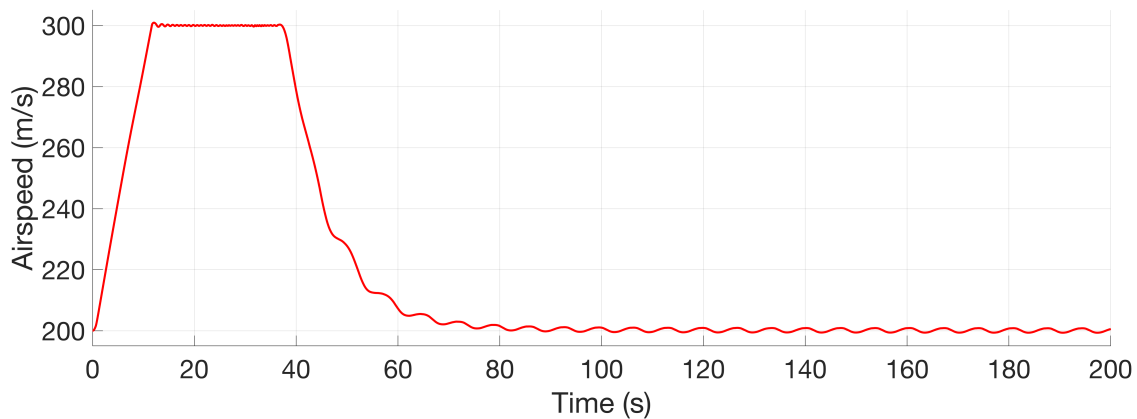
Since the position error between the wingman and lead aircraft consists of large values, the fitness of each bat was measured using a weighted-sum-squared-error (WSSE) fitness function between the actual separation distance and the demanded distance. Use of a WSSE fitness function allowed the large initial position error to be damped, favouring a long-term reduction to the steady-state error. However, to prevent overshoot during the transient period, if the wingman overshoot the demanded separation distance by more than 100 metres a harsh penalty was applied. The optimised position error time response can be seen in Figure 6.16a with the airspeed profile for the wingman throughout the simulation in Figure 6.16b.

From the optimised time response in Figure 6.16a, it can be seen that the response is adequately damped with no overshoot despite the slowly varying dynamics associated with airspeed. However, when comparing the airspeed profile of the wingman during the same manoeuvre, during the steady-state response the airspeed is continuously oscillating around 200 m/s. This oscillatory motion is not considered to be a major issue since the amplitude of the oscillatory motion is less than 0.5 m/s and is not growing over time. Additionally, since the distance controller is actively performing micro-adjustments relative to the separation distance, and the engine dynamics contain a first-order lagged

response, the oscillatory airspeed response can be expected.



(a) Position error time response



(b) Airspeed during optimisation for optimised response

FIGURE 6.16 Distance controller simulation optimisation

Since the implementation of the distance controller is relatively simple when compared to the complexity of the problem, it was expected that operational issues would occur. For example, the work produced in Chapter 6 assumes constant-altitude flight, enabling the controller to not account for the climb dynamics of the aircraft. This means that if the aircraft was to perform a climb, the controller would produce throttle responses which would be unsatisfactory for the dynamics of the aircraft, reducing the robustness of the control system. Despite these factors, the distance controller was merely a tool for demonstrating a catch-up procedure during online path re-planning, with future work advised to increase the robustness of the controller by factoring predictive along-track and cross-track evaluations for the lead aircraft.

6.7 Classification Testing

It was discussed in Chapter 5 that producing three-dimensional paths required two pieces of information: (i) set of body-axis vectors for the initial position; and (ii) a set of body-axis vectors for the final position. The orientation of these body-axis vectors determines how the path is generated between the points, since the initial and final direction of the path is orientation-constrained along the initial and final heading vectors. By considering the path generation problem in this manner, the location of the internal control points was solved using multi-objective optimisation.

During optimisation, both the initial and final locations for the path were considered as static locations. The term static location refers to a position and orientation of the body-axis vector that is not changing over time. However, the purpose of this research was to develop a path planning framework that allowed real-time interaction between a wingman and lead aircraft during cooperative flight missions. This means that both the initial location and waypoint target cannot be considered as static locations, since they are dynamically moving through the inertial frame throughout flight. Therefore, the applied path planning framework needed to be adaptable and monitor changes to the inertial positions, allowing flight paths to be re-planned online. Therefore, testing of the classification network was performed in two stages: (i) a static waypoint analysis; and (ii) dynamic waypoint analysis.

6.7.1 Static Waypoint Analysis

It has already been discussed that the focus of the research in this thesis is towards cooperative flight, which required dynamically moving waypoints. However, since the optimisation process was performed for static locations, performing a static waypoint analysis provides an insight into the overall performance capabilities of the network, and how well the network is able to classify optimised solutions.

Upon initialising the computational model, the location of the aircraft and waypoint target was randomly placed within the inertial frame and with a random orientation. Limits were applied ensuring the waypoint's location relative

to the position vector remained correctly satisfied and within the perimeter of the IRF. Additionally, the initial airspeed of the aircraft was randomly generated between the range of 100 and 300 m/s. The airspeed controller was engaged throughout the duration of the simulation to maintain this airspeed value.

To produce a path between the aircraft and waypoint target, the waypoint was translated into the IRF using the same procedure outlined in Section 6.5, with relative observations used to query the classification network. The classified solutions were applied to the path planning framework, generating the resultant Bézier curve. This path was applied to the guidance control system, allowing the computational model to autonomously fly the proposed path. For the duration of this analysis the look ahead value for the guidance control system was kept constant at a distance of 300 metres. To aid in the repeatability of the results and to prevent background processing tasks from influencing the results, a fixed time-step of 0.01 seconds was used throughout the static waypoint analysis as opposed to a variable time-step.

Algorithm 2: Static waypoint testing algorithm used to determine dynamically feasible and optimal solutions

Variables: Aircraft Position (x_i, y_1, z_i)
Waypoint Position (x_i, y_1, z_i)
Airspeed: u
Simulation Time: T
Predicted Path Time: T_{path}

Output : Feasible (Boolean)
Optimal (Boolean)

Position Error = norm(Aircraft Position - Waypoint Position)
if Position Error < 300 metres **then**
 Feasible = True
 Time Pct Error = $(T - T_{path}) / T \times 100$ **if** Time Pct Error < 10% **then**
 Optimal = True
 else
 Optimal = False
else
 Feasible = False
 Optimal = False

If the computational model successfully reached the end of the path, the simulation commenced and the path was deemed to be dynamically achievable.

Since constant-altitude path planning aimed to create continuously flyable paths based on maintaining a near constant airspeed, the performance of the path was quantitatively measured based on a time analysis. During path generation an estimated time required to complete the path at a constant velocity was computed. This estimated completion time was used as a conditional statement, to determine whether or not the path represented an optimal solution. For example, if the computational model successfully reached the end of the path within 10% of the estimated time required, the solution was deemed to represent an optimal solution as this suggested the path was successfully completed based on a near-constant velocity. Conversely, if the computational model successfully reached the end of the path but exceeded the estimated time required by 10%, the path was deemed dynamically achievable but represented a sub-optimal solution. This algorithmic procedure for quantifying feasible and optimal solutions can be seen in Algorithm 2.

During a simulation, if the aircraft deviated more than 300 metres away from the path, the guidance control systems triggers an alert to the autonomous flight control system. This alert indicates that the current path is not dynamically feasible, and that to proceed with the simulation a new path should be re-planned. This procedure increases the robustness of the path planning framework, since the control system is able to identify whether or not the path is adequately manoeuvrable online. However, since this analysis was not conducted to evaluate the robustness of the path planning framework, but rather the performance of the classification network to classify solutions for observations throughout the IRF, online path re-planning was not enabled. Therefore, during the static waypoint analysis, if this warning was triggered, the simulation would end with the classified solution deemed dynamically unachievable.

Figure 6.17 shows the results from 5,000 simulations. Each marker represents the location of the waypoint target with respect to the IRF, where the location of the aircraft for the simulation is at the origin. During this static waypoint analysis, the aircraft successfully navigated to the waypoint target for 100% of the 5,000 randomly generated waypoint targets, with optimal solutions occurring for 99.52% of simulations. These optimal solutions are denoted as blue markers, as determined using Algorithm 2, whilst red markers indicate sub-optimal

solutions.

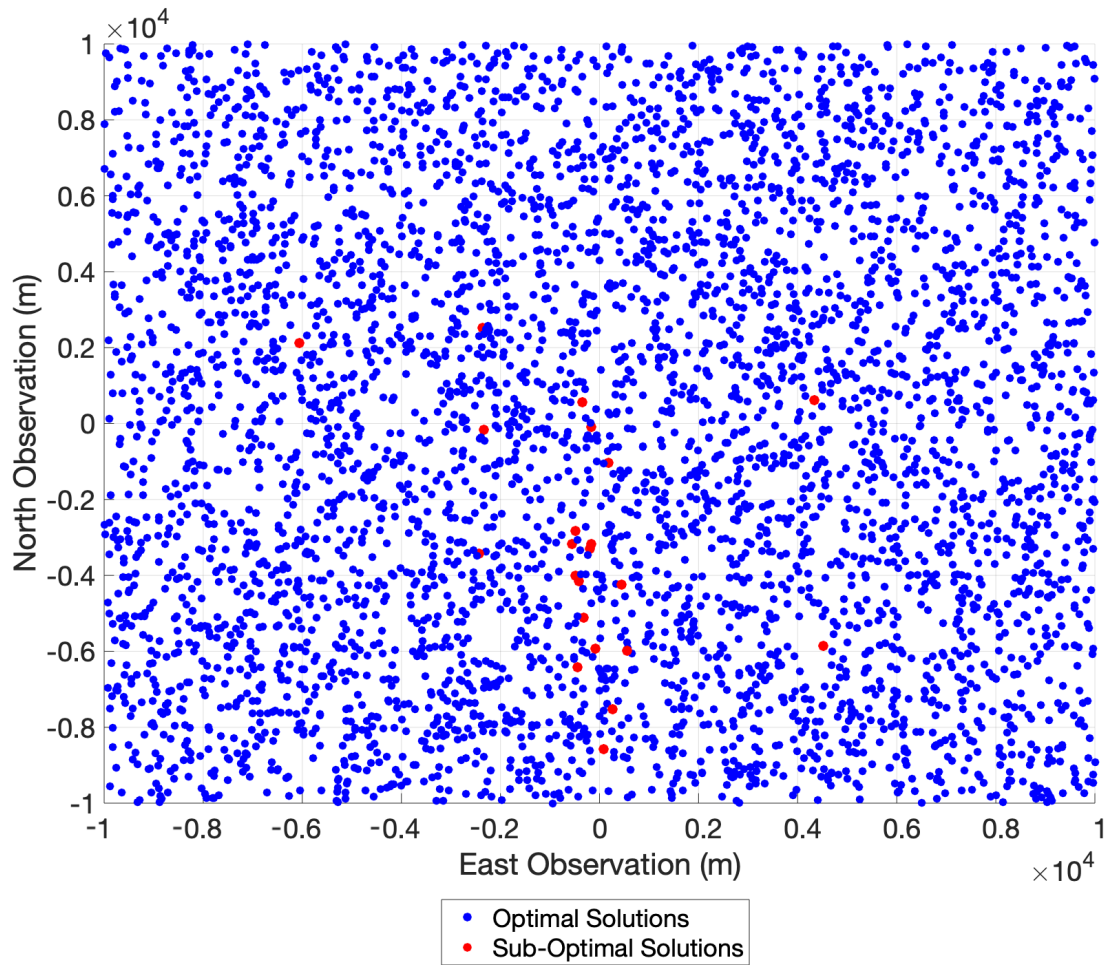


FIGURE 6.17 Classification network static waypoint analysis optimal and sub-optimal solutions displayed for north and east observations with respect to the intermediate reference frame

Although the classification network was successfully able to produce dynamically feasible paths for all the tested scenarios and regardless of the airspeed, it is evident that sub-optimal solutions occur more frequently when the waypoint target is located with a negative north observation in the IRF. More than 80% of sub-optimal solutions occurred within this region, with 90% of these sub-optimal solutions occurring within ± 1 km along the east observation. During these scenarios the placement of the waypoint occurs almost directly behind the initial position of the aircraft, meaning a full directional change is required to reach the intended target. Given that the majority of these solutions also occurred as the east coordinate approached the origin, this indicates that sub-

optimal solutions occur more frequently within this region due to a small euclidean distance between the position of the aircraft of the waypoint target, whilst also demanding a very high directional change, pushing the dynamic limits of the aircraft.

Additionally, it can be seen that sub-optimal solutions within this region occur in almost identical places on either the positive or negative side of the east observation. Since optimisation was only performed for positive east observations with solutions for negative east observations obtained by mirroring the waypoint into the positive domain, this characteristic is to be expected. This means that a solution deemed sub-optimal when located in the positive region will translate into a sub-optimal solution in the negative axis due to dynamic symmetry. Although this doubles the chance of sub-optimal solutions occurring, due to the low number of sub-optimal solutions throughout the static waypoint analysis, this was not deemed to be a concern.

6.7.2 Dynamic Waypoint Analysis

During the static waypoint analysis in Section 6.7.1, the classification network was able to classify dynamically feasible path planning solutions for 100% of the tested scenarios with 99.52% of solutions categorised as optimal. This analysis was a good measure of the network's performance and how well solutions transpired during non-linear flight simulations. To understand the performance of the classification network during cooperative flight with a lead aircraft, a dynamic waypoint analysis was conducted. Unlike the static waypoint analysis, quantitatively measuring the performance of the network based on successfully navigating the aircraft to the waypoint target was not feasible, since the waypoint target was dynamically moving throughout the duration of a flight simulation. In addition, the application of online path re-planning was complicated due to the orientation constraint.

During the development of the path planning algorithm in Chapter 4, the orientation constraints were defined as a set of orientated heading vectors located on both the aircraft and waypoint target body-axis x axes. These orientated vectors allowed a control point to be placed at some distance, along each of the heading vectors. This created a preliminary region along the path that ensured

the initial directions of the path remained parallel to the motion vector, assuming no sideslip. This was a vital aspect for producing dynamically feasible and continuously manoeuvrable paths. However, if path re-planning was to occur whilst the aircraft was located inside this orientation forward region of the path, the new path would only extend this region further, engaging the aircraft in a deadlock and preventing the aircraft from adequately pathing towards the waypoint.

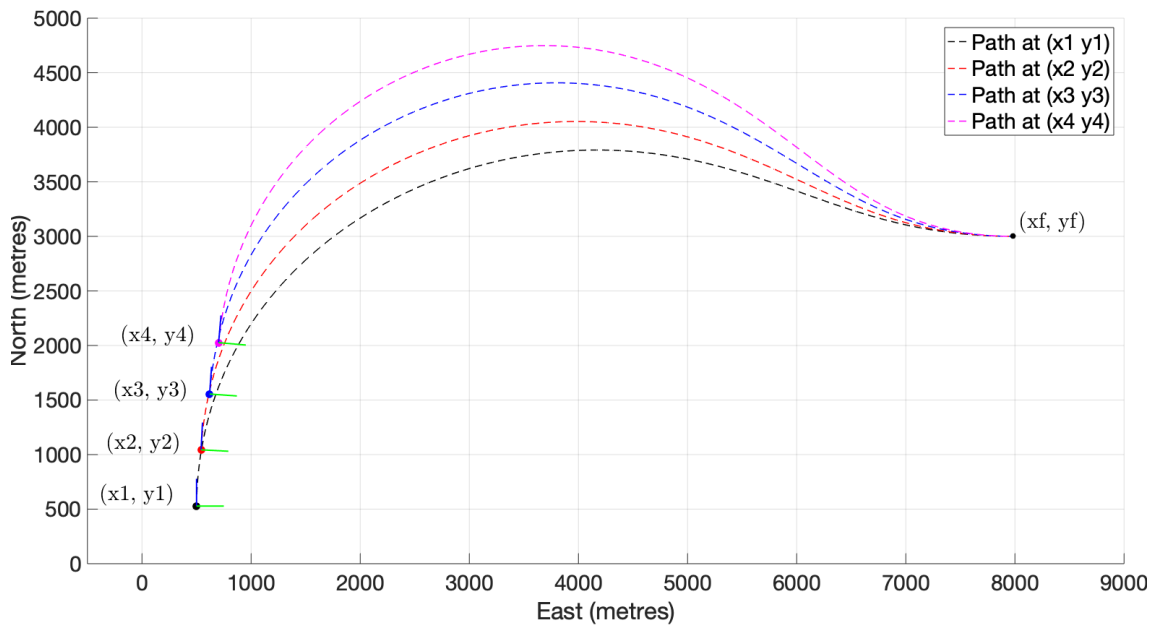


FIGURE 6.18 Online Path re-planning example conducted every 500 metres for four intervals. This example was conducted on a static waypoint target

This can be seen Figure 6.18, where an example simulation is demonstrated. During this demonstration, the aircraft is navigating from point (x_1, y_1) to (x_f, y_f) . For the purpose of clarity, the waypoint target in this example is static and not moving. Despite a static waypoint target, online path re-planning was performed for the aircraft every 500 metres, with the parameters applied to the path planning framework updated based on the aircraft's new state dynamics. Each of the four path re-plans displayed, occurs within this orientated region and stagnates the aircraft from adequately navigating towards the waypoint target. In addition, as the aircraft moves through the inertial frame and its position adjusted, the resultant path becomes more dynamically challenging as a result of path re-planning being performed within this orientated region. To alleviate

this problem a partial path re-planning algorithm was implemented.

The partial path re-planning algorithm takes a path already generated from the classification network, and applies a partition point at some distance along the path. The distance this partition point occurs at can either be determined as a distance measurement or as a percentage based measurement along the total duration of the path. However, under partial path re-planning, this partition point acts as the origin of the IRF as opposed to the current position vector of the aircraft. The waypoint target is then described as relative coordinates to the

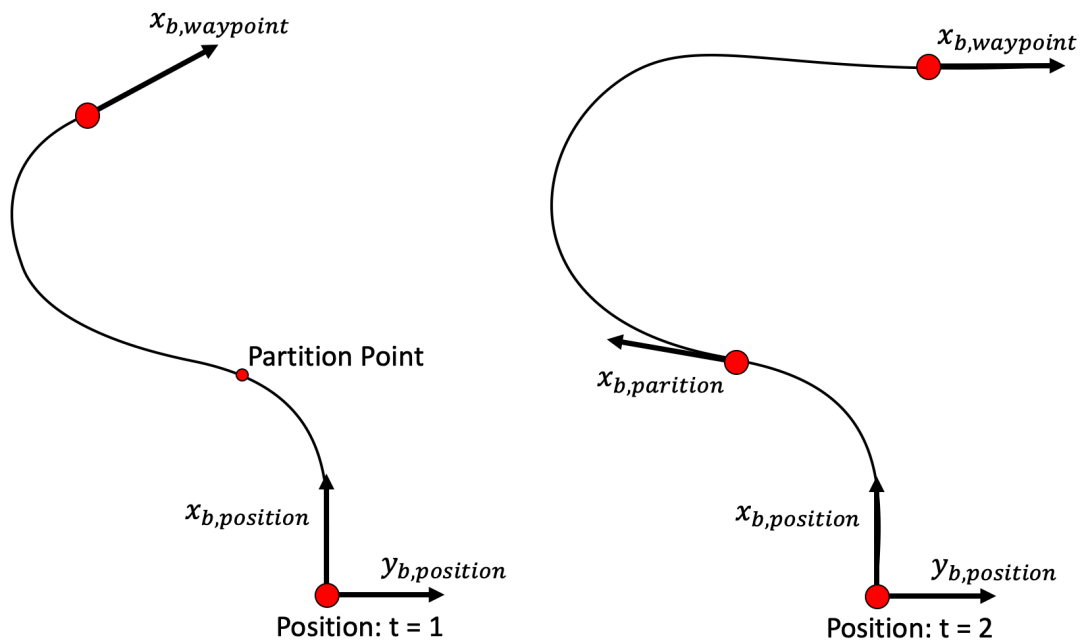


FIGURE 6.19 Partial path re-planning demonstration for moving waypoint target. Partial path re-planning is demonstrated between two time intervals $t = 1$ and $t = 2$

partition point and translated into the IRF using the same procedure as during the static waypoint analysis. The classification network is queried to determine the control point scaling solutions allowing a new path to be produced which originates from the partition point to the updated position of the dynamic waypoint target. Upon generating a path between the partition point and the waypoint target, the newly generated path is concatenated onto the original path.

This prevents path re-planning from being performed within the orientated forward motion zone of the path, allowing the aircraft to continue briefly along the original path until it reaches the partition point. An example of this process

can be seen in Figure 6.19, where the waypoint target between $t = 1$ and $t = 2$ changes, requiring a new path to be generated. A partition point is implemented when $t = 1$, with this location acting as the position vector with an orientation of $x_{b,partition}$ for the partially re-planned path at $t = 2$.

It has already been explained how quantitatively measuring the performance of the classification network during dynamic waypoint simulations is not feasible. Therefore, to measure the performance of the network during cooperative flight with a lead aircraft, a number of cooperative flight missions were created. During these missions, the lead aircraft flew a preset flight path autonomously and at a constant airspeed governed by the airspeed controller. The location and orientation of the lead aircraft was applied to the flight control system of the wingman as the location of the waypoint target. This allowed the waypoint to be a dynamically moving target coincident to the lead aircraft's position at any instance in time. The wingman was instructed throughout each mission to follow the lead aircraft at a set distance using the applied distance controller. This meant the wingman would increase and decrease its airspeed relative to the separation distance to the lead aircraft. Each mission flight path flown by the lead aircraft gradually increased in complexity, with a discussion performed into the development and testing purposes of each profile. Throughout each mission, an inertial frame representation between the lead aircraft and wingman is used at various instances in time to perform a visual analysis. This visual analysis allows the historic position data of both aircraft to be factored into the analysis, with the wingman's current path also displayed for the purpose of the discussion.

Each of the mission flight paths developed for the lead aircraft are comprised of multiple waypoint targets, with path planning performed between each waypoint using the classification network. This demonstrates the robustness of the classification network, since it can be applied to a large number of waypoints, creating more complex paths as opposed to creating a single path between two points. This allows the solution to remain effective even when implemented into more complex mission planning frameworks or path finding algorithms, external to the work conducted in this thesis.

Mission 1

The first cooperative flight mission, denoted as *Mission 1*, was a simple manoeuvre consisting of a large constant radius turn. The flight path for the lead aircraft as shown in Figure 6.20, starts at the location denoted as *Lead Aircraft*, with the aircraft flying counter-clockwise at an airspeed of 200 m/s, passing through each of the defined waypoints, before finishing back at the start location. The orientation of the lead aircraft has been represented by a set of unit vectors, where the blue vector indicates the orientation of the body-axis x -vector and the green vector representing the orientation of the body-axis y -vector.

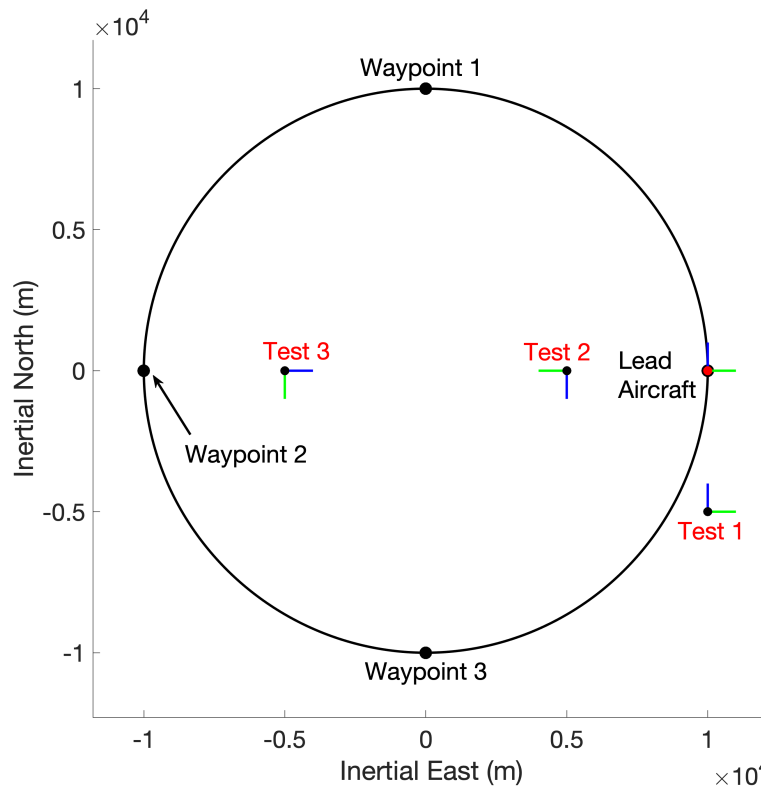


FIGURE 6.20 Classification network dynamic waypoint testing *Mission 1* flight path

It was previously discussed how due to the orientation constraint of the path planning algorithm, online path re-planning was conducted as either a full or partial re-plan. Determining the optimal conditions for performing either a full or partial re-plan could be considered a large scale analysis, which varies depending on the flight conditions and flight dynamics. However, for the purpose of this research, path re-planning was performed based on time intervals,

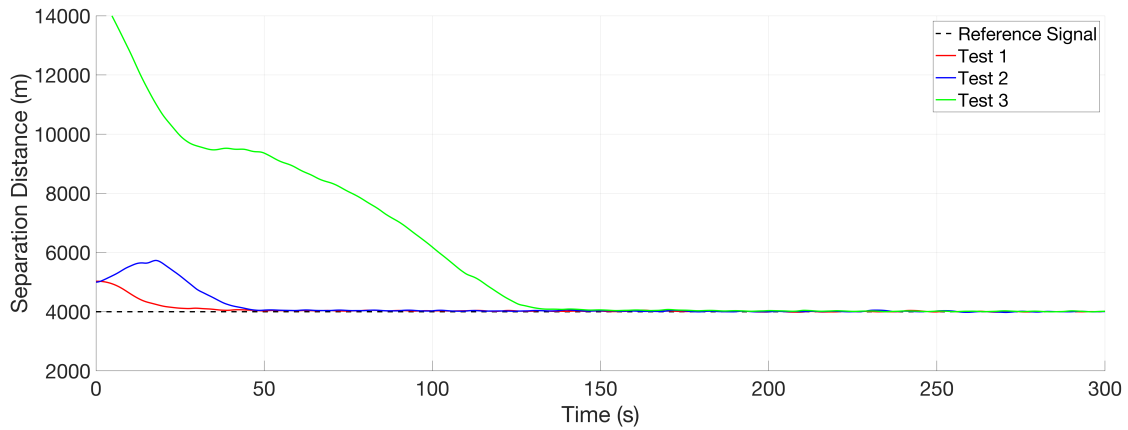
with partial re-planning performed every second and full re-planning every 10 seconds. The only exception to this rule occurred if the aircraft deviated too far from the path. If the wingman exceeded the bounds of the path by more than 300 metres, the wingman was deemed to be off the path, forcing a full path re-plan to be conducted during the subsequent time-step.

Throughout *Mission 1* it was important to understand how the wingman behaved and adjusted its dynamics based on different initial conditions. Therefore, a total of three tests were performed. During each of the three tests, the wingman was initialised at a different location and orientation as depicted by locations: *Test 1*, *Test 2*, and *Test 3* in Figure 6.20. For each starting position the autonomous aircraft was instructed to maintain a separation distance of 4 km relative to the lead aircraft. Since each of the starting positions exceeded this initial requirement, the wingman initially had to chase the lead aircraft and reduce the position error. Upon achieving the desired position error, the wingman was instructed to maintain this separation distance for the remainder of the simulation by adjusting the throttle setting to either increase, decrease or maintain airspeed.

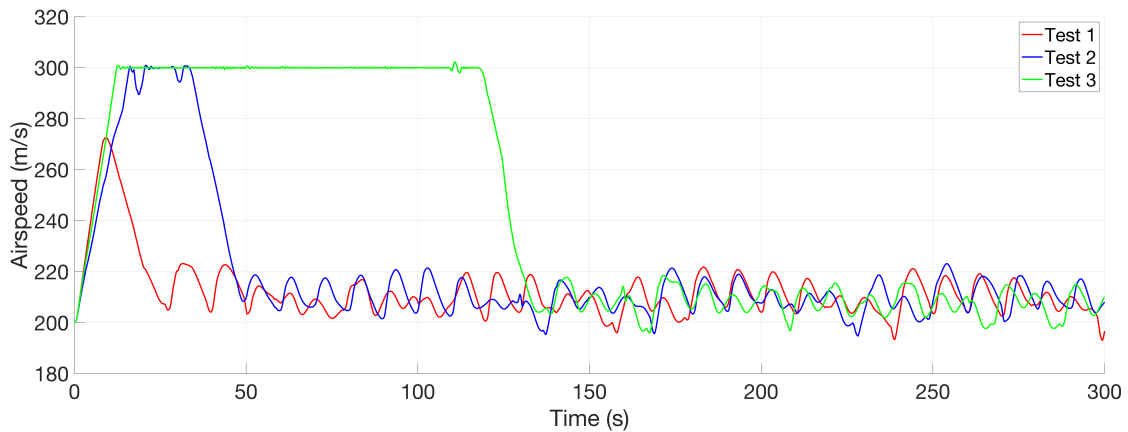
Since the wingman was commissioned to fly at a set separation distance from the lead aircraft, one measure of the flight control system performance was to analyse the separation distance between the two aircraft during each of the tested scenarios. Figure 6.21a shows the separation distance between the lead aircraft and wingman for all three test cases. The reference signal has been included for clarity and indicates the demanded separation distance throughout each test. Based on the separation distance, it can be seen that the wingman was successfully able to catch the lead aircraft and then proceed to maintain the demanded separation distance for the remainder of the mission.

During *Test 1* this catch-up procedure was relatively simple since the wingman started along the same inertial east location as the lead aircraft and with the same orientation. This meant that the wingman pathed in a straight line trajectory towards the lead aircraft, allowing the distance controller to instantly start reducing the separation distance. However, when focusing on *Test 2*, the orientation of the autonomous aircraft meant that initially both aircraft were travelling with heading differences of 90 degrees, with the separation distance between

the two aircraft increasing during the initial stages of flight. This, however, did not affect the performance of the flight control system, with the aircraft able to adequately perform a tight radius turn whilst accelerating up to an airspeed of 300 m/s, as depicted in Figure 6.21b.



(a) Separation distance between the manned aircraft and unmanned aircraft



(b) Airspeed profile for the autonomous aircraft throughout each test conducted in *Mission 1*

FIGURE 6.21 *Mission 1*: separation distance and airspeed time responses

During *Test 3*, quite a large initial separation distance of over 14 km was present, meaning the wingman had to perform a prolonged catch-up at 300 m/s for approximately 130 seconds. Despite this increased airspeed, the wingman was still able to maintain stability with the classification network successfully producing dynamically feasible paths, allowing the wingman to successfully navigate towards the lead aircraft. Additionally, scheduling of the distance controllers internal states prevented integral wind-up from occurring, allowing the wingman to accelerate and decelerate safely, and without overshooting the sep-

aration distance demand.

When focusing on the airspeed profile for the wingman during all three tests (Figure 6.21b), the profile could be interpreted as being undesirable due to its oscillatory acceleration and deceleration during the separation hold phase of the manoeuvre. However, since the wingman had no prior knowledge of the lead aircraft's flight path, the flight control system was unable to damp and maintain airspeed since it was constantly performing micro-adjustments based on the separation distance position error. Additionally, since changes to the throttle are subjected to a first-order time lag as per the engine model, the output received from the distance controller had a lagged response, reducing its immediate effectiveness. Whilst this acceleration/deceleration profile could be negated by introducing an operational dead zone, this was not deemed to be necessary for the purpose of the dynamic waypoint analysis, as this study focuses on the performance and robustness of the classification network as opposed to the distance controller. Nevertheless, the wingman was adequately able to catch the lead aircraft and maintain a near perfect separation distance despite these dynamically challenging factors.

Figure 6.22 shows a set of four time stamped inertial frame representations of the flight paths for both aircraft during the third test, with the time stamped representation for the first and second test in Appendix D.1. These time stamped representations show the formation of the lead aircraft and wingman in the inertial frame at specific locations in time. The historic location of both vehicles as well as the current path they are following have also been depicted for clarity.

Displaying the current path for the wingman was an important factor since the flight control system is performing online path re-planning based on updates received corresponding to the lead aircraft's location and orientation. All information regarding the inertial frame plots can be seen in the caption handle.

Since the primary objective of *Mission 1* was to perform a simple routine catch-and-maintain manoeuvre, use of time-stamped inertial frame references demonstrates to the reader a better representation of how the wingman was situated at various instances in time and showcases the potential of online path re-planning using the classification network.

It can be seen from Figure 6.22, that each aircraft is located at the centre of

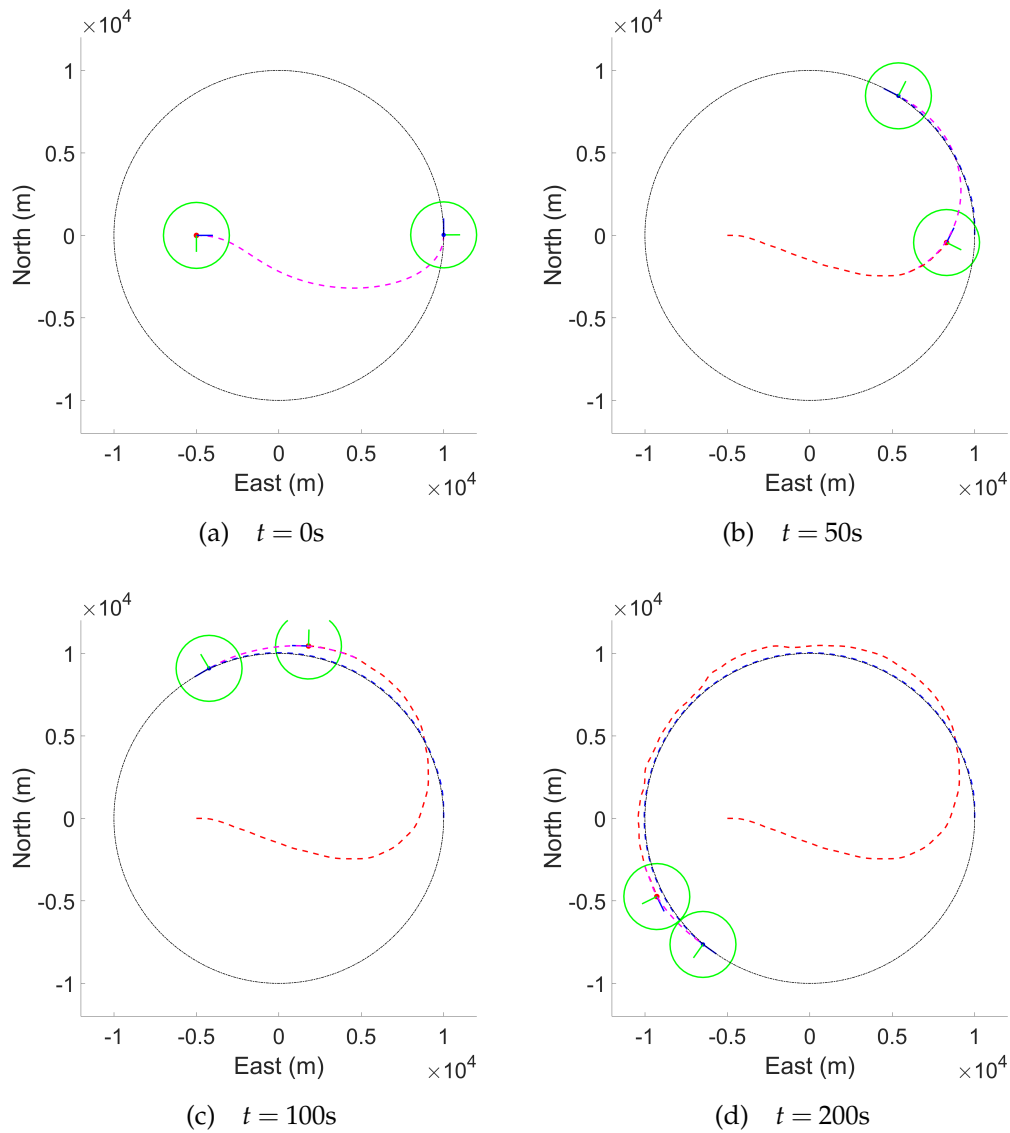


FIGURE 6.22 Mission 1: Test 3 - Time sampled inertial position. Red dotted line indicates historic position data for the wingman with the blue dotted line indicating the historic position data for the lead aircraft. The black line indicates the flight path for the lead aircraft with the magenta dotted line indicating the current path for the wingman developed by the classification network.

a green halo. The purpose of this green halo is to give the reader a better understanding of the demanded separation distance. For *Mission 1*, the wingman was instructed to maintain a separation distance of 4 km, which was successfully achieved during all three tests. This meant that the radius of each halo was 2 km,

meaning that the objective of the wingman was to fly with its halo tangentially connected to the lead aircraft's halo.

Based on the separation distance analysis shown in Figure 6.21a for *Test 1*, it can be seen that after approximately 130 seconds the separation distance approaches the demanded 4 km, indicating the halos are now tangentially connected. It can be seen in Figure 6.22c that after 100 seconds the separation distance between to the lead aircraft is reducing when compared to Figure 6.22b, with the halos tangentially connected in Figure 6.22d after 200 seconds. This demonstrates that although the separation distance was achieved after approximately 130 seconds, 70 seconds later the connection was still present, indicating the distance controller and path planning framework were adequately working together to maintain the demanded separation distance.

Mission 2

During *Mission 1* the primary objective was to determine how well the wingman was able catch the lead aircraft and then maintain a set separation distance during a simple and constant-radius turn manoeuvre. Having successfully completed all of the tests within *Mission 1*, the second mission, denoted as *Mission 2*, contained a significantly more sophisticated manoeuvre, consisting of 12 waypoints as depicted in Figure 6.23.

This manoeuvre was designed to try and deter the wingman from being able to maintain the demanded separation distance by incorporating multiple consecutive and tight constant-radius turns. Although *Mission 2* does not incorporate multiple defined test scenarios as seen in *Mission 1*, due to the complexity of the flight path, this mission was performed multiple times to adjust the path planning internal parameters and the separation distance values.

For the initial test, the wingman was configured to maintain a separation distance of 4 km as seen during *Mission 1*. This meant that the demanded separation distance exceeded the radius of curvature for each of the consecutive turns between waypoints two-to-nine. By configuring the turns to have a smaller radius of curvature, the wingman should suffer from numerous geometric issues. Firstly, as the lead aircraft begins to perform one of the many consecutive turns, the distance between the wingman and lead aircraft will immediately

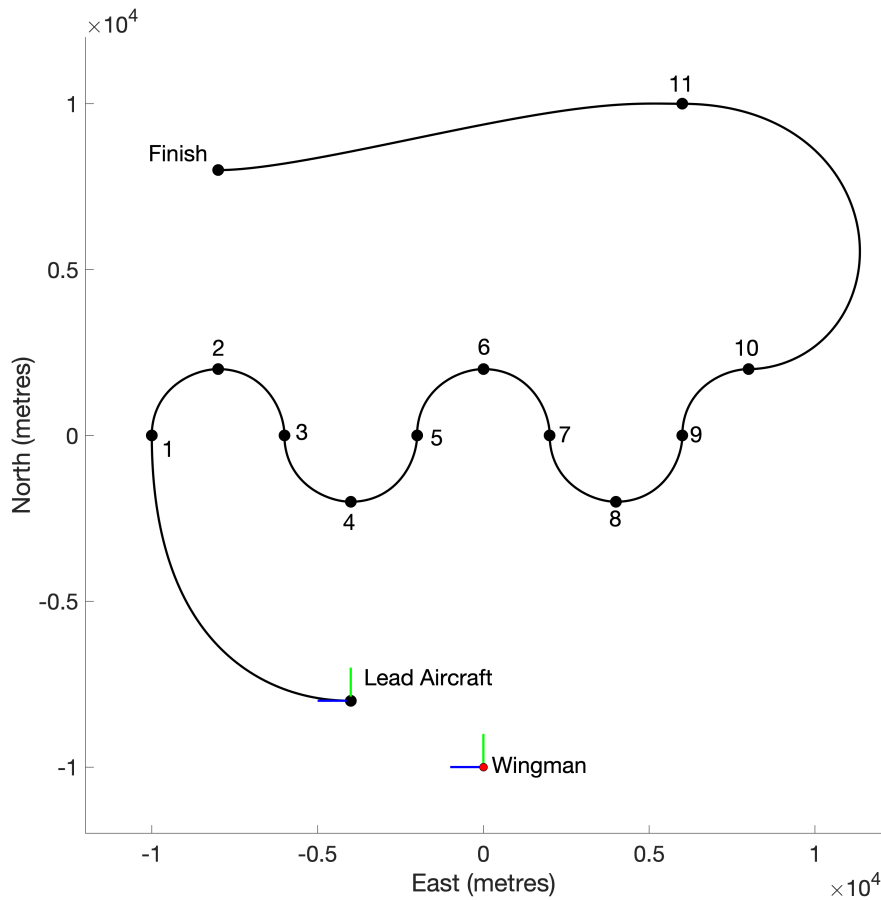


FIGURE 6.23 Classification network with dynamic waypoint path re-planning *Mission 2*

begin to reduce. If the wingman is operating at the demanded separation distance, this creates a rapidly changing negative position error resulting in the wingman reducing its airspeed to try and increase the separation distance and reduce the likelihood of a collision occurring. Secondly, as the lead aircraft performs one of the many tight turns, the heading difference between the lead aircraft and wingman increases whilst the separation distance is reducing. During the static waypoint analysis it was determined that when the Euclidean distance between the position and waypoint was small, sub-optimal paths occurred more frequently. Under this configuration, it can be expected that if the separation distance between the two aircraft becomes too small, the DNN will have an increased probability to produce either a sub-optimal path or a path which contains a dynamic deviation, allowing the wingman to divert away and increase its distance to the lead aircraft, which, is exactly what happened during the first test flight.

During the first test flight of the *Mission 2* profile, the configuration of the path planning algorithm did not adequately allow the wingman to track and follow the manned aircraft. Since the path planning algorithm was configured to perform partial re-planning every second with full re-planning every 10 seconds, the re-planning interval did not allow the wingman to perform the same turn rate as the lead aircraft. This resulted in the wingman performing larger radius turns, as seen during the early stages of flight in Figure 6.24a.

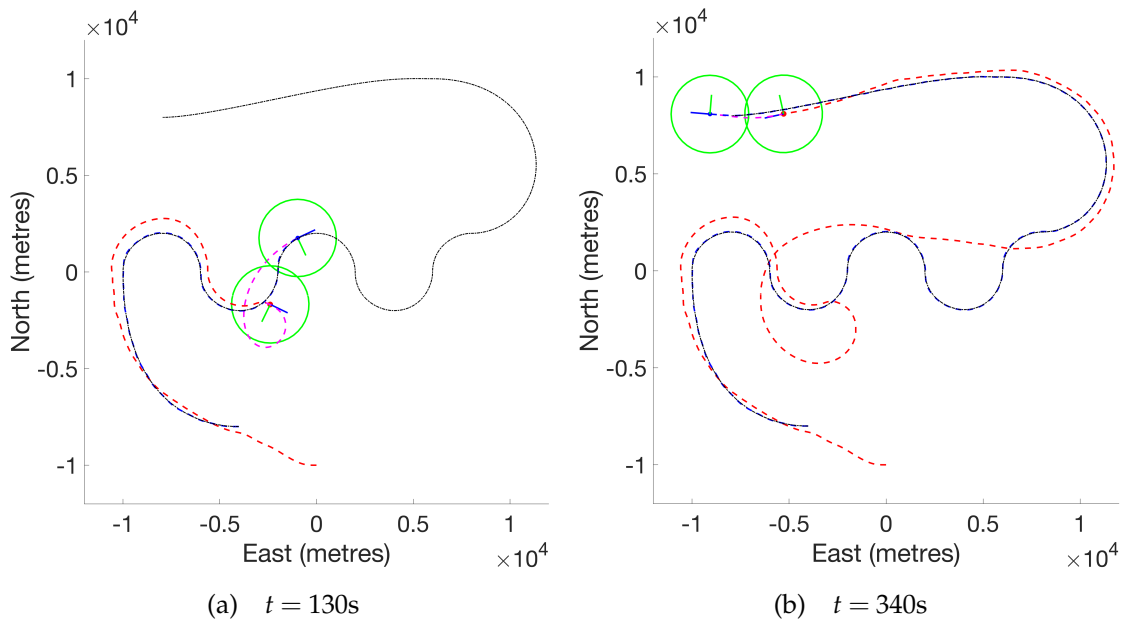


FIGURE 6.24 *Mission 2* Time sampled inertial position. Red dotted line indicates historic position data for the wingman, blue dotted line indicates historic position data for manned aircraft, black line indicates the path of the manned aircraft and the purple dotted line indicates the current path for the wingman

As the wingman was forced to initiate a larger radius turn, this increased the overall distance needed to complete the first constant-radius turn and positioned the wingman slightly further east for the second constant radius turn. As the wingman's position was compromised due to the larger radius initial turn, its position relative to the lead aircraft meant that a dynamic deviation was imminent, as shown in Figure 6.24a. This dynamic deviation caused the wingman to instigate a sharp turn to the right, increasing the relative distance and heading angle to the lead aircraft. As a result of the detour, the position error to the lead aircraft was extremely large and throughout the remainder of the manoeuvre the

wingman engaged in a catch-up procedure, ignoring the remaining tight radial turns.

Despite these factors, the wingman was successfully able to re-track the lead aircraft without flight becoming unstable and maintain the demanded separation distance of 4 km prior to the simulation ending, as seen in Figure 6.24b.

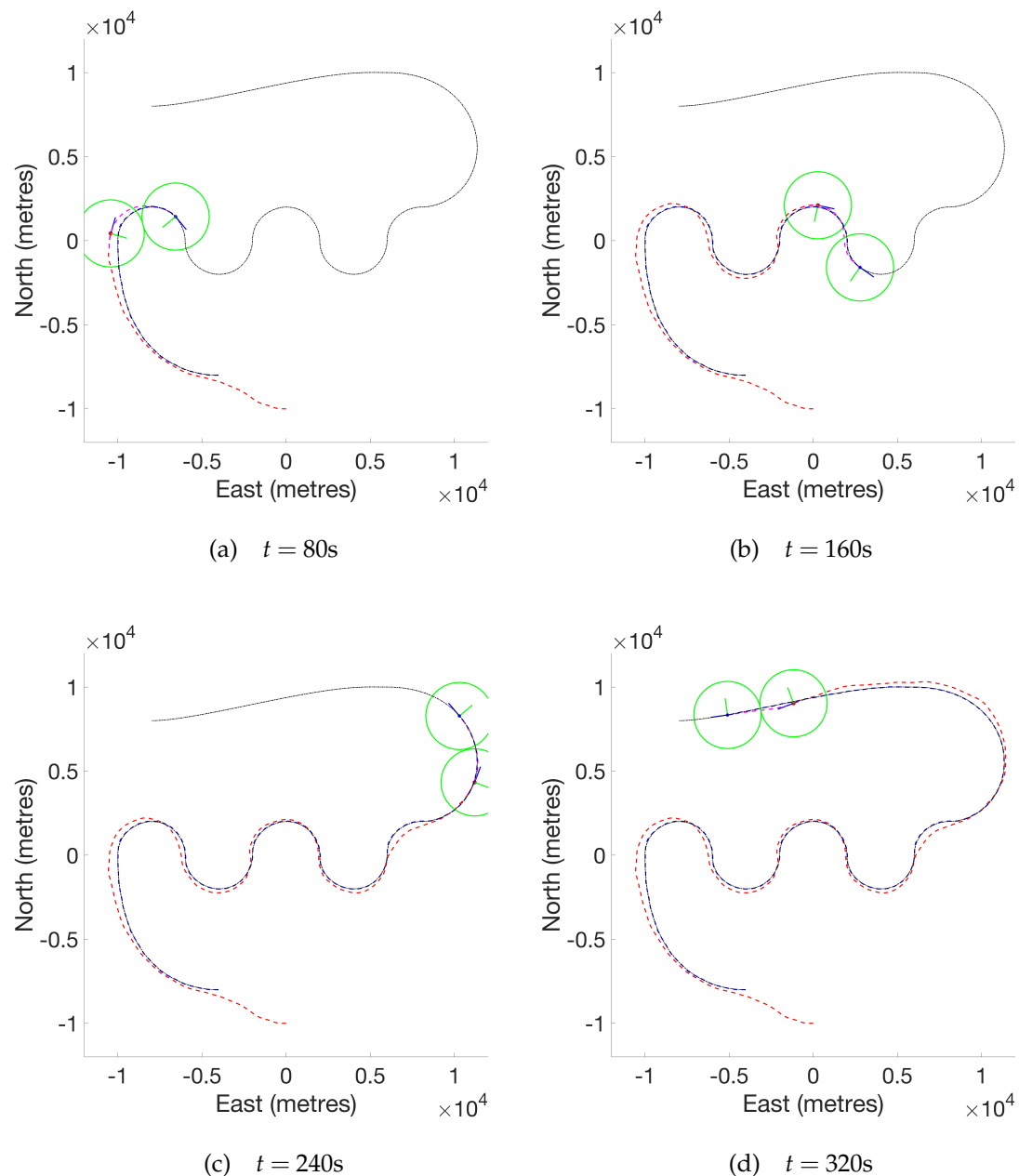


FIGURE 6.25 *Mission 2* time sampled inertial axis positions for second test flight with partial replanning every 0.5 seconds and fully replanning every three seconds

Due to the path re-planning issue seen during the first test flight of *Mission 2*, it was evident that adjustments needed to be made to speed up the rate of re-planning. Therefore, the partial re-plan time was reduced from one second to 0.5 second with full re-plan reduced from 10 seconds to three seconds. A sampled time breakdown of the second test flight for *Mission 2* can be seen in Figure 6.25.

Based on the four sampled inertial positions in Figure 6.25, conducted at various locations in time, it is evident that the changes made to the replanning intervals have provided significant benefits to the tracking performance of the wingman and have aided in its ability to follow and navigate closely to the lead aircraft. By performing path re-planning more frequently, the wingman was able to sustain much tighter radius turns whilst also maintaining a near perfect separation distance despite the complexity of the manoeuvre.

Additionally, the updates performed to the re-planning intervals enabled the same level of tracking accuracy regardless of the initial starting location and orientation of the wingman. For example, Figure 6.26 shows four time history samples from the inertial frame when the wingman started from a north position of -10 km, an east position of 10 km and with an orientation of 0° .

Although it can be seen that the path of the wingman during the initial constant-radius turn does not follow a similar flight path as the lead aircraft, this was solely due to the separation distance between the wingman and manned aircraft being initially large and allowing the autonomous aircraft to perform a different manoeuvre. However, the wingman was able to successfully complete the initial catch-up manoeuvre and then maintain an almost identical trajectory throughout the remainder of the manoeuvre and without any dynamic deviations occurring.

Throughout *Mission 2* the wingman demonstrated extremely robust operation despite the complexity of the mission. Although during the initial analysis the wingman was unable to follow the lead aircraft during the complex portion of the mission, this was solely due to the configuration of the path planning algorithm not performing both a partial and full re-plan at a high enough frequency. By remedying this, the wingman was successfully able to navigate through the complex and quick succession radial turns whilst maintaining a

near constant separation distance. It was also demonstrated how making adjustments to the initial start location of the wingman did not impact performance, with the wingman still able to track and follow the lead aircraft with a high level of accuracy.

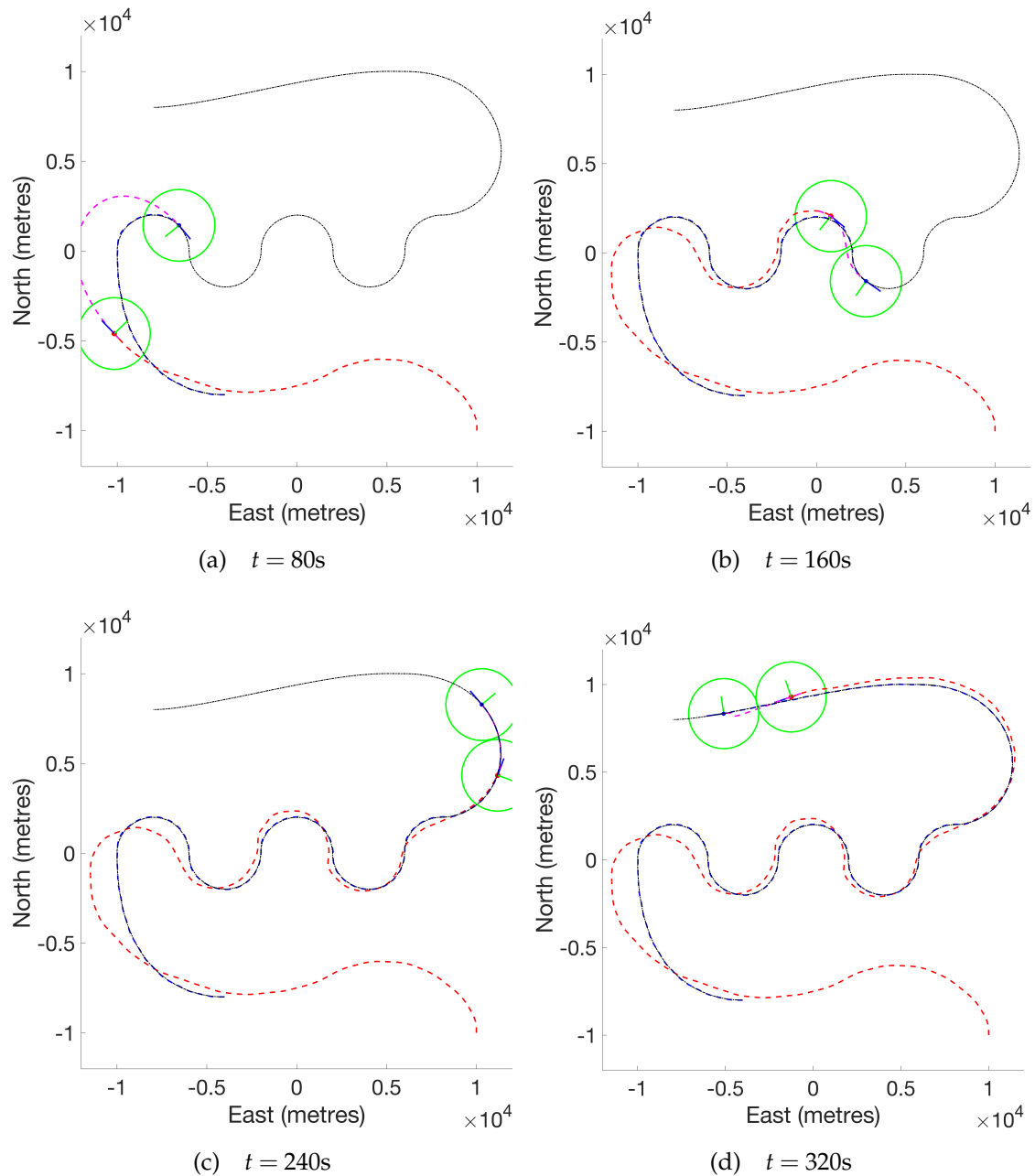


FIGURE 6.26 *Mission 2* time sampled inertial axis positions for second test flight with partial replanning every 0.5 seconds and fully replanning every three seconds

Mission 3

The final mission, denoted as *Mission 3*, was developed based on the fundamental principles behind an evasive fighter manoeuvre known as the flat scissor manoeuvre. During a flat scissor manoeuvre, two fighters occupy the same manoeuvre plane with each pilot attempting to get behind the other aircraft [140]. During this type of manoeuvre, the attacking aircraft initially has a lower airspeed, allowing the attacker to instigate a tighter turn, aiding in aligning its line-of-sight onto the defending aircraft. The defending aircraft then initiates a series of directional changes in an attempt to displace the attacking aircraft, allowing the defender to become the attacker.

To simulate this dog-fight scenario, the lead aircraft operates as the defender for the entire duration of the mission. This means that rather than trying to displace the wingman in order to become the attacker, it follows a pre-determined flight that simulates a displacement attempt, with the wingman attempting to track and follow the lead aircraft throughout. The flight path of the lead aircraft during *Mission 3* can be seen in Figure 6.27.

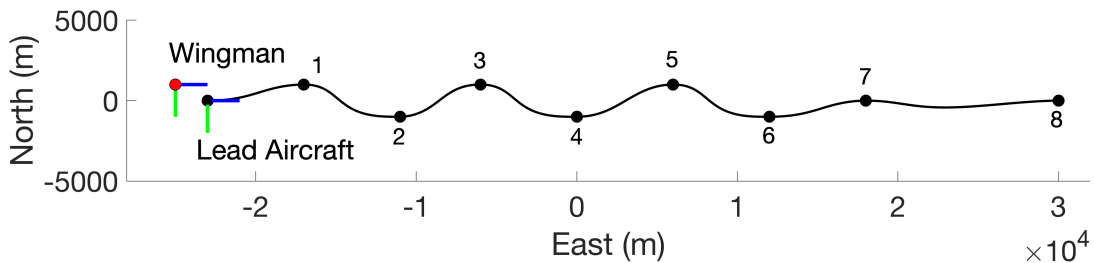


FIGURE 6.27 Classification DNN with dynamic waypoint path replanning *Mission 2*

For the first test flight of *Mission 3*, the lead aircraft was initialised with an airspeed of 275 m/s. This airspeed was chosen since the operational envelope of the computational model's aerodynamics are only valid during subsonic flight conditions. By initialising the lead aircraft with an airspeed of 275 m/s, this pushes the limits of the wingman towards the 300 m/s threshold during any catch-up manoeuvre. It can be seen from the mission schematic in Figure 6.27, that the initial location of the wingman brings both aircraft in close proximity. This aided in simulating the lower airspeed of the attacking aircraft during the early stages of the dog-fight, since the distance controller will reduce the throttle

demand to achieve the demanded separation distance.

From the results of the first test flight, the scissor manoeuvre performed by the lead aircraft proved ineffective when trying to evade the attacking wingman. The wingman was able to reduce its airspeed and then re-accelerate upon achieving the desire separation distance after approximately 20 seconds. This can be seen in Figure 6.28a, where 20 seconds into the simulation both aircraft halos are tangentially connected, indicating the demanded separation distance has been achieved.

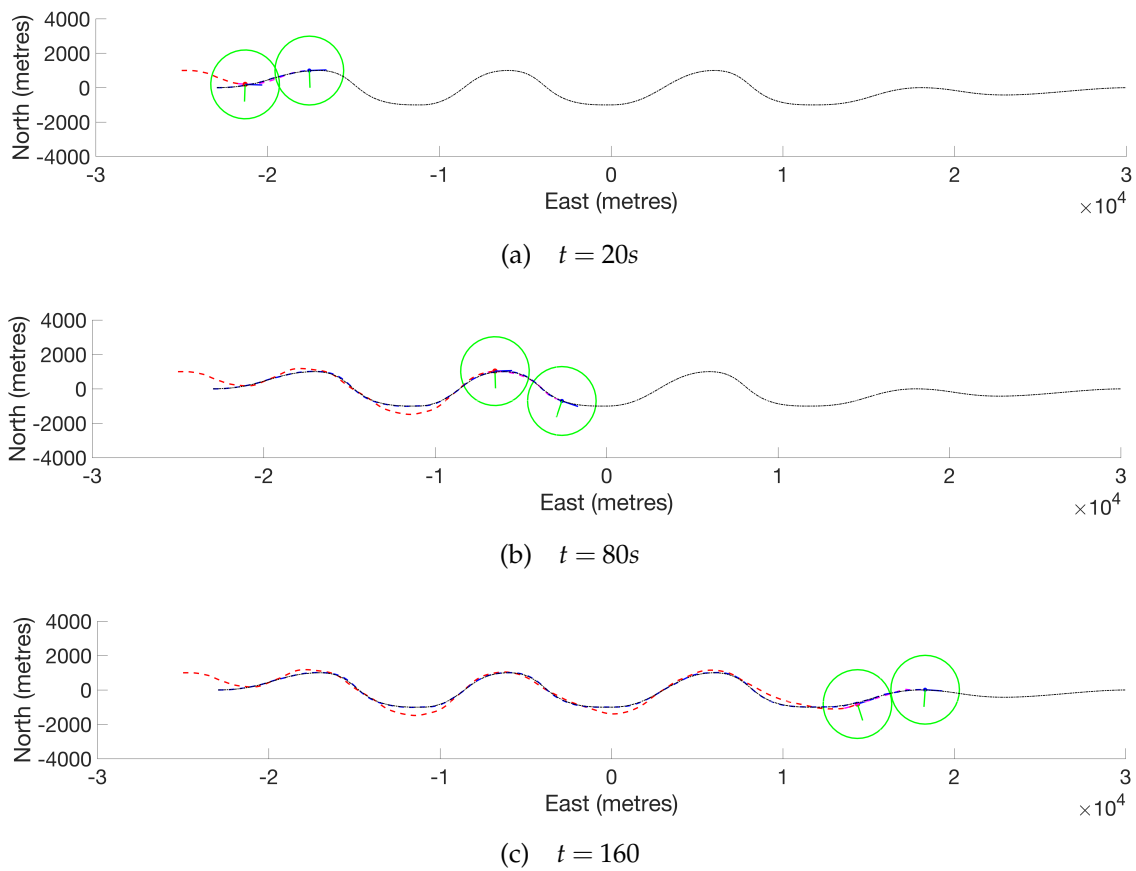


FIGURE 6.28 Mission 3 time sampled inertial axis position for first test flight

Although the wingman was successfully able to track the lead aircraft throughout the course of the entire manoeuvre, it is evident that due to the abrupt directional changes exhibited by the lead aircraft, the wingman was unable to sustain such tight turns, with the wingman performing slightly larger radius turns at waypoints two and four. By performing slightly larger turns at the second waypoint, the separation distance between the two aircraft increased by approximately 500 metres. This can be seen in Figure 6.28b, where both halos are slightly

disconnected due to the increased separation distance. This indicates that the manoeuvre performed by the lead aircraft was beginning to push the dynamic limits of the wingman due to rapid directional changes. Regardless, the wingman was able to regain the separation distance by increasing its airspeed, with the separation distance never exceeding more than 500 metres past the demand during the entire manoeuvre.

Despite the wingman demonstrating robust and stable flight conditions and successfully being able to track the lead aircraft during an evasive-type manoeuvre, the performance of the flight control system was far from perfect. Typically, an evasive-type manoeuvre would be performed at much closer distances during a dog-fight scenario. However, due to limitations in the number of classifications and the observational interval sizes, maintaining a dog-fight separation distance was not possible, since the path planning algorithm maintains a conservative approach to the dynamics of flight.

The applied path planning algorithm does allow paths to be performed which would heavily strain the dynamics of the aircraft and potentially induce instability. This limitation prevents the wingman from being able to path directly to a waypoint target when the euclidean distance is small, and instead opts to perform a much more dynamically conservative detoured manoeuvre. Whilst this does interfere with the wingman's ability to fly at very close separation distances to the lead aircraft, it ensures that the dynamics of the aircraft are not being too heavily utilised, with flight directional changes always within the dynamics of the aircraft.

However, to allow the wingman to operate more closely to the lead aircraft and better simulate a dog-fight encounter, rather than using the location of the lead aircraft as the waypoint target, the waypoint was artificially adjusted and placed at some distance along the lead aircraft's orientation vector. By offsetting the waypoint along the lead aircraft's line-of-sight, the waypoint target maintained the same orientation dynamics, but allowed a dynamically conservative path to be produced by predicting the future location for the lead aircraft.

This has been demonstrated in Figure 6.29, where the same dog-fight simulation was performed. However, the waypoint target was offset 3 km along the lead aircraft's orientation vector. This allowed a separation distance of 1 km to

be administered against the lead aircraft, allowing the wingman to fly four times closer whilst still maintaining similar path planning dynamics.

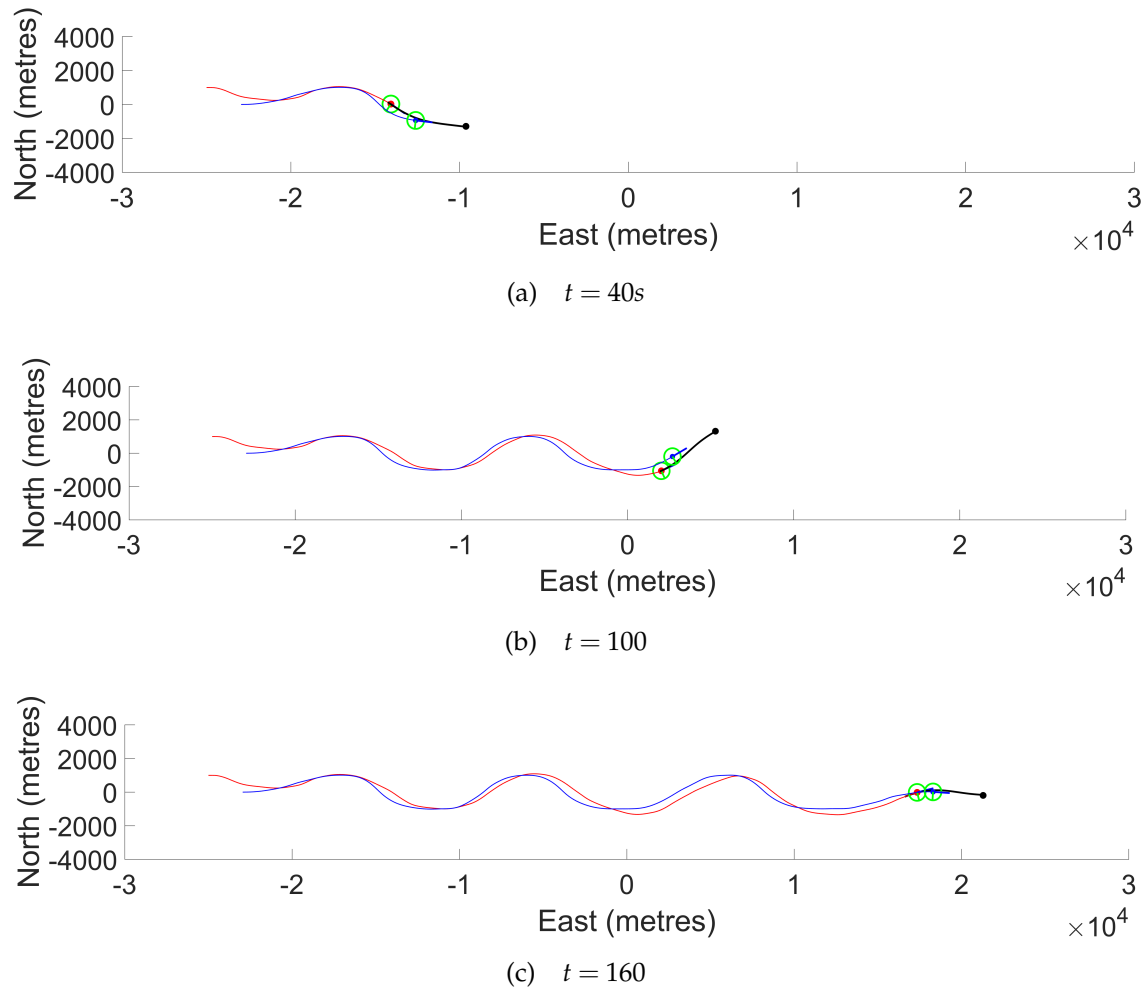


FIGURE 6.29 *Mission 3* time sampled inertial axis position for second test flight with offset waypoint target of 3,000 metres.

Although this has better demonstrated a more typical attacking/defending fighter scenario, it can be seen that the path taken by the wingman represents a lagged response when compared to the path taken by the lead aircraft. This lagged response indicates that due to the small separation distance, the wingman is unable to maintain the same flight path as the lead aircraft, since the path produced is for a point located 3 km ahead of the lead aircraft at any instance in time. Whilst this does not cause any issues to the stability of the wingman during flight, it reduces the necessary response time to react to movements performed by the lead aircraft. In addition, since this analysis does not include

a collision avoidance system, depending on the size of the offset, the path of the wingman could cross in front of the lead aircraft, causing numerous safety concerns and reducing the robustness of the solution.

6.8 Discussion

Throughout Chapter 6 control point optimisation was performed for constant-altitude flight using a bespoke fitness function. The control point optimiser was used to train a classification deep neural network, allowing intermediate observations to be classified as known data samples. The resultant network was trained on a dataset consisting of 2,640 classifications and applied to the computational flight physics model for both a static and dynamic waypoint analysis.

During the static waypoint analysis, random waypoint configurations were computed, with the aircraft model instructed to navigate along the path to the waypoint target. Solutions were identified as being dynamically feasible and optimal based on successfully navigating to the target and within a computed simulation time. Despite this analysis not incorporating online path re-planning, the aircraft successfully navigated to the waypoint target with 100% accuracy for 5,000 tested scenarios. In addition, 99.52% of solutions were deemed optimal, with a correlation between sub-optimal solutions occurring negatively along the north observation and as the east observation approached the origin.

During dynamic waypoint testing, three mission profiles were developed for simulating cooperative flight between a lead aircraft and wingman. During each mission, the location of the waypoint target used for path planning of the wingman, was determined based on the location and orientation of the lead aircraft. However, since this location was now moving through the inertial frame, obtaining a quantitative measure of the aircraft's success rate when navigating to the waypoint target was not possible. Therefore, each of the three missions was broken down into discussions with visual representations depicted of each aircraft's position at different instances in time.

During *Mission 1* it was demonstrated how the distance controller can be used to perform a catch-up manoeuvre based on the separation distance to the lead aircraft. Although this mission was extremely simple, it demonstrated how

robust the path planning methodology is, particularly since the wingman was initialised at multiple different locations and orientations.

During Mission 2, it was demonstrated how if path re-planning is not conducted at a high enough frequency, the wingman's ability to adequately follow the lead aircraft suffers, as dynamic deviations are more likely to occur. It was demonstrated how increasing the frequency allowed the wingman to manoeuvre tight radial turns, whilst also increasing and decreasing the airspeed based on the relative separation distance. This mission aimed to fine tune the internal parameters for path re-planning, and again, demonstrated robust path planning, closed-loop stability and tracking performance regardless of the wingman's initial location and orientation.

During *Mission 3* the wingman was situated as an attacking aircraft, with the lead aircraft acting as a defender during a flat-scissor manoeuvre. The initial analysis demonstrated how the evasive manoeuvre conducted by the lead aircraft was unable to offset the wingman into performing a dynamic deviation. However, despite these factors, the dog-fight scenario was not entirely accurate due to a large separation distance between the two aircraft.

It was discussed how the conservative nature of the path planning was to maintain robust and closed-loop stability during flight conditions. Therefore, exposing the wingman into a dog-fight scenario and operating within close proximity to the lead aircraft was not accomplishable, since dynamic deviations become extremely likely as the separation distance between the two aircraft reduces.

To remedy this, it was demonstrated how offsetting the waypoint target along the lead aircraft's heading vector by 3 km allowed paths to be relaxed, whilst also allowing the wingman to fly closer to the lead aircraft. Despite the wingman being able to successfully maintain a much smaller separation distance of 1 km, this method of operation reduces the robustness of the solution, and leaves little room for error and manoeuvrability of the flight control system to respond to an action performed by the lead aircraft. This is noticeable when analysing the different paths taken by the autonomous aircraft during each flight in *Mission 3*, since the wingman's historic position displayed a lagged flight path. In addition, it was demonstrated how performing this offset occasionally caused

the wingman to create a path ahead of the lead aircraft, potentially resulting in an intersection, and aligning the wingman on a collision course with the lead aircraft.

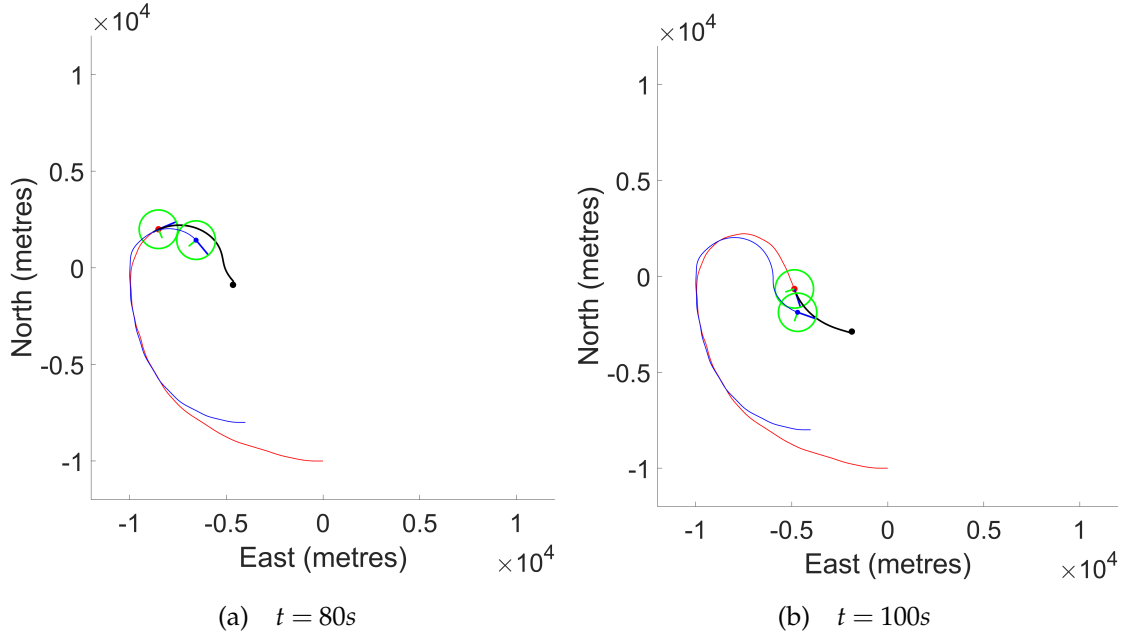


FIGURE 6.30 Classification network dynamic waypoint analysis - *Mission 2* performed with waypoint offset of 3 km ahead of lead aircraft

This can be seen in Figure 6.30 where the a waypoint offset of 3 km was applied to the *Mission 2* flight path. Initially, the wingman is performing adequately and maintaining a separation distance of 2 km to the lead aircraft. However, as the path planning algorithm is focusing on a waypoint target located along the lead aircraft's orientation vector, this creates a path offset and results in the wingman performing a larger radius turn 80 seconds into the simulation, as seen in Figure 6.30a. This larger radius turn positions the autonomous aircraft further east and as the lead aircraft performs a left turn 100 seconds into the simulation, the separation distance between the two aircraft begins to reduce with the path of the wingman intersecting the lead aircraft, as shown in Figure 6.30b.

This demonstrates how offsetting the waypoint does allow the wingman to fly closer to the lead aircraft, but reduces the robustness of the solution, since collision avoidance has not yet been factored into the flight control system.

In Chapter 7, an extension of the constant-altitude path planning framework

is produced, allowing paths to be optimised and produced during variable-altitude scenarios.

Chapter 7

Variable-Altitude Path Planning

During Chapter 6, a solution was created using an evolutionary path optimiser to train a classification deep neural network for constant-altitude flight. In Chapter 7, a climb rate analysis is performed to develop a relationship for predicting specific excess power during an altitude changing manoeuvre. The evolutionary algorithm is used to train a regression deep neural network, allowing path re-planning to be performed online for variable-altitude flight. The robustness and performance of the network is demonstrated during a static and dynamic waypoint analysis on the aircraft model.

7.1 Variable Altitude Problem

In Chapter 6 a solution was proposed and implemented onto the computational model, enabling constant-altitude path planning to be performed in real-time using a classification neural network. Although this approach could be considered as proof of concept as an intelligent path planning solution, the lack of change occurring to the altitude is quite a considerable downfall, given the aircraft is a highly manoeuvrable fast jet. Therefore, throughout Chapter 7, an analysis will be conducted into the dynamics associated with changing the altitude of the aircraft.

When attempting to increase altitude, the aircraft experiences an increase in its potential energy. This increase in potential energy can be achieved in one of three ways. Firstly, the aircraft converts kinetic energy into potential energy, essentially trading airspeed for altitude. Secondly, excess power available from the engine or chemical energy, can be converted into potential energy, allowing an

increase to the altitude level whilst maintaining a near-constant kinetic energy. Finally, the aircraft can use both the excess power and kinetic energy to increase altitude, achieving a high rate of climb.

When considering the first and third scenarios, both require converting kinetic energy into potential energy, which, requires extensive knowledge of the capabilities and performance characteristics of the aircraft. In addition, the number of climb scenarios utilising both kinetic and chemical energy is infinite, making performance predictions before a climb more complex. During manned flight, the pilot predicts how much kinetic energy will be required to complete a specific manoeuvre. If the manoeuvre is deemed too demanding and requires a lot of kinetic energy, the aircraft could become susceptible to stall resulting in dynamic instabilities. During this scenario the pilot may perform an intermediate altitude decreasing manoeuvre, allowing potential energy and chemical energy to be converted into kinetic energy at a high rate. This is a very complex decision making situation and is one of the reasons pilots are subjected to immense training regimes tailored towards operating a specific type of aircraft. These conditions pose numerous complications for autonomous flight, and therefore have not been initially considered during variable-altitude path planning. Therefore, for the purpose of this research, the focus towards altitude-changing manoeuvres focuses on using chemical energy to increase the altitude of the aircraft, whilst maintaining near-constant kinetic energy.

7.2 Climb Rate

To understand the performance capabilities of the aircraft during altitude increasing manoeuvres, a climb rate analysis was conducted at various airspeeds and altitudes. To perform the analysis, a climb rate controller was implemented onto the aircraft model wrapped around the pitch rate controller. The schematic for the climb rate control system can be seen in Figure 7.1.

Since the climb rate controller is controlling a slowly varying state dynamic, integral wind-up is a concern. Integral wind-up was also a concern during the development of the airspeed controller (Section 4.5) and the distance controller (Section 6.6), with the applied solution to schedule the integral component of the

PID controller. This meant that although the general controller architecture was PID, the internal controller states were scheduled based on the position error, meaning the controller would act as a proportional-derivative (PD) controller during high position error scenarios. The integral component only engages during low position errors to reduce the steady-state error.

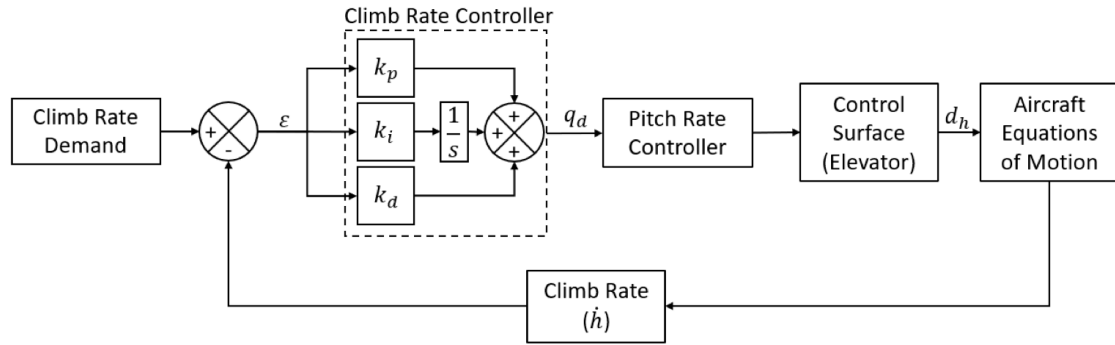


FIGURE 7.1 Climb rate control system block diagram

Additionally, when a climb rate demand is applied to the controller, there exists a large instantaneous position error. This large error drives the output of the climb rate controller and demands a large unsustainable pitch rate, negatively impacting the longitudinal dynamics of the aircraft. Rapid changes to the pitch rate controller can cause a sudden change to the angle of attack, which increases the drag and reduces the airspeed. Whilst the reduction to the airspeed will eventually be alleviated by the airspeed controller, the slow engine dynamics cause a lagged response and also reduce the necessarily excess power available. This produces an undesirable manoeuvre since the corresponding increase in power is being used to compensate for the loss in kinetic energy as opposed to directly increasing the potential energy, leading to a less efficient climb. Therefore, a dynamic limit was imposed onto the climb rate controller, restricting the pitch rate demand to ± 15 deg/s. Whilst this will increase the rise time to achieve the demanded climb rate, it poses more advantages with respect to energy utilisation, particularly when developing a climb rate performance envelope for the aircraft.

The process for optimising the PID controller gains was identical to the methodology performed in Chapter 4. Prior to optimisation, it was observed that when operating at airspeeds lower than 80 m/s the aircraft's ability to climb was

very poor. This was due to the aircraft having to sustain a relatively high pitch angle in order to generate enough lift to achieve equilibrium flight due to the low airspeed. In addition, this became more problematic due to the decreasing density of air as a function of increasing altitude. Therefore, when optimising the climb rate controller, it was not feasible to consider flight below 80 m/s, as this low airspeed not only negatively impacted the longitudinal dynamics but was also not a sustainable airspeed at higher altitude levels.

Optimisation was conducted for the same trim points identified in Chapter 4 during optimisation of all onboard autopilot systems. The bat algorithm (BA) was selected as an appropriate optimisation algorithm, with fitness measured using the sum-squared-error against the applied reference signal. A penalty function was imposed if the steady-state error of the time-response exceeded 0.001. Optimisation was first performed to establish a neutral operating gain surface, enabling the controller to sustain a climb rate of 0 m/s across all flight levels. For trim points, a five second simulation was conducted, where the demanded climb rate was set to be zero for the duration of the simulation.

Upon optimising the neutral surface, the climb rate during the initialisation period was controllable. This enabled optimisation to be performed when specific input demands were applied to the climb rate controller. During optimisation of the pitch rate controller in Section 4.4.3, an analysis was conducted to determine the maximum sustainable pitch rate, which was then applied to the reference signal during optimisation. However, repeating this process is not as straight forward for the climb rate controller. During any simulation where the aircraft is performing a climb, the environmental conditions vary throughout the manoeuvre. As the altitude increases the air become less dense, reducing the lift force produced by the aerodynamic surfaces and thrust produced by the engine. The culmination of these factors means that the longer the sustained climb, less thrust and lift are produced, and eventually, no matter what the climb rate is, the aircraft will become susceptible to stall, particularly if the altitude level approaches the flight ceiling.

Whilst it is not expected for the aircraft to reach these conditions due to the scope of the research, it is something that needs to be considered. Furthermore, when considering the maximum climb rate for a given airspeed and altitude,

the maximum climb rate exists when the aircraft is flying with a pitch angle of 90 degrees. Although this orientation will produce the largest climb rate, the thrust-to-weight ratio of the aircraft does not allow this to be sustained, and the aircraft will rapidly stall.

Therefore, rather than determining a maximum sustainable climb rate, an initial test was conducted to determine a suitable demand. Starting with an airspeed of 80 m/s, optimisation was performed where the reference signal indicated a climb rate demand of 40 m/s, half that of the airspeed. Initially, the reference signal demand was 0 m/s, allowing the aircraft to complete dynamic initialisation. After five seconds, the demanded 40 m/s was applied to the climb rate controller for 15 seconds.

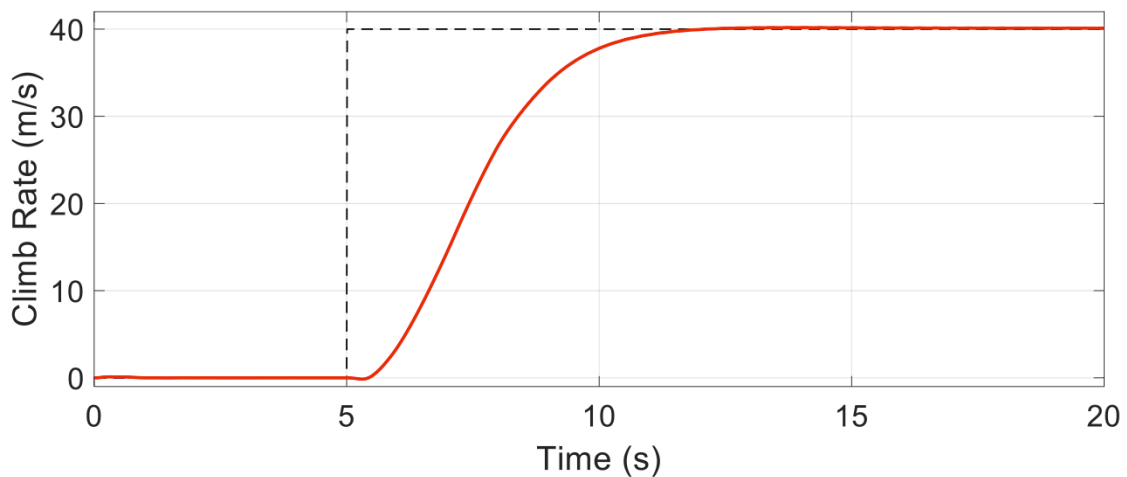


FIGURE 7.2 Climb rate optimised response for an airspeed of 80 m/s starting with an initial altitude of 0 m and a demanded climb rate of 40 m/s

The fitness function for optimising the climb rate controller needed to be carefully selected, since at the point the demanded climb rate was applied a large instantaneous error occurs within the controller. This large error drives the output of the climb rate controller resulting in the pitch rate either increasing or decreasing depending on the applied demand. This meant that during periods where a large error is present, a sum-squared-error fitness function would drive the gains to reduce the error quickly, favouring a fast and aggressive control system. This meant that the BA would favour reducing the rise-time of the response and allowing for moderate overshoot, provided the magnitude of the overshoot outweighed the impact associated with an initially large position error.

In addition, the sum-squared-error fitness function damps the steady-state error, provided it is less than one, and cumulatively outweighed the impact associated with the initial large position error. This meant that use of the sum-squared-error fitness function was not appropriate, with a weighted sum-squared-error (WSSE) fitness function selected. Use of a WSSE fitness function allowed specific regions of the time response to be disregarded, scaled up or scaled down. Additionally, fixed penalties were imposed if overshoot was present, or if the steady-state error of the time-response exceeded 0.001.

The optimised time-response for an airspeed of 80 m/s at a climb rate of 40 m/s is shown in Figure 7.2. It can be seen that the rise-time of the response is not overly aggressive with a near constant acceleration resulting in no measurable overshoot. The steady-state error is practically zero and is sufficiently maintained throughout the manoeuvre, allowing the aircraft to sustain a constant climb.

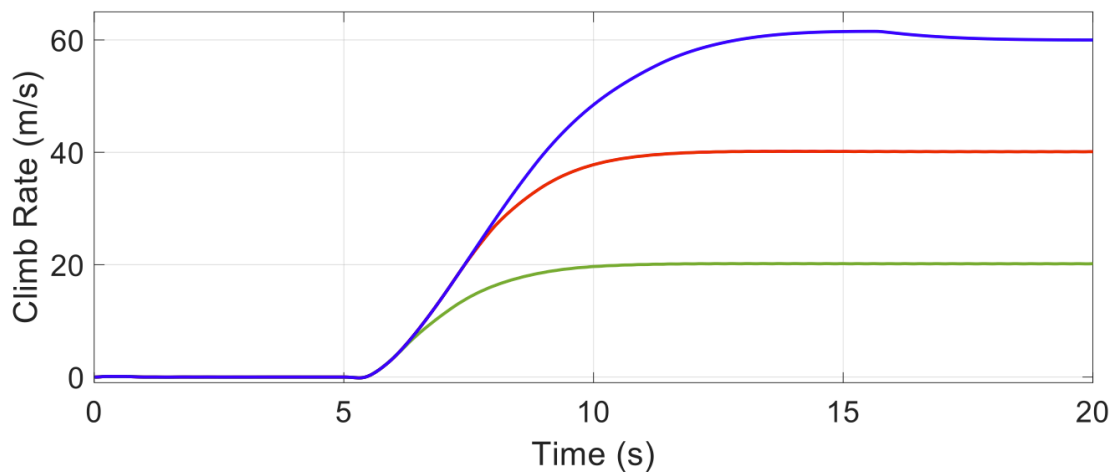


FIGURE 7.3 Climb rate comparison conducted at 80 m/s for input demands 20 m/s, 40 m/s and 60 m/s

When applying input demands of different amplitudes to the climb rate controller whilst using the optimised gains for a climb rate demand of 40 m/s from Figure 7.2, it can be seen in Figure 7.3, that the response is consistent despite optimisation having not been performed for these inputs. When demanding a climb rate of 20 m/s there is no overshoot and the steady-state error is zero. However, when applying a climb rate demand of 60 m/s, which is larger than the optimised demand, a small overshoot is present. This overshoot is less than

3% and is damped relatively quickly, with the controller maintaining a steady-state error of zero.

Although an overshoot is present when applying a demand that is larger than the optimised demand, the response and performance were deemed acceptable, since the climb could be achieved and sustained with very little dynamic impact. In addition, since this climb rate of 60 m/s was achieved for an airspeed of 80 m/s, it can be assumed that achieving a higher rate of climb would not be possible, since this would result in an almost 90 degree climb, and as previously explained, the thrust-to-weight ratio does not allow this.

Having validated the optimisation process for a single airspeed value where the optimised climb rate demand is half that of the airspeed, this same process was repeated for all of the optimisation flight speed trim points, conducted with an initial altitude of 0 metres. All of the optimised responses can be seen in Figure 7.4.

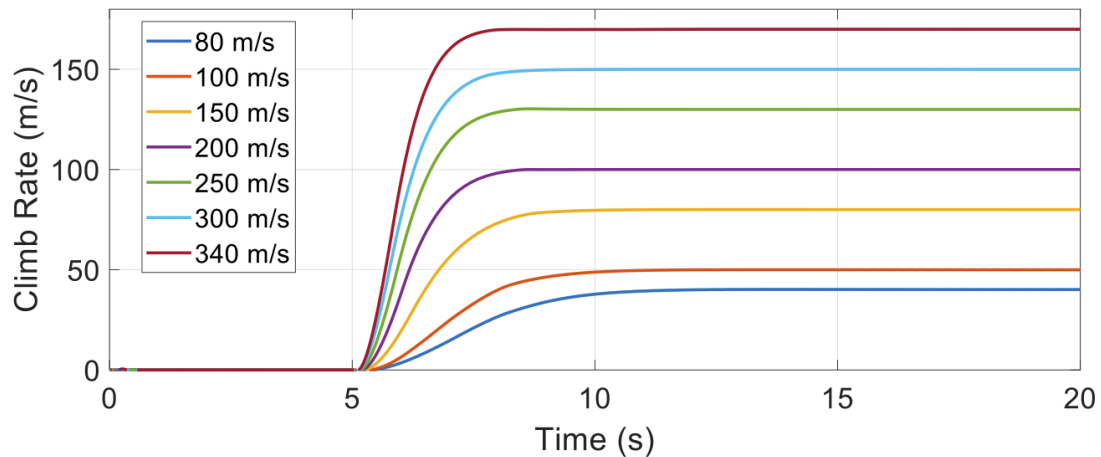


FIGURE 7.4 Climb rate optimised responses for all optimisation flight speed trim points starting with an altitude of 0 metres

Previous optimisation methods aimed at producing a series of gain surfaces that are scheduled based on a set of dynamic states. Since the primary function of the climb rate controller impacts the altitude level of the aircraft, scheduling the gains based on altitude becomes problematic, since there exists a reduction in the specific excess power available as the altitude increases due to reducing air density. Specific excess power (P_s) is the relationship between the thrust available (T_A) and the thrust required (T_R) per unit weight, as seen in Equation

7.1. Specific excess power is often used to describe certain performance characteristics in flight dynamics as this quantity represents the ability of the aircraft to change its total energy per unit weight, hence the weight normalisation in Equation 7.1.

$$P_s = \frac{(T_A - T_R)V}{W} \quad (7.1)$$

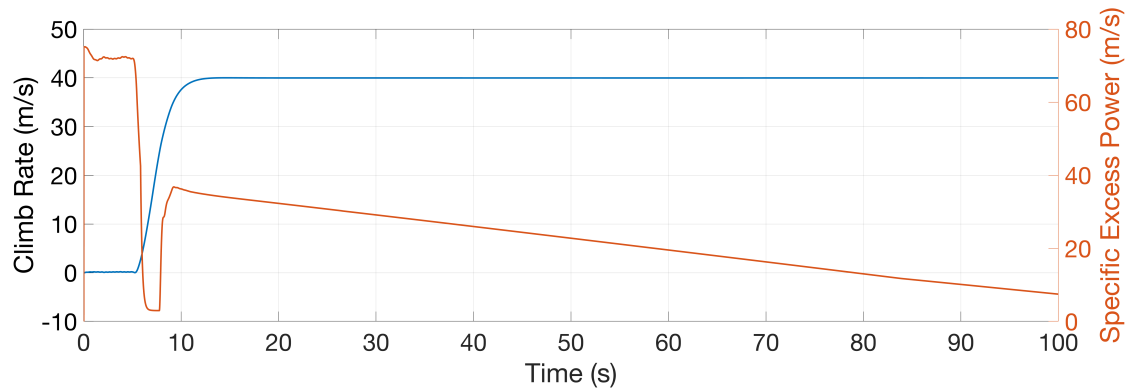
Consider the following scenario. The aircraft is initialised with an airspeed of 80 m/s with an altitude of 0 km. Based on the trim conditions for the aircraft, equilibrium flight is achieved when the pitch angle is approximately 10 degrees and a throttle stick setting at 12%. Based on its current dynamics, the aircraft has a specific excess power (SEP) of approximately 73 m/s. The aircraft is demanded to hold trim conditions for the first 10 seconds before a demanded climb rate of 40 m/s is applied to the controller. The climb rate and SEP time history over the course of the manoeuvre can be seen in Figure 7.5a.

Upon applying a climb rate demand to the controller, a large instantaneous decrease occurs to the SEP. This decrease occurs as the aircraft begins to pitch up, increasing the drag force and resulting in a brief loss of airspeed. This brief loss to the airspeed causes the airspeed controller to increase the throttle towards 100%, reducing the difference between the available thrust and maximum thrust, thus reducing the resultant SEP. Upon achieving the demanded climb rate of 40 m/s, the climb rate controller demands a near-constant pitch angle of 38 degrees.

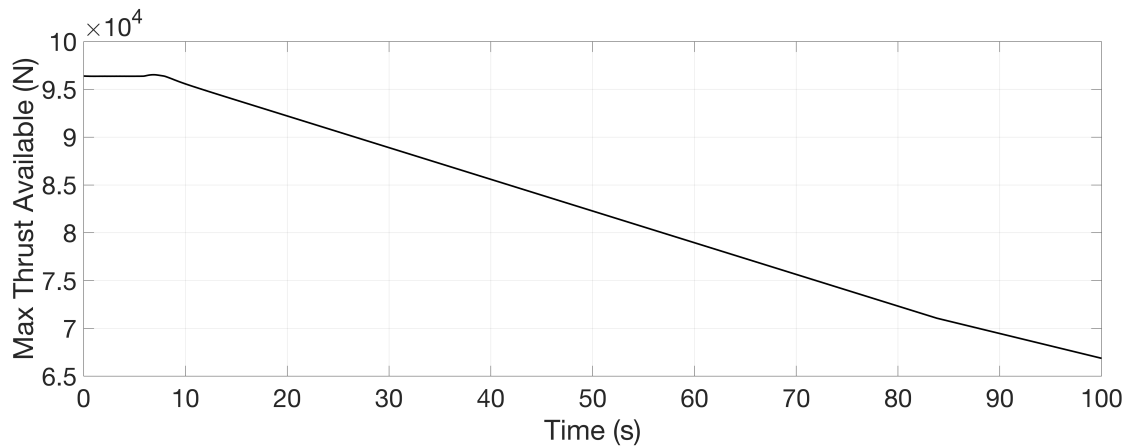
At this stage of flight the airspeed controller is able to maintain the demanded airspeed and reduces the throttle to prevent further translational acceleration. This causes the SEP to suddenly rise as the difference between the maximum thrust and current thrust increases, as depicted in Figure 7.5a, with the peak rise in SEP coinciding with having achieved the demanded climb rate.

As the aircraft begins to settle into a state of constant climb, the SEP available is approximately 40 m/s. This indicates that the aircraft is able to sustain this constant climb rate provided the thrust produced by the engine does not degrade. However, as the aircraft continues to climb the density of the air decreases, reducing the maximum thrust available over time, as seen in Figure 7.5b. As a result of the maximum thrust available decreasing, so does the SEP based on Equation 7.1.

During optimisation of the climb rate controller, it was assumed that the



(a) SEP time response



(b) Maximum thrust available over time

FIGURE 7.5 Initial climb rate manoeuvre conducted with an airspeed of 80 m/s at an initial altitude of 0 metres

aircraft could sustain the demanded climb rate for prolonged periods. However, based on this current analysis, as the aircraft continues to climb the density of air decreases and the maximum thrust available also decreases. This means that regardless of the demanded climb rate, as the aircraft continues to climb the SEP will eventually approach zero. At this point, the aircraft has depleted its excess power and can no longer convert chemical energy into potential energy, resulting in the aircraft beginning to convert kinetic energy into potential energy.

When this occurs, depending on the initial conditions, climb rate demand and current altitude level, stall can be expected to occur quite imminently, especially at lower airspeeds or high climb rate demands. This means that whilst the SEP can be used as a factor for determining whether or not the aircraft can sustain a climb, this climb can not necessarily always be guaranteed depending

on the longevity or severity of the manoeuvre. However, by mapping the SEP at numerous airspeeds and altitudes for various climb rates, the performance of the aircraft can be predicted based on the linearly decreasing slope of the SEP.

To perform a predictive climb rate analysis, the performance was mapped over the flight envelope at various different climb rate demands. Starting with an airspeed of 200 m/s, the aircraft was subjected to a 50 second simulation, which included a 10 second dynamic initialisation period followed by a 40 second constant climb. Between simulations the climb rate was incrementally increased until the maximum climb rate was determined.

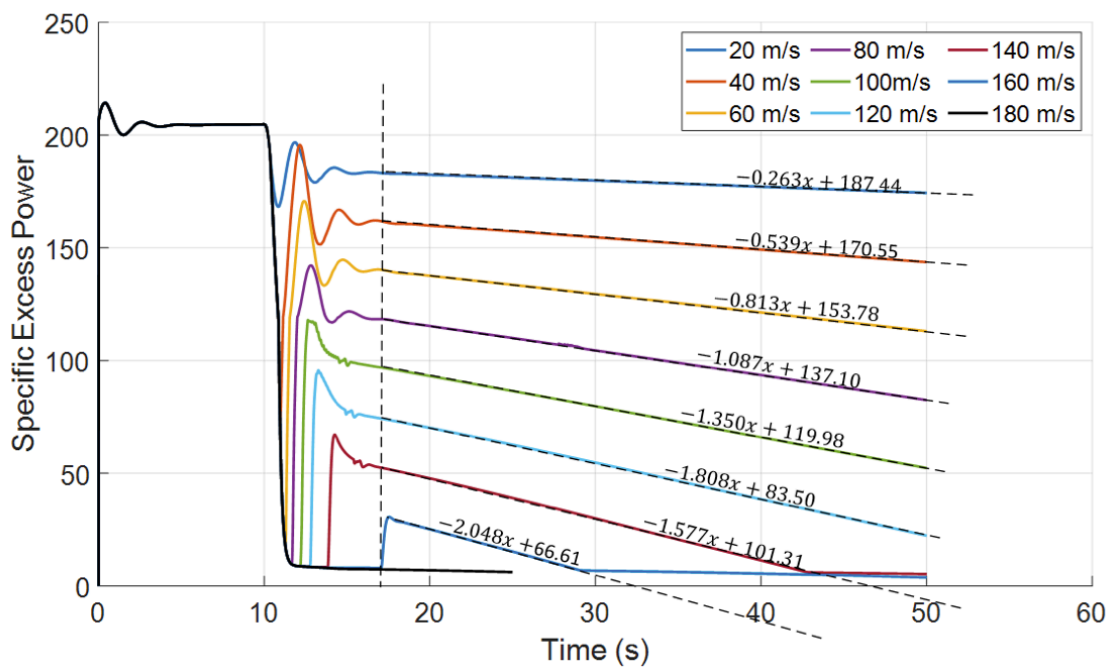


FIGURE 7.6 Specific excess power analysis for an airspeed of 200 m/s with an initial altitude of 0 metres for various climb rate demands

During the climb period the slope of the SEP is determined and used as a predictor for how long the climb can be sustained based on the flight dynamics. The SEP and linear equations corresponding to the SEP degradation over time can be seen in Figure 7.6 for an airspeed of 200 m/s where the climb was performed at an initial altitude of 0 metres. It is important to note that the the equations displayed in Figure 7.6 have been offset by 10 seconds due to the applied dynamic initialisation period at the start of the simulation, plus the rise-time during the climb demand. Therefore, the adjusted components of the equations can be seen

in Table 21.

Using the compensated offset components and applying directly to the straight-line equations, the maximum climb time is determined. It can be seen from climb rates of 140 m/s, 160 m/s and 180 m/s that when the SEP drops below 6 m/s, the SEP is no longer sufficient to increase the potential energy of the aircraft. This indicates that continuing to sustain this climb rate in this regime will result in using kinetic energy to sustain the applied climb.

TABLE 7.1 Compensated offset equations from Figure 7.6

Climb Rate (m/s)	T_1	$P_{s,1}$	T_2	$P_{s,2}$	Gradient	Y-Intercept	Estimated Time (s)
20	0	182.8	32.35	174.3	-0.26275	182.8	672.88
40	0	161.1	32.47	143.6	-0.53896	161.1	287.78
60	0	139.4	32.34	113.1	-0.81323	139.4	164.04
80	0	117.8	32.26	82.73	-1.0871	117.8	102.84
100	0	96.06	32.29	52.47	-1.34995	96.06	66.71
120	0	73.32	32.26	22.45	-1.57688	73.32	42.69
140	0	51.39	32.25	6.19	-1.40155	51.39	25.11
160	0	30.44	32.4	7.42	-0.71049	30.44	11.93
180	0	7.27	32.18	6.2	-0.03325	7.27	8.44

Having established a climb performance for one specific airspeed, it was important to understand how the performance varied when the initial climb started at higher altitudes. Figure 7.7 shows three SEP time plots, where each climb rate manoeuvre was conducted at the same airspeed of 200 m/s, and with the same climb rate demand of 20 m/s. However, the initial altitude for which the climb commenced is set at 0 km, 2 km and 8 km.

From the analysis it is evident that during the dynamic initialisation period where the climb rate demand was 0 m/s ($t < 10$), the nominal SEP value varies due to the altitude level. This is expected since the SEP calculation is a function of current thrust and maximum available thrust, which has already been confirmed to degrade based on altitude due to a reduction in the density of air.

One important characteristic from this analysis occurs when focussing on the

impact the pitching rate has on the SEP. It was previously explained that upon applying a climb rate demand, the aircraft begins to pitch up resulting in a brief loss of airspeed, as indicated by a large dip in SEP. As the aircraft reaches the desired climb rate, the pitch rate reduces to zero and the SEP increases as the airspeed controller reduces the throttle to maintain the airspeed. Although it can be seen that increasing the altitude level increases the severity of the SEP dip, the overall magnitude of the SEP drop between the nominal value and the constant climb value (d_1 , d_2 and d_3), is constant across all three altitudes, with an SEP drop of approximately 20.5 m/s.

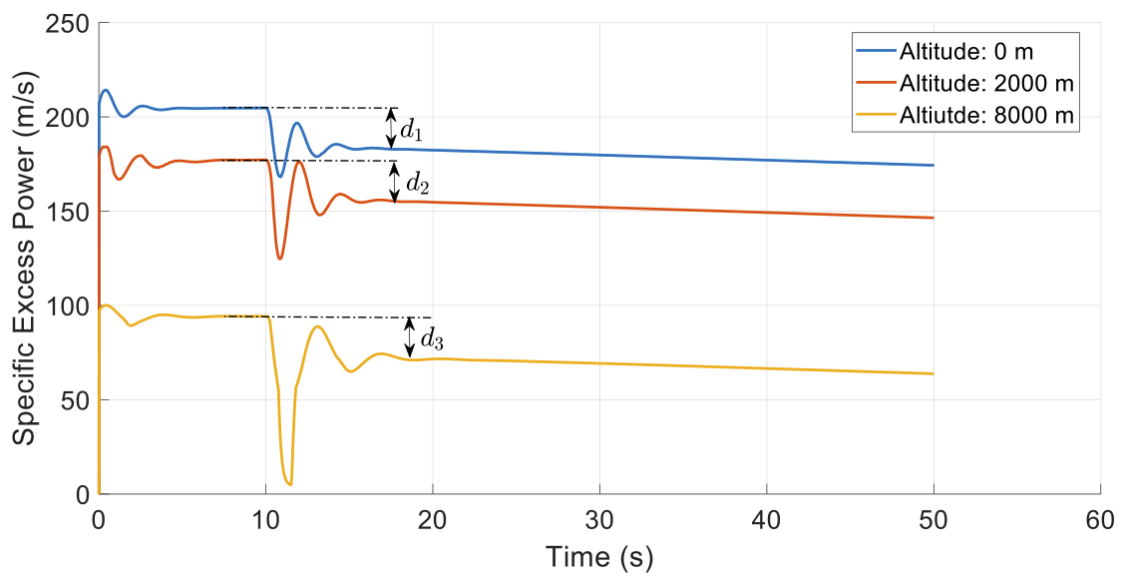


FIGURE 7.7 Specific excess power analysis for a climb rate demand of 20 m/s with an airspeed of 200 m/s. This analysis was conducted at three different initial altitudes, which are indicated in the legend

Furthermore, the gradient of the SEP decay is also near constant for all three simulations, regardless of the altitude, with an approximate gradient of -0.26 m/s^2 . This means that although the SEP power of the aircraft does degrade as the attitude level increases, the analysis performed in Figure 7.6 can be utilised across the flight envelope, as long as the initial starting SEP value, which is determined based on the initial altitude level, is adequately offset and compensated for.

To compensate for the the initial SEP value for all altitudes and airspeeds, an analysis was conducted to determine all of the nominal SEP values at specific

locations throughout the flight envelope. This analysis was conducted by initialising the aircraft with a specific airspeed at a certain altitude. The climb rate controller was set to demand a climb rate of 0 m/s for the entire duration of the simulation.

During the initial stages of the simulation the aircraft is in a state of dynamic initialisation as the trim conditions are loaded, but the applied controllers are attempting to gain control and maintain stability. Therefore the SEP during this period was neglected, and the longitudinal dynamics were allowed to settle, before the SEP value was determined. The results of all the nominal SEP values throughout the operational envelope can be seen in Table 7.2.

TABLE 7.2 Specific excess power values (m/s) for nominal flight conditions at various airspeeds and altitudes. These values were obtained by flying the aircraft at the specified states with a climb rate of 0 m/s for 50 seconds. For any flight conditions that were not attainable, the SEP has been left blank

Altitude (m)	Airspeed (m/s)								
	40	60	80	100	150	200	250	300	340
0	24.83	47.22	72.50	96.90	154.31	204.72	248.64	292.95	325.92
1000		39.23	64.04	86.91	141.49	191.05	234.56	279.28	309.45
2000		35.04	55.40	76.76	128.51	177.17	221.21	264.60	293.62
3000		25.79	46.54	66.42	115.29	163.11	207.45	248.96	274.26
4000			39.05	57.79	103.96	149.29	193.05	233.74	257.67
5000			30.26	49.00	92.44	135.19	178.19	218.91	240.37
6000			22.10	39.92	80.52	120.77	162.88	204.16	222.09
7000			14.24	31.10	69.16	107.70	147.98	189.58	204.30
8000			6.38	21.92	57.60	94.27	132.75	174.45	187.11
9000				11.56	45.67	80.28	115.97	158.66	168.98
10000				2.41	35.47	67.92	100.68	141.93	149.80

Having determined the nominal SEP values for various flight conditions throughout the operational envelope, the same procedure as seen in Figure 7.6 was performed for all of the additional airspeeds. This enabled the performance to be mapped for various climb rates throughout the desired airspeed range,

allowing the performance of the individual state dynamics to be predicted.

To test this methodology an analysis was conducted to predict the performance of the aircraft during a climb at an untested altitude of 3.5 km, where the demanded climb rate was 100 m/s. The airspeed controller was engaged throughout the simulation with a demanded value of 200 m/s. However, prior to the the simulation, the performance was predicted.

Firstly, interpolation was performed on the data in Table 7.1, allowing the initial and final points corresponding to the gradient of linear SEP decay to be calculated for the given climb rate. Since the climb rate being tested was 100 m/s, which, was already an established quantity, the values were looked up. The start and final points in the line are:

$$\begin{aligned} T_1 &= 0 \text{ s} & P_{s,1} &= 96.06 \text{ m/s} \\ T_2 &= 32.29 \text{ s} & P_{s,2} &= 52.47 \text{ m/s} \end{aligned}$$

with a gradient

$$\frac{dP_s}{dt} = -1.35 \text{ m/s}^2$$

Since these values correspond to a climb initialised at 0 metres, the nominal SEP value ($P_{s,nom}$) is obtained from Table 7.2 at these given state dynamics, where ($P_{s,nom}$) = 204.72 m/s. By obtaining this nominal SEP value corresponding for flight initialised with an altitude of 0 metres, the applied offset, τ_1 , is determined based on y_1 . Therefore:

$$\tau = P_{s,nom} - P_{s,1} = 207.74 \text{ m/s} - 96.06 \text{ m/s} = 108.66 \text{ m/s}$$

The second offset (τ_2) requires interpolation of the data in Table 7.2 to obtain the initial SEP value for the initial altitude of 3.5 km, where $\tau_2 = 156.2 \text{ m/s}$. This value is used to predict the SEP constant climb initial value based on the offset.

$$P_{s,alt} = \tau_2 - \tau_1 = 156.2 \text{ m/s} - 108.66 \text{ m/s} = 47.54 \text{ m/s} \quad (7.2)$$

The value $P_{s,alt}$ is the predicted SEP value for a climb rate of 100 m/s, starting with an altitude of 3.5 km and at an airspeed of 200 m/s. This value, $P_{s,alt}$ acts as the initial value for which the SEP decays throughout the climb.

During the initial analysis, it was determined that as the SEP approached a value of 6 m/s, the SEP would level out and maintain a near constant value.

This indicated the aircraft is now converting its kinetic energy to sustain the climb. Therefore, by applying an approximate value of 6 m/s as the SEP limit, the maximum climb time under SEP can be predicted as:

$$T_{\text{climb}} = \frac{6 \text{ m/s} - P_{s,\text{alt}}}{dP_s/dt} = 30.623 \text{ s} \quad (7.3)$$

Having predicted the climb time and the SEP value at the start of the climb for an unknown altitude, a simulation was performed to measure the accuracy of the prediction. The time-response for the simulation is shown in Figure 7.8 with all of the necessary components from the approximation displayed and labelled as an overlay on the time response.

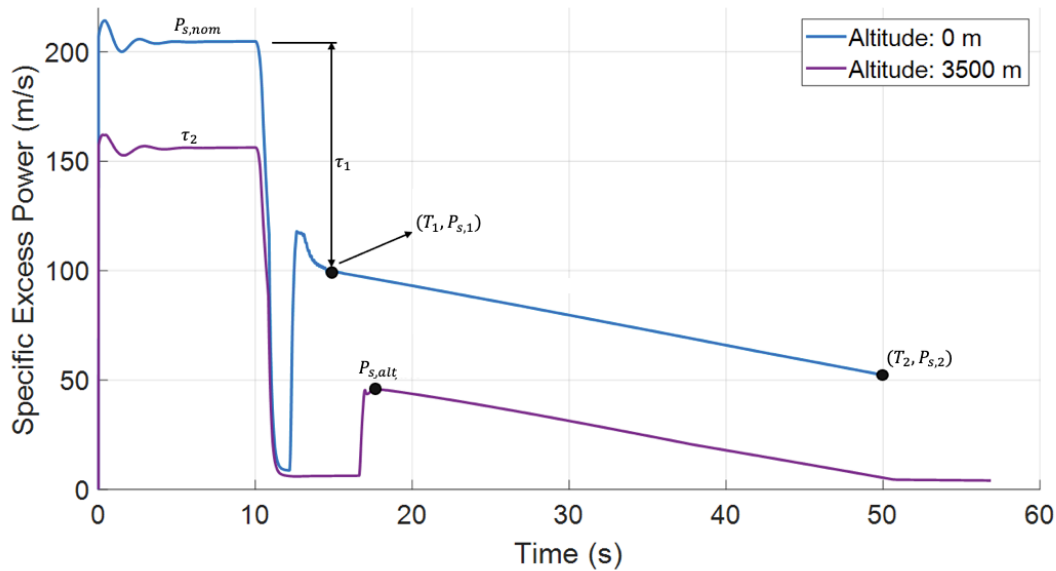


FIGURE 7.8 SEP simulation analysis for an airspeed of 200 m/s conducted with an initial altitude of 3.5 km. The time responses plotted represent the manoeuvre conducted at an initial altitude of 0 metres and then at 3.5 km.

When analysing the results from the simulation, it is evident that the predicted SEP values achieved a high level of accuracy. When the climb rate was applied during the 3.5 km altitude manoeuvre, the SEP value dropped to a value of 46.91 m/s, while the predicted SEP value was 47.54 m/s, giving an error of approximately 1%. However, the estimated climb time was not as accurate.

During the estimation of the climb time a minimum value of 6 m/s was assumed as this was approximately the minimum value that was sustained during the climb time analysis in Figure 7.6. During the manoeuvre at an altitude of

3.5 km, the SEP dropped as low as 3.5 m/s. This discrepancy in the minimum SEP value extended the length of the climb to 33.25 seconds, as opposed to the predicted 30.623 seconds, resulting in an error of approximately 8%. However, when predicting the climb time using an SEP limit of 3.5 m/s, as opposed to a limit of 6 m/s, the predicted climb time is 32.62 seconds, giving an error of less than 2%.

To test the robustness, an additional analysis was conducted, but this time using a climb rate demand not present in Figure 7.6. This meant that for this analysis, knowledge of the aircraft's climb performance was unknown and any predictions required interpolation of the data in Table 7.1. The test airspeed remained at 200 m/s whilst a climb rate demand of 70 m/s was conducted at an initial altitude of 4 km. Therefore:

$$\begin{aligned} T_1 &= 0 \text{ s} & P_{s,1} &= 128.60 \text{ m/s} \\ T_2 &= 32.30 \text{ s} & P_{s,2} &= 97.91 \text{ m/s} \end{aligned}$$

with a gradient

$$\frac{dP_s}{dt} = -0.95 \text{ m/s}^2$$

In the previous analysis the minimum SEP value was set at 6 m/s. However, during that analysis it was found that during the climb period the SEP dropped as low as 3.5 m/s. Therefore using the same procedure as before to determine the nominal values and nominal offsets, but factoring this updated minimum SEP limit into the predicting the climb time:

$$T_{\text{climb}} = \frac{3.5 \text{ m/s} - P_{s,\text{alt}}}{-0.95} = 73.30 \text{ s} \quad (7.4)$$

When comparing the analytical prediction to the actual simulated performance (Figure 7.9), it is evident that the prediction has once again yielded an accurate representation. The predicted SEP value at the first stage of the climb was estimated to be 128.60 m/s, whilst during the simulation the SEP was 128.50 m/s, giving an error of less than 0.1%. This same level of accuracy is also present when attempting to predict the SEP 50 seconds into the simulation. The predictive SEP value was 97.91 m/s compared to the simulation's 97.77 m/s. Knowing that these two components have a high level of accuracy means that the overall climb time can be accurately calculated, with an error of only 0.72% between the predicted and actual climb times.

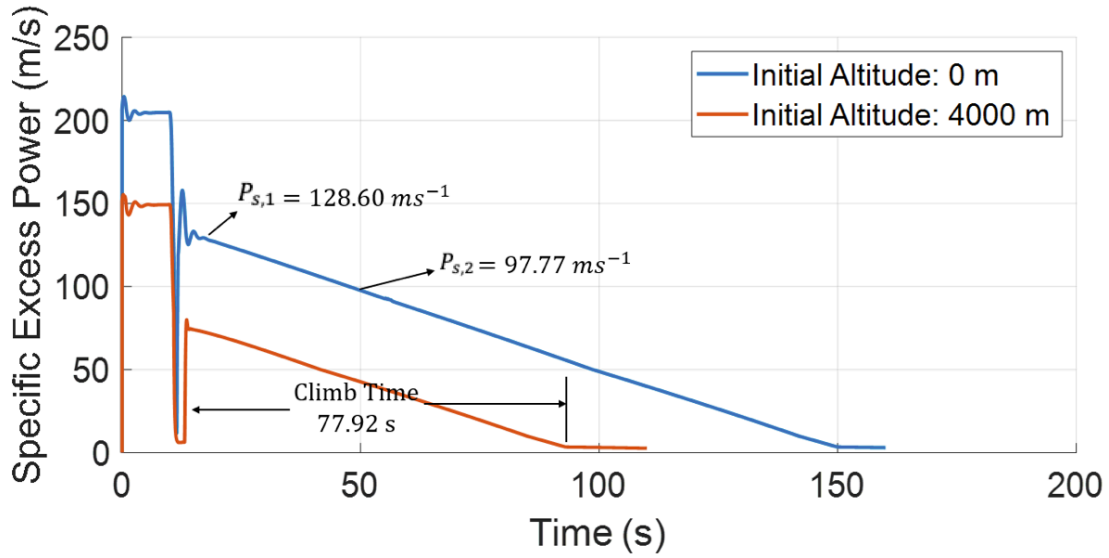


FIGURE 7.9 SEP analysis for a climb rate of 70 m/s with an initial altitude values of 0 km and 4 km and an airspeed of 200 m/s

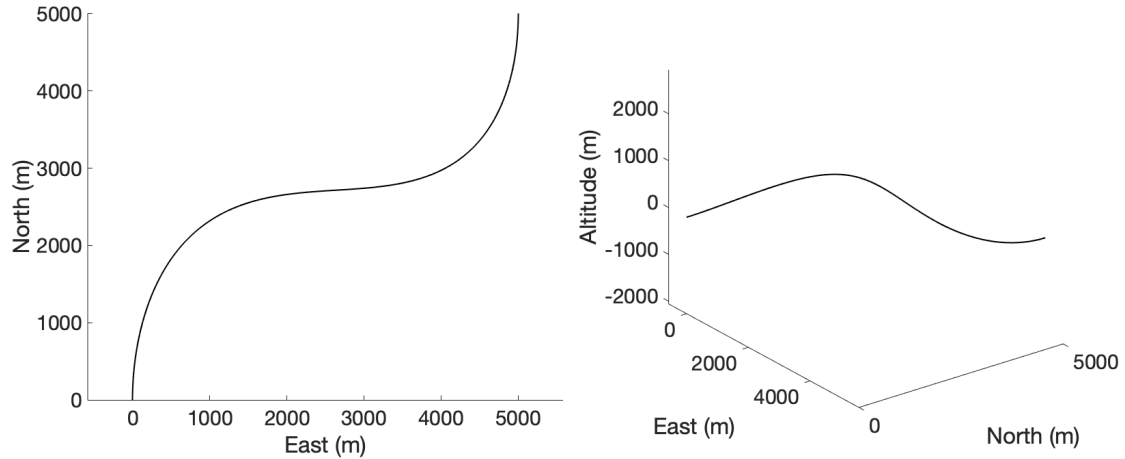
Although the process taken to determine the SEP during a climb has only been demonstrated on constant climb rate manoeuvres, the process can be directly applied to any altitude changing path, based on the path's climb rate. To determine the climb rate of a proposed path, the first derivative of the path in the z -axis ($\dot{z}$) is divided by the overall magnitude of the path and multiplying the answer by the airspeed, as seen in Equation 7.5. By iteratively calculating the offset and nominal values as a piecewise operation for every point along the path using the calculated climb rate, $\dot{h}$, and assuming constant velocity flight, the SEP profile can be predicted for the entire path.

$$\dot{h} = \left(\frac{\dot{z}}{\sqrt{\dot{x}^2 + \dot{y}^2 + \dot{z}^2}} \right) \times V \quad (7.5)$$

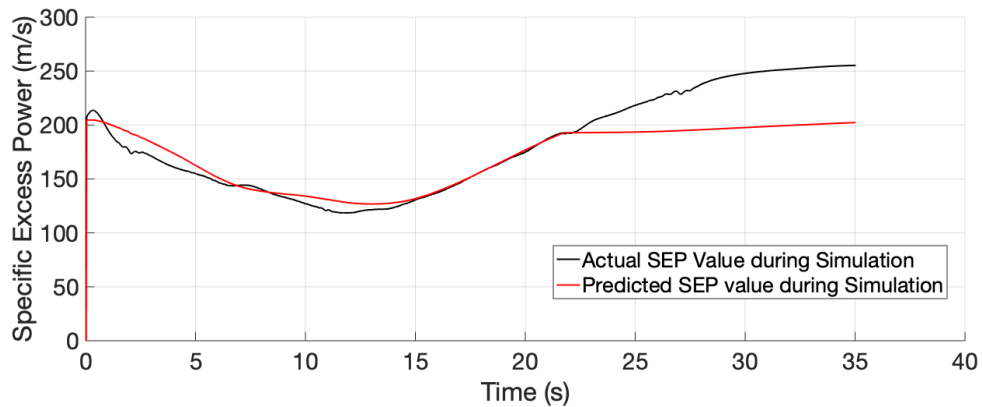
Consider the three-dimensional path shown in Figure 7.10a. This is a very simple path that has no heading changes and navigates from (0 km, 0 km, 0 km) to (5 km, 5 km, 0 km) in the inertial frame. Although the initial position vector and final waypoint vector are located at the same altitude, the altitude varies over the duration of the path, with a maximum altitude change of almost 1 km.

When applying this specific path to the guidance control system on the aircraft model, the aircraft was successfully able to complete the path whilst also maintaining a near-constant airspeed of 200 m/s. At each time-step, the SEP

was calculated based on the relationship in Equation 7.1 and was also predicted based on the SEP climb rate data analysis, with both the actual simulated SEP and predicted SEP plotted over time in Figure 7.10b.



(a) Three-dimensional path with variable altitude component



(b) Actual SEP and predicted SEP over the course of the simulation

FIGURE 7.10 Specific excess power prediction analysis on three-dimensional altitude changing paths for an airspeed of 200 m/s

Based on the path in Figure 7.10a, the altitude-changing component has two distinct features: an altitude gain period; an altitude fall period. When analysing the SEP of the aircraft during the simulation and predicted SEP values during the altitude gain period of the path, it is evident that the prediction accurately captures the climb dynamics of the aircraft, whilst assuming a constant velocity climb. This enables a prediction to be applied to any path generated determining whether or not the altitude component of the path is achievable through utilising the SEP and maintaining a near-constant kinetic energy.

However, during the altitude fall period, the predictive behaviour was not adequately captured. Throughout the altitude fall period, a conversion from potential to kinetic energy caused the aircraft to accelerate. This acceleration cannot be compensated for by the aircraft without performing some sort of kinetic energy controlling manoeuvre, which, is out of the scope of this research. Therefore, as the aircraft accelerates, the airspeed controller reduces the throttle to zero, with the aircraft now following the path by performing a controlled glide.

By setting the throttle position to zero whilst still accelerating due to gravitational acceleration, the actual SEP of the aircraft increases, as seen in Figure 7.10b. However, since the SEP prediction analysis assumes a constant velocity and does not capture changes to the velocity due to gravitational acceleration, the predicted SEP does not exhibit the same characteristics.

Assuming that the flight control system always maintains stability during a path descent phase, a path that results in a loss of altitude will always be achievable, given the dynamic constraints discussed in Chapters 4, 5 and 6 remain satisfied. This means that the large offset between the simulated SEP and predicted SEP in Figure 7.10, was not considered to be a concern and can be neglected provided the airspeed does not exceed the flight envelope.

7.3 Fitness Function

During Section 6.2 the fitness function for constant-altitude paths aimed to describe a path that was dynamically achievable whilst also maintaining a near-continuous average curvature profile. The components of the fitness function, defined as: $\omega_{penalty}$, $g_{penalty}$ and $P_{penalty}$ perform similar actions for the variable-altitude fitness function but with different weightings.

During constant-altitude path planning, the curvature of the path was only analysed in the x - y plane since there was no change in altitude along the path. However, for variable-altitude paths this was no longer the case, with the curvature profile having to be analysed in three-dimensions. Harris and Stocker [141] describe the general expression for the curvature of a curve in three-dimensions, as per Equation 7.6, where $\dot{x}$, $\dot{y}$ and $\dot{z}$ are the first derivatives of the path and $\ddot{x}$,

$\ddot{y}$ and $\ddot{z}$ are the second derivatives. The radius of curvature is then determined by taking the reciprocal of the curvature (κ).

$$\kappa = \frac{\sqrt{(\ddot{z}\dot{y} - \dot{y}\ddot{z})^2 + (\ddot{x}\dot{z} - \dot{z}\ddot{x})^2 + (\dot{y}\ddot{x} - \ddot{x}\dot{y})^2}}{(\dot{x}^2 + \dot{y}^2 + \dot{z}^2)^{\frac{3}{2}}} \quad (7.6)$$

Similar to Section 6.2, the dynamics of the path were determined based on the angular velocity demand and acceleration forces due to gravity. However, during the development of the fitness function for constant-altitude path planning in Section 6.2, the angular velocity limit was held constant at 20 deg/s. Since all of the applied flight velocities could sustain this pitch rate, and given the performance of the aircraft did not vary over the duration of a manoeuvre due to altitude, this was deemed an acceptable assumption. However, during altitude-changing manoeuvres, the turn-rate capabilities of the aircraft change with altitude, meaning that a maximum turn-rate at the start of the manoeuvre may not be achievable by the end of the manoeuvre.

To ensure the required turn rate was achievable throughout the entire path, the angular velocity was an adaptable parameter based on the demanded airspeed (u) and the maximum altitude (h_{max}) attained during the manoeuvre. This was achieved by utilising the maximum sustainable pitch rates (ω_{sus}) determined in Section 4.4.3, during optimisation of the pitch rate controller, with linear interpolation used to obtain the angular velocity at intermediate state dynamics. This value was then halved to manipulate the fitness function to not operate on the upper most limit of the flight envelope unless necessary. Therefore,

$$F_{\omega_{lim}} = \frac{\omega_{sus} f(u, h_{max})}{2} \quad (7.7)$$

and similar to Section 6.2, a gravitational acceleration limit (g_{lim}) of 9g was applied.

During altitude-changing manoeuvres, exceeding the imposed limits have larger repercussions on the robustness and stability of flight when compared to constant-altitude manoeuvres. For example, if a path exceeds or approaches the applied limits during the early stages of flight, the dynamic repercussions could cause a loss in airspeed, hindering dynamic performance throughout the remainder of the path, especially during a large climb. To prevent the optimiser from creating paths that cause the aircraft to operate at the upper limits of the

available dynamics, a harsh set of penalty factors were imposed, as shown in Equation 7.8. Hence, the general expression for the path dynamics portion of the fitness function can be seen in Equation 7.8.

$$F_{Dynamics} = \left(|\omega_{max}| - |\omega_{avg}| \right) \times \begin{cases} 1 & \text{if } |\omega_{max}| \leq F_{\omega_{lim}} \\ 500 & \text{if } |\omega_{max}| > F_{\omega_{lim}} \end{cases} + g_{max} \times \begin{cases} 0 & \text{if } |g_{lim}| \leq 9 \\ 1000 & \text{if } |g_{lim}| > 9 \end{cases} \quad (7.8)$$

Optimal paths should ideally have a near-constant curvature profile. This is achieved in Equation 7.8 by ensuring the difference between the absolute maximum angular velocity (ω_{max}) and the absolute average angular velocity (ω_{avg}) is small. However, this relationship causes the length of the path to extend towards infinity, reducing the difference between $|\omega_{max}|$ and $|\omega_{avg}|$.

To prevent this from occurring, the ratio between the total path length and point-to-point distance was directly applied to the fitness function, and then scaled by a factor of 100. The scaling factor had to be substantially larger than the scaling factor imposed during the constant-altitude analysis due to the increased number of coefficients formulating the variable-altitude fitness function.

$$F_{Geometry} = \left(\frac{100 \times P_{Total}}{(P_i^2 + P_f^2)^{1/2}} \right) \quad (7.9)$$

When considering variable-altitude path planning, various other considerations needed to be accounted for which were not present or discussed during composition of the fitness function in Section 6.2. These additional components were considered as three individual contributions: the SEP profile of the path, which was discussed in Section 7.2; the total rate of climb for the path; and where along the path the climb occurred.

The SEP is amongst one of the most important components applied to the fitness function, as this enabled a direct prediction of how much power would be required to complete a proposed climb. By analysing the SEP for each specific location along the path, a predicted SEP profile was established, enabling a quantitative measure.

During a climb, it was previously explained that as the SEP value approaches zero, this indicated insufficient power available to continue climbing, resulting in

a loss of kinetic energy. Therefore, a good measure of the SEP is achieved when the minimum SEP value (SEP_{min}) is large or close to the initial nominal value. Additionally, if the path has a gradual climb as opposed to multiple high and low climb regions, the decay of the SEP is predictable with a linear relationship.

To quantitatively measure the decay, the average SEP over the entire path was determined and compared to the initial SEP. The initial SEP (SEP_{init}) was denoted as the nominal value and based on the altitude level and airspeed of the aircraft. If the path has a gradual climb period as opposed to abrupt climbs, the difference between the initial SEP and average SEP should be relatively small. This led to the expression seen in Equation 7.10, where the fitness is determined as the ratio between the initial/average difference and the minimum SEP. Additionally, a large penalty was imposed for paths resulting in a minimum SEP that was less than 10.

$$F_{SEP} = \frac{(SEP_{init} - SEP_{avg})}{SEP_{min}} + \begin{cases} 0 & \text{if } |SEP_{min} \geq 10| \\ 500 & \text{if } |SEP_{min} < 10| \end{cases} \quad (7.10)$$

During development of the autopilots and controllers in Chapter 4, an operational flight envelope was used for optimisation, which was described in Section 3.11. This operational envelope gave the flight physics models a ceiling height of 10 km. Although this altitude range was used during optimisation of the controllers, actually climbing from 0 km to 10 km using a single fourth-order Bézier path was not necessarily a feasible objective, due to the SEP decay.

If the aircraft was to climb from 0 km to 10 km using a single path, the path would require a relatively large climb to be conducted at a near constant rate. This would result in a linear decay of the SEP, and depending on the initial conditions, state dynamics and climb demand, the SEP would begin to approach zero. During this scenario, it would be desirable for the path to adjust the rate of climb based on the local altitude and the corresponding SEP, resulting in a path containing multiple climb stages.

Whilst it would be desirable to produce a single path containing multiple climb stages based on altitude, these paths would require a higher level of path manipulation. This can be achieved either through the implementation of additional control points, which could be optimally placed to manipulate the climb profile at different regions along the point, or through the use of multiple Bézier

curves, each with different climb characteristics. Additionally, this could be considered as a problem where the solution is obtainable through an external path finding algorithm. A path finding algorithm would identify multiple waypoints, allowing individual paths between each waypoint and establishing a more complex climb manoeuvre. However, this formulation was deemed to be outside the scope of this research, as this research focuses on developing single dynamically achievable paths in three-dimensions between two waypoints, where the waypoint has already been determined as a feasible location.

Due to these factors, it was determined that the variable-altitude optimiser would reduce the ceiling height from 10 km to 5 km. This reduction in altitude allowed the aircraft to navigate from 0 km to 5 km using a single path across the entire airspeed operational envelope.

Having reduced the operating domains for the variable-altitude optimiser, a climb rate specification was imposed. This climb rate specification analysed the climb rate demand over the duration of the path and comparing it against the maximum achievable climb rate obtained in Section 7.2, as per Equation 7.11.

$$\dot{h}_T = \frac{\dot{h}_{max}}{\dot{z}_{max}} \quad (7.11)$$

When considering the climb rate analysis in Equation 7.11, the variable $\dot{z}_{max}$ represents the maximum climb rate that can be dynamically sustained by the aircraft based on altitude and airspeed. The term $\dot{h}_{max}$ represents the maximum climb rate demand over the path. By applying this analysis, it can be expected that as $\dot{h}_{max}$ approaches and exceeds $\dot{z}_{max}$, the output $\dot{h}_T$ increases, which decreases the overall fitness of the path. Use of $\dot{h}_T$ prevents unnecessarily high climbs being performed, since paths resulting in a climb rate exceeding the dynamic capabilities, amplify the resultant output.

However, for certain scenarios a high rate of climb that exceeds the limits, but only for a brief period, could be advantageous. For example, during constant-altitude path planning in Chapter 6, as the point-to-point distance between the position and waypoint was relatively low, the location of the control points expanding away from the origin, created a dynamic deviation. Being able to change altitude by performing an inclined loop reduces the the need to perform a dynamic deviation.

This can be seen in Figure 7.11a where a loop-like manoeuvre has been performed enabling a dynamically feasible path which otherwise wouldn't have been possible when applied solely to the x - y plane as seen in Figure 7.11b. However, when performing a loop-type manoeuvre, there exists a point along the path where the pitch angle required approaches 90 degrees, causing a very large, but brief climb rate demand.

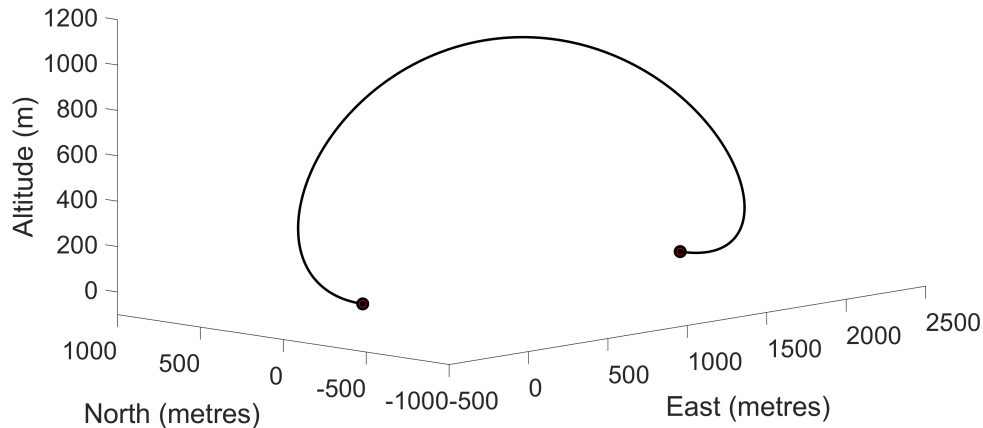
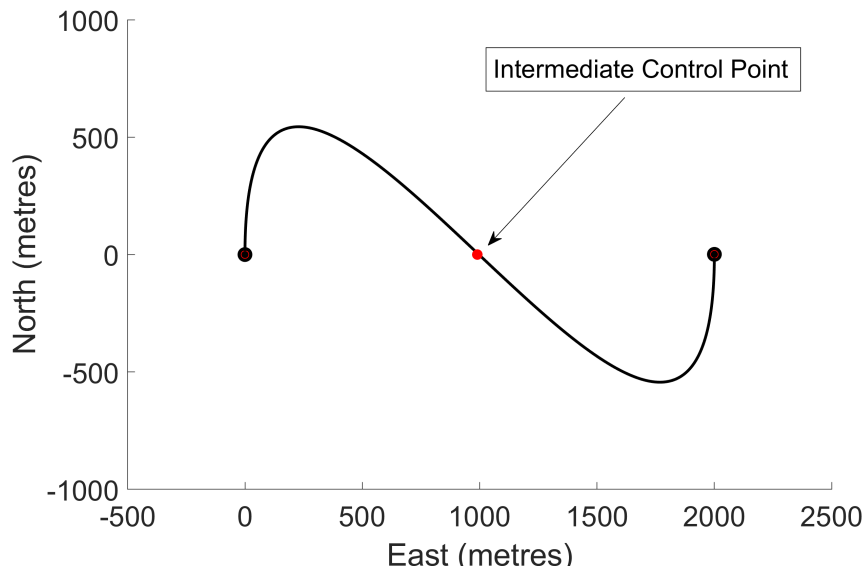
(a) x - y - z view(b) x - y view

FIGURE 7.11 Variable altitude changing path performing a loop-like manoeuvre to avoid the dynamics constraints of an x - y path

Additionally, it was important that when performing a climb, such as a loop manoeuvre, the climb portion was extended, allowing the most gradual climb

possible. Consider the three-dimensional path shown in Figure 7.11a. The location of the intermediate control point is displayed as a red coloured circular marker, and occurs at the mid-point between the position and waypoint along the east axis. This results in a path that has almost identical ascending and descending periods. Whilst this path maintained a consistent curvature profile, with acceptable dynamic performance characteristics, the rate of climb can be adjusted by locating the intermediate control point closer to the initial position or waypoint target.

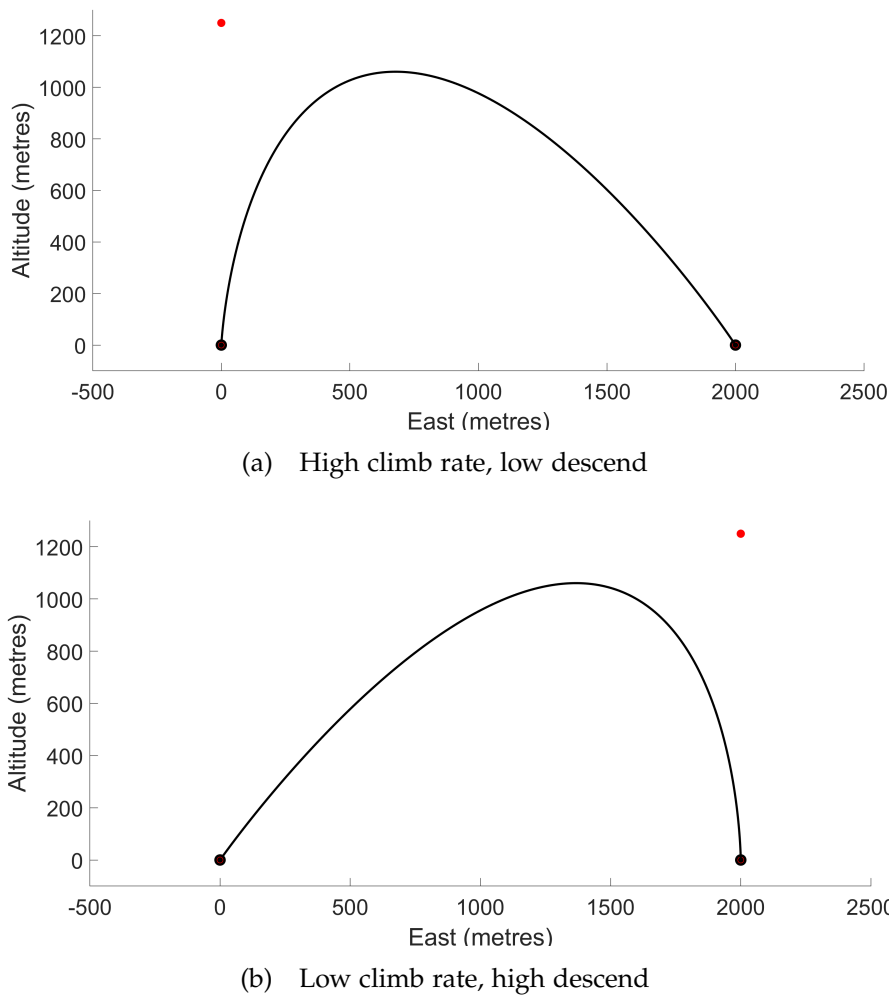


FIGURE 7.12 Intermediate control point placement in the y axis indicating the climb rate variances in the $y - z$ plane

An example of this can be seen in Figure 7.12a, where the location of the intermediate control point is located towards the origin in the y -axis (east), resulting in a steep and high climb rate. However, by moving the intermediate con-

trol point along the y -axis (east) towards the waypoint target as seen in Figure 7.12b, the required climb levels out, whilst the fall period becomes a lot steeper. This is a more desirable characteristic since it reduces the climb demand whilst achieving the same dynamic characteristics.

To ensure that paths are produced that favour a reduced climb rate, a climb rate penalty coefficient ($\dot{h}_{penalty}$) was applied to the fitness function. This was performed by partitioning the path into two equal sections ($\dot{h}_1$ and $\dot{h}_2$), with the altitude change determined individually for each section. If the magnitude of the altitude's rate of change in the first section was larger than the rate of change in the second section, a scalar penalty factor was imposed, as per the conditions in Equation 7.12.

$$\dot{h}_{penalty} = \begin{cases} 0 & \text{if } \dot{h}_1 \leq \dot{h}_2 \\ 1000 & \text{if } \dot{h}_1 > \dot{h}_2 \end{cases} \quad (7.12)$$

Finally, to prevent altitude-changing paths from being performed negatively through the z -axis, a sea-level penalty was applied. This penalty function was extremely severe as any paths which resulting in an altitude level below sea-level should be completely disregarded.

$$z_{sea-level} = \begin{cases} 0 & \text{if } z \geq 0 \\ 1000 & \text{if } z < 0 \end{cases} \quad (7.13)$$

The climb rate portion of the fitness function can be represented by summing all of the corresponding components, as seen in Equation 7.14.

$$F_{Climb} = F_{SEP} + \dot{h}_T + \dot{h}_{penalty} + z_{sea-level} \quad (7.14)$$

To complete the fitness function, the dynamic analysis in Equation 7.8 and climb analysis from Equation 7.14 are summed.

$$F_T = F_{Geometry} + F_{Dynamics} + F_{Climb} \quad (7.15)$$

Testing of the fitness function is performed in Section 7.4 after introducing the optimisation algorithm.

7.4 Control Point Optimisation

The optimisation algorithm developed for variable-altitude path planning was the same genetic algorithm (GA) used during constant-altitude path planning in

Section 6.3. The main modification to the GA was to increase the input and output dimension to allow optimisation to be performed on five scaling functions as opposed to four, where the fifth scale represented the location of the intermediate control point along the z-axis. Additionally, changes were made to the fitness function as discussed in Section 7.3 to accommodate the specific excess power and climb rate analysis required for altitude changing manoeuvres.

A population consisting of 100 individuals was initialised using mitosis, based on a single individual with a default set of values. The default set of values remained consistent with those used in Section 6.3, and was determined based on a point-to-point straight line path. For each generation, a total of 50 individuals were selected for reproduction using a tournament selection process, where each tournament was filled randomly with five individuals. Although having tournaments comprised of five individuals did create a high selection pressure, it aided in convergence towards a solution, creating a more reliable and repeatable evolutionary process. The winner of each tournament would enter the reproduction pool and be paired with the winner of subsequent tournament, whilst the tournaments unselected individuals returned to the general population.

Crossover was performed by initialising a binary matrix of all possible genome combinations between each couple. Since each individual is comprised of five genes, each couple has the potential to create 32 offspring per generation. If each couple was allowed to create 32 offspring per generation, the size of the population would increase from 100 to 975 per generation. Therefore, the number of offspring produced by each couple was limited to 10, where each combination of offspring was randomly selected. Similarly to the constant-altitude optimisation algorithm, an exact replica of each parent was also applied to the offspring, bringing the total number produced by each couple, to 12. By duplicating each parent and performing a slight mutation to each gene, the population receives further diversification and increases the probability of convergence occurring towards a better solution in the proceeding generation.

Mutation was applied to each offsprings gene with a probabilistic chance of 90% and an initial mutation range of 0.5. The mutation range decayed proportionally over time, by a coefficient of 0.8 and to a minimum value of 5×10^{-3} .

Allowing the mutation range to decay over time, but restricting the size from diminishing too much, allowed convergence to occur more quickly and more frequently, whilst the diversity of the population could also be prolonged for more generations. Due to crossover, at the end of each generation the population increased from 100 individuals to 400, as shown in Equation 7.16. At the end of each generation, the same selection process as used during reproduction, was applied to determine which individuals would be retained into the next generation, reducing the population size from 400 back to 100. This process was then repeated for a maximum of 200 generations, or until the standard deviation of the population dropped below 5×10^{-3} .

$$\begin{aligned} \text{Population Size} &= 100 + (\text{Offspring Size} \times \text{Number of Couples}) \\ &= 100 + \left(12 \times \frac{50}{2}\right) = 400 \end{aligned} \quad (7.16)$$

Testing of the GA and fitness function was conducted applying the GA directly to the guidance and navigation control system on the aircraft model. At the start of each simulation, a random start location for the aircraft and orientation was computed, as well as a waypoint location and orientation. Both the initial aircraft position and waypoint position were specified with respect to the inertial frame. For the purpose of this analysis, the position and waypoint location within the inertial frame was limited to ± 10 km in both the x (north) and y (east) axes, and between 0 km to 5 km in the z -axis (altitude). The initial airspeed of the aircraft was also limited to between 100 m/s and 300 m/s, and was randomly generated at the start of each simulation. The limits applied to the airspeed were based on the dynamic limitations of the aircraft's aerodynamics, as well as the limits imposed during the applied climb rate analysis in Section 7.2.

The computed position and orientation vectors for the aircraft and waypoint target were then applied as inputs to the GA, with optimisation performed to determine the optimal control point scaling functions and corresponding path. Upon completing the optimisation process, the path was applied to the guidance and navigation flight control system. The aircraft model attempted to complete the optimised path whilst maintaining a near constant airspeed throughout. If the aircraft was successful in making it to the end of the path, the solution was

deemed to be dynamically feasible. Additionally, during the development of the path, an estimated completion time was computed based on completing the path with a constant airspeed. If the aircraft was successful in reaching the end of the path without exceeding the estimated completion time by more than 10%, the solution was deemed to be optimal since this indicates the path is flyable with a near-constant airspeed. On the other hand, since paths incorporated a change in altitude, the proposed path could result in an altitude descending trajectory, causing numerous issues for the time estimation analysis. For example, if the waypoint target was located at a lower altitude level than the initial start location, due to the guidance law aligning the aircraft's LOS through points located on the path, the aircraft is unable to maintain a near constant airspeed due to gravitational acceleration and lack of airspeed braking. A solution to this would be to traverse the descending portion of the path with a greater positive pitch angle and using the angle of attack as opposed to LOS. This would increase the surface area exposed to the free-stream velocity and increase drag on the aircraft to prevent rapid acceleration. However, this is proposed as future research as it would require adaptation to the guidance law, and an additional analysis to understand the resilience of the implementation as the aircraft levels out. For this reason, the proposed time analysis was not administered to paths which were completed in less time than the paths estimated completion time, and only to those that exceeded the estimated completion time.

After completing 25,000 randomly generated simulation scenarios, the optimiser achieved an accuracy of approximately 99.55% when developing dynamically feasible paths. Although it is desirable for the optimiser to develop dynamically feasible paths for 100% of the tested scenarios, the complexity of the non-linear system as well as the proposed fitness function methodology, makes this difficult to achieve. Additionally, when analysing the un-flyable path produced by the optimiser, it was evident that these paths occurred more frequently when paths originated at the upper altitude levels (> 4 km) and during low velocity simulations (< 150 m/s) as shown in Figure 7.13. Whilst this indicates that the dynamics of the aircraft during low-velocity and high-altitude flight have not been fully captured by the fitness function, it is important to note that this does not occur during every scenario, and can also be attributed to the stochastic

nature of the optimiser. Therefore during training of the deep neural network, solutions that are dynamically unachievable were redacted from the training dataset.

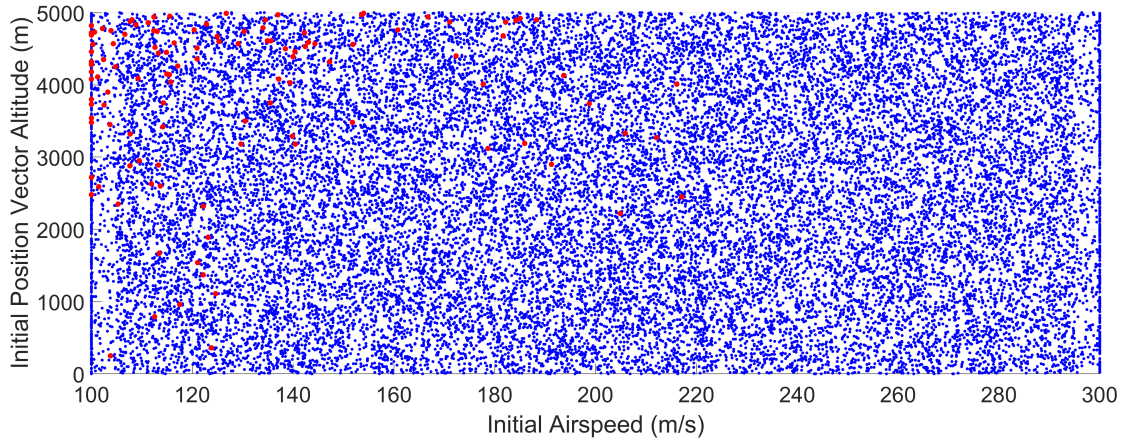


FIGURE 7.13 Optimisation algorithm and fitness function testing for variable-altitude path planning. Blue markers indicate solutions which were dynamically achievable, with red markers indicating solutions that were not dynamically achievable

7.4.1 The Curse of Dimensionality

During constant-altitude path optimisation in Chapter 6, it was assumed that the aircraft would be in constant velocity flight whilst also maintaining a constant altitude throughout the entire duration of any path. This allowed the dimensions of the problem to be reduced, since changes in altitude and velocity were not factored into the analysis. Additionally, these assumptions enabled the position of the aircraft to be described using an intermediate reference frame (IRF), unique to the constant-altitude path planning algorithm. Within the IRF the aircraft was assumed to always be fixed at the origin with a heading angle of zero degrees. By fixing the location of the aircraft to the origin of the temporary axis system and with a heading angle of zero degrees, the waypoint was described as a relative location during optimisation.

The IRF for constant-altitude path planning was partitioned as a grid-based environment, with nodes located at set intervals along the north and east axes. Since the aircraft has symmetrical dynamics about the x - y and y - z axes, the IRF

was mirrored about the x -axis, halving the required number of observations and optimisations as depicted in Figure 6.14. A breakdown of the required observations for the constant-altitude path planning analysis can be seen in Table 7.3. This breakdown of observations gave a total of 2,640 classification solutions, with exhaustive optimisation performed to obtain all of the necessary solutions to train a classification deep neural network.

TABLE 7.3 Observation breakdown for constant-altitude path planning

Axis/Rotation	Range of Values	Intervals	Observations
x -axis (North)	-10 km to +10 km	1 km	21
y -axis (East)	0 m to +10 km	1 km	10
Azimuth	0 deg to 330 deg	30 deg	12
Total Classifications			2640

Although the IRF had a distance-restriction in the x and y axes as seen in Table 7.3, since the IRF moved through the inertial frame with the same velocity and orientation as the aircraft model, the operating conditions were essentially infinite. Not only did the IRF reduce the number of optimisations and classifications required, it also expanded the operating potential, providing numerous benefits for operation.

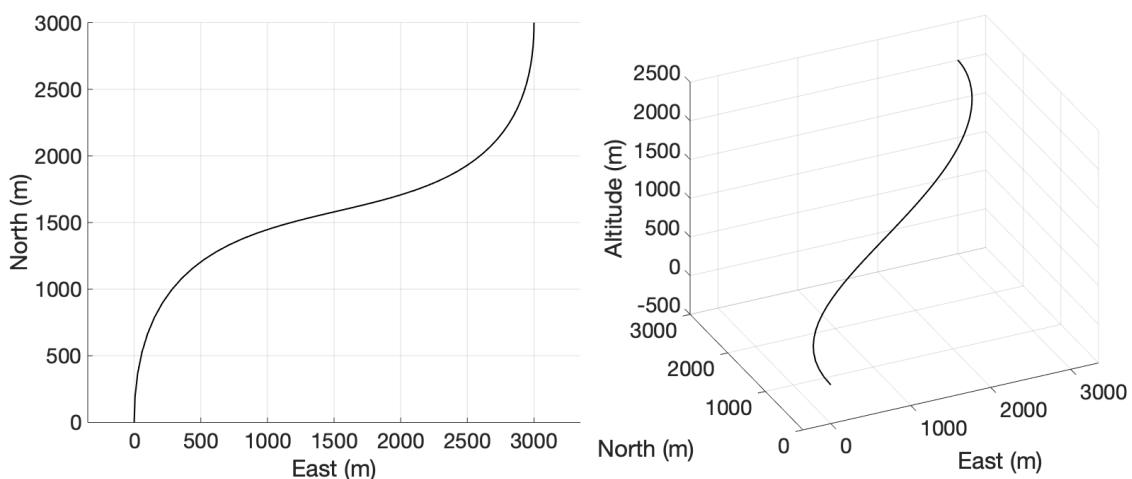


FIGURE 7.14 Optimised 3D path originating from position (North: 0 km, East: 0 km, Altitude: 0 km) to a waypoint target (N: 3 km, E: 3 km, Alt: 2 km)

However, when considering three-dimensional path optimisation, the implementation of varying altitude causes numerous issues. Consider the aircraft located with a north coordinate of 0 km, east coordinate 0 km, altitude 0 km and with a heading angle of zero degrees. The path planning algorithm is instructed to navigate to a waypoint target located 3 km in the north direction, 3 km in the east direction and with an altitude level 2 km higher than the start position. The final orientation of the waypoint is identical to the position of the aircraft. The optimised path for this configuration can be seen in Figure 7.14 for an airspeed of 200 m/s.

The position/waypoint configuration for this path demands a climb of 2 km to be conducted over a euclidean distance of 4.7 km. Since the initial altitude of the path originates from 0 km, the performance capabilities of the aircraft allows a manoeuvre to be conducted without requiring an intermediate climb phase. This enables the aircraft to path directly to the waypoint with a near-constant climb rate, by performing an S-curve manoeuvre.

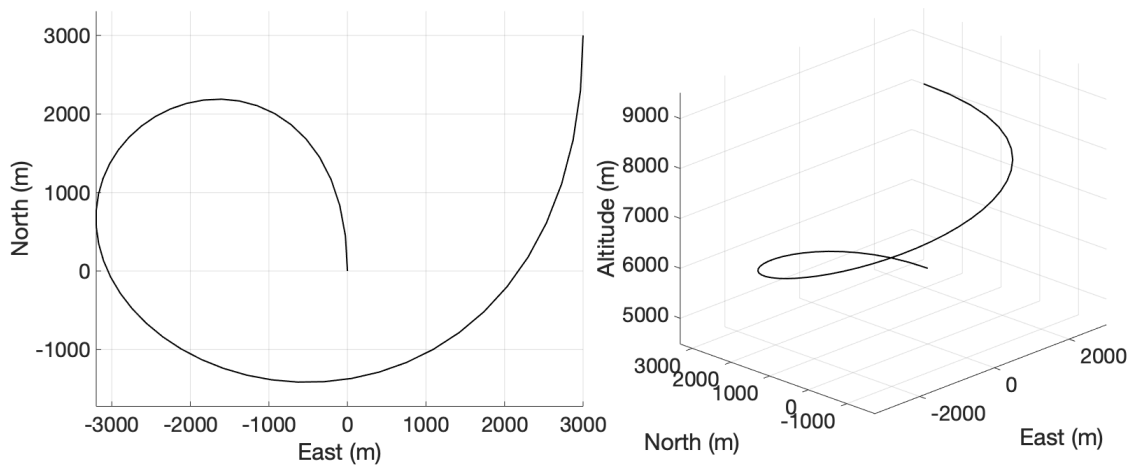


FIGURE 7.15 Optimised three-dimensional path originating from position (North: 0 metres, East: 0 metres, Altitude: 6 km) to a waypoint target (North: 3 km, East: 3 km, Altitude: 8 km)

However, when optimising the same configuration, but with an initial altitude of 6 km as opposed to 0 km, the optimised solution as shown in Figure 7.15 does not allow the same climb to be performed. Since the density of air is lower at 6 km, the thrust generated from the engine does not allow the same 2 km climb to be conducted at the same climb rate as Figure 7.14. Due to this, an

intermediate climb stage has been implemented in Figure 7.15 with the optimised solution incorporating a helical climb, increasing the length of the path to reduce the climb rate.

Whilst this has demonstrated how the fitness function and SEP analysis has enabled path optimisation to adapt to different environmental conditions, it means the methodology proposed in Chapter 6 for constant-altitude path optimisation is not a suitable solution. This is due to the implementation of the IRF not adequately compensating or considering dynamic performance discrepancies associated with climb performance at various altitude levels. This was not a problem for constant-altitude path planning, since the applied angular velocity limit during optimisation was achievable throughout each flight level.

TABLE 7.4 Observation breakdown for variable-altitude path planning given the same observational area and intervals used for constant-altitude path planning

Axis/Rotation	Range of Values	Intervals	Observations
Position x -axis (North)	-10 km to +10 km	1 km	21
Position y -axis (East)	-10 km to +10 km	1 km	21
Position z -axis (Altitude)	0 m to + 5 km	1 km	5
Position Azimuth	0 deg to 330 deg	30 deg	12
Waypoint x -axis (North)	-1 km to +10 km	1 km	21
Waypoint y -axis (East)	-1 km to +10 km	1 km	21
Waypoint z -axis (Altitude)	0 m to + 5 km	1 km	5
Waypoint Azimuth	0 deg to 330 deg	30 deg	12
Velocity	100 m/s to 300 m/s	10 m/s	20
Total Classifications			1.4×10^{10}

For this reason, path optimisation for variable-altitude path planning needed to be performed for absolute position/waypoint configurations, as opposed to relative positions with respect to a IRF. This increased the number of required solutions to train a classification neural network, since the observations now included both position and waypoint targets, as shown in Table 7.4.

Although optimisation was required based on absolute positions with respect to the inertial frame, an optimisation zone needed to be identified since optim-

isation could not be performed over an infinite axis system for obvious reasons. Based on the climb rate analysis, and the development of the fitness function, it was explained how the operational envelope for path optimisation had been reduced from 10 km to 5 km, allowing robust optimised paths to be obtained using single waypoint information. Table 7.4 shows the required observations needed to perform control point classification for variable-altitude path planning, provided the operating range and classification interval sizes were kept consistent with constant-altitude path planning in Chapter 6.

Based on the information in Table 7.4, the increased dimensionality and the requirement for absolute position and waypoint configurations, meant that the total number of classifications increased to 1.4×10^{10} . This large number of classifications made it impossible to implement a classification network based on the information in Table 7.4. Additionally, the information in Table 7.4 contains large intervals between nodes determined during the constant-altitude analysis. Whilst these intervals were acceptable for the constant-altitude analysis, it does not indicate that these intervals will transfer into acceptable solutions for variable-altitude analysis, given the increased complexity. Finally, during the constant-altitude conclusions, it was determined that sub-optimal solutions occurred more frequently as the waypoint approached the position vector. This indicates that to increase the accuracy of the solution, as the waypoint approaches the position vector, more classifications are required, decreasing the interval size and increasing the observations. Under this analysis, this would only increase the number of classifications further. Whilst decreasing the optimisation area from ± 10 km in both the x and y axes to 0 - 10 km does decrease the number of classifications by 92%, a total of 1,054,152,000 classifications still exist, indicating use of a classification network with exhaustive optimisation is not an appropriate solution for variable-altitude path optimisation due to the increased dimensionality. Therefore, a regression deep neural network was implemented, replacing the need to performed exhaustive control point classification, through optimisation.

7.5 Regression Deep Neural Network

During Chapter 6, a classification deep neural network was used to predict the control point scales for Bézier curve path planning, based on waypoint observation classification. It was discussed during Chapter 6 that use of a classification network was suitable because the environment could be discretised whilst still remaining tractable. However, for reasons which will become apparent throughout Section 7.5, the increased dimensionality associated with variable-altitude path planning increased the number of discrete classifications to 7.0546×10^{17} . Obtaining an optimised solution for each discrete classification was not possible given the time-constraints of the thesis as well as the computational feasibility of the solution, leading to use of a regression model. A regression model, similar to a classification model, is a type neural network trained using supervised learning. However, rather than assigning a discrete classification to specific data points, the neural network outputs a multi-dimensional continuous function, fitted to the training data. Whilst this does pose numerous advantages, those advantages can be challenging to obtain robustly and demonstrate resilience with high certainty.

7.5.1 Preliminary Study

Consider the following simplified problem. Whilst assuming a constant velocity manoeuvre, optimisation was performed over a small region of the inertial frame. The start location for the path was held constant and initialised at the origin (0 km, 0 km, 0 km) with an orientation of $(0^\circ, 0^\circ, 0^\circ)$. The north coordinate of the waypoint target remained fixed at a location of 1 km, whilst the east coordinate ranged from 0 km to 10 km, with optimisation performed every 1 km, giving a total of 11 waypoint targets. Each of the identified waypoints was optimised once using the optimisation algorithm outlined in Section 7.4, with the solutions forming a small dataset comprised of waypoint target locations and control point scaling solutions. These solutions can be seen displayed as red markers in Figure 7.16, where each control point scaling solution obtained through optimisation, is represented on a specific subplot, as indicated by the y -axis label.

When analysing the solutions for each of the individual control point scaling functions in Figure 7.16, it is visually apparent that a relationship is present between the location of the waypoint target along the east axis and the size of the corresponding five control point scaling functions. This relationship allows a regression network to be trained on the dataset, with the network predicting intermediate solutions based on the relationship in the trained data.

A regression neural network was created with one hidden layer comprised of 30 neurons fully-connected to the input and output. The activation function for the input layer and hidden layer was a rectified linear unit (ReLU), with a total of 1295 trainable parameters. Since the training data for this example contained only one relevant piece of information, the location of the waypoint target, the model only had three inputs, with five outputs. Since this problem

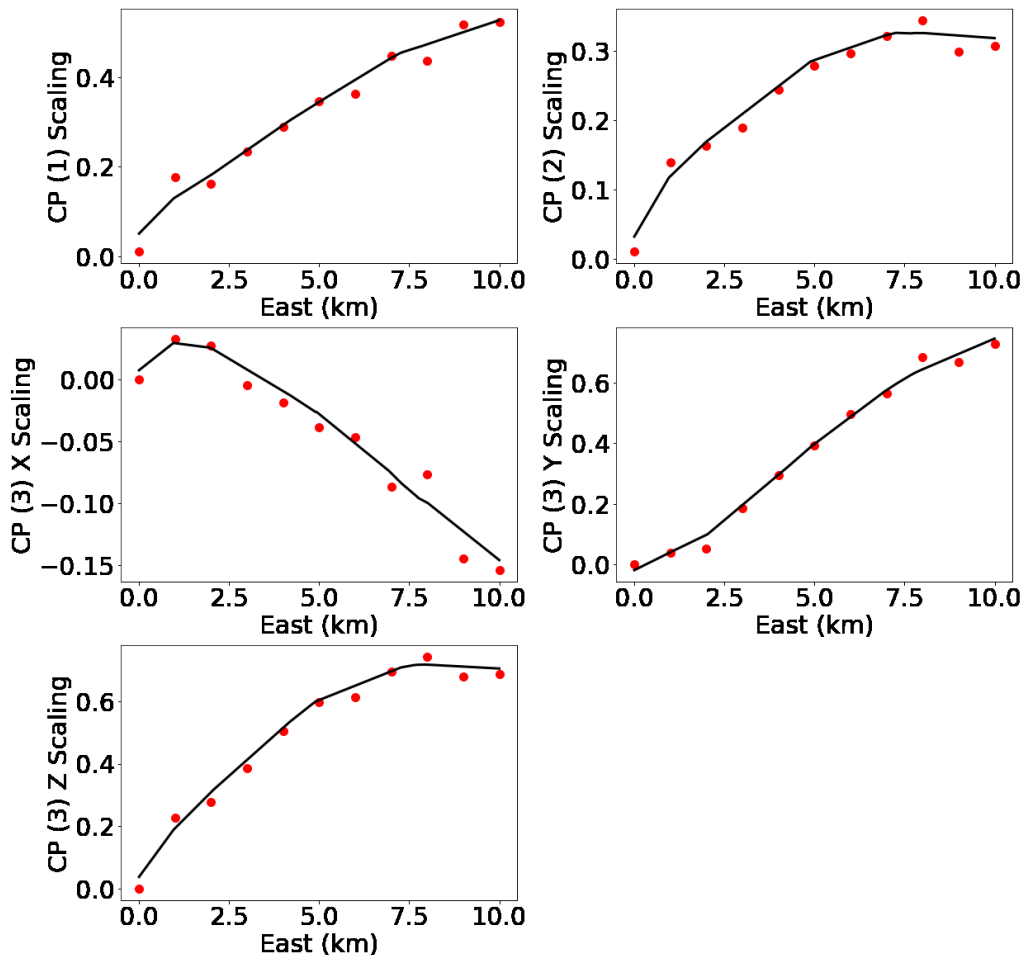


FIGURE 7.16 Five optimised control point scaling factors for a simply regression model.

is simplistic with low inputs and outputs, the choice of optimiser and internal

network structure was not considered with great significance. Therefore, training was conducted over 500 epochs using the RMSProp optimiser. For each epoch the network was measured using the mean absolute error (MAE) and mean squared error (MSE), forming the overall loss. Upon training the network, a prediction was performed evaluating the network over the entire range of east coordinates every 10 metres. The network's prediction can be seen in Figure 7.16 as the black line-of-best-fit.

Although this model was a very simplistic representation with reduced input dimensionality and optimisation only performed over a reduced operating range, this demonstrated how a regression network can be used to predict the control point scaling functions for the purpose of path planning. In addition, although optimisation was performed for evenly spaced discrete waypoint locations, this is not a requirement for a regression network, with optimisation capable of being performed for random waypoint locations.

7.5.2 North-East (NE) Model

Although the preliminary study proved that a regression neural network can be applied as a suitable alternative to a classification network, the solution was simplistic and not representative of a higher-order problem. To incrementally

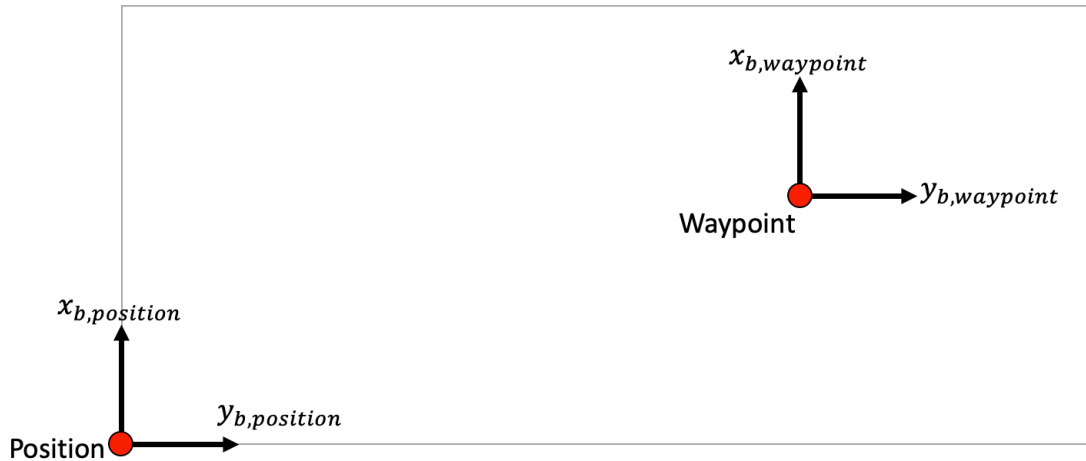


FIGURE 7.17 North-East model position/waypoint configuration

increase the complexity of the model, a second study was performed. This model denoted as the North-East (NE) model allowed the location of the waypoint

target to vary not only along the y -axis (east), but also along the x -axis (north).

The initial position vector remained fixed at the origin for the duration of this analysis, and both the orientation of the position and waypoint vectors remained constant at $(0^\circ, 0^\circ, 0^\circ)$, as depicted in Figure 7.17. Similar to the previous study, the position and waypoint vectors were located with an altitude of 0 metres. Although no change in altitude occurred between the position and waypoint vectors, the altitude was allowed to vary throughout a generated path based on the applied fitness function. Optimisation was performed 5,000 times for random waypoint north and east coordinates rounded to the nearest 100 metres, using the same procedure outlined in Section 7.4.

Since the optimised solutions were obtained for random waypoint north and east coordinates rounded to the nearest 100 metres, the dataset is considered as being comprised of discrete observations. Considering the north and east coordinates of each observation were confined between 0 and 10 km, for this analysis a total of 10,000 discrete observation combinations existed. Given that optimisation was performed 5,000 times, it was expected that due to the random observation assignment, many discrete intervals may have been optimised more than once, whilst some locations may not have been optimised at all. Whilst the continuous output function of the regression model should aid in obtaining accurate predictions for locations which did not obtain an optimised solution, locations which obtained more than one solution could become a hindrance, particularly if the optimised solutions have some variance.

During optimisation for constant-altitude path planning in Section 6.3, due to the aircraft's symmetrical dynamics, more than one global solution occasionally existed. However, when considering three-dimensional paths with a fitness function containing as many parameters as described in Equation 7.15, it would be naive to suggest that only one global solution exists, when in reality, many global solutions may exist. Whilst it would be ideal for optimisation to be 100% repeatable, with a single global solution for each observation, this was simply not feasible, especially when trying to produce an optimiser capable of performing across many dimensions. However, it should be expected that a robust optimiser should produce similar solutions when subjected to identical input observations. Whilst this was demonstrated during testing of the optimiser and

fitness function, occasionally the optimiser would produce a contrasting solution due to the stochastic element of the optimiser. Additionally, if an observation was to be repeated more than once within the training dataset, this would introduce training bias. For these reasons it was important to remove duplicated observations from the training data and apply a method that would analyse the optimised data and identify outliers. These outliers were identified as being solutions that were sub-optimal due to premature convergence, or solutions that were not characteristic of previously obtained local solutions.

For the NE model the optimised dataset consisted of two inputs: the waypoint north observation; and the waypoint east observation, with five outputs, each corresponding to a control point scaling function. To identify outliers within the data, the data was first arranged into discrete samples based on a single dimension of data which, for the NE model, was the north coordinate. Since each optimised solution was comprised of five control point scaling solutions, each scale was analysed individually, forming five data samples per discrete observation. Given a specific sample of data, (x) , the mean value, (μ) , and standard deviation (σ) of the sample was used and compared to each individual solution within the sample, forming a numerical metric known as a z-score, as seen in Equation 7.17. This same procedure was also used in the identification of outliers during constant-altitude path planning in Chapter 6. Based on the z-score a threshold was then applied to each discrete sample, aiding in the identification of solution outliers.

$$Z = \frac{x - \mu}{\sigma} \quad (7.17)$$

Consider the data shown in Figure 7.18. This represents a sample of data extracted from the training dataset where the waypoint north coordinate ranges from 0 metres to 1 km. For clarity, each of the five subplots displays one of the control point scaling solutions as indicated by the y -axis. The sample of data shown in Figure 7.18 is considered as one of the most challenging waypoint configurations to obtain a dynamically feasible solution due to the small euclidean distance between the position and waypoint. During optimisation the fitness function initially attempts to produce a simple point-to-point shortest path between the two points, which is constrained by average curvature. However, as the euclidean distance between the position and waypoint approaches

zero, producing a simple point-to-point path generally exceeds the dynamic limits of the aircraft, causing the path to have a poor fitness. As a result, the fitness function prioritises producing a path that operates within the dynamic limits of the aircraft by increasing the magnitude of the first and second control point scaling functions. This characteristic is indicated by a protuberance to the data as the east coordinate approaches the origin. This characteristic was also present

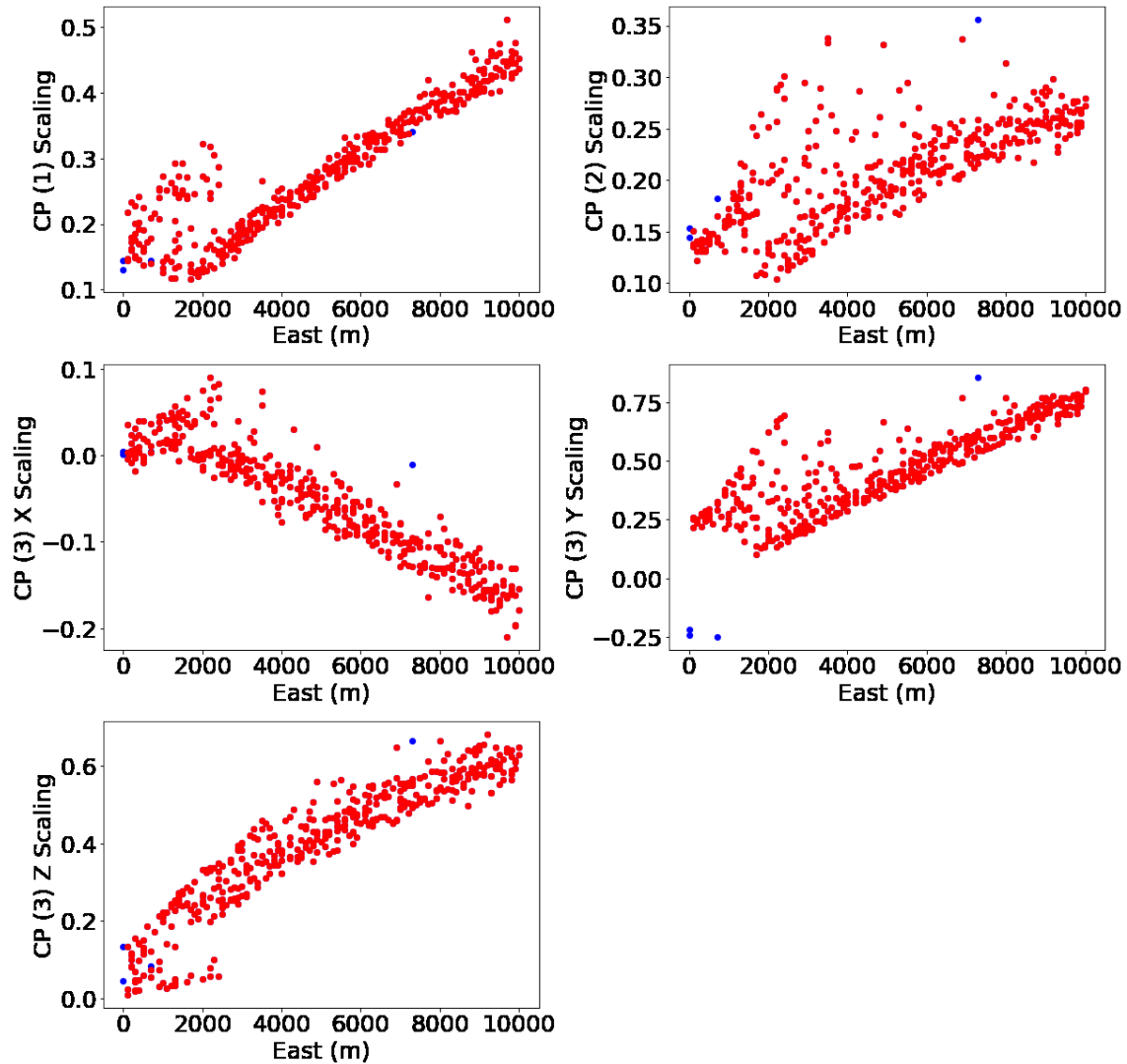


FIGURE 7.18 Control point scaling solutions for NE model where the waypoint north coordinate is located between 0 km and 1 km. Optimisation was performed for a velocity of 200 m/s with an initial and final heading angle of 0° and an initial and final altitude of 0 km. Data points assigned as blue markers have been identified as outliers using a z-score measurement and removed from the training sample.

during constant-altitude path planning in Chapter 6 as demonstrated in Figure 6.2.

When analysing the solution for the fourth control point scaling factor, corresponding to the location of the intermediate control point along the y -axis (east), as the waypoint east location approaches the origin a few solutions become negative. Whilst these solutions are most likely dynamically achievable, convergence has occurred in this instance onto a local solution, or onto a secondary global solution. It was previously discussed that when this occurs, it can have implications on the accuracy measurement of a regression network, as contrasting solutions will result in a large loss factor. However, by calculating a z -score for all displayed solutions, these secondary solutions have successfully been identified as data outliers and removed from the dataset, as indicated by the blue marker.

A second example of this data outlier identification process can be seen in Figure 7.19. This data represents a sample of data extracted from the training dataset where the waypoint north coordinate is located between 8 km and 9 km. Unlike the previous data sample, which consisted of position/waypoint configurations with relatively small euclidean distances, the euclidean distance in this instance is large.

The trend of data in Figure 7.19 indicates that as the distance between the position and waypoint increases, the topology of the fitness function no longer contains multiple regions of local minima, but rather represents a shallow gradient consisting of multiple global minima. Multiple global solutions exist because the euclidean distance between the initial position and waypoint has become so large and the fitness of the path is no longer heavily constrained by the dynamics of the aircraft. Since the dynamics of the path are no longer considered to be a major factor during this configuration, the fitness function attempts to determine the shortest path constrained by average curvature, creating two possible solutions. The first solution is that the fitness function creates a path with a constant-altitude profile. This path will increase the curvature resulting in a slightly higher dynamic loading but still within the dynamic limits of the aircraft. The second solution is that the fitness function creates a path that allows for a small change in altitude to occur over the duration of the path. This solution

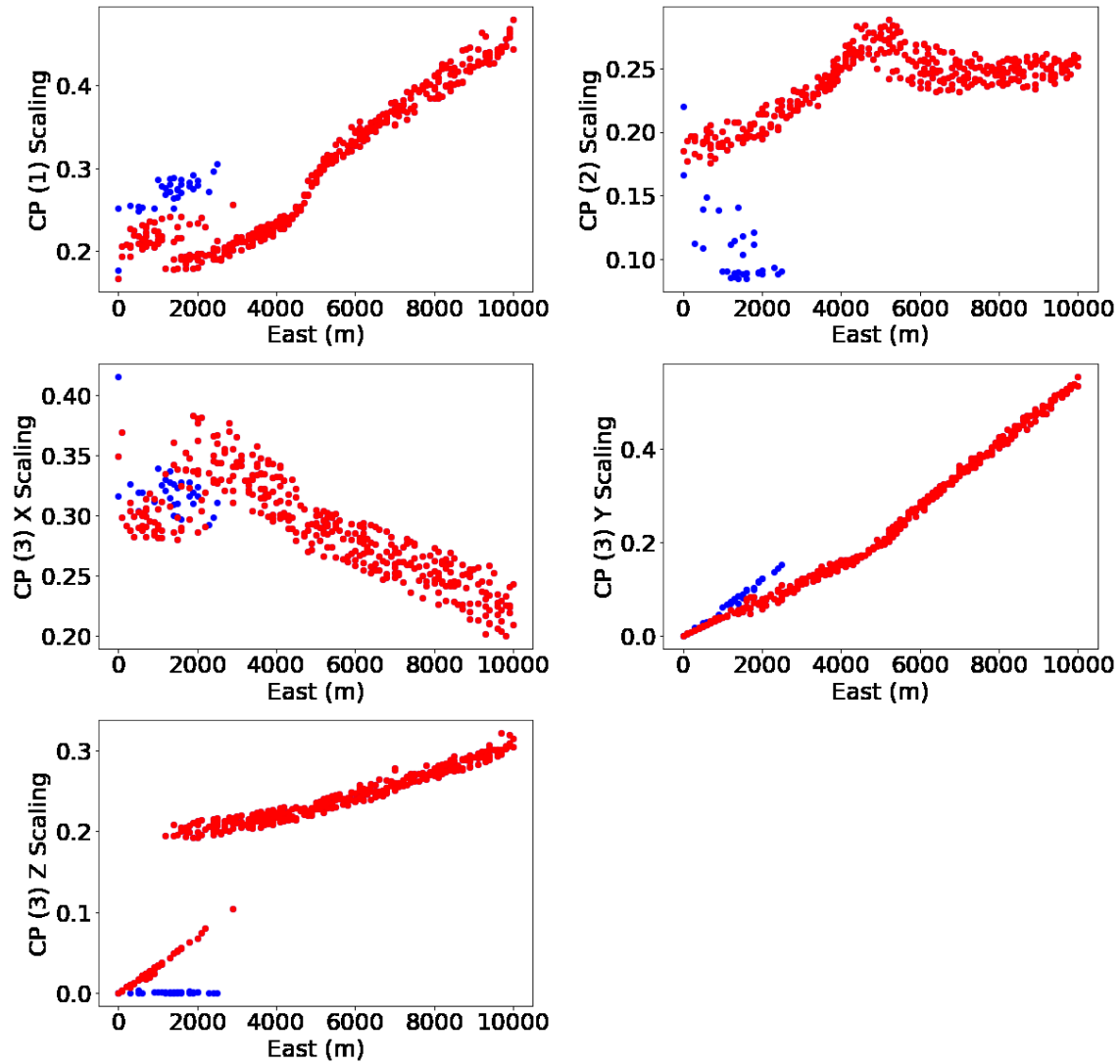


FIGURE 7.19 Optimised control point scaling functions for north coordinates between 8 km and 9 km. Optimisation was performed for a velocity of 200 m/s with an initial and final heading angle of 0° and an initial and final altitude of 0 km. Data points assigned as blue markers have been identified as outliers using a z-score measurement and removed from the training sample.

creates a trade-off between performing a slight and gradual climb and reducing not only the curvature profile but the overall distance as the optimiser can produce a straighter path to the waypoint. When analysing the fifth control point scaling function in Figure 7.19, which corresponds to the location of the intermediate control point in the z-axis and is responsible for changes to the altitude, it is evident that optimisation has occurred for both of these scenarios, with the

majority favouring a slight altitude change. By measuring individual solutions against the overall dataset and calculating the z-score, solutions that converged onto a constant-altitude path have been identified as outliers, as indicated by the blue marker and removed from the dataset.

Having established and demonstrated a method of identifying and removing solution outliers within the training dataset, the number of data points reduced from 5,000 observations to approximately 3,900. However, a large quantity of the data removed was due to repetition between observations as opposed to contrasting optimised solutions. The waypoint information and corresponding solutions were partitioned into a training and validation dataset with a ratio of 20:1, with data points selected randomly. The structure of the deep neural network was comprised of two hidden layers where each layer contained 100 fully connected neurons, giving a total of 21,005 trainable parameters. Training was conducted for a total of 50,000 epochs with a learning rate of 1×10^{-4} , with the model iteratively saved every 200 epochs. The performance of the network was described during training using the same loss metric as the preliminary study. After training for 50,000 epochs the MSE between the network and training data measured 0.0017 and 0.0055 between the network and validation data.

Since the dimensionality of the NE model is relatively low with only two input observations, a visual analysis of the model's overall fit to the dataset was produced as a three-dimensional surface. This was performed by initialising a set of north and east arrays with values ranging from 0 to 10 km with discrete intervals every 10 metres. The NE model was then queried for all combinations of the applied north and east values within the computed arrays, allowing a three-dimensional predicted surface to be generated for all five control points.

Figure 7.20 shows a three-dimensional plot where the solutions for the first control point scaling function used to train the neural network are displayed as a scatter plot. The model prediction has been applied to the scattered data, creating a three-dimensional surface. A visual analysis of the surface indicates the predictions have successfully enveloped the optimised solutions and adequately captured the dynamic variances occurring in the solution based on the local observation. This is evident by the somewhat three-dimensional linear relationship between the north and east coordinate, originating from the presence of the

solution protuberance as the east coordinate approaches zero.

Although the visual analysis enables the reader to understand the overall topology of the model predictions, as well as understand how well the model visually fits the data during training, this is far from a robust analysis and is purely applied for aesthetics. Additionally, since the NE model consisted of only two observations, a three-dimensional representation of the surface was achievable without having to implement additional constraints to the data. This process will not be possible for future models consisting of increased observations or will require multiple observations to be held constant.

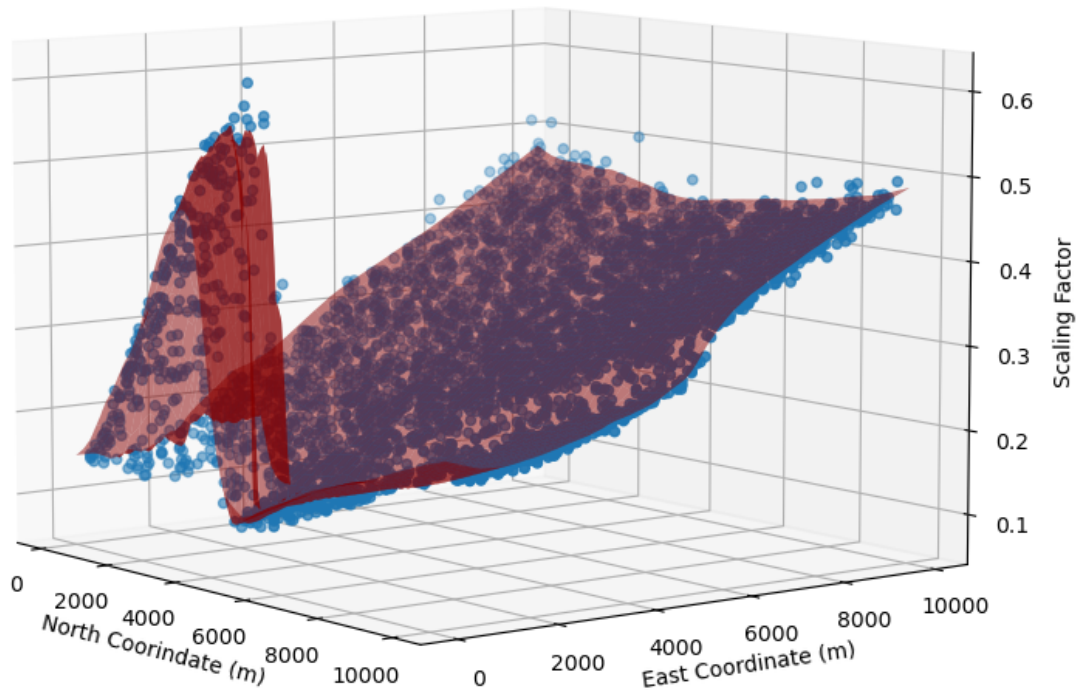


FIGURE 7.20 North-East DNN model surface prediction for the first control point scaling function

A better measure of how well the model predictions accurately fit the training data is achieved by performing a comparative study between the predictions made by the model and known optimised solutions. Since this analysis aimed to determine the fit of the network relative to the training data, the training data was iteratively applied as an observation to the model. The predictions made by the model were analysed for numerous factors with each solution assigned two classifications. Firstly, the predicted solutions were applied to the

path generation algorithm with the fitness of the resultant path analysed using the same fitness function used during optimisation. However, the fitness function was solely used to determine whether or not the dynamics of the path were within the dynamic limits of the aircraft. Therefore, if a predicted solution exceeded the dynamic limits, the path was classified as being dynamically unachievable. Secondly, if a path adhered to the dynamic constraints of the aircraft, the predicted solution was deemed dynamically achievable, and to understand how closely the predicted solution resembled the optimised solution, a direct comparison was administered. This comparison would first determine the absolute error between each of the five predicted control point scaling solutions and the five optimised control point scaling solutions. If the predicted solution was within 10% of the optimised solution, the predicted solution was categorised as representing the optimal solution.

Whilst this percentage-based analysis does allow for a moderate representation between the predicted and optimal solutions, it is flawed and does not provide a consistent metric. For example, if any of five optimised solutions approached zero, the magnitude of the scaling function would be very small. This small scaling function amplifies the percentage change due to control point rounding. Whilst this large percentage error does reflect the error between the two solutions, in reality the absolute difference between the two solutions could be negligible. To reduce this effect, a secondary assessment was conducted using the overall path length. If the predicted path exceeded the allocated 10% solution error, the path length between the predicted and optimised solutions was analysed. If the difference between the path lengths also exceeded 10%, this implies a deviation along the path occurred, revealing the solution error percentage as being a valid metric and predicted solution representing a sub-optimal path. However, if the length of the path was within the allocated 10% error, this implied that the solution percentage error was incorrect and most likely being amplified due to a small magnitude solution scaling factor. The algorithmic approach to determining feasible, optimal and sub-optimal paths, can be seen in Algorithm 3.

After training for 10,000 epochs, the comparative study indicated the DNN produced dynamically feasible solutions for 99.26% of the training data, with op-

Algorithm 3: DNN regression model path accuracy grading algorithm

```

Variables:  $x$  - Array of training observations
Output : Feasible (Boolean)
          Optimal (Boolean)

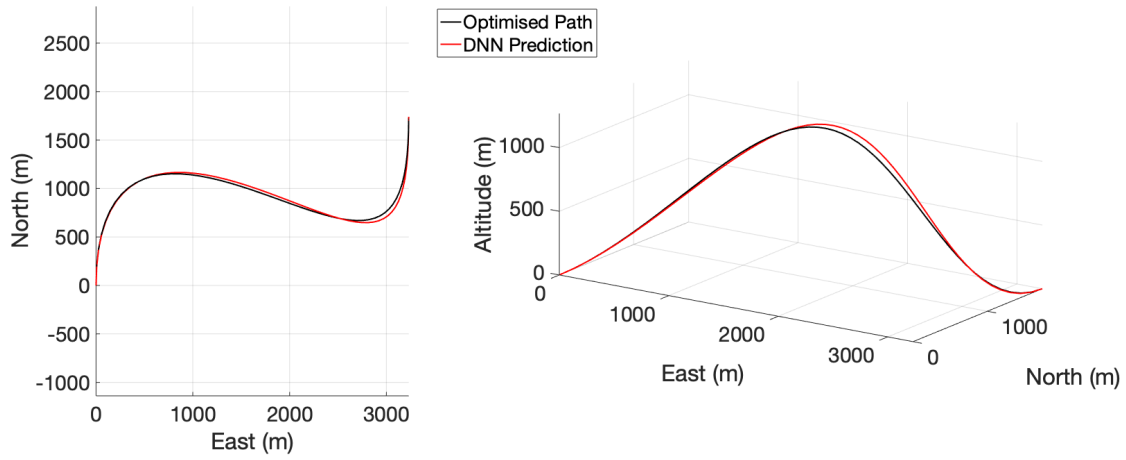
// iterate over all training examples
for Observation in  $x$  do
    1) Predict the solution from the deep neural network
    2) Use the solution to produce the corresponding path
    3) Determine the fitness of the path and whether it is dynamically achievable

    if Path is not dynamically achievable then
        Feasible = false
        Optimal = false
    else
        Feasible = true
        // Perform first metric based on control point scaling error
        if abs percentage difference between optimised solution and predicted solution > 10% then
            // Perform second metric based on path length difference
            if abs percentage difference between the optimised path length and predicted path length > 10%
                then
                    Optimal = false
                else
                    Optimal = true
            else
                Optimal = true
    return Feasible, Optimal
  
```

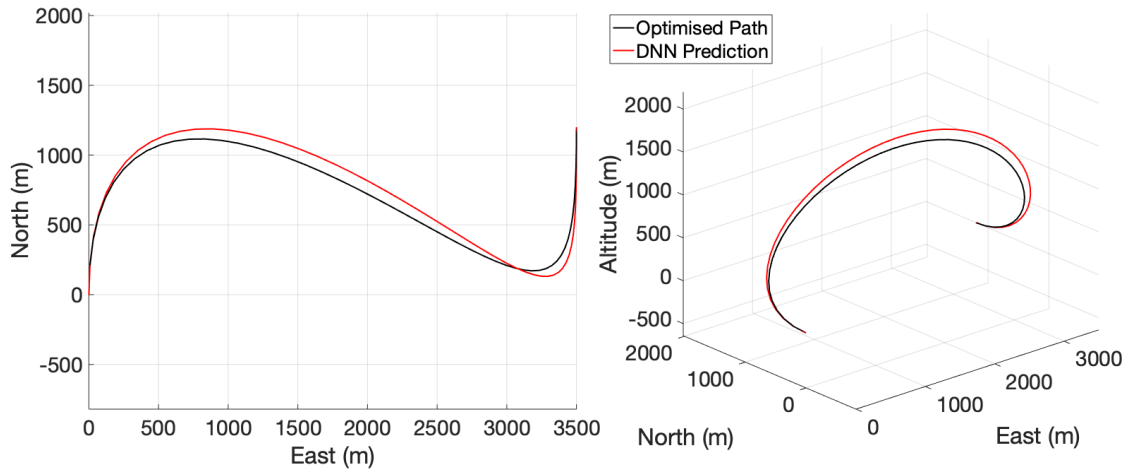
timal solutions produced for 95.42% of the training data. It is important to note that despite a reduction in accuracy when attaining optimal solutions, it does not indicate that a failed solution has occurred. An optimal solution simply implies that the predicted solution is within a certain tolerance of the optimised solution and does not necessarily constitute overall path performance. Additionally, a path can be both sub-optimal whilst also being dynamically achievable.

Since this study was conducted by comparing the output of the model against the same observations used to train the model, the output from the model had a high level of bias. Whilst this does confirm the model's fit relative to the training data, it cannot be used as definitive evidence for its performance on observations it has never seen before. To test the model's output on observations it has never seen before, optimisation was performed on 1,000 observations not present in the training dataset. By repeating this comparative study on unknown data, it can

be assured that the model performance translates accurately from the training dataset onto unknown samples.



(a) Waypoint - North: 1740 metres, East: 3231 metres - Solution optimal



(b) Waypoint - North: 1200 metres, East: 3500 metres - Solution sub-optimal but dynamically achievable

FIGURE 7.21 North-East DNN predicted paths against optimised paths for various north and east coordinate waypoints

By following the same procedure outlined in Algorithm 3, the DNN output dynamically feasible solutions for 98.4% of the unknown data samples, with 92.10% of solutions deemed optimal. Although this represents a reduction of over 3% when predicting optimal solutions, the comparative studies imposed stringent measures when classifying optimal solutions. Figure 7.21a shows a waypoint observation where the DNN predicted an optimal solution as determined using Algorithm 3, with Figure 7.21b displaying waypoint observations

where the DNN predicted a sub-optimal solution.

Although the solution in Figure 7.21b was classified as being sub-optimal, the DNNs prediction closely resembles the optimised solution, emphasising how stringent the comparative measurement in Algorithm 3 is. In addition, during the development of the NE model the structure of the neural network was not heavily considered due to the low-dimensionality of the problem. This meant a generic number of hidden layers and fully-connected neurons were used. If the internal structure of the network was to be more heavily analysed, the network performance could potentially be improved. Finally, when analysing paths deemed dynamically unachievable, all of the solutions occurred during waypoint observations where the euclidean distance between the position and waypoint was less than 1.5 km, resulting in paths pushing the imposed dynamic limits of the vehicle.

7.5.3 North-East-Heading Model (NEH)

The North-East-Heading (NEH) model was the final incremental model before the overall solution. The observations remained largely the same as the NE model, with the only exception being that the heading angle of the waypoint was now considered to be a variable, as seen in Figure 7.22. The heading angle for the

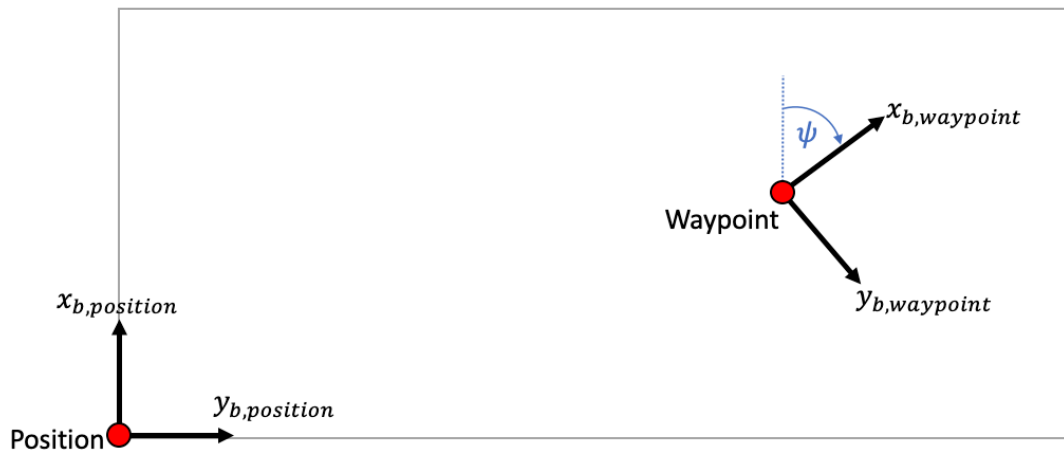


FIGURE 7.22 North-East-Heading model observational configuration

waypoint ranged from 0° to 360° and was rounded to the nearest degree. The constraints applied to the position vector remained the same as those applied

to the NE model. This gave the NEH model a total of three input dimensions: (i) waypoint north coordinate; (ii) waypoint east coordinate; and (iii) waypoint heading angle.

By increasing the number of observations from two to three, the dimensionality of the problem became more complex. During optimisation of the NE model, it was discussed how due to rounding of the individual components forming the observation, a total of 10,000 discrete observations existed. Since the same observational rounding was administered to the NEH model with the addition of a heading angle, the total number of discrete observations increased from 10,000 to 3,600,000. This is quite a significant increase given that only one additional observation has been introduced, and that path planning is still being performed with major assumptions, such as only one considered velocity (200 m/s), fixed initial position and orientation, and no demanded altitude change.

The training dataset for the NEH model comprised 60,000 randomly generated observations with optimised solutions obtained using the same optimisation process as the NE model. Outliers were removed from the training data based on their respective z-score as well as the removal of repeated observations, as per the methodology outlined during the analysis of the NE model. The resultant dataset was partitioned into a training and validation array, with a ratio of 20:1. The DNN was configured with six hidden layers each consisting of 300 fully connected neurons with no dropout between each layer, resulting in a total of 544,505 trainable parameters. Training was performed with a learning rate of 1×10^{-4} using the RMSprop optimiser. To prevent the model from over-fitting the training data it was periodically saved every 500 epochs. Additionally, a model would be saved and training ended if the loss metric during training fell below 0.001.

Due to the additional dimension of the NEH model, producing a three-dimensional surface of the model's predictions could only be performed if at least one of the dimensions was held constant or representing a certain range of values. Therefore, to obtain a three-dimensional visual representation of the model's output, the data was partitioned into multiple samples based on a set of discrete heading values, described by intervals of 10° . Since the range of heading orientations for the training dataset was comprised of values between 0°

and 360° , there existed a total of 36 samples. Therefore a single arbitrary heading range was selected for investigation between 170° and 180° . This heading range was chosen because a heading angle of 180° results in the largest absolute heading difference between the initial position and waypoint target since the orientation was wrapped between 0 and 2π .

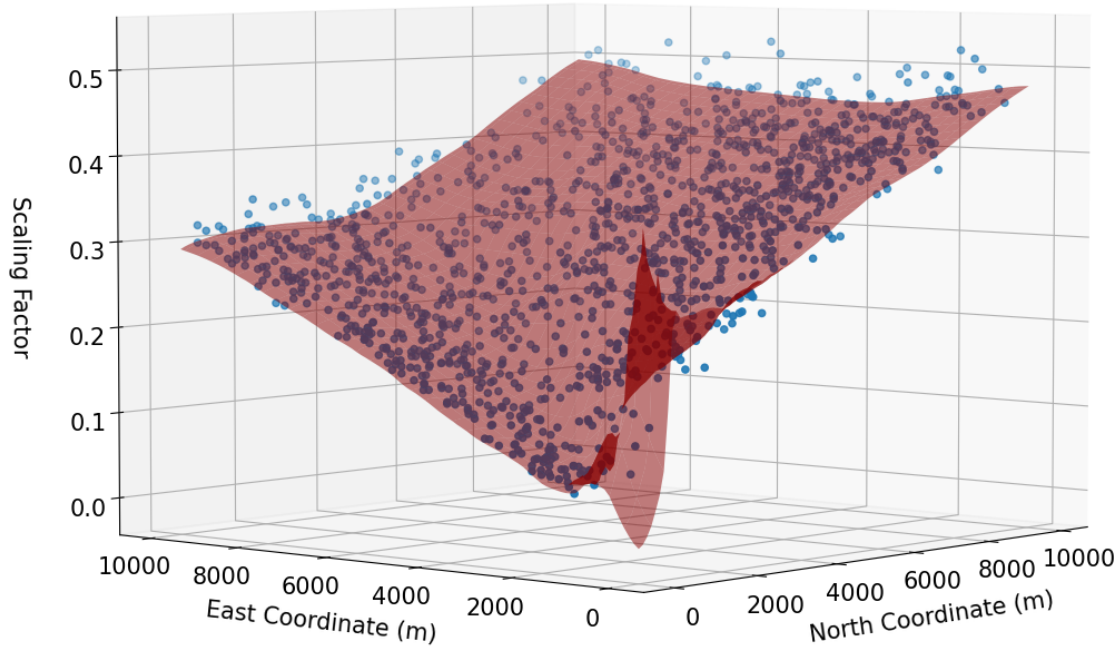


FIGURE 7.23 North-East-Altitude DNN model surface prediction for the second control point scaling function for heading angles 170° to 180°

Based on a sample of data corresponding to all north and east waypoint observations where the heading angle of the waypoint was between 170° and 180° , two linearly spaced arrays containing north and east coordinate observations between 0 and 10 km were created with intervals every 100 metres. The model was then iteratively queried for all north and east coordinate observations with a set heading observation of 180° , allowing a three-dimensional surface to be developed for each of the five control point solutions. Figure 7.23, shows the optimised sample solution for the second control point scaling factor displayed as a scatter. It is important to note that these data points only represent solutions where the observational heading was within 170° to 180° . Similar to the NE analysis, it is visually apparent that the three-dimensional predicted surface accurately represents the trend of the training data, across the entire operating

range.

Similar to the NE model, it was important to obtain a more robust understanding of the prediction accuracy of the DNN as opposed to visualising the model's fit to the training data. Therefore, the same comparative study performed for the NE model was applied to the NEH model, with predicted solutions assigned two classifications as per Algorithm 3. After training for 100,000 epochs the DNN was able to predict dynamically feasible paths for 98.67% of the training data with 96.53% solutions deemed optimal. When tested on 5,000 unknown observations, the DNN was able to predict dynamically feasible solutions for 98.33% of observations with 91.48% of solutions categorised as optimal.

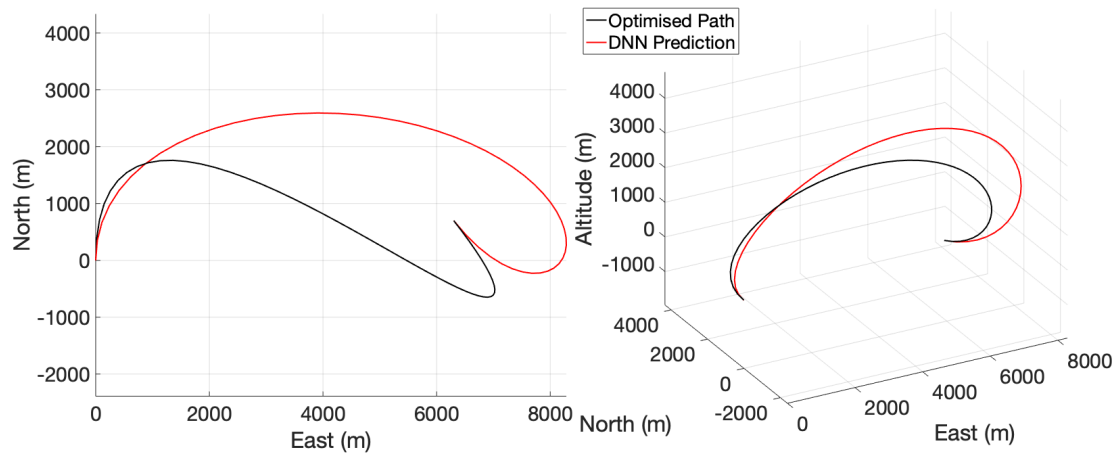


FIGURE 7.24 Waypoint - North: 3233 metres, East: 4620 metres, Heading: 4.25 radians
- Path classified as sub-optimal but dynamically feasible

During the analysis on unknown observations there existed noticeable conditions where sub-optimal paths appeared to be produced more frequently. For example, the path shown in Figure 7.24 maintains a rather unique configuration, where the north coordinate and heading angle change between the position and waypoint target is small, whilst the change to the east coordinate is large.

Due to the large change in the east coordinate, the curvature of the path results in a large right turn, meaning that locally along the path, the direction of travel approaches 90° . As the path approaches the waypoint target the difference between the path heading and waypoint heading is larger than the initial observation, and to satisfy the orientation constraint, the path loops around the back of the waypoint to align the headings. Due to the dynamics of the path under

this configuration, it was evident there exists multiple stages to performing this manoeuvre, which increased the potential number of solutions for the optimiser. For example, the path could perform the final heading realignment solely in the x - y plane, reducing the necessary climb but in the process and restricting the curvature. Additionally, orientation realignment could be performed as a descent period, where during the initial stages of the path, a brief climb occurs. This exact formation can be seen in Figure 7.24, where both the predicted and optimised solutions perform this type of manoeuvre.

Due to the various solutions available under this configuration, it is apparent that the optimised solutions are inherently not as repeatable when compared to much simpler observation configurations, resulting in quite a substantial error between the optimised and predicted solutions. However, regardless of the error, the paths produced remain dynamically achievable based on the fitness function, albeit sub-optimal against the criteria defined in Algorithm 3.

7.5.4 Final Solution

Having established that a regression model offers accurate predictions for multi-dimensional problems, as demonstrated during the NE and NEH model analysis, the full solution was formulated. Unlike the NE and NEH models, the full solution needed to incorporate a variable position vector, with observations corresponding to north, east and altitude coordinates as well as the orientation of the position vector. By incorporating a set of position vector observations, the number of observations increased from three to seven. Additionally, the full model must also incorporate a variable altitude observation for the waypoint target as well as various initial airspeeds, increasing the total number of observations to nine.

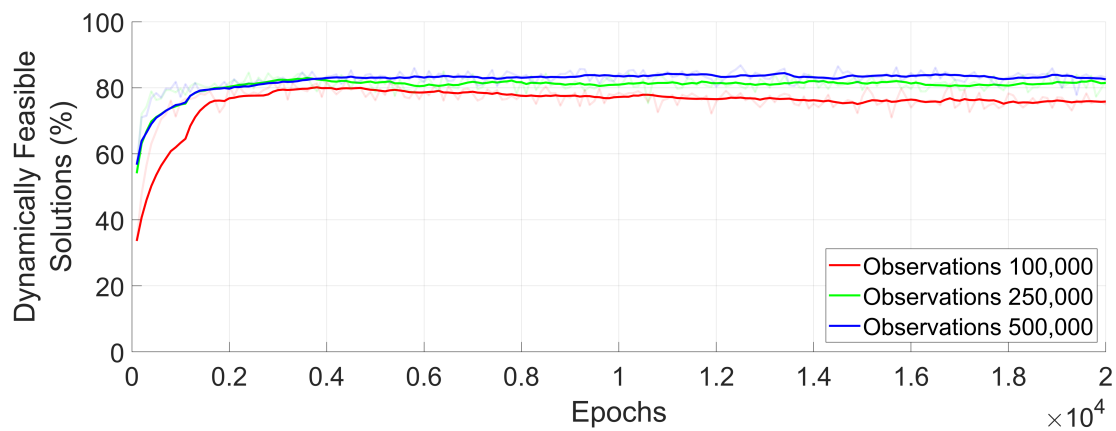
During the development of the NE and NEH models, these models had a relatively small number of observations when compared to the full model. This small number of observations allowed a large percentage of discrete observations to be accounted for within the optimised training data. For example, the NE model had a total of 10,000 discrete observations with the training data comprised of approximately 4,000 unique observations. This meant that the NE model was trained using data covering approximately 40% of the optimised op-

erating region. The NEH model was trained on 60,000 unique observations with a total of 3,600,000 discrete observations identified. This meant training was performed using 2% of the total observational area, whilst maintaining similar levels of accuracy when compared to the NE model.

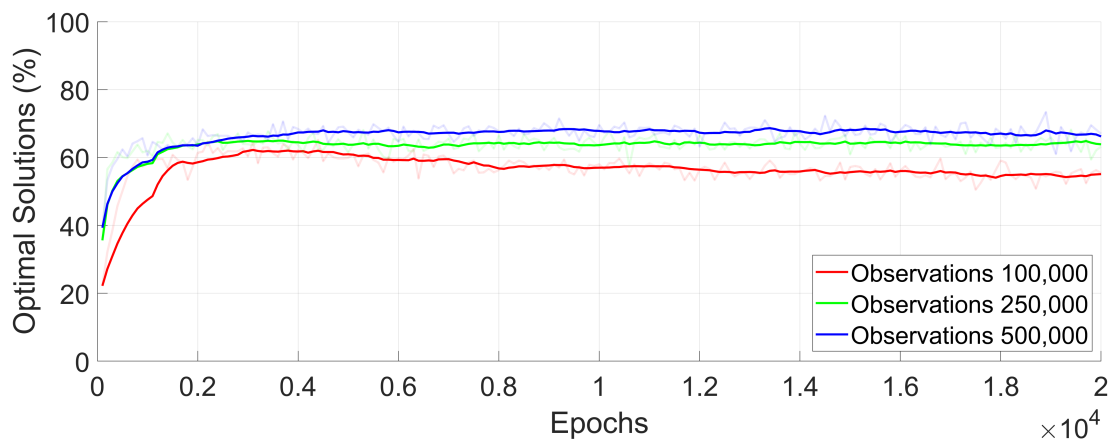
When considering an optimised observational area for the full model occupying the same region of space as the NE and NEH models and with the same interval sizes, there exists a total of 7.0546×10^{17} discrete observations. Whilst it was known that a high number of observations existed, since this was the main reason for using a regression network as opposed to a classification network, training the model on similarly sized optimised regions was not feasible. To train the model on 2% of the optimised region, a total number of 1.411×10^{16} unique optimised solutions are required. When performing parallel path optimisation using the applied GA with the workload distributed evenly over eight processing cores, approximately 2700 solutions were obtainable every hour. To optimise the number of observations required to complete 2% of the optimised space, would require an estimated 598,171,889 years of constant optimisation. Whilst this value could be reduced by increasing the interval size between observations or reducing the size of the optimised operational area, these were not appropriate solutions since the dimensionality of the problem was still extremely high and became a bottleneck for performance.

Due to these factors, it was determined that the training dataset for the full model needed to be populated with enough optimised solutions as to sufficiently capture the overall dynamics of path planning, but not large enough as to become unfeasible and unnecessarily time consuming. Additionally, as the size of the training array increased in size, the time required to train the neural network over a single epoch also increased. Therefore, the required number of training observations was considered to having an optimal solution, determined as a trade-off between improvements made to the model's predication accuracy and the time requirements to obtain additional optimised solutions as well the required training time. However, determining the optimal number of observations was not a problem solvable analytically. Therefore, an analysis was conducted to identify an adequate number of solutions based on understanding the accuracy of trained models.

To perform this analysis a network was created with an internal structure comprised of five hidden layers with each layer containing 250 fully connected neurons. The size of this network was considered to be small and selected for this reason, since this could also act as a lower limit for determining the optimal size of the network. Initially, the network was trained for 20,000 epochs with 100,000 unique optimised observations. The number of observations was increased until the accuracy of the network started to produce diminishing returns. For all tested observation sizes, the optimised data was partitioned into training and validation datasets with a ratio of 10:1. Throughout the training



(a) Dynamically feasible solutions (moving average)



(b) Optimal solutions (moving average)

FIGURE 7.25 Final model observations analysis conducted with 100,000, 250,000 and 500,000 observations on five-layer, 250 neuron DNN

process, the network was periodically saved every 20 epochs to allow an accuracy analysis to be conducted. This accuracy analysis followed the same meth-

odology as the comparative study for the NE and NEH models, with solutions assigned classifications based on whether or not they are dynamically feasible and represent optimal solutions based on Algorithm 3.

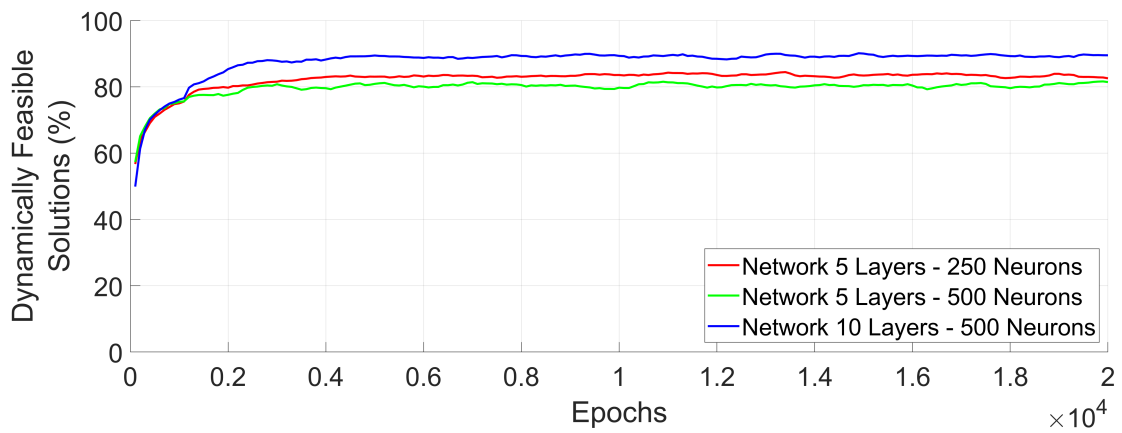
Figure 7.25a-b shows the accuracy of the network to predict dynamically feasible paths and optimal solutions when trained using 100,000, 250,000 and 500,000 observations. When comparing the accuracy values of the networks trained with different size observations, one interesting characteristic is the degradation of the network trained with only 100,000 observations. This degradation is likely due to the low observation size resulting in unstable model optimisation, causing abrupt fluctuations in performance, the longer training is conducted.

When observing the model trained with 250,000 observations, although the training process stagnates very quickly as indicated by the accuracy level, the accuracy remains stable with small improvements occurring over time. Although the model trained with 500,000 epochs does produce higher levels of accuracy, the improvements began to diminish with an accuracy improvement of around 2% for dynamically feasible solutions. This indicates that further improvements to the network's prediction accuracy will not occur through simply increasing the number of observations, with poor accuracy due to a low number of trainable parameters. For this reason, it was determined that the low level of accuracy displayed by the network, was most likely not due to the number of observations, but rather the structure of the model and the training process.

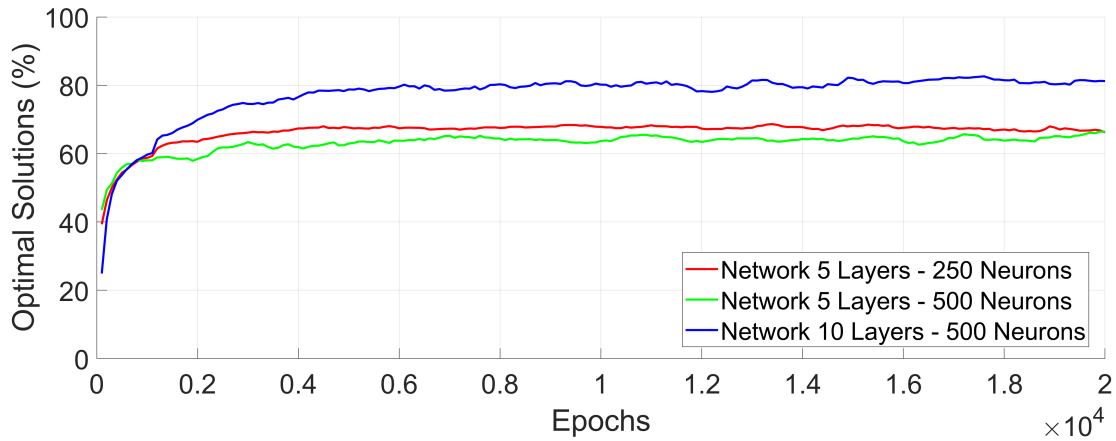
To see if improvements could be made to the network by adjusting the internal structure, multiple networks were trained using 500,000 observations, with a varying number of hidden layers and fully connected neurons. By increasing the number of hidden layers and fully connected neurons, it can be expected that the model's fit to the training data can become more accurate due to an increased number of trainable parameters, allowing the characteristics of the training data to be more highly distinguished and represented. The results from this analysis can be seen in Figure 7.26.

Although it was expected that increasing the number of trainable parameters would improve the performance of the networks predictions, this was not experienced when only increasing the number of neurons. Increasing the number of fully connected neurons from 250 to 500 whilst retaining a hidden layer

size of five, negatively impacted the accuracy of the network, despite doubling the number of trainable parameters. However, by increasing the number of hidden layers from five to 10 and increasing the number of neurons to 500, significant improvements were measured to the accuracy. The network predictions improved dynamically feasible paths by more than 6%, with optimal solutions increasing by almost 15%. This large improvements to the network predictions provided dynamically feasible paths for almost 90% of the unknown tested solutions, with approximately 83% of the predicted solutions deemed optimal when compared to the optimised solution.



(a) Dynamically feasible solutions (moving average)



(b) Optimal solutions (moving average)

FIGURE 7.26 Various network structures for final model trained with 500,000 observations

Although the accuracy of the network remains relatively low when compared to previously trained networks, and remains substantially lower than the

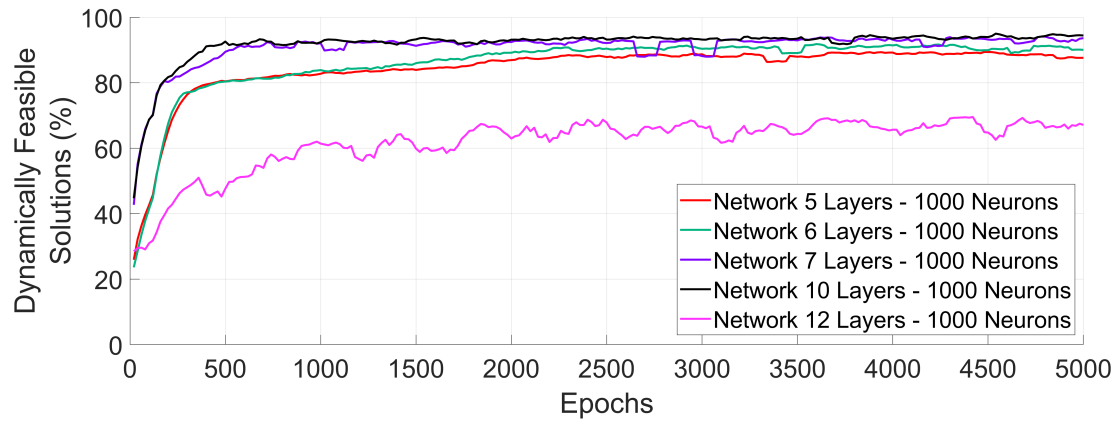
99.55% accuracy of the optimiser when applied directly to the aircraft model, this analysis confirmed that improvements to the network predictions can be established without having to increase the size of the training data. The accuracy improvements obtained in the network predictions shifted the focus of the network structure around 10 to 20 hidden layers, and over 500 fully connected neurons.

Having established that the a 10 layer network where each layer consisted of 500 fully connected neurons provided a prediction accuracy of over 90% for dynamically feasible paths and 83% optimal solutions, the number of neurons was increased to 1000. Increasing the number of hidden neurons to 1000 provided a substantial increase to the number of trainable parameters, posing numerous issues. One of the main issues with further increasing the internal structure of the network, occurred due to the increased computation time required for training the network.

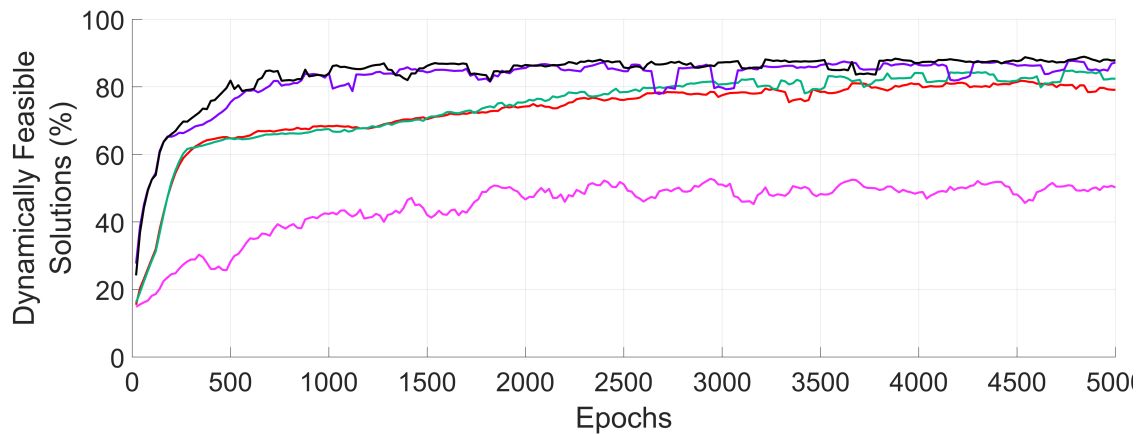
During training, the trainable parameters within the network are adjusted every epoch using optimisation through back-propagation. As the number of trainable parameters increases, the computation time required to complete a single epoch also increases, prolonging the training process. During training of the network consisting of five hidden layers and 250 neurons (Figure 7.25), the network consisted of approximately 317,000 trainable parameters. When training this network on 500,000 observations, a single training epoch required approximately 0.6 seconds. However, by increasing the hidden layer size to 10 and number of neurons to 1000, the number of trainable parameters exceeded 10 million. This increased the time required to complete a single epoch more than 10 times, to over eight seconds. Due to the time constraints of this research, this made it difficult to increase the size of the network structure any further due to the time complexity associated with training such a large and complex neural network.

Additionally, due to limitations in the hardware available, the memory allocation during training was limited to 8 GB. During training of the network, the network is stored in the GPU's random access memory. For large models consisting of millions of trainable parameters, loading these models into the allocated memory occupied a large quantity of the available resources. Increasing the size

of the network further, would exceed the memory allocation and not allow the training data to be stored. For these reasons, the number of hidden neurons within the network was limited to 1,000 with the final study used to determine the optimal number of hidden layers.



(a) Dynamically feasible solutions (moving average)



(b) Optimal solutions (moving average)

FIGURE 7.27 Network performance analysis for various number of hidden layers, where each layer was comprised of 1,000 fully connected neurons. Training was performed for 500,000 observations.

To determine the optimal number of hidden layers, a network was configured with 1,000 fully connected neurons with an initial layer size of five. During the previous studies, training was administered for 20,000 epochs, with the model incrementally saved every 20 epochs. However, based on these studies, it is evident that training each model for 20,000 epochs was not necessary, given that the progress of the network began to stagnate after only a few thousand epochs. Therefore, training for the layer size analysis was conducted over 5,000 epochs.

Upon completing the training process, the layer size was increased, and training conducted once again. This occurred until the performance of the network peaked. The performance of the network was evaluated using the loss metric during training as an initial benchmark, before performing same comparative study as used during previous analyses, and outlined in Algorithm 3. The results from this study can be seen in Figure 7.27, for multiple layer sizes ranging from five to 12.

When comparing dynamically feasible paths predicted by the networks in Figure 7.27a, it is evident that incrementally increasing the hidden layer size has gradually began to improve the networks performance. However, improvements to the performance begin to diminish as the number of hidden layers approaches 10. Both networks trained with seven and 10 hidden layers, predicted dynamically feasible paths for 96.1% of the tested paths during the comparative study.

Although Figure 7.27a indicates the performance of the network to predict dynamically feasible paths stagnates around a layer size of 7, the networks ability to predict optimal solutions continues to show improvements until a layer size of 10. The network trained with 10 hidden layers achieved 91% accuracy when predicting optimal solutions, compared to the 89% accuracy achieved for a layer size of 7. Although this accuracy level is lower than previously optimised and lower-dimensional networks, given the complexity of the optimisation problem and the large network structure, this level of accuracy is quite impressive.

As the number of hidden layers exceeded 10, the performance of the network quickly degraded, with the network consisting of 12 hidden layers achieving a performance accuracy of around 65% for dynamically feasible paths. To ensure that this was not an anomaly, training was performed for a new network consisting of 12 hidden layers, with the performance of the network yielding consisted results with those displayed in Figure 7.27. This indicates that as the number of hidden layers exceeds 10, the structure of the network does not allow for stable training, with large fluctuations occurring during the training process.

Having determined that a network structure consisting of 1000 neurons with 10 hidden layers produced the best accuracy in terms of dynamically feasible and optimal solutions, this network architecture was selected as the final solution. During the hidden layer analysis, each model was trained for 5,000 epochs as this

allowed the general performance and capabilities to be understood. Although this was correct, the training progress of the selected model structure indicated slight improvements over time, with the current best model achieved after 4,800 training epochs. Given that the best model occurred towards the end of training and the general trend indicates small improvements can be made, the model was trained for an additional 15,000 epochs. This additional training period allowed the model to adjust the internal network parameters to make small improvements to the model accuracy. During this training period the model was able to improve the accuracy for predicting dynamically feasible solutions up to 97% with optimal solutions remaining consistent, at approximately 91%.

7.6 Model Implementation and Testing

Similar to the constant-altitude model in Chapter 6, testing of the variable-altitude model was performed in two stages: static waypoint testing and dynamic waypoint testing.

7.6.1 Static Waypoint Testing

To test the network's ability to create dynamically feasible and optimal paths, testing of the DNN during static waypoint scenarios was conducted first. Similar to the constant-altitude path planning, static waypoint testing was performed for randomly generated position and waypoint targets, where the waypoint is at a fixed location in the inertial frame throughout each evaluation. Although the DNN was trained on data corresponding to a specific region within the inertial frame, the network can be queried for any inertial frame observations, with an output determined based on the internal parameters of the network. However, the accuracy of the network to predict dynamically feasible and optimal paths cannot be guaranteed for path planning observations located externally to the optimised region.

To test this, the static waypoint analysis was performed in two stages: (i) operation with observations solely located in the optimised region; (ii) operation with observations located externally to the optimised region. Figure 7.28 shows an x - y view of the inertial frame, where the illustrated regions are defined from

–10 km to +20 km along both the inertial north (N_i) and east (E_i) axes. Although these axes have been denoted as N_i and E_i , they refer to the inertial x_i and y_i axes respectively.

This inertial frame representation has been partitioned into nine individual regions, where each region occupies a 100 km² area along the inertial frame x - y plane. Although not visually apparent in Figure 7.28, each operating region expands along the z_i axis from 0 km to 5 km. Each region has been assigned a numerical identifier, allowing each region to be individually referenced throughout the discussion. As well as a numerical identifier, each region has been assigned a colour code of either green, yellow or red.

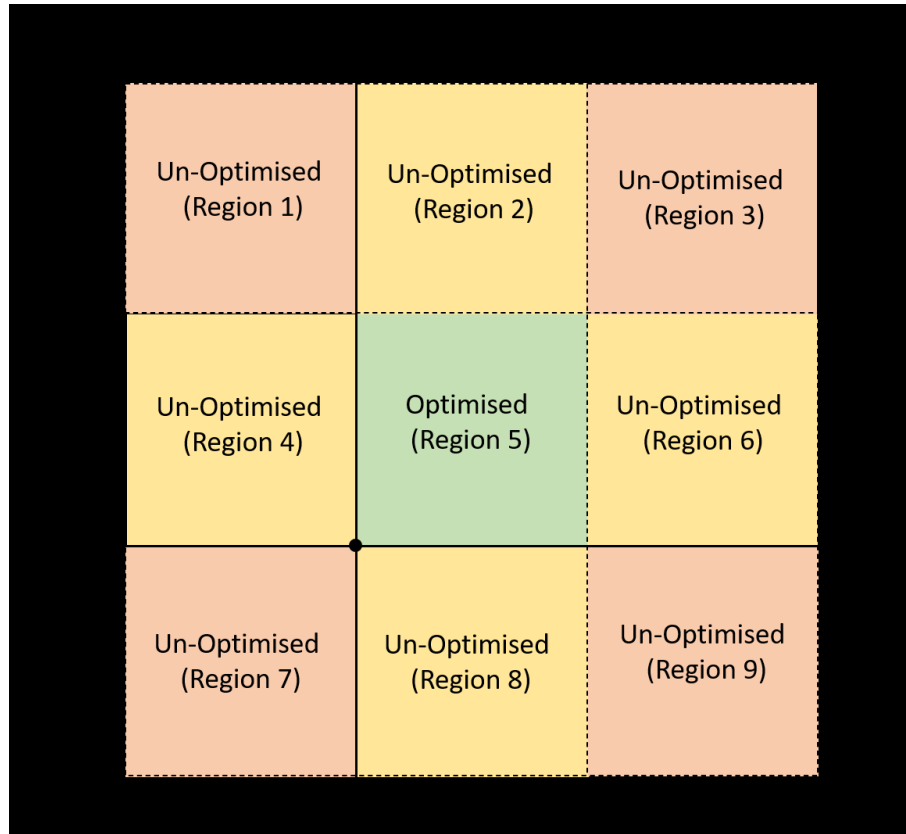


FIGURE 7.28 Variable-altitude DNN optimised space in the north/east axes with respect to the inertial frame.

These colour codes correspond to the number of unknown observations within the model. The green colour code represents the optimised region, where all path planning observations are known to the model and were used during the training process. The yellow region indicates that either one of the north or east

observations is unknown, whilst the red region indicates that both north and east observations are unknown. For example, the un-optimised region holding the numerical identifier number two, occupies the inertial frame from a north coordinate of +10 km to +20 km and an east coordinate of 0 km to +10 km. Since the DNN was optimised for data corresponding to 0 km to +10 km along the north and east axes, the north observation exceeds this condition, meaning the north observation is unknown for region two, whilst the the east observation is within the optimised observation range. This means that one observation is unknown, hence the yellow colour code.

Optimised Space

Static testing of the variable-altitude model within the optimised space was performed by randomly initialising the position and orientation of the aircraft and waypoint within the optimised region, as indicated in Figure 7.28. For this analysis, both the waypoint target and location of the aircraft were placed within the optimised region, with flight not considered between regions. However, if either the position of the aircraft or waypoint was located near the boundaries of the optimised region, the aircraft was allowed to navigate into neighbouring regions provided this was as a result of the predicted DNN solution. The initial airspeed of the aircraft was randomly generated between 100 m/s to 300 m/s, with the airspeed controller set to maintain this velocity throughout the duration of the simulation.

Upon initialising the position and orientation of the aircraft and waypoint targets, the DNN was queried to predict the control point scaling coefficients, with the resultant path generated. The aircraft model manoeuvred along the path until it reached the waypoint. Initially, path re-planning was not performed meaning that if the aircraft deviated more than 300 metres away from the path, the simulation ended. If this occurred the path was deemed to be dynamically unfeasible since the aircraft model was unable to adequately follow the path. If the aircraft successfully made it to the waypoint, the path was deemed to be dynamically achievable and the methodology used during constant-altitude static waypoint tested outlined in Algorithm 2 used to determined whether the solution predicted by the network was optimal. After completing over 7,000

simulations, the DNN predicted dynamically feasible solutions with an accuracy of 95%, with 94% of solutions deemed optimal (Table 7.5).

During training of the variable-altitude DNN in Section 7.5.4, a methodology was outlined (Algorithm 3) using a test dataset to evaluate the performance of the network. This test dataset contained optimised path solutions for observations within the optimised region. However, the information within this test dataset was not used during training of the neural network. Since this data was unknown to the network but corresponded to observations located within the optimised region, the performance of the network predictions were measured by comparing the predicted regression solution against the known optimised solution. During that analysis, it was found that the neural network achieved an accuracy of approximately 97% when predicting dynamically achievable solutions, with 91% of solutions optimal. These results quite closely resemble the accuracies of the solutions predicted by the DNN when applied to the aircraft model, adding validity to the testing procedure.

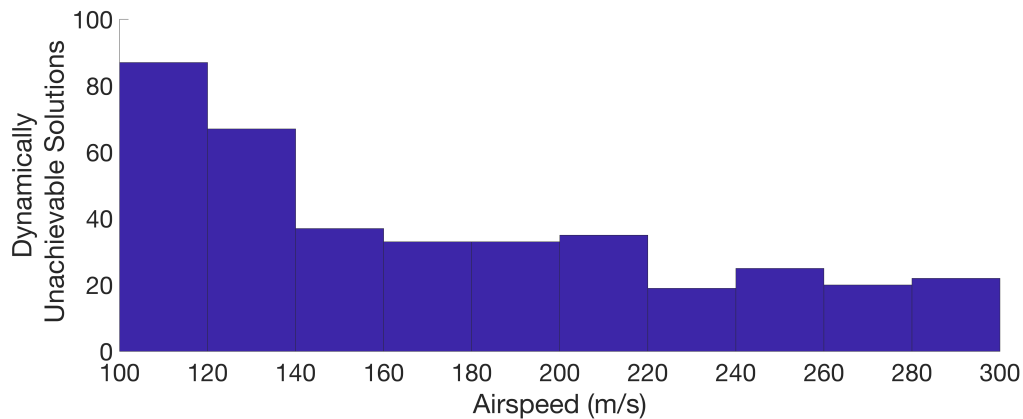


FIGURE 7.29 Distribution of dynamically unachievable solutions for static waypoint analysis with no path replanning

Despite a high level of accuracy, a drop-off of around 2% was experienced between dynamically feasible solutions during the analytical analysis and computational study. This drop-off indicates that the criteria used for analytically establishing dynamically feasible solutions in Section 7.5.4 did not account for all non-linearity within the aircraft model. In addition, when analysing simulations where the predicted solution from the DNN was deemed dynamically unachievable, a correlation between unachievable solutions and airspeed was

present. More than 40% of solutions deemed dynamically unachievable occurred for an airspeed between 100 m/s and 120 m/s, with more than 60% occurring below 150 m/s. This trend can be seen in Figure 7.29, where the distribution of dynamically unachievable solutions is displayed against airspeed.

Additionally, approximately 60% of dynamically unachievable solutions occurred when the euclidean distance between the position and waypoint target was less than 4 km. During constant-altitude path planning, it was found that as the euclidean distance between the aircraft and waypoint decreased, the optimiser began to trade-off between increasing the length of the path and ensuring the dynamics of the path are within the limits of the aircraft. This often pushed paths to extract the highest level of performance from the aircraft, whilst maintaining the shortest path. Although analytically the fitness function during optimisation categorised these solutions as dynamically feasible paths, when applying the paths on the non-linear computational model, this did not always translate into a feasible path. This factor became more prevalent as euclidean distance between the aircraft and waypoint reduced and the corresponding path started to increase the dynamic demand of the vehicle.

Finally, upon closer investigation of the individual scenarios deemed dynamically unachievable, it was evident that many solutions had a demanding turn during the early stages of the path. This initial demanding turn caused a conflict as during this region of operation the dynamics of the aircraft are still initialising and the autopilots gathering control. This often resulted in inadequate dynamic performance during early stages of the simulation, meaning the aircraft was unable to follow the path closely enough to satisfy the applied tolerances. This type of operation would be unexpected during routine real world flight, since the aircraft would most likely not be initialised into the environment with a set of dynamic states consisting of no time history and with an initial demanding turn, therefore, a second analysis was conducted. This analysis followed the same procedure, but rather than prematurely ending the simulation if the aircraft deviated too far from the path, the path was allowed to be re-planned.

During many of the scenarios categorised as being dynamically unachievable during the first study, if the aircraft was unable to sustain a demanded turn rate, it gradually drifted away from the path. This increased the distance between the

aircraft and the selected reference point applied to the guidance law, resulting in an even tighter turn rate to orientate the aircraft along the line-of-sight. This often left the aircraft dynamically compromised particularly at low airspeeds. To prevent this behaviour from occurring during the path re-planning analysis, once the aircraft deviated more than 300 metres away from path, the autonomous flight control system engaged orientation recovery mode. During orientation recovery mode, the angular rate controllers no longer receive inputs from the guidance control system, but rather from the angular position controllers. The angular position controllers demand a roll and pitch angle of 0 degrees for a total of five seconds. Orientation recovery is purely used to re-establish dynamic control of the aircraft and prevent path re-planning from engaging the aircraft in an unwarranted climb or demand a tighter turn. After five seconds, a new path is planned based on the new position of the aircraft and its state dynamics and the guidance control system provides new inputs to the angular rate controller.

During path re-planning of the static waypoint analysis, obtaining optimal solutions for these simulations was not feasible due to the way optimal solutions were determined. Since optimal solutions incorporated a predictive path time analysis as well as a velocity over the path analysis (Algorithm 2), if a path was re-planned the predictive time analysis became redundant. In addition, during unfeasible paths it has already been discussed that often the guidance law demands larger turn dynamics to align the LOS as the aircraft deviates away from the path, dynamically compromising the aircraft. This compromise often reduced the airspeed of the aircraft and indicates the path is sub-optimal. Therefore, during path re-planning the only metric used was the success rate of the aircraft navigating to the waypoint target, with dynamically feasible and optimal paths not the focus of this aspect of the study.

After performing an additional 5,000 evaluations with orientation recovery and online path re-planning, the aircraft successfully navigated to the waypoint for 99.90% (Table 7.5) of the randomly generated scenarios, with only five scenarios deemed dynamically unachievable. Out of these five scenarios, all five incorporated a high altitude change, with either the position of the aircraft or waypoint located at the upper end of the altitude range, where the control surfaces of the aircraft are less effective. Additionally, four of the five solutions

occurred for airspeeds lower than 120 m/s. This indicates that although the aircraft was able to sustain near perfect dynamically achievable solutions, there is still some correlation between solutions being dynamically unachievable at low airspeeds and higher altitude. These characteristics were also exhibited during testing of the variable-altitude optimisation algorithm in Section 7.4.

Un-Optimised Space

Having determined that the neural network is able to predict dynamically feasible and optimal solutions within the optimised space with a high level of accuracy, understanding how this performance translated to paths located externally was important. It has already been discussed that the inertial frame represented in Figure 7.28 has been partitioned into nine equal regions, with eight of these regions categorised as un-optimised regions. Similar to the analysis on the optimised region, each of the un-optimised regions were simulated using the same procedure. The position and orientation of the aircraft and waypoint was randomly initialised within a specific region and with a randomly generated airspeed between 100 m/s and 300 m/s. Each region was analysed 1,000 times for static waypoints with path re-planning initially disabled. The results from each region can be seen in Table 7.5.

Based on the results in Table 7.5, it is evident that the accuracy of the network degrades as either the north or east coordinate becomes negative, as seen in regions one, four, seven, eight and nine. This is expected since the network was only trained on positive north and east observations, meaning that as either value tends towards -10 km, the deviation between the input and output changes quite drastically. This degradation results in dynamically feasible paths only being predicted for around 75% of the tested scenarios, with sub-optimal solutions occurring approximately 72% of the time. However, when operating in the seventh region, both the north and east observation are negative, resulting in poorer performance. The probability of predicting dynamically feasible and optimal paths within this region becomes little more than a coin toss, with only 58% of solutions feasible.

When operating in the second and sixth region, the performance of the predictions remains very good, with dynamically feasible and optimal solutions

TABLE 7.5 Variable-altitude static waypoint analysis for operation in all identified regions depicted in Figure 7.28

Operating Region (Colour)	Without Path Re-planning		With Path Re-planning
	Dynamically Feasible (%)	Optimal Solution (%)	Waypoint Success Rate (%)
1 (R)	74.40	72.20	98.81
2 (Y)	91.37	90.36	99.59
3 (R)	85.20	84.20	99.53
4 (Y)	83.03	81.22	99.70
5 (G)	95.18	94.58	99.90
6 (Y)	90.10	88.60	99.51
7 (R)	58.79	56.87	99.10
8 (Y)	75.69	72.92	99.32
9 (R)	73.49	70.88	99.02

experienced for 91% and 90% of simulations. The retention of this high performance is due to all observations being positive with at least one of the observations being an optimised quantity, resulting in only a 4-5% performance drop-off when compared to the optimised region. Since one of the observations within these regions is an optimised quantity, the network was trained on information partially related to these regions, allowing more suitable solutions to be predicted. Although region three has both observations unknown to the network, due to the fact both observations are positive, the network was able to produce adequate predictions, with feasible solutions occurring for 85% of simulations with optimal solutions for 84%.

Despite the performance degradation experienced when operating outside of the optimised region, by enabling path re-planning, the aircraft was successfully able to navigate to the waypoint target for over 99% of simulations. This demonstrates the robustness of the flight control system, that even when operating in poorly performing regions, adequate performance characteristics and closed-loop stability is maintained, allowing paths to be re-planned and successfully navigate the aircraft to the waypoint target.

7.6.2 Dynamic Waypoint Testing

Having completed the static waypoint analysis with online path re-planning both enabled and disabled, the DNN was tested during cooperative flight with a lead aircraft acting as a moving waypoint target. During the dynamic waypoint analysis for constant-altitude path planning in Section 6.7, a distance controller was used, allowing the wingman to engage in a catch-up manoeuvre and then maintain the demanded separation distance to the lead aircraft. The distance controller adjusts the throttle demand for the engine model, allowing the wingman to either accelerate, decelerate or maintain separation distance by maintaining a constant airspeed. During constant-altitude path planning, the functionality of the distance controller was not an issue since decreasing the airspeed did not negatively impact the performance of the aircraft provided it remained larger than the stall speed. However, during variable-altitude path planning, this was no longer true.

During cooperative flight, the wingman tracks the position and orientation of the lead aircraft and creates paths based on these observations. As the lead aircraft performs a manoeuvre, such as pitch up, depending on the severity of the turn rate the separation distance between the two aircraft begins to reduce quite rapidly. This rapid reduction causes the distance controller to reduce the throttle, allowing the wingman to reduce its airspeed, preventing further reductions to the separation distance. However, if this deceleration occurs during a climb or just before a climb, this impacts the performance of the wingman and increases the probability of stall occurring. This negatively impacts the stability of the aircraft and reduced the robustness of the path planning solution. For these reasons, the distance controller was removed from operation during variable-altitude flight. Therefore, throughout the remainder of Section 7.6.2, both the lead aircraft and wingman are initialised with the same airspeed and demanded to maintain that airspeed throughout any simulation. Although this prevents the wingman from maintaining a set separation distance, it still enables the wingman to track and follow the lead aircraft whilst flying its own flight plan.

Similar to constant-altitude path planning, obtaining a quantitative measure

of the wingman's performance during cooperative flight is not feasible. In addition, since the distance controller was not available for variable-altitude path planning, analysing the performance of the wingman was performed solely as a visual analysis. This meant that as the lead aircraft flies a pre-determined mission flight path, the position of the wingman at various instances in time is used to understand the performance of the system, by analysing its relative position, tracking capabilities and path generation.

The flight path for the lead aircraft was formed from numerous waypoint targets with individual Bézier curves connecting the waypoints, allowing longer and more complex flight plans to be computed. A total of three missions were performed with the parameters for each mission detailed in Table 7.6. All three missions were conducted within the optimised region used to train the neural network.

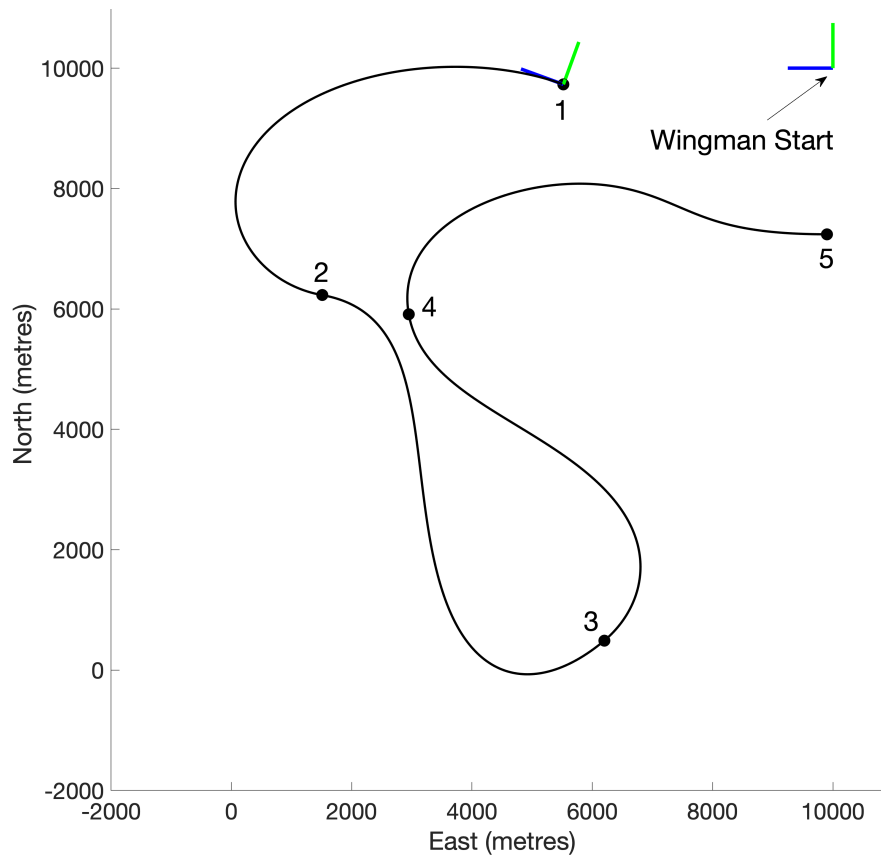
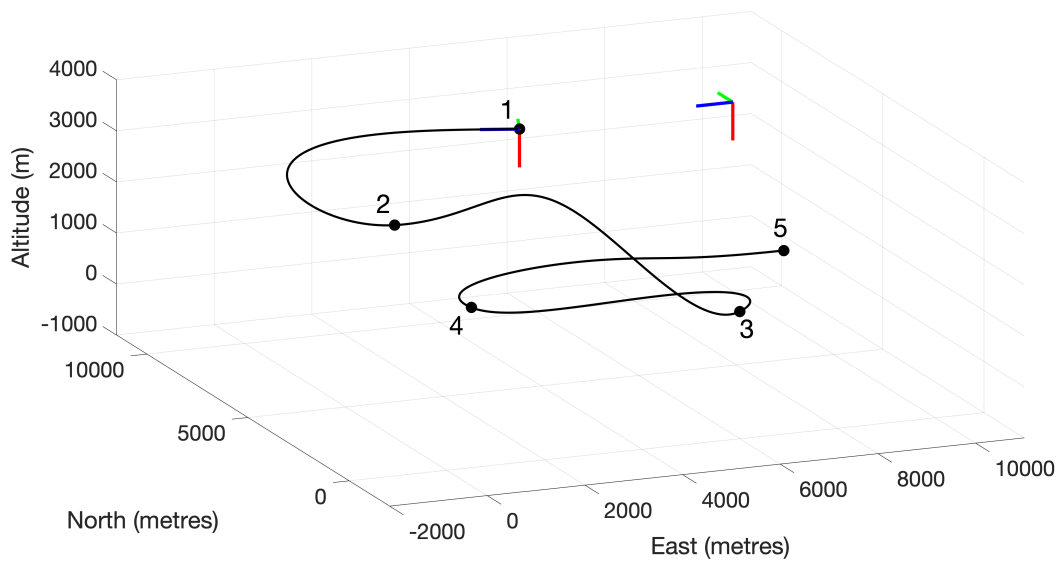
TABLE 7.6 Mission parameters for dynamic waypoint analysis of the variable-altitude path planner

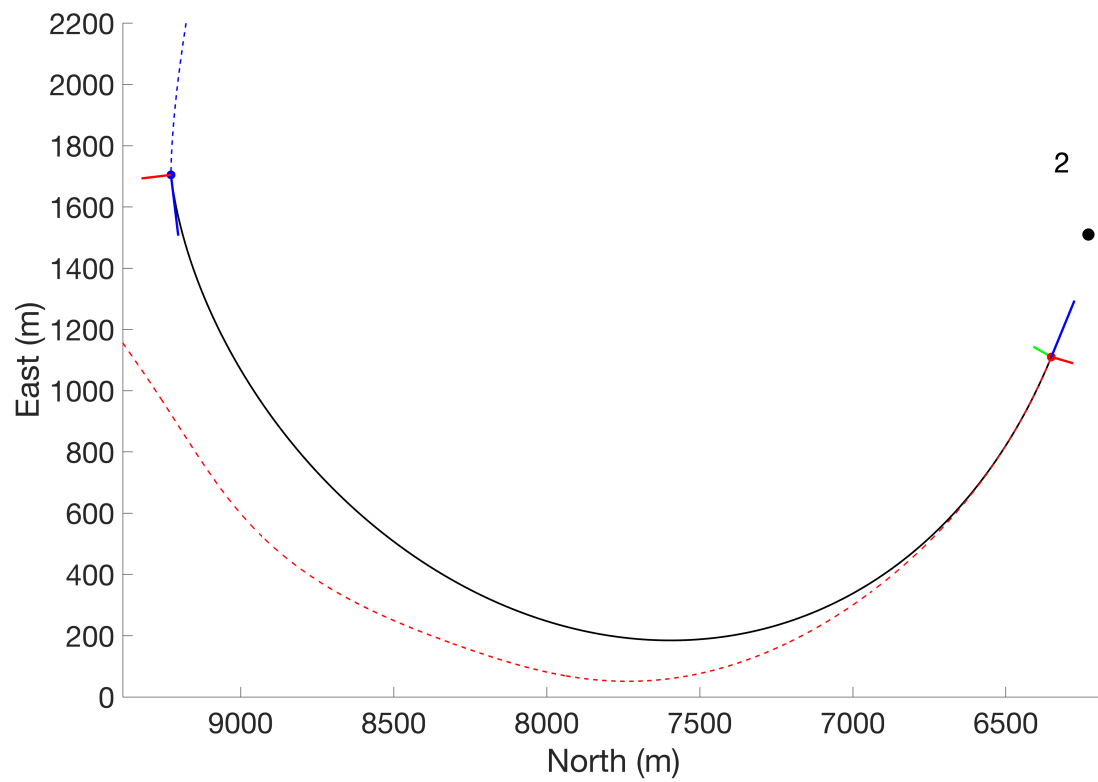
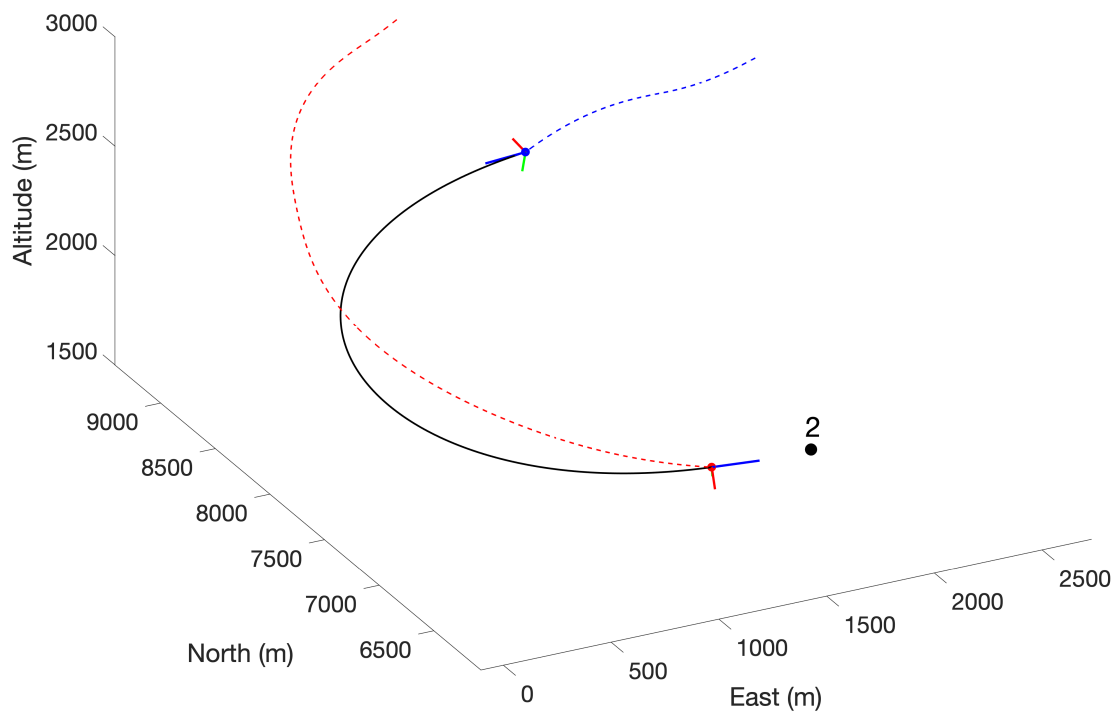
Mission	Waypoints	Lead Aircraft Start (Inertial Frame)	Wingman Start (Inertial Frame)	Airspeed (m/s)
1	5	North: 9.74 km East: 5.5 km Altitude: 2.7 km $\psi_i : 290^\circ$	North: 10 km East: 10 km Altitude: 2.7 km $\psi_i : 270^\circ$	140
2	8	North: 3.2 km East: 9.1 km Altitude: 2 km $\psi : 270^\circ$	North: 10 km East: 10 km Altitude: 2 km $\psi : 180^\circ$	200
3	8	North: 3.2 km East: 4.4 km Altitude: 4 km $\psi : 300^\circ$	North: 0 km East: 10 km Altitude: 4 km $\psi : 0^\circ$	175

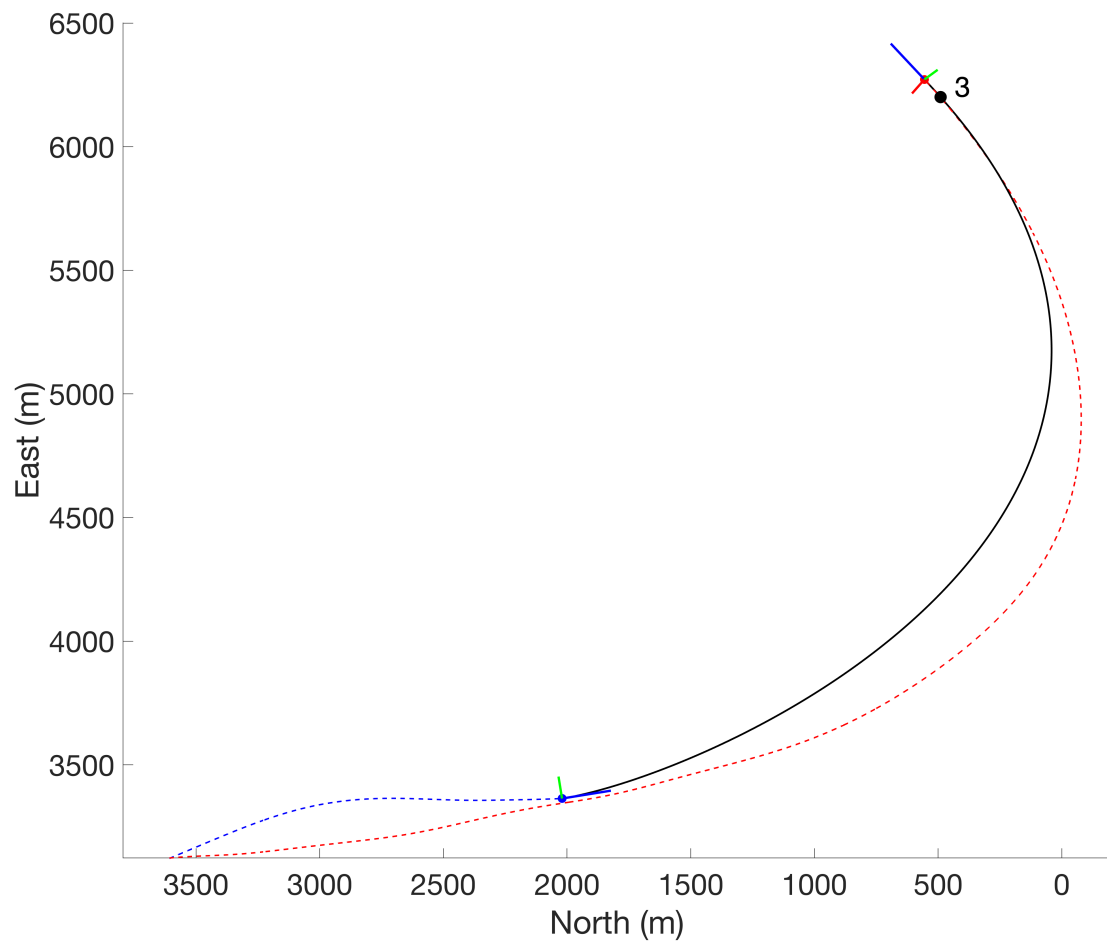
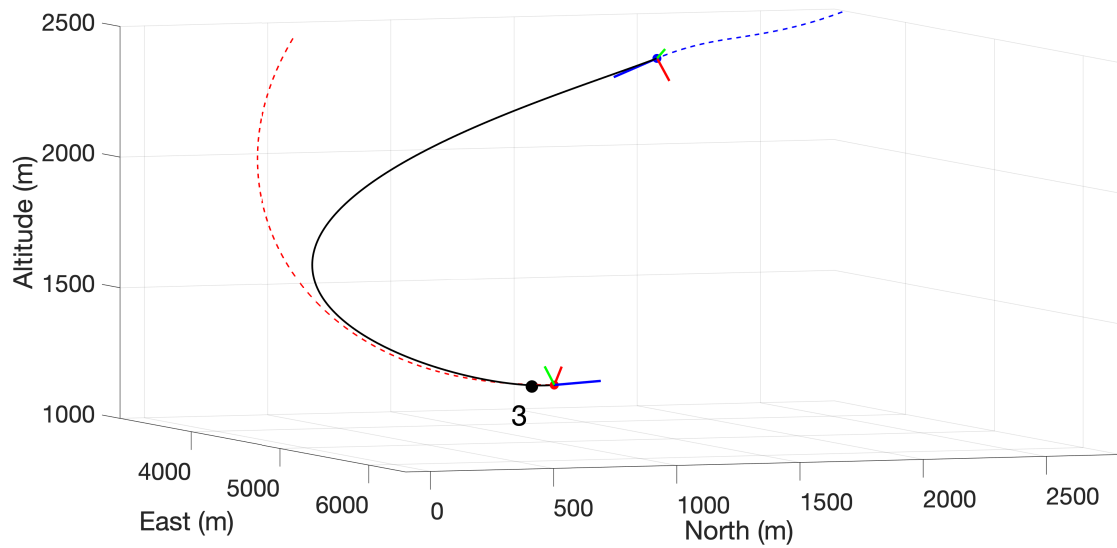
For all three missions, a visual representation of the lead aircraft and wingman are displayed as x - y and x - y - z inertial plots at a specific instance in time.

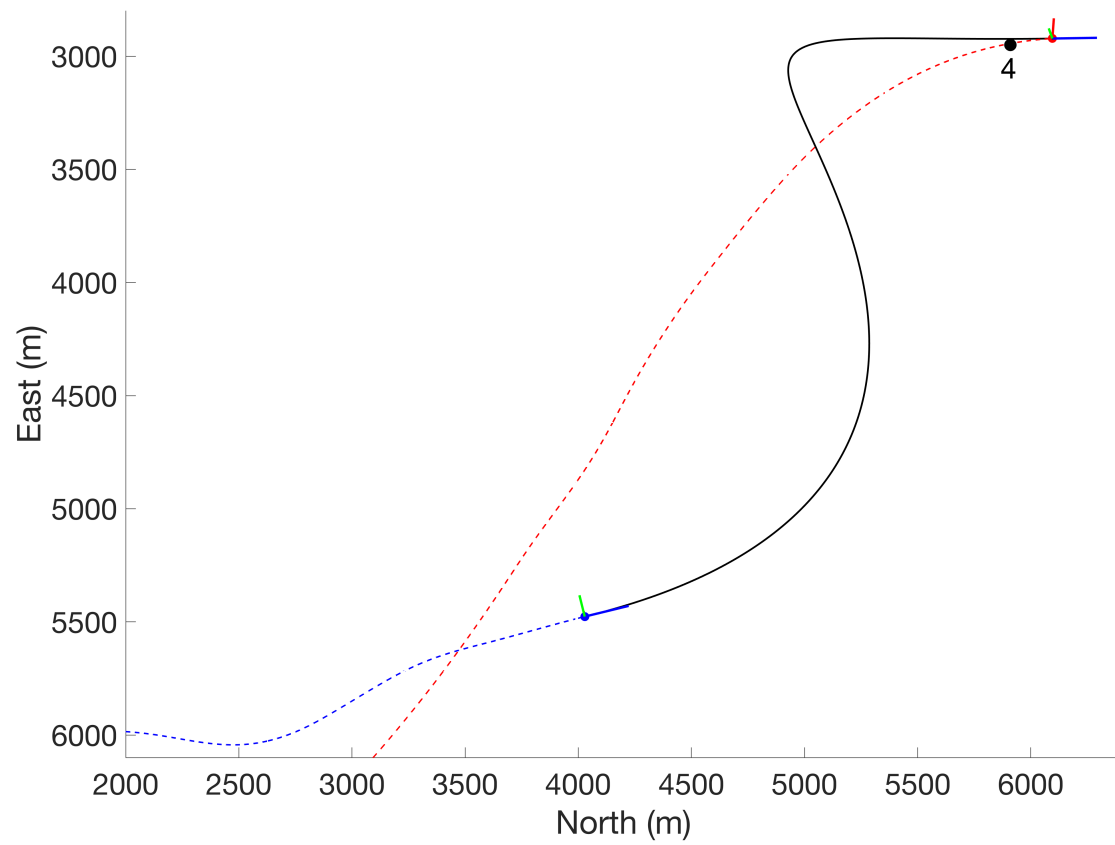
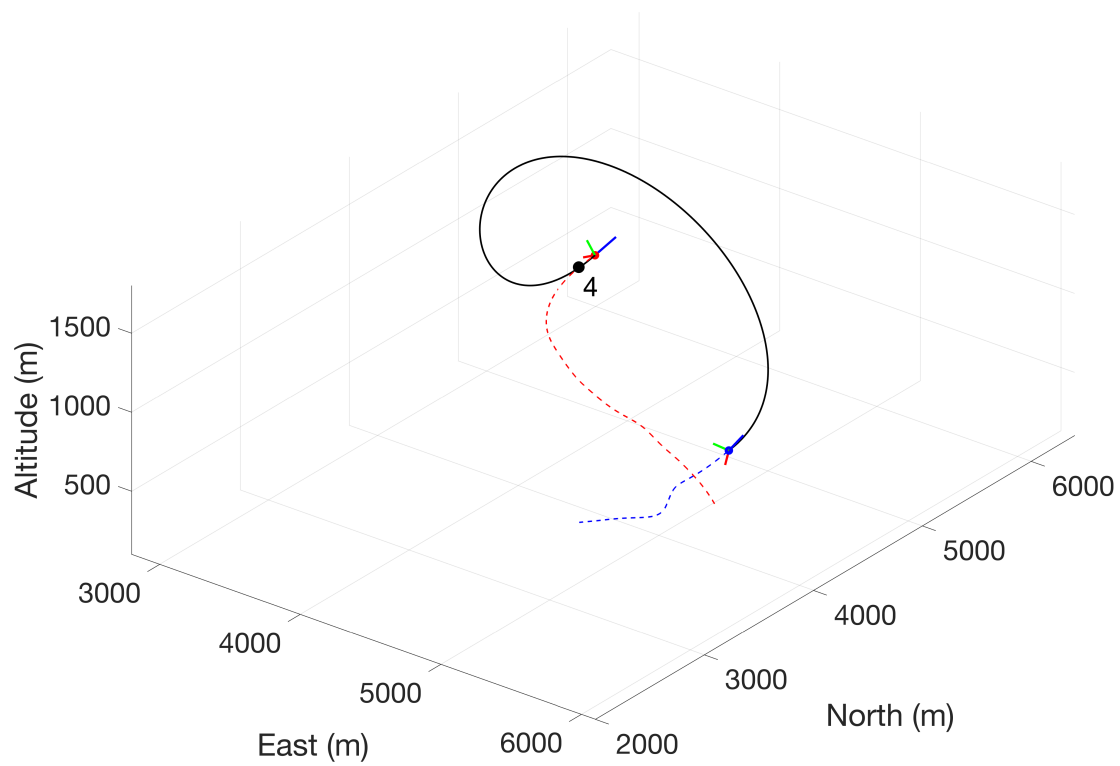
This visual representation gives an overview of the inertial frame, illustrating both the position and orientation of the lead aircraft and wingman, as well as historic position and live path information. These representations allow the reader to follow the progress of both aircraft throughout the duration of each mission. At each displayed instance the current path developed by the wingman is also displayed.

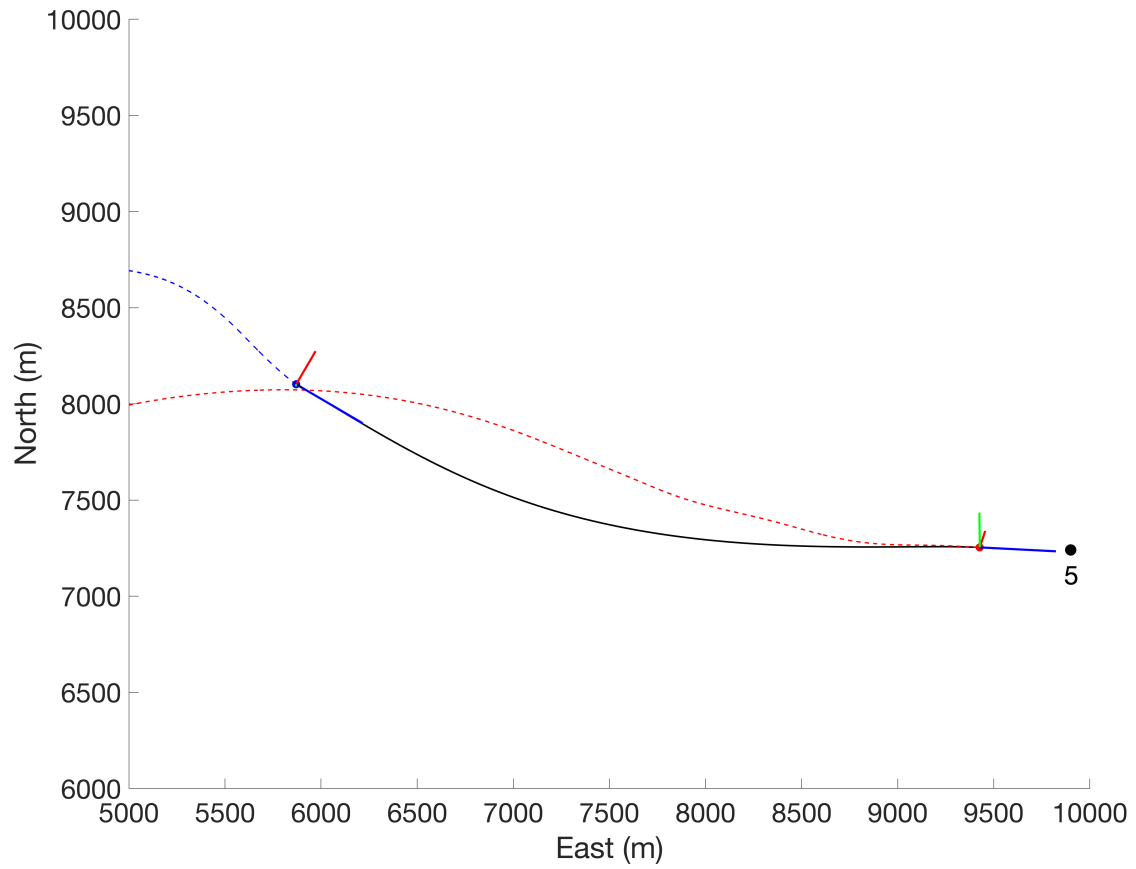
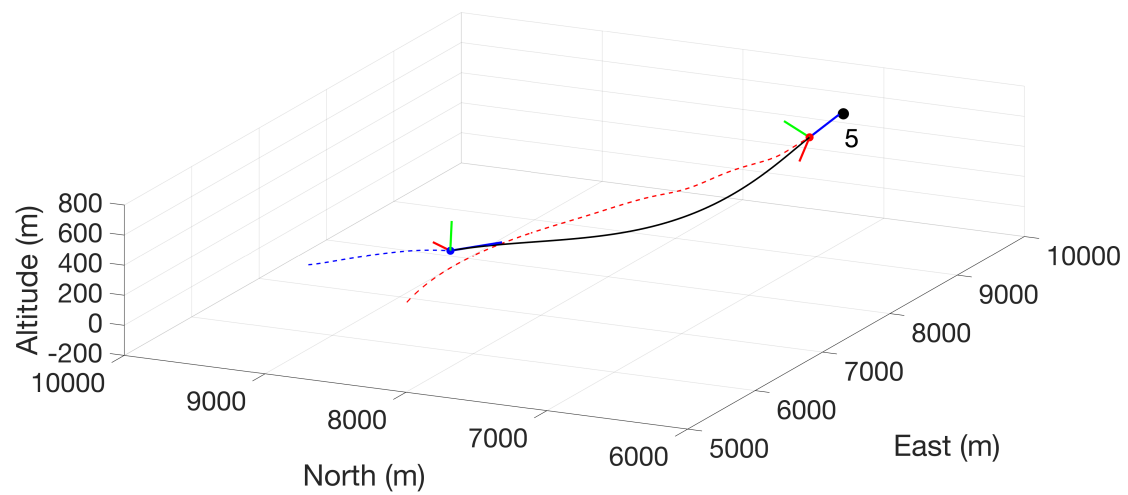
All three missions, including an introduction of the entire mission with start locations and orientations for both aircraft, time sampled inertial representations and an overview of the final mission status are displayed in the following section, with an overall discussion about each mission conducted in Section 7.6.3. The time sampled inertial representations depict the location of each aircraft as they either approach or pass through one of the defined mission waypoints.

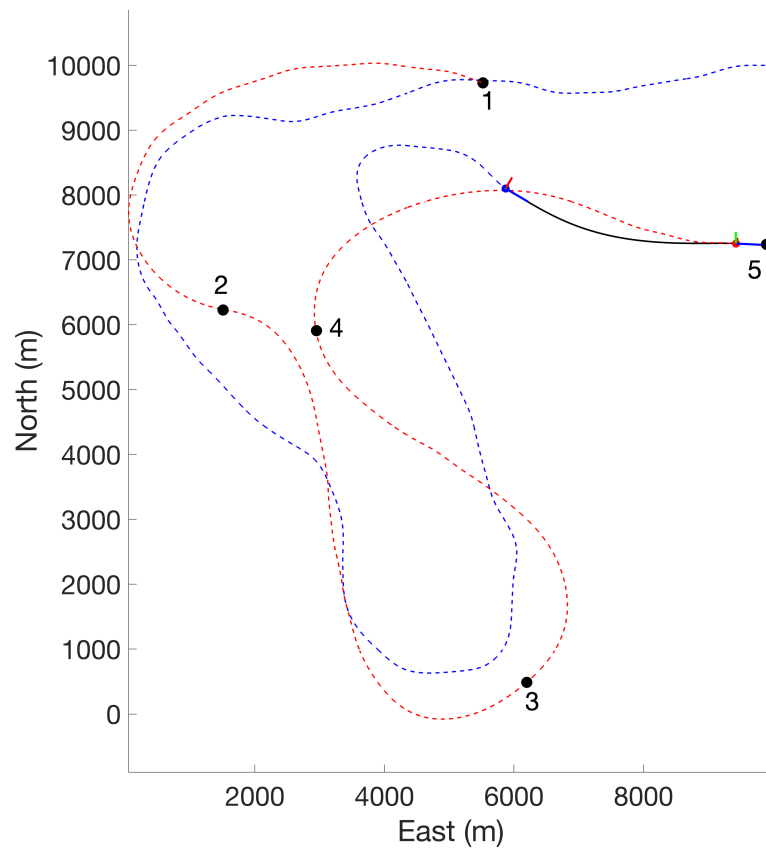
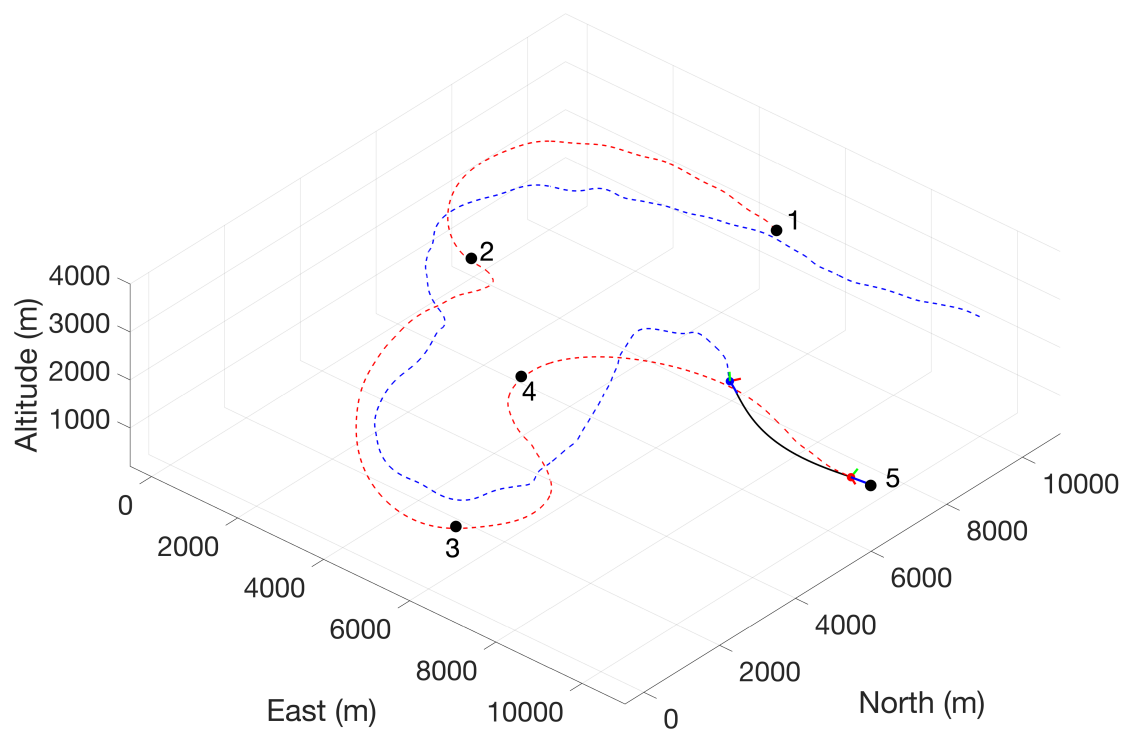
Mission 1(a) x - y representation(b) x - y - z representationFIGURE 7.30 *Mission 1* flight plan

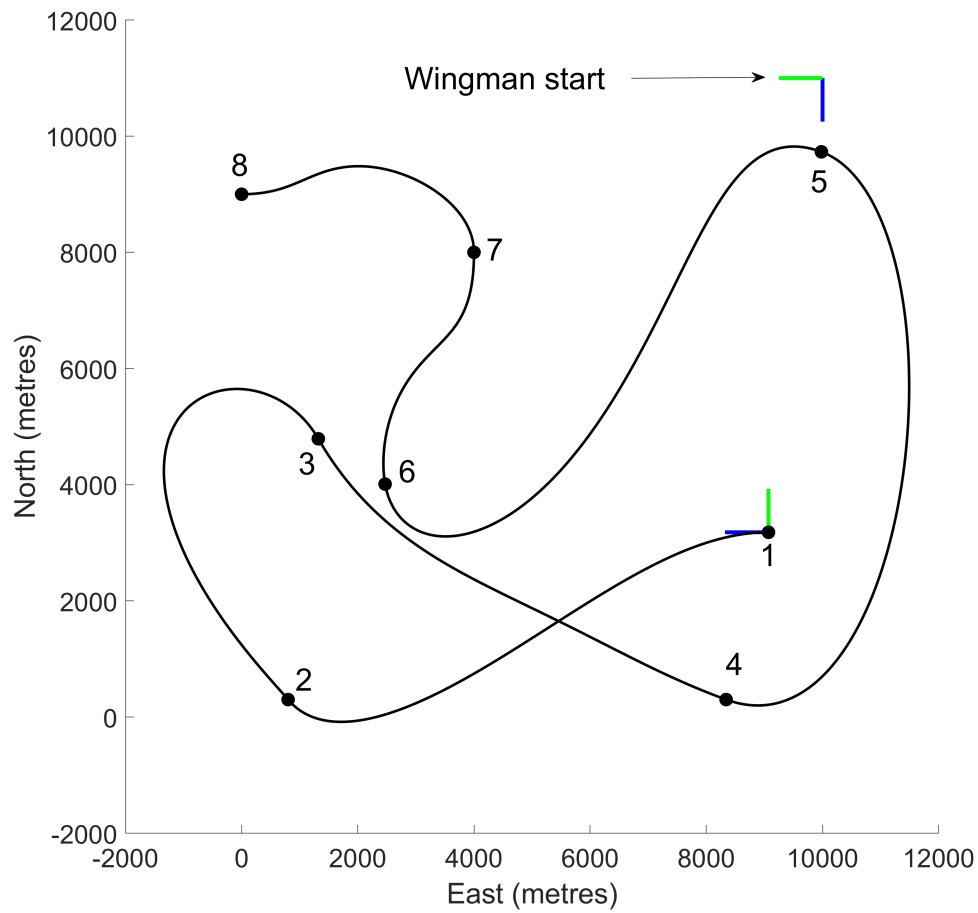
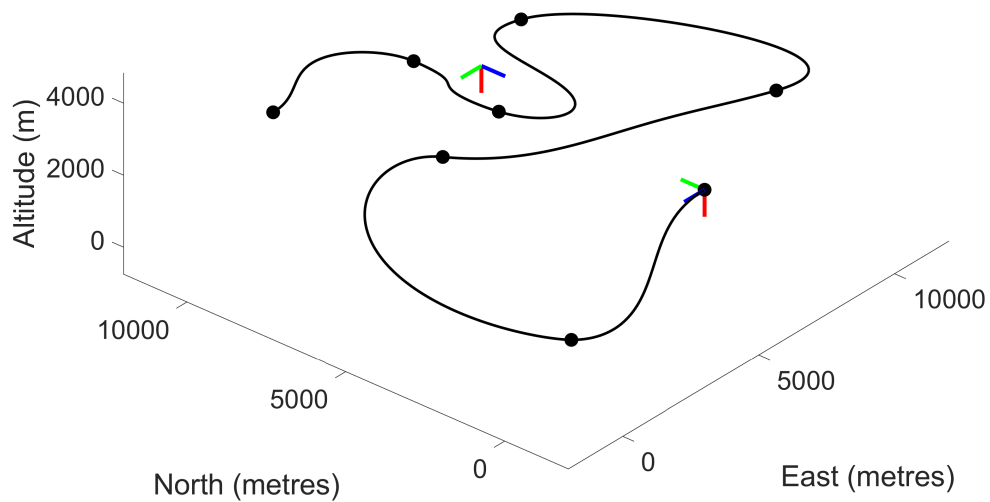
(a) x - y representation(b) x - y - z representationFIGURE 7.31 *Mission 1* inertial representation at Waypoint 2

(a) x - y representation(b) x - y - z representationFIGURE 7.32 *Mission 1* inertial representation at Waypoint 3

(a) x - y representation(b) x - y - z representationFIGURE 7.33 *Mission 1* inertial representation at Waypoint 4

(a) x - y representation(b) x - y - z representationFIGURE 7.34 *Mission 1* inertial representation at Waypoint 5

(a) x - y representation(b) x - y - z representationFIGURE 7.35 *Mission 1* final inertial flight paths

Mission 2(a) x - y view(b) x - y - z viewFIGURE 7.36 Variable-altitude model, dynamic waypoint target analysis *Mission 2*

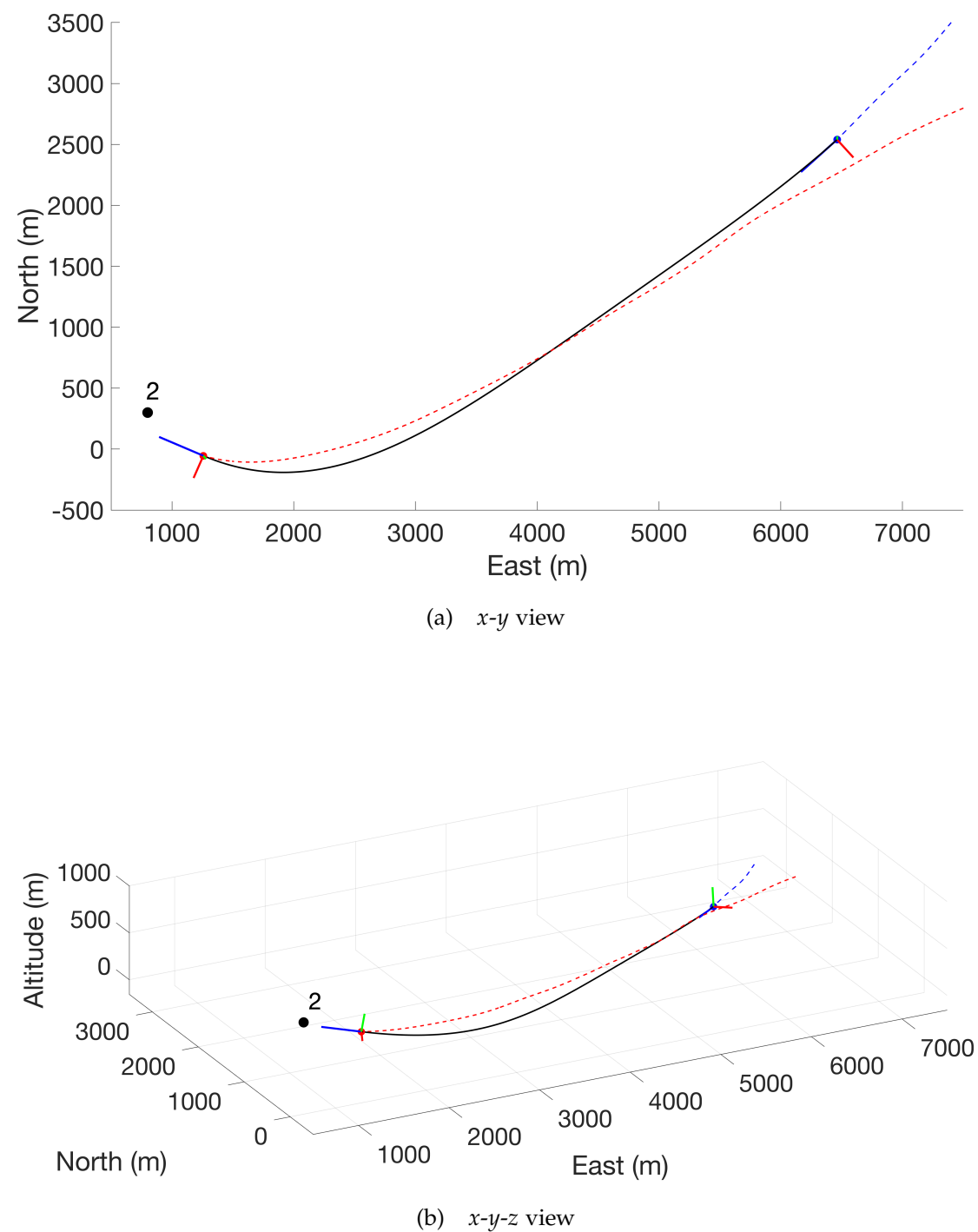
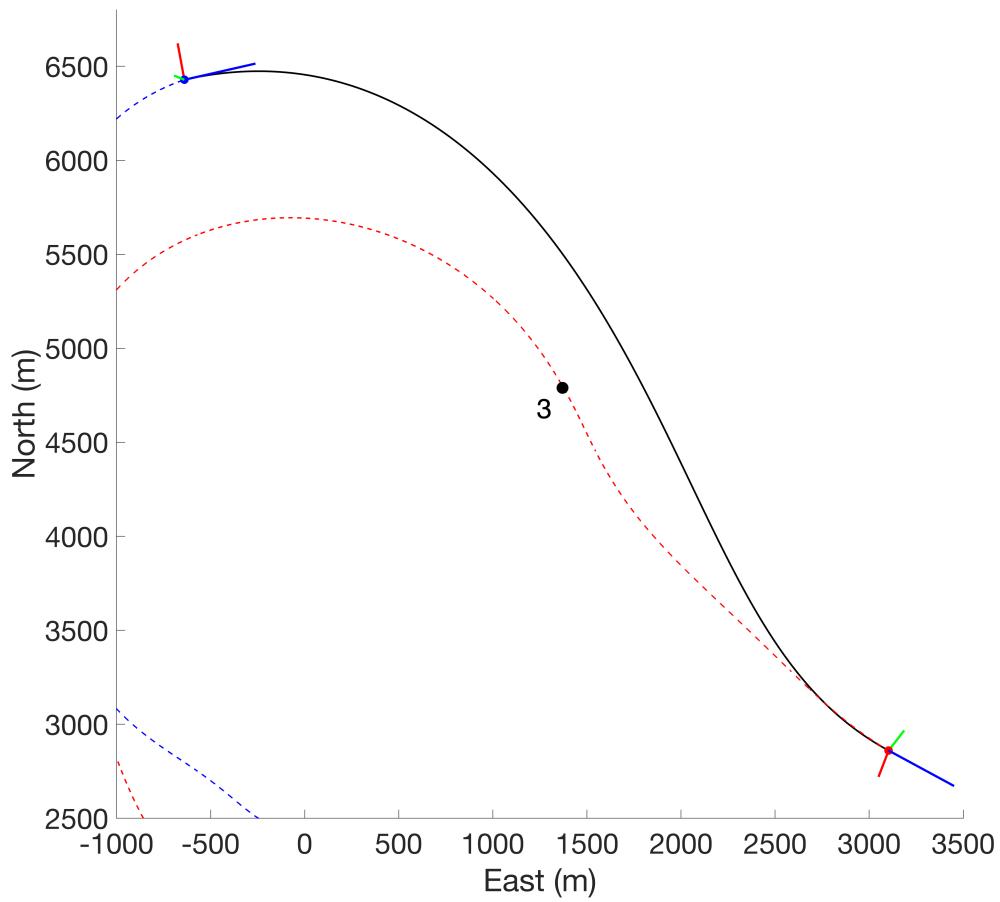
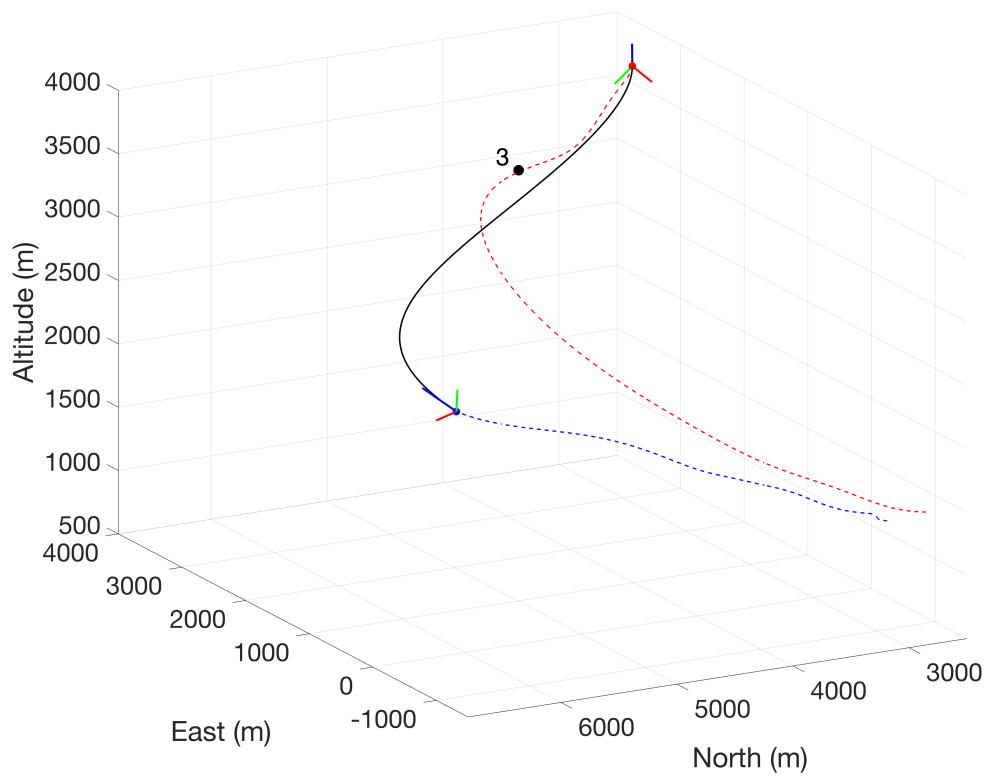
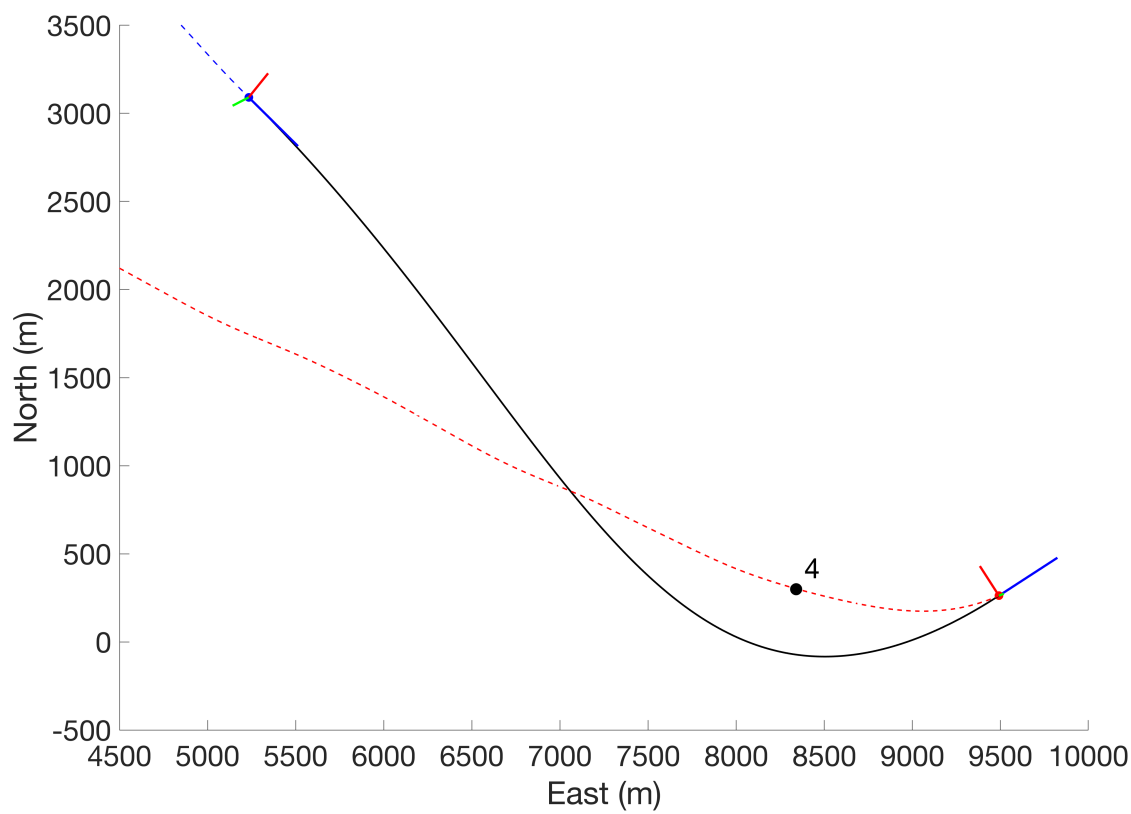
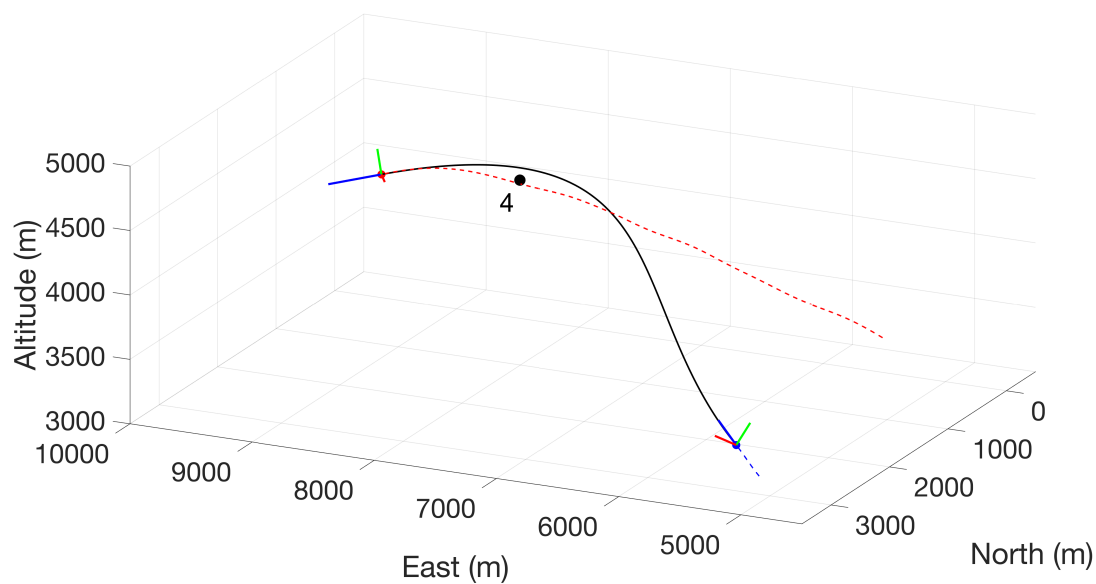


FIGURE 7.37 *Mission 2* inertial representation at Waypoint 2

(a) x - y view(b) x - y - z viewFIGURE 7.38 *Mission 2* inertial representation at Waypoint 3



(a) x - y view



(b) x - y - z view

FIGURE 7.39 *Mission 2* inertial representation at Waypoint 4

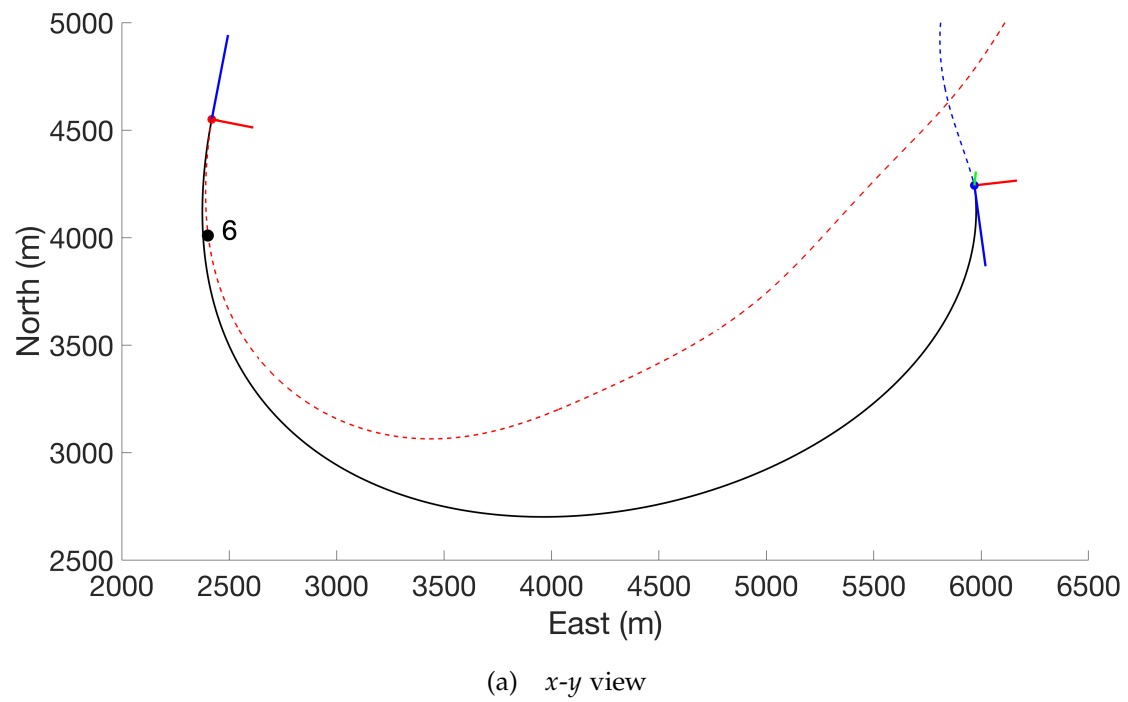
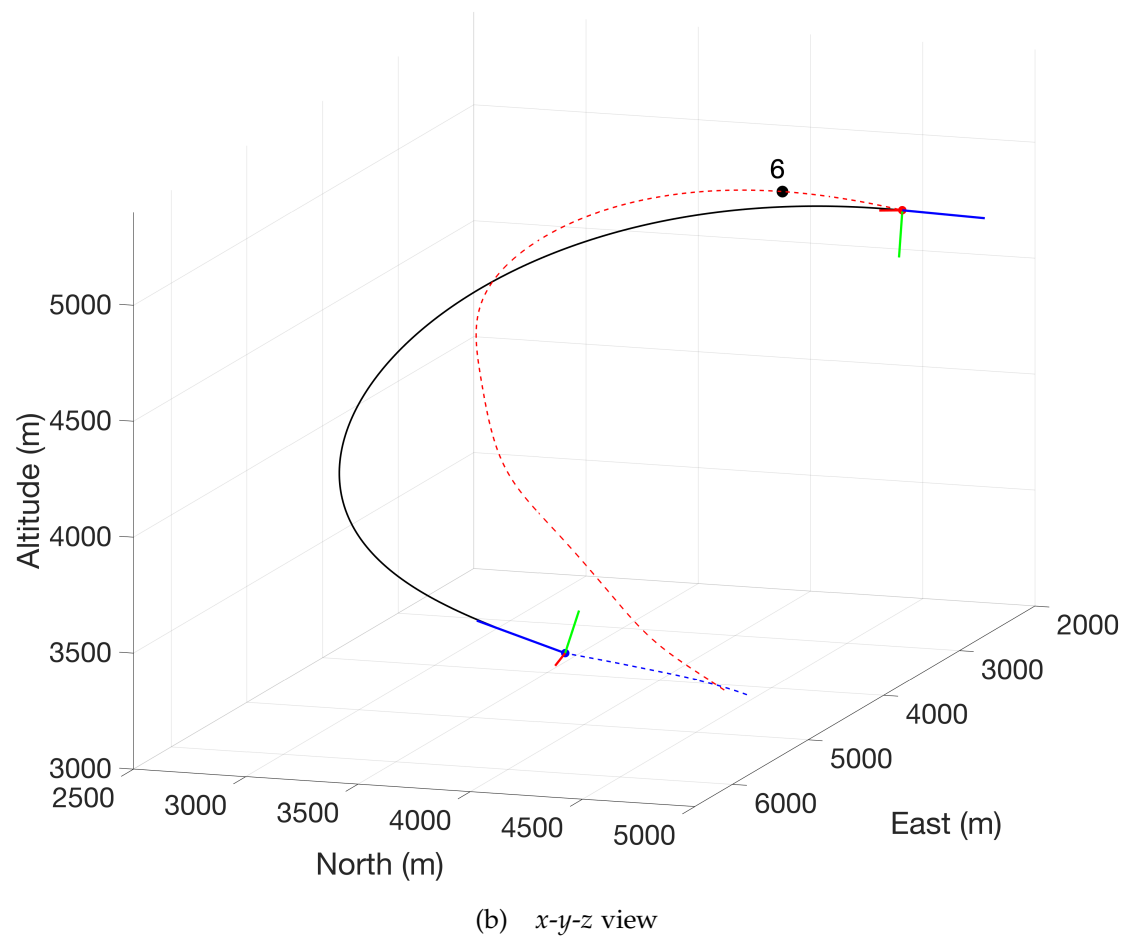
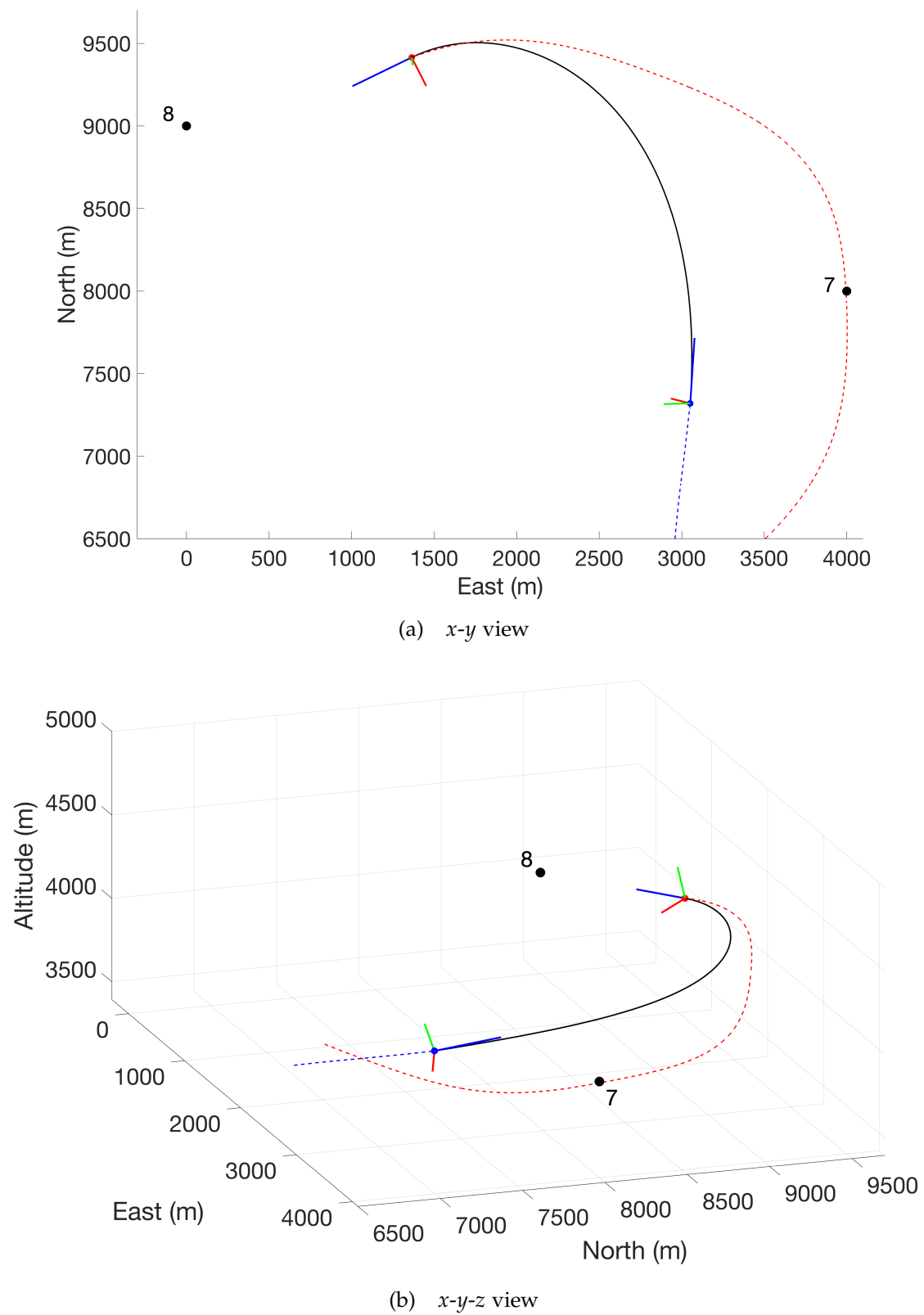
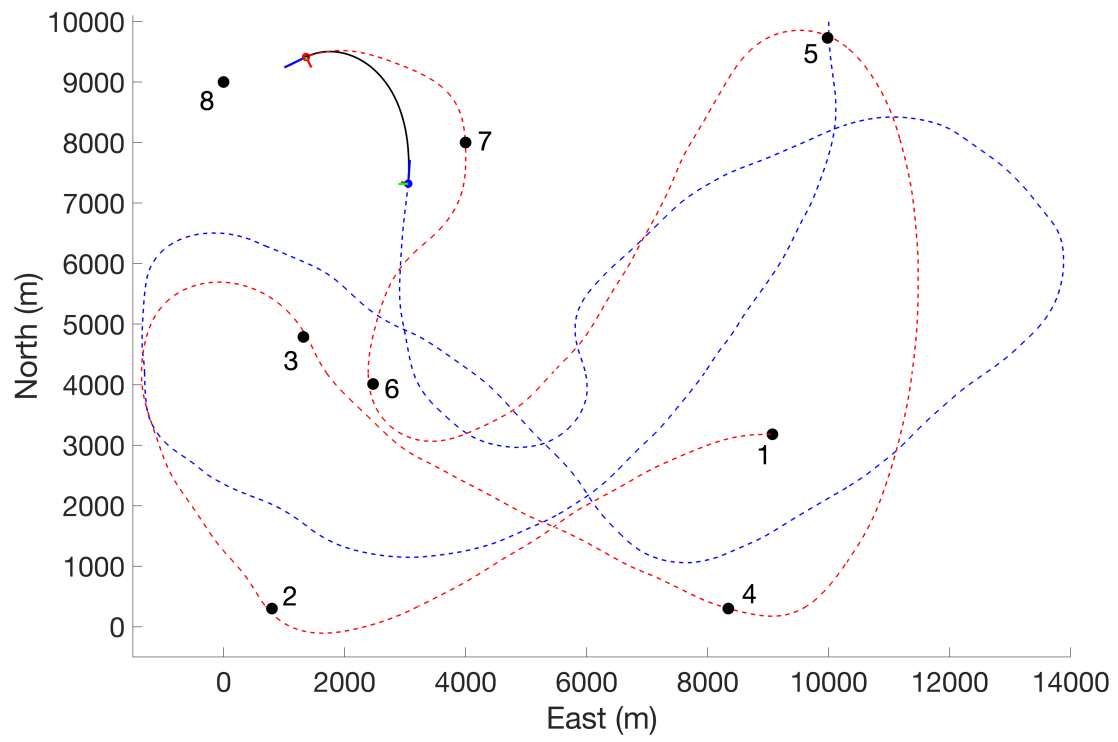
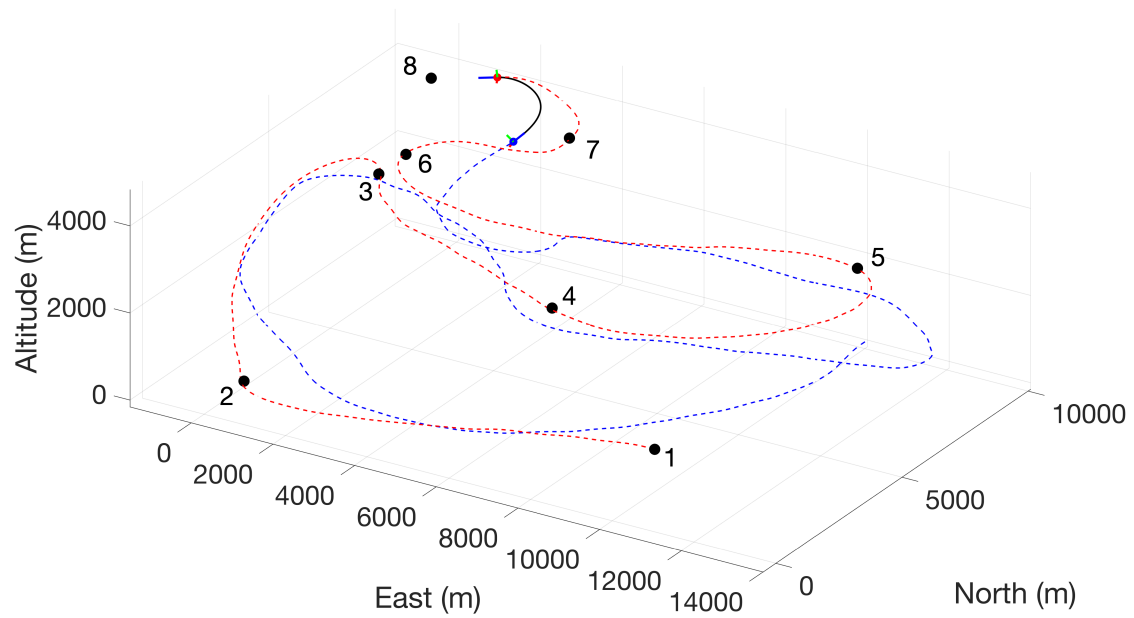
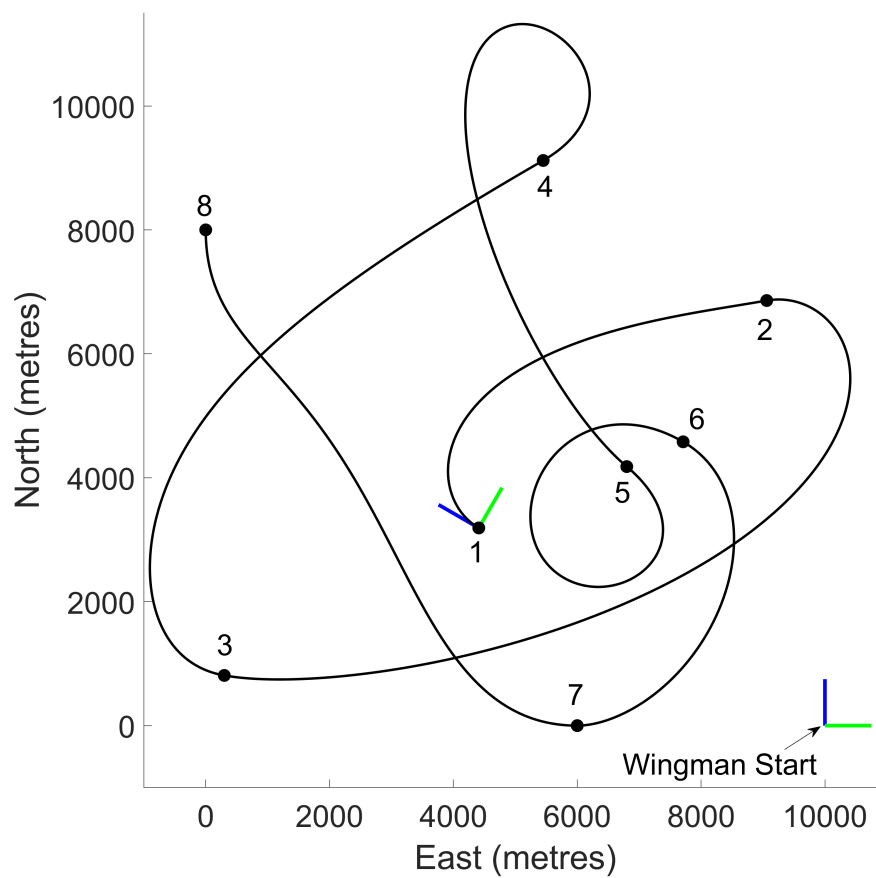
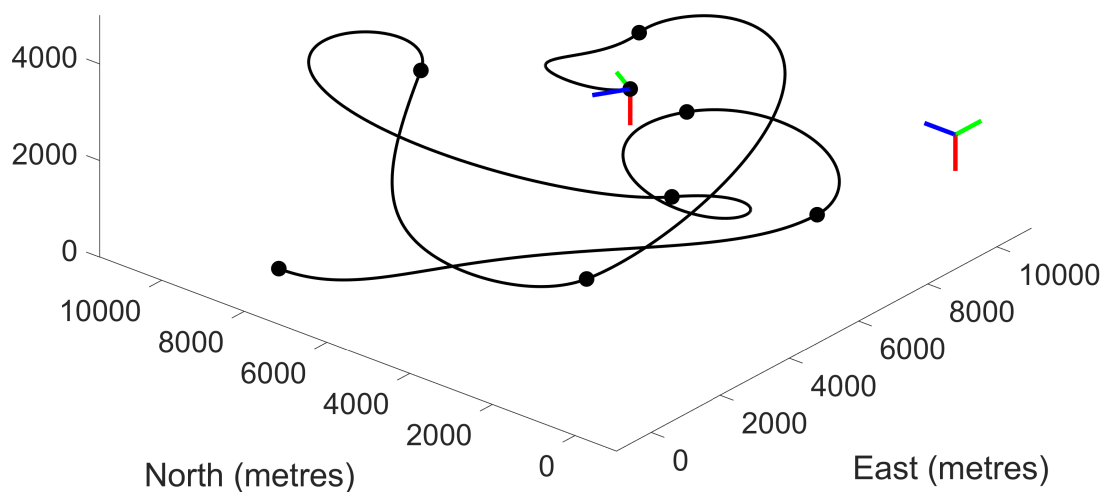
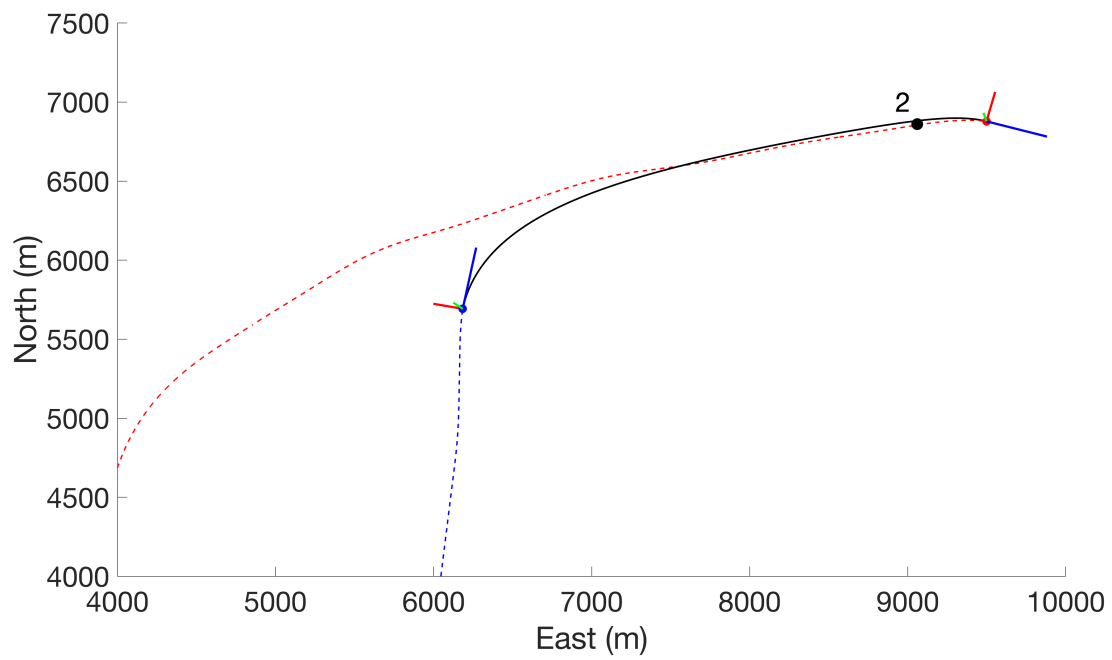
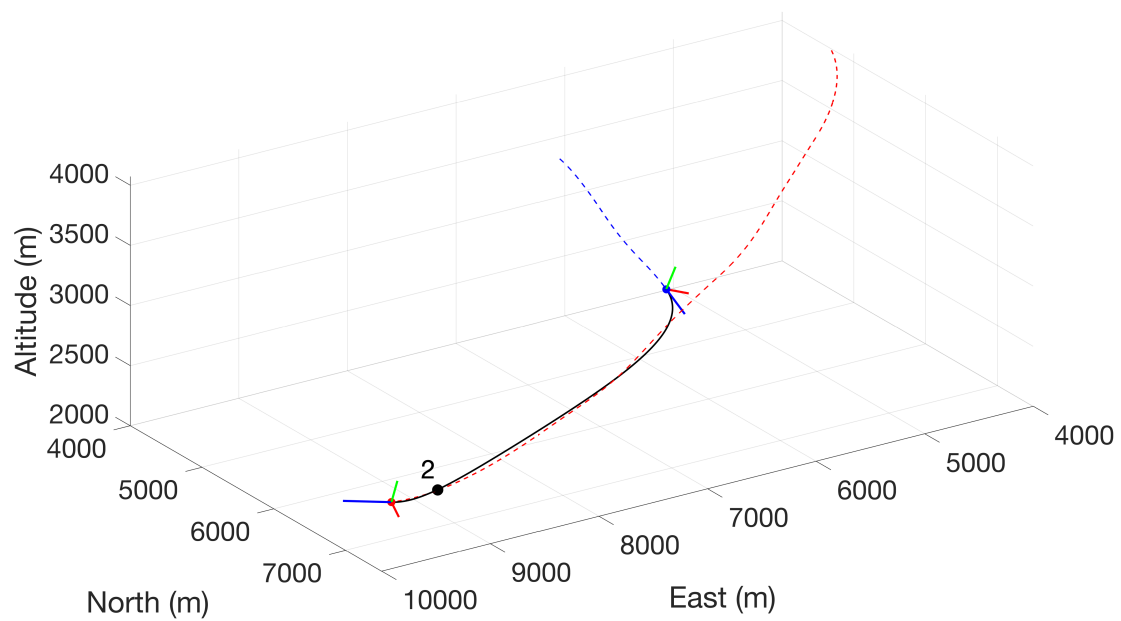


FIGURE 7.40 Mission 2 inertial representation at Waypoint 6

FIGURE 7.41 *Mission 2* inertial representation at Waypoints 7 and 8

(a) x - y view(b) x - y - z viewFIGURE 7.42 *Mission 2* inertial

Mission 3(a) x - y view(b) x - y - z viewFIGURE 7.43 Variable-altitude model, dynamic waypoint target analysis *Mission 3*

(a) x - y view(b) x - y - z viewFIGURE 7.44 *Mission 3* inertial representation at Waypoint 2

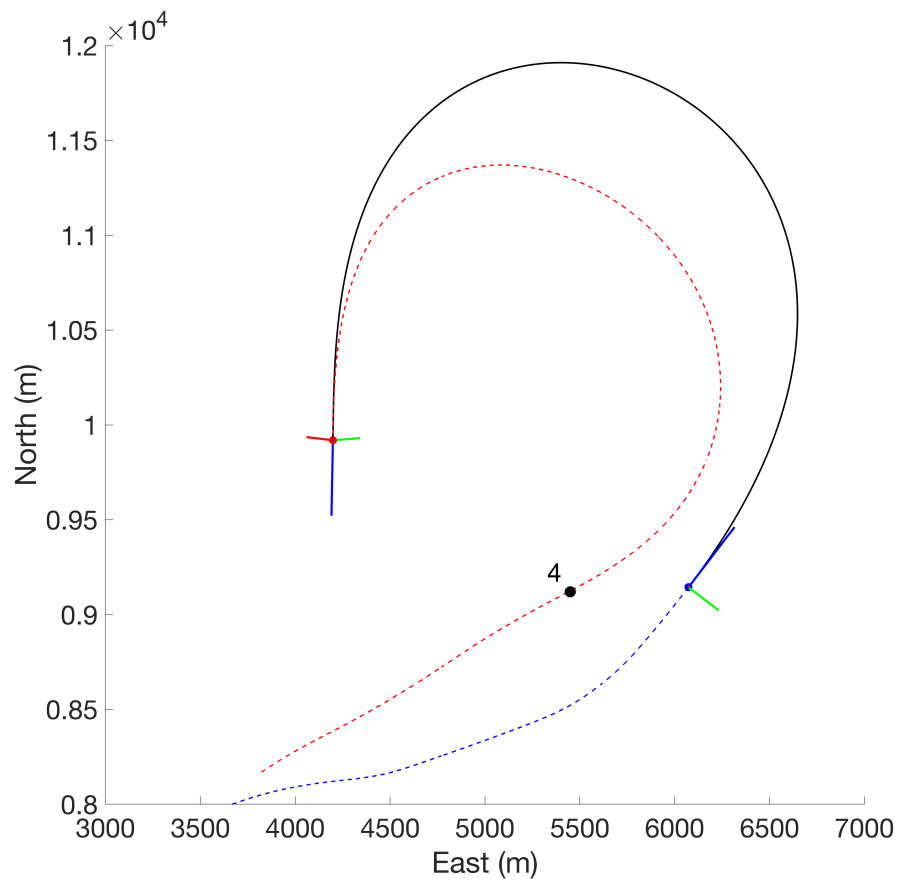
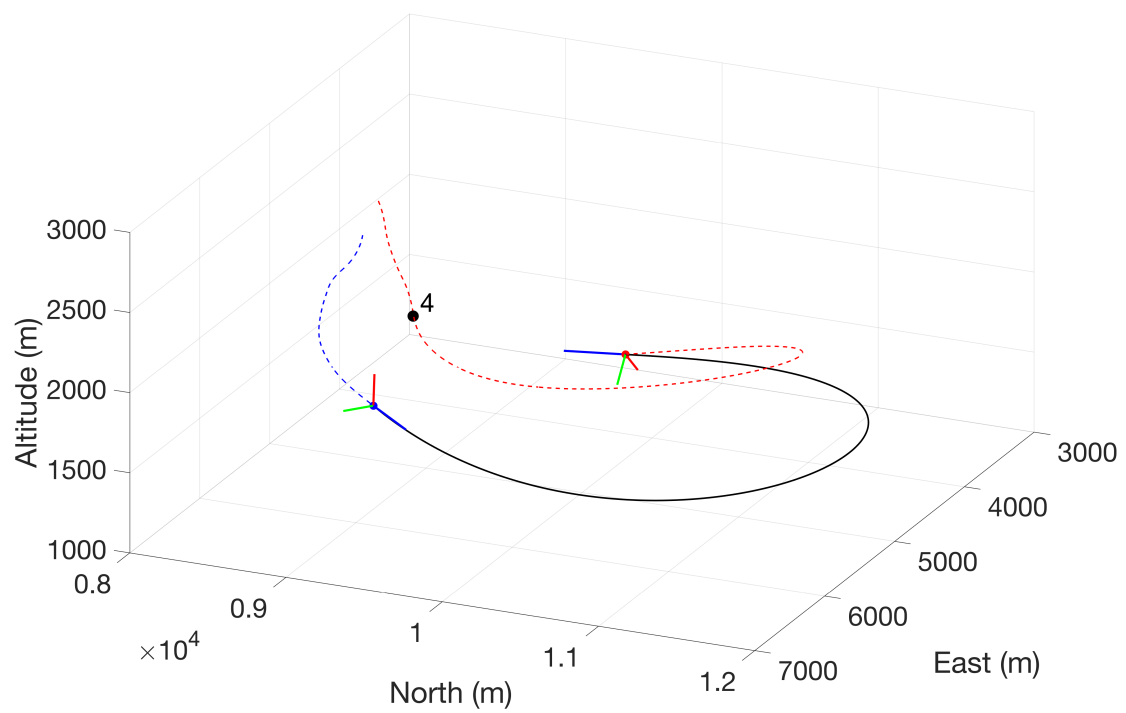
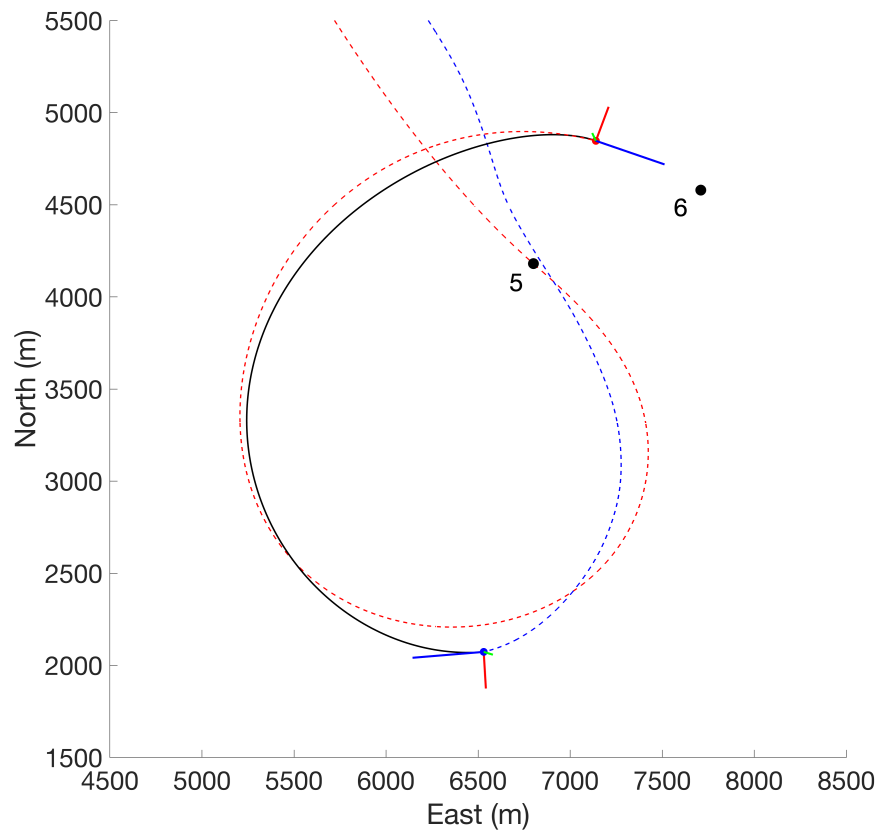
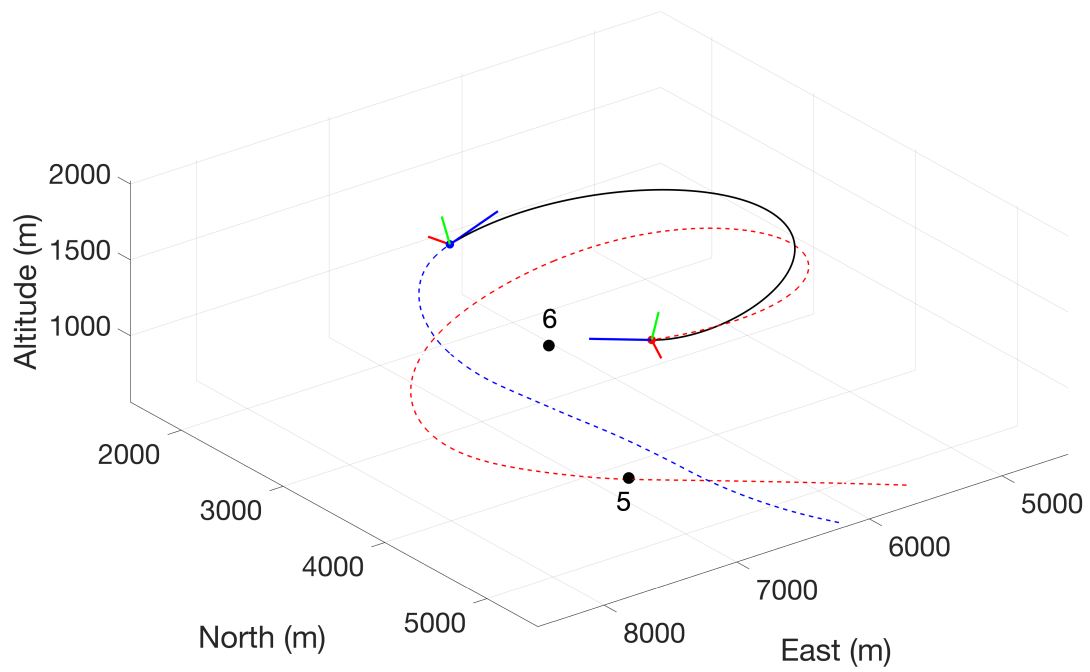
(a) x - y view(b) x - y - z view

FIGURE 7.45 Mission 3 inertial representation at Waypoint 4

(a) x - y view(b) x - y - z viewFIGURE 7.46 *Mission 3* inertial representation at Waypoints 5 and 6

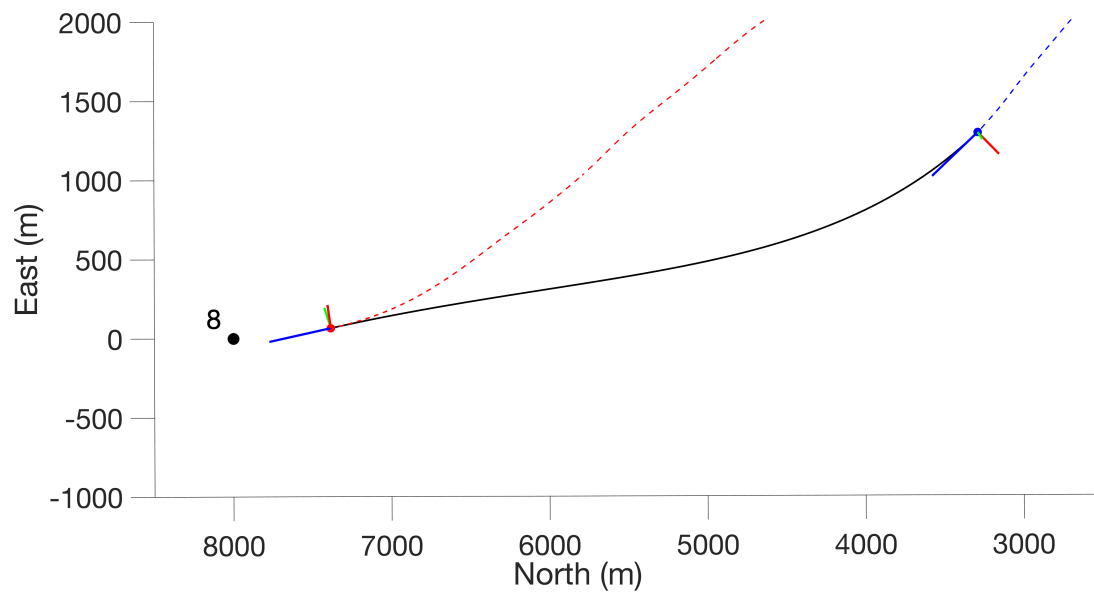
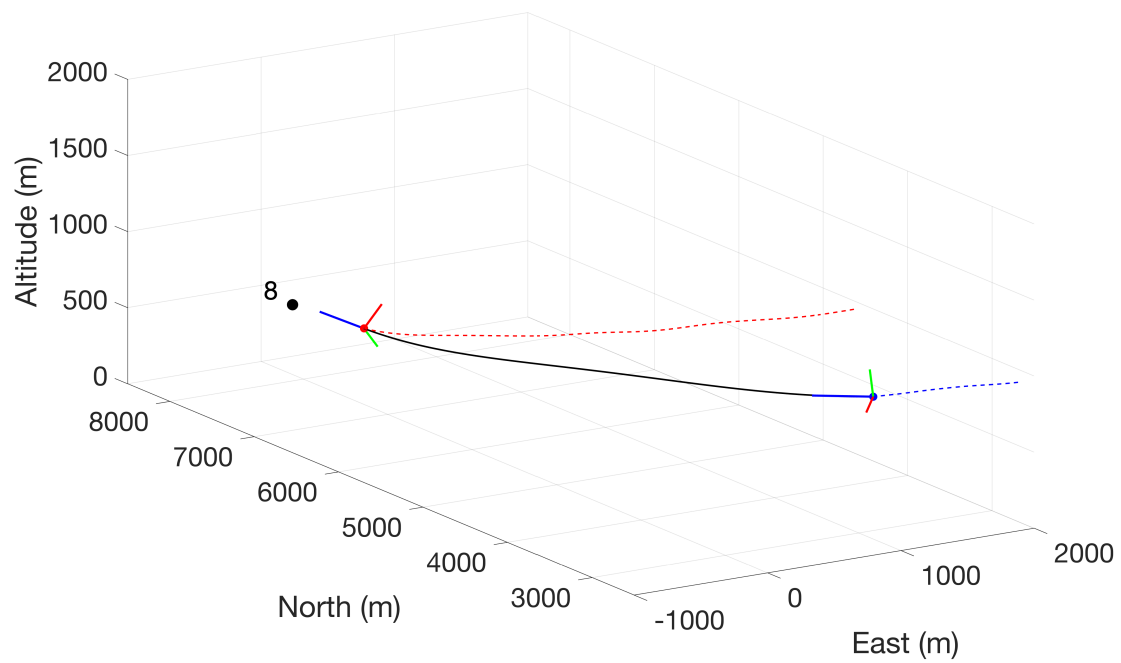
(a) x - y view(b) x - y - z view

FIGURE 7.47 Mission 3 inertial representation at Waypoints 8

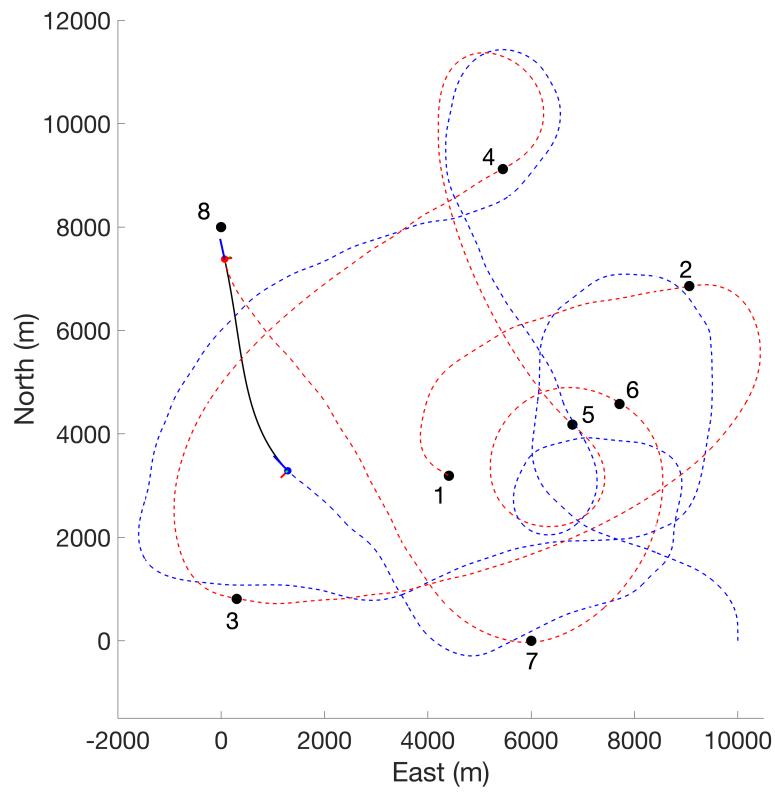
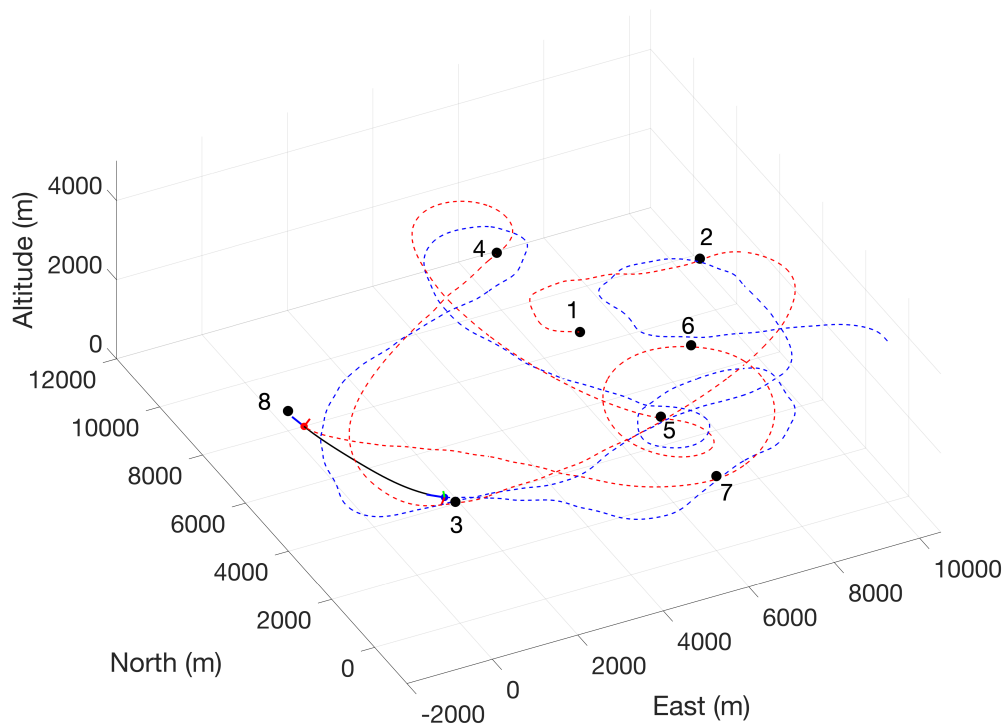
(a) x - y view(b) x - y - z view

FIGURE 7.48 Mission 3 inertial flight paths

7.6.3 Discussion

Having implemented the final iteration of the DNN in Section 7.6, the performance and robustness of the solution was demonstrated for both static and dynamic waypoint scenarios.

During static waypoint testing, the DNN predicted dynamically feasible and optimal solutions for 95% and 94% of the tested scenarios when operating within the optimised region. Although the performance of the network degraded as the operation exceeded the optimised region, the path planning algorithm was able to retain quite a high level of accuracy, particularly for positive coordinate regions. In addition, by enabling path re-planning, the aircraft successfully navigated to the waypoint target with over 99% accuracy, regardless the operating region. This demonstrates that the path planning solution and autonomous flight control system are robust, as the aircraft does not go unstable when exposed to dynamically unachievable paths and when operating in unknown environments.

During dynamic waypoint testing, three cooperative flight missions were performed. These missions contained quite complex flight plans, utilising the entire optimised region with altitude ascends and descends. Although these mission profiles could be considered as overly complex manoeuvres for conventional and routine cooperative flight missions, they have demonstrated the capabilities of the path planning algorithm and its robustness during complex cooperative missions.

The first cooperative flight denoted as *Mission 1*, was relatively simple and consisted of five waypoints. The first two waypoints were located at similar altitudes, with a descending manoeuvre between waypoints two and three. Although this descent was relatively benign, it was initiated by performing a small climb, as depicted in Figure 7.30b. This small climb created a rapid pitch change for the lead aircraft. Although this rapid change did not cause the lead aircraft to suffer any dynamic disadvantage, the path planning orientation constraint caused the wingman to deviate away from the lead aircraft for a few seconds. This deviation is visible in Figure 7.35, where the path taken by the wingman does not pass near the second waypoint.

Although this is not a major concern as the aircraft remained stable and maintained a near constant airspeed, the orientation and position of the wingman was compromised for the third waypoint, resulting in a tighter turn when compared to the lead aircraft. To compensate for this offset, the path planning algorithm proposed an altitude climbing deviation ahead of the fourth waypoint, as depicted in Figure 7.33. Although this altitude changing manoeuvre was never conducted since the path was eventually re-planned, it demonstrates how a small offset induced earlier in the mission can compromise the position of the wingman for the remainder of the mission.

The flight plan for *Mission 2* contained numerous large radius turns, with each waypoint gradually increasing or decreasing the altitude. These large radius turns and gradual changes to the altitude level, demonstrated the wingman's ability to track the lead aircraft during slightly more aggressive manoeuvres when compared to *Mission 1*. Despite the more aggressive turns, the paths produced by the DNN ensured that the wingman remained within its dynamic limits and did not demand unnecessarily high turn rates. The inertial frame representations in Figures 7.37 - 7.41 indicate that paths proposed for the wingman maintain a near constant curvature and are within the dynamic limits of the aircraft. The only noticeable deviation by the wingman occurred at approximately 180 seconds into the mission and is visible in Figure 7.42a, between waypoints four and five. This deviation most likely occurred because at this instance in time the wingman exceeded the optimised region, resulting in a higher probability that paths are sub-optimal. Although this results in the wingman flying further east than necessary, the network does eventually bring the wingman back on course and compensates for this increased distance by performing a slightly shallower turn at the fifth waypoint.

The final mission, denoted as *Mission 3* increased the aggressiveness of the flight path even further, by demanding tighter turns and larger climb periods, with a brief helical climb between waypoints five and six. Despite the flight plan containing tighter turns and more aggressive altitude changes, the path taken by the wingman remained consistent with the lead aircraft, with no dynamic deviations required. The wingman was able to engage in tight radial turns around waypoint four (Figure 7.45), five and six (Figure 7.46), both of which also

incorporated altitude climbs. During the helical climb, it is visually apparent that the path produced by the DNN (Figure 7.46) maintains a near constant radius of curvature and a near constant climb, allowing the dynamics of the wingman to remain relaxed, despite the challenging scenario.

The flight path taken by the wingman during *Mission 3* demonstrates the robustness and high performance of the solution. The final flight plan conducted by the wingman closely resembles that of the lead aircraft, with the wingman having to perform no dynamic deviations despite dynamically challenging turns and climbs with high heading change differences between the two aircraft.

Despite each of the three mission containing complex flight paths, the wingman was successfully able to track the lead aircraft for the duration of each mission, with no concerning deviation from the flight path occurring. This has demonstrated the robustness of the path planning solution and autonomous flight control system during cooperative flight missions, since the wingman was able to maintain a near-constant velocity throughout each of the missions despite having no prior knowledge of the mission flight plans. The fact that throughout all of the missions the wingman was able to maintain a near constant airspeed, indicates that the trained neural network is producing optimal paths within the dynamic confines of the aircraft, whilst the airspeed controller also demonstrates a high level of performance, allowing the aircraft to ascend and descend with little impact to the translational dynamics.

Chapter 8

Conclusions

This thesis started by identifying a need and desire for highly autonomous aerial vehicles, with current path planning methodologies identified as a major limitation for fixed-wing autonomous flight, particularly during real-time and cooperative flight. Therefore, the aim of the thesis was to:

Develop an online path planning algorithm enabling continuously flyable paths to be developed for a fixed-wing autonomous fighter jet during cooperative flight.

Chapter 8 summarises each preceding chapter and concludes its findings, highlighting the contributions of the chapter to the overall thesis. Chapter 8 closes by concluding the entire thesis and discussing the limitations of the research solutions.

8.1 Conclusions from Chapter 2

In Chapter 2, literature was reviewed corresponding to path finding and path generation algorithms for autonomous systems. The review identified that solely using a path finding algorithm does not guarantee paths will be continuously flyable and manoeuvrable for fixed-wing flight. Despite this limitation, path finding algorithms remain a well defined area of research and allow collision-free paths to be established through complex and cluttered environments. In addition, methods using artificial intelligence for path finding were investigated, enabling paths to be solved in more complex environments, as well as

coordinating the movement of multiple systems working synergistically.

Due to these factors, the focus of the research shifted from path finding algorithms towards path generation and path smoothing algorithms. Path smoothing and path generation algorithms are a requirement for many dynamically restricted autonomous systems, such as fixed-wing aircraft. These algorithms utilise information obtained from a path finding algorithm and allow path optimisation to be performed, creating a path that can be manoeuvred along within the dynamical limits of the vehicle.

Although path optimisation is also a well defined area of research, computational limitations often do not allow complex problems to be solved online, severely impacting the practicality of an autonomous system. In addition, the literature reviewed failed to demonstrate a robust optimisation technique that allowed flight to be performed throughout the entire operating flight envelope, especially during cooperative flight.

This chapter concluded by identifying Bézier curves as a suitable path smoothing algorithm due to computational efficiency to obtain a solution, as well as the ability to manipulate the curvature of the path by optimally placing the internal control points. This enables path optimisation to be performed, ensuring that paths remain dynamically feasible, continuously flyable and manoeuvrable. To allow path re-planning to be performed online, deep neural networks were identified as appropriate solutions for representing a multi-dimensional dataset as a non-linear continuous function.

8.2 Conclusions from Chapter 3

In Chapter 3, the flight physics model was introduced. This flight physics model was developed from first principles, with six non-linear equations of motion derived for modelling flight with rigid body dynamics. Non-linear subsonic aerodynamic data was obtained from Nguyen et al. [16] as well as first-order models for the engine and aerodynamic control surfaces. At the end of the chapter, a brief study was performed to identify the operational flight envelope for the flight physics model

8.3 Conclusions from Chapter 4

In Chapter 4, control was applied to the flight physics model, governing all rotational and translational motion. After identifying PID control as a suitable control algorithm, a study was performed into various meta-heuristic optimisation algorithms that allowed PID controller tuning.

The resultant controller gains were applied by expanding upon a novel gain scheduling technique described by Hamilton et al. [19], forming a normalised hypersurface of controller gains that are not only scheduled against the system states but also against the magnitude of the input reference signal and controller position error. This novel gain scheduling approach ensures adequate dynamic performance and closed-loop stability throughout the entire flight envelope, whilst also demonstrating improvements to the system response, with reductions to the transient response, steady-state error and controller effort.

8.4 Conclusions from Chapter 5

In Chapter 5, the path planning algorithm was formulated. This was performed by first identifying the order of the Bernstein polynomial that would not only satisfy the orientation constraints, but also allow for enough fidelity of control point movement that the dynamics can also be satisfied. It was determined that a minimum of a fourth-order polynomial was required to satisfy both the orientation and dynamic constraints of the path.

Although increasing the order of the Bernstein polynomial further does enhance path manipulation and allow the dynamics of the path to be further relaxed, this unnecessarily increases the computational requirement to optimise a solution, since additional control points require optimisation. Chapter 5 concludes by introducing a pure pursuit guidance law, enabling the aircraft to autonomously fly any path deemed dynamically achievable.

8.5 Conclusions from Chapter 6

In Chapter 6 path optimisation was performed using an evolutionary algorithm for constant-altitude path planning. A bespoke fitness function was developed that utilised the curvature profile of the paths to ensure dynamic feasibility. This evolutionary optimiser was used to train a classification deep neural network, allowing online path re-planning to be performed during cooperative missions. An observational size analysis was performed to locate suitable classification intervals for both the north and east observations as well as the heading orientation for the waypoint target. An intermediate reference frame was implemented to allow the waypoint observations to be described as relative coordinates to the location of the aircraft as opposed to absolute coordinates with respect to the inertial frame. This not only reduced the number of classifications, but allowed operation of the aircraft to be performed throughout the entirety of the inertial frame without additional optimisation or network classifications.

The performance of the classification deep neural network was demonstrated on the dynamic model for both static and dynamic waypoint targets. During the static waypoint analysis, the aircraft model successfully navigated to the waypoint target for 100% of the tested simulations, indicating a high level of accuracy when producing dynamically feasible paths. Despite this high level of accuracy, it was found that although dynamically feasible solutions were obtained, sub-optimal solutions occurred more frequently when the waypoint was located with a negative north observation.

Additionally, this was amplified by the fact that optimisation was only performed for positive east observations, with negative east observations translated into the positive axis for the purpose of classification. This meant that if a solution was sub-optimal as a positive east observation it would translate into the negative east observation as a sub-optimal solution, doubling the probability of this occurring. Despite these factors, throughout the static waypoint analysis it was found that 99.52% of solutions were deemed optimal, with 100% of solutions dynamically achievable, exemplifying the robustness of the optimisation process and the ability to train a classification deep neural network for constant-altitude path planning.

Dynamic waypoint testing was conducted for the constant-altitude path planner by performing three individual cooperative flight missions. Each mission consisted of a flight path flown by the lead aircraft, with the autonomous aircraft acting as a wingman, tracking and following the lead aircraft for the duration of the mission. During each mission, the wingman was instructed to maintain a set separation distance from the lead aircraft using a distance controller. The distance controller adjusts the airspeed of the wingman, allowing the demanded separation distance to be achieved. The implementation of the distance controller enabled the performance of the autonomous flight control system and path planning framework to be measured based on separation performance as opposed to the success rate of navigating to the waypoint target.

Additionally, use of the distance controller aided in fine-tuning the path re-planning parameters. For example, during *Mission 1* a set of arbitrarily relaxed path planning parameters were introduced. These parameters saw both partial and full path re-planning performed every three and 10 seconds, which was deemed satisfactory for the purpose of a simple constant-radius manoeuvre. However, during *Mission 2* the complexity of the mission required path re-planning to be performed at a higher frequency, meaning the parameters from *Mission 1* were no longer optimal. This meant the wingman struggled to maintain the demanded separation distance during a series of repetitive and tight radial turns, causing paths to be planned with a dynamic deviation present. Although this dynamic deviation is necessary to ensure robust and stable flight when navigating over short distances, it results in less than ideal wingman tracking capabilities. By increasing the rate at which paths were re-planned, the wingman was able to adequately track and follow the lead aircraft at the demanded separation distance for the duration of *Mission 2*. This mission demonstrated the performance capabilities of the autonomous flight control system and path planning framework, as well as its robustness during unfeasible and challenging scenarios.

Finally, during *Mission 3*, a dog-fight scenario between the wingman and lead aircraft was performed, with the lead aircraft attempting to evade the wingman using a flat-scissor manoeuvre. Despite the evasive techniques of the lead aircraft, the wingman was successfully able to track and follow the lead air-

craft through a series of fast and aggressive banked turns conducted at high airspeeds. Although this mission demonstrated the quick re-planning capabilities and high level of performance extracted from the flight control system and guidance laws, limitations to classifications meant that the separation distance was relatively large for an evasive type manoeuvre, with the wingman following the lead aircraft at a distance of 3 km. Although offsetting the waypoint target along the lead aircraft's heading vector did allow the wingman to fly much more closely, this introduced a higher degree of uncertainty, particularly if the wingman begins to path ahead of the lead aircraft during medium turn manoeuvres.

When comparing the proposed solution to the state-of-the-art, the demonstrated path re-planning methodology provides significant improvement since many path optimisation problems are performed a-priori [51, 52, 54, 55, 56]. In addition, the robustness of the solution was demonstrated throughout the optimised operational area and for exhaustive position and waypoint pose configurations, as opposed to demonstrations being for a specific set of abstracted flight conditions.

Based on the limitations of the distance controller in terms of both positional and dynamic predictability of the lead aircraft's flight path, future work should focus on solidifying the implementation. Improvements to the distance controller would enable better catch-up procedures, whilst also demonstrating better certainty when operating in close proximity to an allied aircraft performing aggressive manoeuvres.

8.6 Conclusions from Chapter 7

In Chapter 7, path optimisation was performed using an evolutionary algorithm for variable-altitude path planning. A bespoke fitness function was developed, utilising the curvature profile of the paths to ensure dynamic feasibility, as well as performing a climb rate analysis to predict the specific excess power during altitude climbing manoeuvres. By predicting the specific excess power, it can be determined whether or not a climb, which is induced by the path, is achievable using only power available by the engine.

Due to the dimensionality of the variable-altitude path planner, an interme-

diated reference frame was not implemented due to discrepancies with climb performance at various stages of flight. This meant that all observations were described as absolute coordinates with respect to the inertial frame, with an almost exhaustive optimisation process required throughout the inertial frame in order to allow for classification. Since this approach was unfeasible, it was determined that as a proof of concept a reduced operating zone was identified and optimised using a Monté Carlo simulation. The resultant data set was then used to train a regression deep neural network as opposed to a classification network, where the structure of the network was determined through an in-depth analysis, allowing the optimal number of hidden layers and the number of neurons within each layer to be identified. The resultant network consisted of 10 hidden layers with each layer consisting of 1000 fully connected neurons, amounting to a network with over 20 million trainable parameters.

Similar to the analysis in Chapter 6, the performance of the regression deep neural network was demonstrated on the aircraft model for both static and dynamic waypoint targets. During static waypoint testing within the optimised region and with online path re-planning disabled, the aircraft successfully navigated to the waypoint target for 95% of the tested scenarios, with 94% of solutions deemed optimal. For the purpose of this analysis, path re-planning was disabled as this gave a better overview of the network's ability to develop dynamically feasible paths.

Although dynamically feasible paths were only produced for 95% of simulations, the complexity of the problems accounts for this, particularly when attempting an altitude changing path at low speeds. From the 5% of simulations conducted where the solution resulted in a dynamically unachievable path, more than 40% of these occurred for airspeeds less than 120 m/s with 60% less than 150 m/s. This indicates that when operating at lower airspeeds, the robustness of the solution degrades, due to reduced aircraft dynamic capabilities.

Despite this solution degradation at lower airspeed, by allowing paths to be re-planned online should the aircraft deviate away from the flight path, the aircraft successfully navigated to the waypoint target for 99.90% of simulations. This demonstrates how, although dynamically feasible paths are not guaranteed with 100% certainty when using this methodology, the autonomous flight con-

trol system is able to maintain stable flight conditions with path re-planning performed based on the aircraft's new state dynamics.

In addition to static waypoint testing being conducted within the optimised region, an analysis was conducted to understand how operating in unknown environments impacted the model's prediction accuracy. For this analysis a total of eight regions were tested. Although all eight regions occupied the same volumetric space as the optimised regions, their respective north and east observations were either partially or completely unknown to the model. Although the performance of the network did degrade when operating in all eight regions, the aircraft was successfully able to navigate to the waypoint target for 99% of simulations whilst path re-planning was enabled. This demonstrated how robust the flight control system is in ensuring adequate dynamic performance and closed-loop stability even when exposed to unknown path planning environments.

Dynamic waypoint tested was conducted for the variable-altitude path planner by performing three individual cooperative flight missions. Each mission consisted of a complex flight path flown by the lead aircraft, with the autonomous aircraft acting as wingman, tracking and following the lead aircraft for the duration of the mission. During this analysis, it was found that use of the distance controller severely impacted the performance and stability of the wingman, since the distance controller adjusts the throttle setting of the aircraft based on the separation distance position error between the lead aircraft and wingman. During climbs, if the separation distance reduced, the distance controller would reduce the throttle, impeding the wingman's ability to climb and becoming susceptible to stall. For these reasons, the distance controller was disengaged for the duration of dynamic waypoint tested for variable-altitude path planning, with the wingman's airspeed controller engaged to maintain a constant airspeed for the duration of the mission.

Despite each mission being designed to destabilise or impede the performance of the wingman by performing quick and rapid altitude changing manoeuvres, the wingman never tended to instability and successfully tracked and followed the lead aircraft for the duration of each mission. The path planning framework's ability to quickly re-calculate and re-plan paths online with a relatively high frequency enabled adequate tracking of the lead aircraft regardless

of its manoeuvre. In addition, the flight control system and guidance law ensured closed-loop stability and adequate dynamic performance regardless of the mission demands. Finally, the wingman was able to successfully complete each mission with a near constant airspeed. This indicates that the specific excess power analysis and predictions made during the optimisation process, successfully captured the climb dynamics and allowed the aircraft to increase altitude using excess power available from the engine as opposed to trading kinetic energy for potential energy.

When comparing the methodology to the current state-of-the-art, a method for performing altitude changing manoeuvres under constant kinetic energy is presented. This method is currently under explored in the literature and poses novelty to the proposed methodology. Although altitude changing paths are only demonstrated for altitude climbs up to 5 km, this limitations was partly due to creating a tractable solution for the optimiser. Despite this, the current state-of-the-art for altitude changing paths for autonomous aircraft are limited usually to multi-rotor designs which have the potential to increase altitude without requiring path planning in the conventional manner, or low-speed/low-altitude UAVs performing benign manoeuvres. However, by using the specific excess power to perform path climb predictions, the full performance envelope of the aircraft is utilised for flight throughout the subsonic flight regime, providing significant benefits to the validity of the solution.

Furthermore, use of a regression deep neural network demonstrated online re-planning during solo and cooperative flight. This use of artificial intelligence for online path predictions is not demonstrated in the current state-of-the-art, adding additional novelty over those whom perform path planning offline and a-priori.

8.7 Limitations of the Methodology

One of the main limitations of the technique is the reduced operating space for the variable-altitude regression network. When training a deep neural network using supervised learning, the training dataset and the quality of data is the ultimate factor in the network predictive output performance. Due to the curse

of dimensionality, the size of the domain and number of observations limited the solution space, as the optimisation process would have become unfeasible.

Although this is a limitation, it was demonstrated in Chapter 7 that the network was still able to predict the control point scaling factors with quite a high level of accuracy even when operating outside of the trained area. Although it can be expected that system stability and performance will reduce when operating outside of the trained area, the system was still able to function and fly adequately although not optimally in these regions. This limitation can be overcome using one or more of the following methods:

1. Perform optimisation and training on more powerful machine clusters
 - Although this does not solve the core issue, being able to expand the hardware capabilities available provides a significant benefit to the size of the optimised area.
 - For example, having two machines running simultaneously to obtain optimised solutions would double the size of the operating area whilst maintaining the same solution density.
 - Performing deep learning on multiple GPUs would expand memory allocation, allowing for larger training datasets and larger neural networks, without negatively impacting the training procedure.
2. Introduce an intermediate reference frame, similar to the constant-altitude solution, allowing observations to be defined at relative locations as opposed to absolute
 - This solution is less computationally expensive, but requires more complex integration of the network into the flight control system.
 - Additionally, this solution would need to work across three-dimensions and factor changes associated to climb performance degradation at increased altitude levels

8.8 Future Work

Based on the work presented in this thesis, a path planning framework for a large-scale, fixed-wing, highly manoeuvrable fast jet has been achieved, with online path re-planning using a deep neural network for cooperative flight missions. A number of areas of future work are discussed below, which would usefully expand the scope of the study.

8.8.1 Flight Physics Model

Future work to expand the scope of this research should incorporate an expansion of the flight physics model, specifically regarding the aerodynamics, autonomous flight control system and distance controller.

Aerodynamics Dataset

Currently, the flight physics model is limited by the aerodynamic dataset to subsonic flight. Ideally, expanding this dataset to allow accurate flight to be modelled in the supersonic region would benefit this research, as it would allow defensive and evasive manoeuvres to be better simulated by the flight control system.

Multi-Purpose and Adaptable Flight Control System

The autonomous flight control system for the purpose of this research was focused towards maintaining stability during routine cooperative flight missions. This often resulted in the dynamics of the aircraft being relaxed, particularly during path planning. However, since the aircraft is representative of a highly-maneuvrable fast jet, it would be advantageous to have the autonomous flight control system have different modes of operation, such as routine flight, offensive flight and defensive flight.

During offensive and defensive flight, the flight control system would adapt the angular rate autopilots to prioritise performance over stability, allowing the wingman to expand the flight dynamics. Additionally, each of these modes of operation should influence the path planning framework, allowing more ag-

gressive chase procedures to be performed during offensive modes of operation, and more evasive manoeuvres to be conducted during defensive flight.

Distance Controller

Despite the author's efforts, the distance controller was not able to adequately function for variable-altitude path planning due to limitations with the implementation. These limitations prevented the distance controller from being able to predict when additional power was required, relative to different stages of flight. As a result, the wingman became susceptible to stall during cooperative missions when the lead aircraft performed an abrupt altitude changing manoeuvre.

8.8.2 Path Planning Framework

The path planning framework for the scope of this research focused on routine flight operations, where paths generated for the wingman would guarantee they were a dynamically achievable, and continuously flyable manoeuvre. However, expansion of the operating modes in alignment with future plans for the autonomous flight control system, would benefit operation and allow the wingman to successfully perform more complex combat-style missions.

Operational Area

Due to time constraints with the project and the overheads associated with performing path optimisation, the variable-altitude model was limited to where paths could be planned. Although it was demonstrated during the static way-point analysis in Chapter 7, that paths could still be planned outside of the optimised area with quite a high level of accuracy retained, it would be advantageous to adjust the path planning framework to incorporate an intermediate reference frame similar to that implemented in Chapter 6. This would enable observations to be identified as relative coordinates, with predictions outputted from the neural network translated more appropriately to larger regions of the inertial frame.

Additionally, due to the climb performance analysis, paths were limited in

altitude between 0 km and 5 km. During future work, optimisation of higher operating regions should be performed, potentially incorporating interim waypoints or higher order polynomial optimisation, allowing the aircraft to perform more appropriate two or three stage climbs during high altitude changing manoeuvres.

Classification Observations

During development of the classification network for constant-altitude path planning in Chapter 6, the north and east observations were equally spaced throughout the intermediate reference frame. Although this did not hinder the performance of the network to generate classifications overall, it limited the wingman to a relatively large separation distances during cooperative flight of around 3-4 km. If the euclidean distance between the wingman and waypoint approached 2 km, the dynamics begin to become strained and the classification network favoured performing a dynamic deviation.

By introducing more densely populated classifications surrounding low euclidean observation regions, it is expected that this can be remedied, since classifications during these high dynamic regions would retain more appropriately optimised solutions. In addition, adjusting the fitness function to constrain direct pathing as opposed to dynamic deviation would also aid in alleviating this issue.

References

- [1] N. Sharkey. The automation and proliferation of military drones and the protection of civilians. *Law, Innovation and Technology*, 3(2):229–240, December 2011. doi: 10.5235/175799611798204914. URL <https://doi.org/10.5235/175799611798204914>.
- [2] The Bureau of Investigative Journalism. Strikes in Afghanistan, Sep 2020. URL <https://www.thebureauinvestigates.com/projects/drone-war>.
- [3] United States Army. “Eyes of the Army” - U.S. Army Roadmap for UAS 2010-2035, 2010.
- [4] United States Army. Unmanned aircraft systems roadmap, 2005. URL https://fas.org/irp/program/collect/uav_roadmap2005.pdf. [Accessed: 2020].
- [5] United States Air Force. Skyborg program seeks industry input for artificial intelligence, Sep 2019. URL <https://www.af.mil/News/Article-Display/Article/1796930/skyborg-program-seeks-industry-input-for-artificial-intelligence-initiative>. [Accessed: 2020].
- [6] R. Scott. UK Pursues low-cost unmanned combat aircraft, Aug 2019. URL <https://www.janes.com/defence-news/news-detail/uk-pursues-low-cost-unmanned-combat-aircraft>.
- [7] D. Donald. Project Mosquito: UK’s ‘Loyal Wingman’ program moves ahead, Jul 2020. URL <https://www.ainonline.com/aviation-news/defense/2020-07-17/project-mosquito-uks-loyal-wingman-program-moves-ahead>. [Accessed: 2020].

- [8] Royal Air Force. Team Tempest, 2020. URL <https://www.raf.mod.uk/what-we-do/team-tempest/>. [Accessed: 2020].
- [9] Defence World. UK air force reveals 3d model of 'Tempest' fighter jet, LANCA unmanned aircraft, 2020. URL https://www.defenseworld.net/news/27446/UK_Air_Force_Reveals_3D_Model_of__Tempest__Fighter_Jet__LANCA_Unmanned_Aircraft#.X3xsIS2ZNTY. [Accessed: 2020].
- [10] K. Mizokami. China's loyal wingman drone flies alongside manned fighters, Aug 2019. URL <https://www.popularmechanics.com/military/aviation/a28845055/china-loyal-wingman-drone/>.
- [11] M. A. Boon, A. P. Drijfhout, and S. Tesfamichael. Comparison of a fixed-wing and multi-rotor UAV for environmental mapping applications: A case study. *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, XLII-2/W6:47–54, August 2017. doi: 10.5194/isprs-archives-xlii-2-w6-47-2017. URL <https://doi.org/10.5194/isprs-archives-xlii-2-w6-47-2017>.
- [12] M. Protti and R. Barzan. UAV Autonomy - which level is desirable? - which level is acceptable? Alenia Aeronautica Viewpoint. Technical report, NATO, May 2007.
- [13] National Institute of Standards and Technology. Autonomy levels for unmanned systems (ALFUS) framework, Aug 2004.
- [14] United States Army - Department of Defense. Pentagon unmanned systems integrated roadmap 2017 - 2042, 2017. URL <https://news.usni.org/2018/08/30/pentagon-unmanned-systems-integrated-roadmap-2017-2042>. [Accessed: January 2021].
- [15] M. V. Cook. *Flight Dynamics Principles*. Elsevier, 1997. doi: 10.1016/c2009-0-23635-x. URL <https://doi.org/10.1016/c2009-0-23635-x>.
- [16] L. Nguyen, M Ogburn, W. P. Gilbert, K. S. Kibler, P. W. Brown, and P Deal. Simulator Study of Stall Characteristics Fighter Airplane Static Stall / Post-of a Relaxed With Longitudinal Stability. *NASA Technical Paper 1538*, (December 1979), 1979.

- [17] E. A. Morelli. Global nonlinear parametric modelling with application to f-16 aerodynamics. In *Proceedings of the 1998 American Control Conference. ACC (IEEE Cat. No.98CH36207)*. IEEE, 1998. doi: 10.1109/acc.1998.703559. URL <https://doi.org/10.1109/acc.1998.703559>.
- [18] B. L. Stevens, F. L. Lewis, and E. N. Johnson. *Aircraft Control and Simulation: Dynamics, Controls Design, and Autonomous Systems*. John Wiley & Sons, Inc, November 2015. doi: 10.1002/9781119174882. URL <https://doi.org/10.1002/9781119174882>.
- [19] J. Hamilton, D. Galvão Wall, A. J. Saddington, and J. T. Economou. Hypersurface normalised gain-scheduled controller for a non-linear 6-DOF fast jet. *Aerospace Science and Technology*, 106:106155, November 2020. doi: 10.1016/j.ast.2020.106155. URL <https://doi.org/10.1016/j.ast.2020.106155>.
- [20] B. Lu. *Linear Parameter-Varying Control of an F-16 Aircraft at High Angle of Attack*. PhD Thesis - North Carolina University, 2004.
- [21] M. Shanmugavel. Path planning of multiple autonomous vehicles. *PhD Thesis - Cranfield University*, July 2007.
- [22] F. Duchoň, A. Babinec, M. Kajan, P. Beňo, M. Florek, T. Fico, and L. Jurišica. Path planning with modified a star algorithm for a mobile robot. *Procedia Engineering*, 96:59–69, 2014. doi: 10.1016/j.proeng.2014.12.098. URL <https://doi.org/10.1016/j.proeng.2014.12.098>.
- [23] A. Felner. Position paper: Dijkstra’s algorithm versus uniform cost search or a case against dijkstra’s algorithm. In *SOCS*, 2011.
- [24] B. M. Sathyaraj, L. C. Jain, A. Finn, and S. Drake. Multiple UAVs path planning algorithms: a comparative study. *Fuzzy Optimization and Decision Making*, 7(3):257–267, June 2008. doi: 10.1007/s10700-008-9035-0. URL <https://doi.org/10.1007/s10700-008-9035-0>.
- [25] Y. Qu, Y. Zhang, and Y. Zhang. A global path planning algorithm for fixed-wing UAVs. *Journal of Intelligent & Robotic Systems*, 91(3-4):691–707,

- November 2017. doi: 10.1007/s10846-017-0729-9. URL <https://doi.org/10.1007/s10846-017-0729-9>.
- [26] S. A. Bortoff. Path planning for UAVs. In *Proceedings of the 2000 American Control Conference. ACC (IEEE Cat. No.00CH36334)*. IEEE, 2000. doi: 10.1109/acc.2000.878915. URL <https://doi.org/10.1109/acc.2000.878915>.
- [27] L. Liu and S. Zhang. Voronoi diagram and GIS-based 3d path planning. In *2009 17th International Conference on Geoinformatics*. IEEE, August 2009. doi: 10.1109/geoinformatics.2009.5293350. URL <https://doi.org/10.1109/geoinformatics.2009.5293350>.
- [28] K. Judd and T. McLain. Spline based path planning for unmanned air vehicles. In *AIAA Guidance, Navigation, and Control Conference and Exhibit*. American Institute of Aeronautics and Astronautics, August 2001. doi: 10.2514/6.2001-4238. URL <https://doi.org/10.2514/6.2001-4238>.
- [29] Y. Ho and J. Liu. Collision-free curvature-bounded smooth path planning using composite bezier curve based on voronoi diagram. In *2009 IEEE International Symposium on Computational Intelligence in Robotics and Automation - (CIRA)*. IEEE, December 2009. doi: 10.1109/cira.2009.5423161. URL <https://doi.org/10.1109/cira.2009.5423161>.
- [30] Y. Deng, Y. Chen, Y. Zhang, and S. Mahadevan. Fuzzy dijkstra algorithm for shortest path problem under uncertain environment. *Applied Soft Computing*, 12(3):1231–1237, March 2012. doi: 10.1016/j.asoc.2011.11.011. URL <https://doi.org/10.1016/j.asoc.2011.11.011>.
- [31] L. Wang and D. Zhou. Effective path planning method for low detectable aircraft. *Journal of Systems Engineering and Electronics*, pages 784–789, June 2009.
- [32] J. Cui, Y. Zhang, S. Ma, Y. Yi, J. Xin, and D. Liu. Path planning algorithms for power transmission line inspection using unmanned aerial vehicles. In *2017 29th Chinese Control And Decision Conference (CCDC)*. IEEE, May 2017. doi: 10.1109/ccdc.2017.7978899. URL <https://doi.org/10.1109/ccdc.2017.7978899>.

- [33] Y. Hao, Z. Shen, and Y. Zhao. Path planning for aircraft based on MAK-LINK graph theory and multi colony ant algorithm. In *2009 International Joint Conference on Computational Sciences and Optimization*. IEEE, April 2009. doi: 10.1109/cso.2009.315. URL <https://doi.org/10.1109/cso.2009.315>.
- [34] H. Li, F. Karray, O. Basir, and I. Song. A neural network based intelligent planner for the coordinated hybrid agent framework. In *IEEE International Conference Mechatronics and Automation, 2005*. IEEE, 2005. doi: 10.1109/icma.2005.1626631. URL <https://doi.org/10.1109/icma.2005.1626631>.
- [35] S. Li, X. Xu, and L. Zuo. Dynamic path planning of a mobile robot with improved q-learning algorithm. In *2015 IEEE International Conference on Information and Automation*. IEEE, August 2015. doi: 10.1109/icinfa.2015.7279322. URL <https://doi.org/10.1109/icinfa.2015.7279322>.
- [36] L. E. Dubins. On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents. *American Journal of Mathematics*, 79(3):497, July 1957. doi: 10.2307/2372560. URL <https://doi.org/10.2307/2372560>.
- [37] Y. Zhao and P. Tsiotras. A quadratic programming approach to path smoothing. In *Proceedings of the 2011 American Control Conference*. IEEE, June 2011. doi: 10.1109/acc.2011.5990880. URL <https://doi.org/10.1109/acc.2011.5990880>.
- [38] X.-N. Bui, J.-D. Boissonnat, P. Soueres, and J.-P. Laumond. Shortest path synthesis for dubins non-holonomic robot. In *Proceedings of the 1994 IEEE International Conference on Robotics and Automation*. IEEE Comput. Soc. Press, 1994. doi: 10.1109/robot.1994.351019. URL <https://doi.org/10.1109/robot.1994.351019>.
- [39] G. Parlangeli and G. Indiveri. Dubins inspired 2d smooth paths with bounded curvature and curvature derivative. *IFAC Proceedings Volumes*, 43(16):252–257, 2010. doi: 10.3182/20100906-3-it-2019.00045. URL <https://doi.org/10.3182/20100906-3-it-2019.00045>.

- [40] N. Karapetyan, J. Moulton, J. S. Lewis, A. Q. Li, J. M. Okane, and I. Rekleitis. Multi-robot dubins coverage with autonomous surface vehicles. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, May 2018. doi: 10.1109/icra.2018.8460661. URL <https://doi.org/10.1109/icra.2018.8460661>.
- [41] H. Chitsaz and S. M. LaValle. Time-optimal paths for a dubins airplane. In *2007 46th IEEE Conference on Decision and Control*. IEEE, 2007. doi: 10.1109/cdc.2007.4434966. URL <https://doi.org/10.1109/cdc.2007.4434966>.
- [42] J. Reeds and L. Shepp. Optimal paths for a car that goes both forwards and backwards. *Pacific Journal of Mathematics*, 145(2):367–393, October 1990. doi: 10.2140/pjm.1990.145.367. URL <https://doi.org/10.2140/pjm.1990.145.367>.
- [43] C. Siedentop, R. Heinze, D. Kasper, G. Breuel, and C. Stachniss. Path-planning for Autonomous Parking with Dubins curves. 01 2015.
- [44] M. Owen, R. W. Beard, and T. W. McLain. Implementing dubins airplane paths on fixed-wing UAVs AST. In *Handbook of Unmanned Aerial Vehicles*, pages 1677–1701. Springer Netherlands, August 2014. doi: 10.1007/978-90-481-9707-1_120. URL https://doi.org/10.1007/978-90-481-9707-1_120.
- [45] M. Shanmugavel, A. Tsourdos, B. White, and R. Zbikowski. 3d dubins sets based coordinated path planning for swarm of UAVs. In *AIAA Guidance, Navigation, and Control Conference and Exhibit*. American Institute of Aeronautics and Astronautics, August 2006. doi: 10.2514/6.2006-6211. URL <https://doi.org/10.2514/6.2006-6211>.
- [46] W. Cai, M. Zhang, and Y. Zheng. Task assignment and path planning for multiple autonomous underwater vehicles using 3d dubins curves †. *Sensors*, 17(7):1607, July 2017. doi: 10.3390/s17071607. URL <https://doi.org/10.3390/s17071607>.
- [47] R. D. Hurley, R. C. Lind, and J. Kehoe. A torus based three dimensional motion planning model for very maneuverable micro air vehicles. In *AIAA*

- Guidance, Navigation, and Control (GNC) Conference*. American Institute of Aeronautics and Astronautics, August 2013. doi: 10.2514/6.2013-4938. URL <https://doi.org/10.2514/6.2013-4938>.
- [48] R. T. Farouki and V. T. Rajan. On the numerical condition of polynomials in bernstein form. *Computer Aided Geometric Design*, 4(3):191–216, November 1987. doi: 10.1016/0167-8396(87)90012-4. URL [https://doi.org/10.1016/0167-8396\(87\)90012-4](https://doi.org/10.1016/0167-8396(87)90012-4).
- [49] K. Weierstrass. Über die analytische darstellbarkeit sogenannter willkürlicher functionen einer reellen veränderlichen. In *Verl. d. Kgl. Akad. d. Wiss. Berlin* 2(1885), pages 633–639. Königlich Preussischen Akademie der Wissenschaften zu Berlin, 1885.
- [50] R. T. Farouki. The bernstein polynomial basis: A centennial retrospective. *Computer Aided Geometric Design*, 29(6):379–419, August 2012. doi: 10.1016/j.cagd.2012.03.001. URL <https://doi.org/10.1016/j.cagd.2012.03.001>.
- [51] R. Choe, V. Cichella, E. Xargay, N. Hovakimyan, A. C. Trujillo, and I. Kammer. A trajectory-generation framework for time-critical cooperative missions. In *AIAA Infotech Aerospace (IA) Conference*. American Institute of Aeronautics and Astronautics, August 2013. doi: 10.2514/6.2013-4582. URL <https://doi.org/10.2514/6.2013-4582>.
- [52] R. Choe, R. Puig, R. Cichella, R. Xargay, and R. Hovakimyan. Trajectory generation using spatial pythagorean hodograph bezier curves. In *AIAA Guidance, Navigation, and Control Conference*. American Institute of Aeronautics and Astronautics, January 2015. doi: 10.2514/6.2015-0597. URL <https://doi.org/10.2514/6.2015-0597>.
- [53] V. Cichella, R. Choe, B. S. Mehdi, E. Xargay, N. Hovakimyan, A. C. Trujillo, and I. Kammer. Trajectory generation and collision avoidance for safe operation of cooperating UAVs. In *AIAA Guidance, Navigation, and Control Conference*. American Institute of Aeronautics and Astronautics, January 2014. doi: 10.2514/6.2014-0972. URL <https://doi.org/10.2514/6.2014-0972>.

- [54] S. B. Mehdi, R. Choe, and N. Hovakimyan. Avoiding multiple collisions through trajectory replanning using piecewise bézier curves. In *2015 54th IEEE Conference on Decision and Control (CDC)*. IEEE, December 2015. doi: 10.1109/cdc.2015.7402633. URL <https://doi.org/10.1109/cdc.2015.7402633>.
- [55] S. B. Mehdi, R. Choe, V. Cichella, and N. Hovakimyan. Collision avoidance through path replanning using bézier curves. In *AIAA Guidance, Navigation, and Control Conference*. American Institute of Aeronautics and Astronautics, January 2015. doi: 10.2514/6.2015-0598. URL <https://doi.org/10.2514/6.2015-0598>.
- [56] R. Choe, R. Puig-Navarro, V. Cichella, E. Xargay, and N. Hovakimyan. Cooperative trajectory generation using pythagorean hodograph bézier curves. *Journal of Guidance, Control, and Dynamics*, 39(8):1744–1763, August 2016. doi: 10.2514/1.g001531. URL <https://doi.org/10.2514/1.g001531>.
- [57] S. Song, Z. Li, M Q.-H. Meng, H. Yu, and H. Ren. Real-time shape estimation for wire-driven flexible robots with multiple bending sections based on quadratic bézier curves. *IEEE Sensors Journal*, 15(11):6326–6334, November 2015. doi: 10.1109/jsen.2015.2456181. URL <https://doi.org/10.1109/jsen.2015.2456181>.
- [58] K. Yang and S. Sukkarieh. 3d smooth path planning for a UAV in cluttered natural environments. In *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, September 2008. doi: 10.1109/iro.2008.4650637. URL <https://doi.org/10.1109/iro.2008.4650637>.
- [59] I. Bae, J. Moon, H. Park, J. H. Kim, and S. Kim. Path generation and tracking based on a bezier curve for a steering rate controller of autonomous vehicles. In *16th International IEEE Conference on Intelligent Transportation Systems (ITSC 2013)*. IEEE, October 2013. doi: 10.1109/itsc.2013.6728270. URL <https://doi.org/10.1109/itsc.2013.6728270>.
- [60] L. Yang, J. Qi, J. Xiao, and X. Yong. A literature review of UAV 3d path planning. In *Proceeding of the 11th World Congress on Intelligent Control*

- and Automation*. IEEE, June 2014. doi: 10.1109/wcica.2014.7053093. URL <https://doi.org/10.1109/wcica.2014.7053093>.
- [61] T. Kito, J. Ota, R. Katsuki, T. Mizuta, T. Arai, T. Ueyama, and T. Nishiyama. Smooth path planning by using visibility graph-like method. In *2003 IEEE International Conference on Robotics and Automation (Cat. No.03CH37422)*. IEEE, 2003. doi: 10.1109/robot.2003.1242175. URL <https://doi.org/10.1109/robot.2003.1242175>.
- [62] T. Fraichard and A. Scheuer. From reeds and shepp's to continuous-curvature paths. *IEEE Transactions on Robotics*, 20(6):1025–1035, December 2004. doi: 10.1109/tro.2004.833789. URL <https://doi.org/10.1109/tro.2004.833789>.
- [63] N. Nilsson. *Principles of Artificial Intelligence*. Elsevier, 1980. doi: 10.1016/c2009-0-27546-5. URL <https://doi.org/10.1016/c2009-0-27546-5>.
- [64] T. L. Fine. Fundamentals of artificial neural networks [book reviews]. *IEEE Transactions on Information Theory*, 42(4):1322, July 1996. doi: 10.1109/tit.1996.508868. URL <https://doi.org/10.1109/tit.1996.508868>.
- [65] J. Schmidhuber. Deep learning in neural networks: An overview. *Neural Networks*, 61:85–117, January 2015. doi: 10.1016/j.neunet.2014.09.003. URL <https://doi.org/10.1016/j.neunet.2014.09.003>.
- [66] A. Atakulreka and D. Sutivong. Avoiding local minima in feedforward neural networks by simultaneous learning. In *AI 2007: Advances in Artificial Intelligence*, pages 100–109. Springer Berlin Heidelberg, 2007. doi: 10.1007/978-3-540-76928-6_12. URL https://doi.org/10.1007/978-3-540-76928-6_12.
- [67] A. Calise and R. Rysdyk. Nonlinear adaptive flight control using neural networks. *IEEE Control Systems*, 18(6):14–25, December 1998. doi: 10.1109/37.736008. URL <https://doi.org/10.1109/37.736008>.
- [68] S. A. Gautam and N. Verma. Path planning for unmanned aerial vehicle based on genetic algorithm and artificial neural network in 3d. In *2014*

- International Conference on Data Mining and Intelligent Computing (ICDMIC)*. IEEE, 2014. doi: 10.1109/icdmic.2014.6954257. URL <https://doi.org/10.1109/icdmic.2014.6954257>.
- [69] R. Parmar. Training deep neural networks, Sep 2018. URL <https://towardsdatascience.com/training-deep-neural-networks-9fdb1964b964>. [Accessed: May 2020].
- [70] H. Li, S. X. Yang, and M. L. Seto. Neural-network-based path planning for a multirobot system with moving obstacles. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 39(4):410–419, July 2009. doi: 10.1109/tsmcc.2009.2020789. URL <https://doi.org/10.1109/tsmcc.2009.2020789>.
- [71] S. X. Yang and M. Meng. Real-time collision-free path planning of robot manipulators using neural network approaches. *Autonomous Robots*, 9(1): 27–39, 2000. doi: 10.1023/a:1008920117364. URL <https://doi.org/10.1023/a:1008920117364>.
- [72] K. D. Julian and M. J. Kochenderfer. Neural network guidance for UAVs. In *AIAA Guidance, Navigation, and Control Conference*. American Institute of Aeronautics and Astronautics, January 2017. doi: 10.2514/6.2017-1743. URL <https://doi.org/10.2514/6.2017-1743>.
- [73] B. Osinski. What is reinforcement learning? the complete guide, Jul 2020. URL <https://deepsense.ai/what-is-reinforcement-learning-the-complete-guide/>.
- [74] R. Moni. Reinforcement learning algorithms - an intuitive overview, Feb 2019. URL shorturl.at/bhyDS.
- [75] A. L. Strehl, L. Li, E. Wiewiora, J. Langford, and M. L. Littman. PAC model-free reinforcement learning. In *Proceedings of the 23rd international conference on Machine learning - ICML '06*. ACM Press, 2006. doi: 10.1145/1143844.1143955. URL <https://doi.org/10.1145/1143844.1143955>.

- [76] L. P. Kaelbling, M. L. Littman, and A. W. Moore. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, 4:237–285, May 1996. doi: 10.1613/jair.301. URL <https://doi.org/10.1613/jair.301>.
- [77] P. Dayan and Y. Niv. Reinforcement learning: The good, the bad and the ugly. *Current Opinion in Neurobiology*, 18(2):185–196, April 2008. doi: 10.1016/j.conb.2008.08.003. URL <https://doi.org/10.1016/j.conb.2008.08.003>.
- [78] J. Achiam. Part 2: Kinds of RL algorithms, 2020. URL https://spinningup.openai.com/en/latest/spinningup/rl_intro2.html.
- [79] C. J. C. H. Watkins and P. Dayan. Q-learning. *Machine Learning*, 8(3-4): 279–292, May 1992. doi: 10.1007/bf00992698. URL <https://doi.org/10.1007/bf00992698>.
- [80] A. Violante. Simple reinforcement learning: Q-learning, Jul 2019. URL <https://towardsdatascience.com/simple-reinforcement-learning-q-learning-fcddc4b6fe56>.
- [81] C. Watkins. Learning from delayed rewards. University of Cambridge, 1989.
- [82] H. Hasselt. Double q-learning. In J. Lafferty, C. Williams, J. Shawe-Taylor, R. Zemel, and A. Culotta, editors, *Advances in Neural Information Processing Systems*, volume 23, pages 2613–2621. Curran Associates, Inc., 2010. URL <https://proceedings.neurips.cc/paper/2010/file/091d584fced301b442654dd8c23b3fc9-Paper.pdf>.
- [83] I. Goswami, P. K. Das, A. K., and R. Janarthanan. Extended q-learning algorithm for path-planning of a mobile robot. In *Lecture Notes in Computer Science*, pages 379–383. Springer Berlin Heidelberg, 2010. doi: 10.1007/978-3-642-17298-4_40. URL https://doi.org/10.1007/978-3-642-17298-4_40.
- [84] E. S. Low, P. Ong, and K. C. Cheah. Solving the optimal path planning of a mobile robot using improved q-learning. *Robotics and Autonomous*

- Systems*, 115:143–161, May 2019. doi: 10.1016/j.robot.2019.02.013. URL <https://doi.org/10.1016/j.robot.2019.02.013>.
- [85] Z. Yijing, Z. Zheng, Z. Xiaoyi, and L. Yang. Q learning algorithm based UAV path learning and obstacle avoidance approach. In *2017 36th Chinese Control Conference (CCC)*. IEEE, July 2017. doi: 10.23919/chicc.2017.8027884. URL <https://doi.org/10.23919/chicc.2017.8027884>.
- [86] A. Choudhary. Deep q-learning: An introduction to deep reinforcement learning, Apr 2020. URL <https://www.analyticsvidhya.com/blog/2019/04/introduction-deep-q-learning-python/>.
- [87] S. Zhou, X. Liu, Y. Xu, and J. Guo. A deep q-network (DQN) based path planning method for mobile robots. In *2018 IEEE International Conference on Information and Automation (ICIA)*. IEEE, August 2018. doi: 10.1109/icinfa.2018.8812452. URL <https://doi.org/10.1109/icinfa.2018.8812452>.
- [88] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller. Playing atari with deep reinforcement learning, 2013.
- [89] C. Yan, X. Xiang, and C. Wang. Towards real-time path planning through deep reinforcement learning for a UAV in dynamic environments. *Journal of Intelligent & Robotic Systems*, 98(2):297–309, September 2019. doi: 10.1007/s10846-019-01073-3. URL <https://doi.org/10.1007/s10846-019-01073-3>.
- [90] D. Xin, C. Hua-hua, and G. Wei-kang. Neural network and genetic algorithm based global path planning in a static environment. *Journal of Zhejiang University-SCIENCE A*, 6(6):549–554, June 2005. doi: 10.1631/jzus.2005.a0549. URL <https://doi.org/10.1631/jzus.2005.a0549>.
- [91] B. N. Pamadi. *Performance, Stability, Dynamics and Control of Airplanes, Third Edition*. American Institute of Aeronautics and Astronautics, Inc., March 2015. doi: 10.2514/4.102745. URL <https://doi.org/10.2514/4.102745>.
- [92] J. Blakelock. *Automatic Control of Aircraft and Missiles*. Wiley Interscience, November 1991.

- [93] W. R. Hamilton. Theory of quaternions. *Proceedings of the Royal Irish Academy (1836 - 1869), 1844 - 1847*, 3:1–16.
- [94] R. Goldman. Understanding quaternions. *Graphical Models*, 73(2):21–49, March 2011. doi: 10.1016/j.gmod.2010.10.004. URL <https://doi.org/10.1016/j.gmod.2010.10.004>.
- [95] S. Bennet. Development of the PID controller. volume 13, pages 58–62. Institute of Electrical and Electronics Engineers (IEEE), December 1993. doi: 10.1109/37.248006. URL <https://doi.org/10.1109/37.248006>. [Accessed: 23rd March 2021].
- [96] G. A. Bestard and S. C. A. Alfaro. Automatic control of the weld bead geometry. In *Welding - Modern Topics*. IntechOpen, January 2021. doi: 10.5772/intechopen.91914. URL <https://doi.org/10.5772/intechopen.91914>. [Accessed: 23rd March 2021].
- [97] N. Wahid and N. Hassan. Self-tuning fuzzy PID controller design for aircraft pitch control. In *2012 Third International Conference on Intelligent Systems Modelling and Simulation*. IEEE, February 2012. doi: 10.1109/isms.2012.27. URL <https://doi.org/10.1109/isms.2012.27>.
- [98] M. L. Forushani, B. Karimi, and G. Shahgholian. Optimal PID controller tuning for multivariable aircraft longitudinal autopilot based on particle swarm optimization algorithm. *Journal of Intelligent Procedures in Electrical Technology*, 3(9), 2012.
- [99] N. Beygi, M. Beigy, and M. Siahi. Design of fuzzy self-tuning PID controller for pitch control system of aircraft autopilot. *CoRR*, abs/1510.02588, 2015. URL <http://arxiv.org/abs/1510.02588>.
- [100] B. Kada and Y. Ghazzawi. Robust PID controller design for an UAV flight control system. *Proceedings of the World Congress on Engineering and Computer Science*, 2, 2011.
- [101] K. Papakollu, P. Singru, and N. S. Manjarekar. Comparing PID and H-infinity controllers on a 2-DoF nonlinear quarter car suspension system. In *Journal of Vibroengineering*, volume 16. JVE International, 2014.

- [102] G. Zames. Input-output feedback stability and robustness, 1959-85. *IEEE Control Systems*, 16(3):61–66, June 1996. doi: 10.1109/37.506399. URL <https://doi.org/10.1109/37.506399>.
- [103] H. Kwakernaak. Robust control and h_∞ -optimization—tutorial paper. *Automatica*, 29(2):255–273, March 1993. doi: 10.1016/0005-1098(93)90122-a. URL [https://doi.org/10.1016/0005-1098\(93\)90122-a](https://doi.org/10.1016/0005-1098(93)90122-a).
- [104] L. Rollins. Robust control theory, 1999. URL https://users.ece.cmu.edu/~koopman/des_s99/control_theory/. [Accessed: 23rd March 2021].
- [105] R. Kaur and J. Ohri. H-infinity controller design for pneumatic servosystem: a comparative study. *International Journal of Automation and Control*, 8(3):242, 2014. doi: 10.1504/ijaac.2014.064162. URL <https://doi.org/10.1504/ijaac.2014.064162>.
- [106] M. S. Saad, H. Jamaluddin, and I. Z. M. Darus. PID controller tuning using evolutionary algorithms. In *International Journal of Innovative Computing, Information and Control*, volume 8, pages 7781–7779. WSEAS, 2012. [Accessed: 23rd March 2021].
- [107] W. J. Rugh and J. S. Shamma. Research on gain scheduling. *Automatica*, 36(10):1401–1425, October 2000. doi: 10.1016/S0005-1098(00)00058-3. URL [https://doi.org/10.1016/S0005-1098\(00\)00058-3](https://doi.org/10.1016/S0005-1098(00)00058-3).
- [108] J. S. Shamma and M. Athans. Guaranteed properties of gain scheduled control for linear parameter-varying plants. *Automatica*, 27(3):559–564, May 1991. doi: 10.1016/0005-1098(91)90116-j. URL [https://doi.org/10.1016/0005-1098\(91\)90116-j](https://doi.org/10.1016/0005-1098(91)90116-j).
- [109] R. T. Reichert. Dynamic scheduling of modern-robust-control autopilot designs for missiles. *IEEE Contr. Syst. Mag.*, 12(5):35–42, Oct 1992. doi: 10.1109/37.158896.
- [110] T. Richardson, P. Davison, M. Lowenberg, and M. di Bernardo. Control of nonlinear aircraft models using dynamic state-feedback gain scheduling. In *AIAA Guidance, Navigation, and Control Conference and Exhibit*. American

- Institute of Aeronautics and Astronautics, August 2003. doi: 10.2514/6.2003-5503. URL <https://doi.org/10.2514/6.2003-5503>.
- [111] M. W. McConley, B. D. Appleby, M. A. Dahleh, and E. Feron. A computationally efficient Lyapunov-based scheduling procedure for control of nonlinear systems with stability guarantees. *IEEE T. Automat. Contr.*, 45(1): 33–49, 2003. ISSN 00189286. doi: 10.1109/9.827354.
- [112] D. J. Stilwell and W. J. Rugh. Interpolation of observer state feedback controllers for gain scheduling. *IEEE Transactions on Automatic Control*, 44(6):1225–1229, June 1999. doi: 10.1109/9.769379. URL <https://doi.org/10.1109/9.769379>.
- [113] Y. Choi, H. Jimenez, and D. N. Mavris. Statistical gain-scheduling method for aircraft flight simulation. *Aerospace Science and Technology*, 46:493–505, October 2015. doi: 10.1016/j.ast.2015.08.011. URL <https://doi.org/10.1016/j.ast.2015.08.011>.
- [114] A. Barfield and J. Dazzo. Multivariable control laws for the AFTI/f-16. In *22nd Aerospace Sciences Meeting*. American Institute of Aeronautics and Astronautics, January 1984. doi: 10.2514/6.1984-237. URL <https://doi.org/10.2514/6.1984-237>.
- [115] C. Knospe. PID control. *IEEE Control Systems*, 26(1):30–31, February 2006. doi: 10.1109/mcs.2006.1580151. URL <https://doi.org/10.1109/mcs.2006.1580151>.
- [116] J. G. Ziegler and N. B. Nichols. Optimum settings for automatic controllers. *Journal of Dynamic Systems, Measurement, and Control*, 115(2B):220–222, June 1993. doi: 10.1115/1.2899060. URL <https://doi.org/10.1115/1.2899060>.
- [117] G. H. Cohen and G. A. Coon. Theoretical consideration of retarded control. In *Intelligent Control and Automation, Vol.2 No.2, June 3, 2011*, volume 75, pages 827–834. Transactions of the ASME, 1952.
- [118] F. Isdaryani, F. Feriyonika, and R. Ferdiansyah. Comparison of ziegler-nichols and cohen coon tuning method for magnetic levitation control

- system. *Journal of Physics: Conference Series*, 1450:012033, February 2020. doi: 10.1088/1742-6596/1450/1/012033. URL <https://doi.org/10.1088/1742-6596/1450/1/012033>.
- [119] S. N. Deepa and G. Sudha. Longitudinal control of aircraft dynamics based on optimization of PID parameters. *Thermophysics and Aeromechanics*, 23(2):185–194, March 2016. doi: 10.1134/s0869864316020049. URL <https://doi.org/10.1134/s0869864316020049>.
- [120] M. Abdel-Basset, L. Abdel-Fatah, and A. K. Sangaiah. Metaheuristic algorithms: A comprehensive review. In *Computational Intelligence for Multimedia Big Data on the Cloud with Engineering Applications*, pages 185–231. Elsevier, 2018. doi: 10.1016/b978-0-12-813314-9.00010-4. URL <https://doi.org/10.1016/b978-0-12-813314-9.00010-4>.
- [121] S. Pareek, M. Kishnani, and R. Gupta. Optimal tuning of pid controller using genetic algorithm and swarm technologies. In *International Journal of Electronic and Electrical Engineering*, pages 189–194. International Research Publication House, 2014.
- [122] J. F. M. Amaral, R. Tanscheit, and M. A. C. Pacheco. Tuning pid controllers through genetic algorithms. In *2001 WSES International Conference on Evolutionary Computation, Puerto De La*, 2001.
- [123] Andri Mirzal, Shinichiro Yoshii, and M. Furukawa. Pid parameters optimization by using genetic algorithm. *ArXiv*, abs/1204.0885, 2012.
- [124] U. Arora, E. L. Aslan, and T. K. Saxena. Pid parameter tuning using modified bat algorithm. *Journal of Automation and Control Engineering*, 2016.
- [125] B. Omar and I. B. Saida. Bat algorithm for optimal tuning of pid controller in an avr system. 2014.
- [126] X.-S. Yang. A new metaheuristic bat-inspired algorithm. In *Nature Inspired Cooperative Strategies for Optimization (NICSO 2010)*, pages 65–74. Springer Berlin Heidelberg, 2010. doi: 10.1007/978-3-642-12538-6_6. URL https://doi.org/10.1007/978-3-642-12538-6_6.

- [127] D. A. Coley. *An Introduction to Genetic Algorithms for Scientists and Engineers*. WORLD SCIENTIFIC, January 1999. doi: 10.1142/3904. URL <https://doi.org/10.1142/3904>.
- [128] K. Jebari. Selection methods for genetic algorithms. In *International Journal of Emerging Sciences*, pages 333–344. IJES, 2013.
- [129] B. L. Miller and D. E. Goldberg. Genetic algorithms, tournament selection, and the effects of noise. *Complex Systems*, 9:193–212, 1995.
- [130] J. Zhong, X. Hu, J. Zhang, and M. Gu. Comparison of performance between different selection strategies on simple genetic algorithms. In *International Conference on Computational Intelligence for Modelling, Control and Automation and International Conference on Intelligent Agents, Web Technologies and Internet Commerce (CIMCA-IAWTIC'06)*. IEEE, 2005. doi: 10.1109/cimca.2005.1631619. URL <https://doi.org/10.1109/cimca.2005.1631619>.
- [131] T. Pencheva, K. Atanassov, and A. Shannon. Modelling of a roulette wheel selection operator in genetic algorithms using generalized nets. 2009.
- [132] N. Saini. Review of selection methods in genetic algorithms. *International Journal of Engineering and Computer Science*, 6(12):22261–22263, Dec. 2017. URL <http://www.ijecs.in/index.php/ijecs/article/view/2562>.
- [133] D. E. Goldberg and K. Deb. A comparative analysis of selection schemes used in genetic algorithms. In *Foundations of Genetic Algorithms*, pages 69–93. Elsevier, 1991. doi: 10.1016/b978-0-08-050684-5.50008-2. URL <https://doi.org/10.1016/b978-0-08-050684-5.50008-2>.
- [134] E. W. Weisstein. Bézier curve', . URL <https://mathworld.wolfram.com/BezierCurve.html>. [Accessed: September 2019].
- [135] MathWorks. Pure pursuit controller, February 2021. URL <https://uk.mathworks.com/help/robotics/ug/pure-pursuit-controller.html>.
- [136] D. R. Baughman and Y. A. Liu. Classification: Fault diagnosis and feature categorization. In *Neural Networks in Bioprocessing and Chemical Engineering*,

- pages 110–171. Elsevier, 1995. doi: 10.1016/b978-0-12-083030-5.50009-6. URL <https://doi.org/10.1016/b978-0-12-083030-5.50009-6>.
- [137] H. K. D. H. Bhadeshia. Neural networks in materials science. In *Encyclopedia of Materials: Science and Technology*, pages 1–5. Elsevier, 2008. doi: 10.1016/b978-008043152-9.02201-6. URL <https://doi.org/10.1016/b978-008043152-9.02201-6>.
- [138] O. Knocklein. Classification using neural networks, Jun 2019. URL <https://towardsdatascience.com/classification-using-neural-networks-b8e98f3a904f>.
- [139] D. Tomar and S. Agarwal. A survey on data mining approaches for health-care. *International Journal of Bio-Science and Bio-Technology*, 5(5):241–266, October 2013. doi: 10.14257/ijbsbt.2013.5.5.25. URL <https://doi.org/10.14257/ijbsbt.2013.5.5.25>.
- [140] R. Shaw. *Fighter combat: Tactics and maneuvering*. Naval Institute Press, M. D. Annapolis, 1985.
- [141] J. W. Harris and H. Stocker. *Handbook of Mathematics and Computational Science*. Springer New York, 1998. doi: 10.1007/978-1-4612-5317-4. URL <https://doi.org/10.1007/978-1-4612-5317-4>.
- [142] Wiki Creative Commons. File:f16 drawing.svg. URL https://commons.wikimedia.org/wiki/File:F16_drawing.svg.
- [143] S. Alrasheed. Newton’s laws. In *Principles of Mechanics*, pages 37–51. Springer International Publishing, 2019. doi: 10.1007/978-3-030-15195-9_3. URL https://doi.org/10.1007/978-3-030-15195-9_3.
- [144] J. A. Thelander. Aircraft motion analysis. Air Force Flight Dynamics Laboratory, Research and Technology Division, 1965.
- [145] E. Duke, R. Antoniewicz, and K. Krambeer. Derivation and definition of a linear aircraft model. National Aeronautics and Space Administration (NASA) - Scientific and Technical Information Division, 1988.

- [146] M. V. Cook. *Flight Dynamics Principles: A Linear Systems Approach to Aircraft Stability and Control (Aerospace Engineering)*. Butterworth-Heinemann, dec 2012. ISBN 0080982425. URL <https://www.xarg.org/ref/a/0080982425/>.
- [147] E. Weisstein. Vector triple product, . URL <https://mathworld.wolfram.com/VectorTripleProduct.html>.
- [148] D. Williams. Planetary fact sheet - metric, 2018. URL <https://nssdc.gsfc.nasa.gov/planetary/factsheet/>.
- [149] Wikipedia Creative Commons License. U.s. air force standard aircraft characteristics, f-101b voodoo, 2020. URL <https://commons.wikimedia.org/w/index.php?curid=11349451>.

Appendix A

Computational Model

A.1 Aircraft Axis Systems

When formulating problems in flight dynamics, many coordinate or reference systems exist for describing the position, velocity, acceleration, forces and moments upon an aircraft [91]. One cannot simply specify a single axis system, as no single axis system exists that is suitable for the determination of all required quantities.

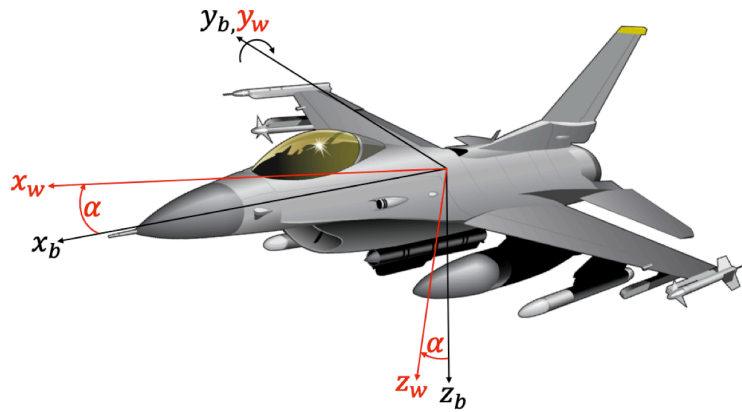


FIGURE A.1 Aircraft body-axis system adapted from Wiki Creative Commons [142]

A.1.1 Inertial Axis Systems

The equations of motion for the aircraft model presented herein are derived using Newtonian Mechanics, where Newton's second law of motion is only valid

when the acceleration is being observed with respect to an inertial frame [143]. Pamadi [91] describes the inertial frame as being a frame of reference where the acceleration is deemed to be zero. Whilst this is difficult to categorise in real-world applications, for flight dynamic problems the inertial frame is assumed to be coincident with the centre of a non-rotating Earth [144]. This neglects orbital motion of the Earth and Sun around the common centre of mass. Since the research presented in this thesis is not interplanetary, and the conducted flight simulations are relatively small in comparison, this is a safe assumption to make. Therefore, using Newton's second law of motion:

$$F = m \left(\frac{dV}{dt} \right)_i = m \begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix}_i \quad (\text{A.1})$$

$$M = \left(\frac{dH}{dt} \right)_i \quad (\text{A.2})$$

Additionally, the inertial axis system can be described as an Earth-fixed axis system which where the spherical coordinates are located at the centre of the earth and rotate with the earth. General nomenclature for an Earth-fixed axis system are (x_e, y_e, z_e) . The origin of the orthogonal system is coincident to the centre of the Earth, with the z_e expanding from the centre, vertically upwards towards the North pole [145]. The axis, x_e expands from the centre, horizontally and inline with the equator with a longitude of 0° [145], also known as the zero meridian or prime meridian [91]. Finally, the axis, y_e , expands from the origin horizontally along the same equatorial plane but perpendicular to the x_e axis.

A.1.2 Body Axes

A body-axis system is a right-handed orthogonal axis system which moves freely with an object with observed from an inertial frame. Generally, the origin of a body-axis system is coincident with the objects centre of gravity, or centre of mass [91]. The x_b axis is directly towards the nose of the aircraft [145] along the longitudinal centreline, whilst the y_b axis points starboard with the z_b axis pointing downwards [145], as depicted in Figure A.1.

In a body-axis system, all forces and moments act along each axis, with the axis systems rotating relative to the object. This allows forces such as thrust to oc-

cur solely along the x -axis, provided the engine is located along the longitudinal centre line. In addition, aerodynamic forces correspond to total aerodynamic forces along each of the body-axis, with aerodynamic models occurring about each of the three body-axes.

A.1.3 Wind Axes

For some flight dynamics problems, it is advantageous to define a set of fixed aircraft axes, such that the x -axis is always located parallel to the aircraft velocity vector, V , known as wind or stability axes [146]. During equilibrium and symmetrical flight, the wind axes are denoted as (x_w, y_w, z_w) , and represent a similar form to traditional body-axes. However, wind axes are rotated about the y_b by the specific angle of attack (α), as seen in Figure A.1.

In a wind-axis system, since the the orientation of the axis system rotates relative to the angle of attack and angle of sideslip, the aerodynamic forces and moments are more appropriately related to the coefficient of lift (C_L) and coefficient of drag (C_D).

According to [146], there are advantages for the use of wind axes, and a set of equations derived from body-axes can be "easily simplified to wind-axes form", depending on the application [146]. However, a set of wind axes equations cannot be easily extended to cater for more generic dynamic applications, making body axes the more preferable option.

A.2 Equations of Motion

When developing the equations of motion, assumptions were made which not only simplified the derivation of the equations, but enable derivation to be performed based on Newton's second law of motion. These assumptions include:

1. The aircraft exists as a rigid-body, meaning that any two points on or within the aircraft remain fixed relative to each other,
2. The equations of motion represent a right-handed orthogonal body-axis system. The body-axis system has a line of symmetry located along the x and z axes [145], meaning that the moment of inertia I_{xy} and I_{yz} are zero.

3. Equation A.1 states that the acceleration of the mass is being observed with respect to an inertial frame. This generality of this expression remains valid if the mass of the object remains constant. Therefore, it has been assumed that throughout the dynamic analysis, the mass of the aircraft remains constant. This adds further simplification to the equations of motion and internal parameters since the location of the centre of gravity does not change.
4. The aircraft is modelled using a body-axis system being observed from a non-rotating Earth-fixed inertial axis system as described in Section A.1.1.

A.2.1 Velocity and Acceleration Analysis

Chapter 3.3 discussed how the since the Earth is non-rotating, the equations of motion neglected the angular velocity of the Earth or any induced angular accelerations as a function of Coriolis. According to Blakelock, this assumption is valid as many on-board sensors, such as gyroscopes and accelerometers do not detect the effects of the Earth's angular velocity [92]. However, to validate this claim an analysis was conducted to determine the effect of negating the angular velocity of the earth and the Coriolis acceleration on the aircraft.

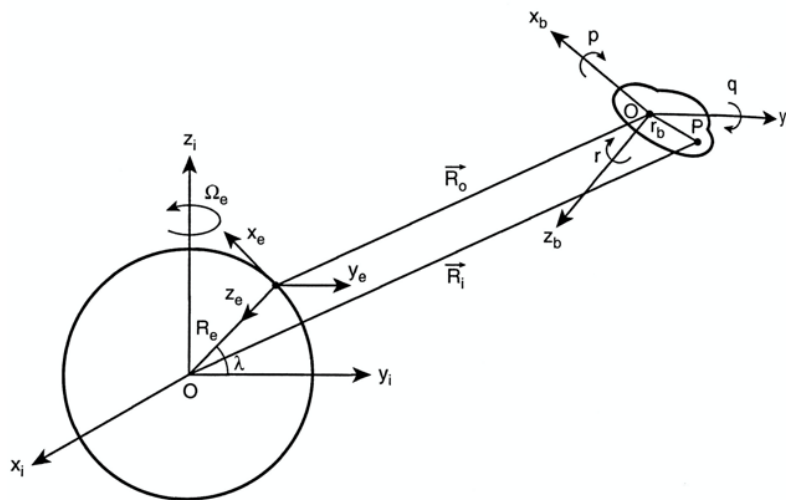


FIGURE A.2 Multi-axis system made up of an inertial frame, Earth-axis and body-axis system [91]

Consider the multi-axis system, as seen in Figure A.2. This multi-axis system

depicts an Earth-fixed axis system acting as an inertial frame of reference, where this axis system is rotating about the z_i -axis by Ω_e . The axes for the reference system are described as (x_i, y_i, z_i) , in accordance to Appendix A.1.1. However, located on the surface of the Earth is a navigational axis system, where x_e extends towards the north, y_e extends towards the east and z_e extends towards the centre of the Earth. During this example, the origin of the body-axis system is offset from the aircraft's centre of gravity by position vector r_b . From this multi-axis representation:

1. The inertial axis system has been placed at the centre of a rotating Earth. As the Earth rotates about the z_i axis, the inertial axis system also rotates with an angular velocity Ω_e .
2. The navigation axis system is located on the surface of the Earth with a position vector magnitude of R_e from the centre along the z_e axis. The magnitude R_e relates to the radius of the Earth, and is therefore approximated to 6,371 km.
3. The body-axis system of the aircraft is offset from the centre of gravity by position vector r_b . Rotation about the body-axis occurs as follows:
 - A positive roll rate (p) occurs due to a clockwise rotation about the x_b axis, rotating the starboard wing downwards.
 - A positive pitch rate (q) occurs due to a clockwise rotation about the y_b axis, rotating the nose of the aircraft upwards.
 - A positive yaw rate (r) occurring due to a clockwise rotation about the z_b axis, rotating the nose of the aircraft towards the navigational y -axis, sometimes referred to as the east axis.
4. The aircraft's body-axis is described by the distance vector $\vec{R}_0$ when observed from the navigational axis, and by the distance vector $\vec{R}_i$ when observed from the Earth-fixed inertial axis system.

Using vector addition, the position vector $\vec{R}_i$ is equal to the magnitude of $\vec{R}_e$, $\vec{R}_0$ and r_b . Hence,

$$\vec{R}_i = \vec{R}_e + \vec{R}_0 + \vec{r}_b \quad (\text{A.3})$$

Therefore the first derivative,

$$\begin{aligned} \left(\frac{d\vec{R}_i}{dt} \right)_i &= \frac{d}{dt}(\vec{R}_e + \vec{R}_o + \vec{r}_b) \\ &= \left(\frac{d\vec{R}_e}{dt} \right)_i + \left(\frac{d\vec{R}_o}{dt} \right)_i + \left(\frac{d\vec{r}_b}{dt} \right)_i \end{aligned} \quad (\text{A.4})$$

Each of the three position vectors were derived individually. Pamadi [91] has outlined a series of derivations using moving axis theorem. The moving axis theorem allows a position vector to be observed whilst in motion by another reference frame and derived over time [91]. The moving axis theorem is defined as:

$$\left(\frac{d\vec{A}}{dt} \right)_i = \left(\frac{d\vec{A}}{dt} \right)_b + \vec{\omega}_{i,b}^b \times \vec{A}_b \quad (\text{A.5})$$

Where $\vec{A}_b$ represents the position vector being observed in the body-axis system, and $\left(\frac{d\vec{A}}{dt} \right)_b$ is the first derivative. Finally, $\vec{\omega}_{i,b}^b$ is the angular velocity of the body-axis system with respect to the inertial frame. All components of angular velocity remain in the body-axis system and are only observed by the inertial frame. Applying moving theorem to the first term in Equation A.4

$$\left(\frac{d\vec{R}_e}{dt} \right)_i = \left(\frac{d\vec{R}_e}{dt} \right)_e + \vec{\Omega}_e \times \vec{R}_e$$

Since $\left(\frac{d\vec{R}_e}{dt} \right)_e$ is the position vector in the earth axis system, and the length of this vector is not changing over time, it becomes zero. Therefore

$$\left(\frac{d\vec{R}_e}{dt} \right)_i = \vec{\Omega}_e \times \vec{R}_e \quad (\text{A.6})$$

Applying vector differentiation to Equation A.6 with respect to time

$$\begin{aligned} \left(\frac{d^2\vec{R}_e}{dt^2} \right)_i &= \frac{d\vec{\Omega}_e}{dt} \times \vec{R}_e + \vec{\Omega}_e \times \frac{d\vec{R}_e}{dt} \\ &= \frac{d\vec{\Omega}_e}{dt} \times \vec{R}_e + \vec{\Omega}_e \times \vec{\Omega}_e \times \vec{R}_e \end{aligned} \quad (\text{A.7})$$

Since the angular velocity of the Earth is constant and there is no acceleration observed in the inertial frame, $\left(\frac{d\vec{\Omega}_e}{dt} \right)_i$ is zero. Therefore Equation A.7 becomes

$$\left(\frac{d^2\vec{R}_e}{dt^2} \right)_i = \vec{\Omega}_e \times \vec{\Omega}_e \times \vec{R}_e \quad (\text{A.8})$$

The same process was then repeated for the second term in Equation A.4

$$\left(\frac{d\vec{R}_0}{dt}\right)_i = \left(\frac{d\vec{R}_0}{dt}\right)_e + \vec{\Omega}_e \times \vec{R}_0$$

Since $\left(\frac{d\vec{R}_0}{dt}\right)_e$ is the rate of change over time of the position for the body axis, with respect to the navigational axis system, this term can be denoted as the translational velocity of the body with respect to the navigational axis system.

Therefore

$$\left(\frac{d\vec{R}_0}{dt}\right)_i = \vec{V}_0^e + \vec{\Omega}_e \times \vec{R}_0 \quad (\text{A.9})$$

Applying vector differentiation to Equation A.9 with respect to time

$$\frac{d^2\vec{R}_0}{dt^2}_i = \frac{d}{dt}\vec{V}_0^e + \frac{d}{dt}(\vec{\Omega}_e \times \vec{R}_0)$$

Where $\frac{d}{dt}(\vec{V}_0^e) = \vec{\dot{V}}_0^e + \vec{\Omega}_e \times \vec{V}_0^e$. Therefore

$$\frac{d^2\vec{R}_0}{dt^2}_i = \vec{\dot{V}}_0^e + \vec{\Omega}_e \times \vec{V}_0^e + \frac{d}{dt}(\vec{\Omega}_e \times \vec{R}_0)$$

Using the chain rule on $\frac{d}{dt}(\vec{\Omega}_e \times \vec{R}_0)$

$$\frac{d^2\vec{R}_0}{dt^2}_i = \vec{\dot{V}}_0^e + \vec{\Omega}_e \times \vec{V}_0^e + \frac{d\vec{\Omega}_e}{dt} \times \vec{R}_0 + \vec{\Omega}_e \times \frac{d\vec{R}_0}{dt}$$

Where $\frac{d}{dt}(\vec{R}_0) = \vec{\dot{R}}_0 + \vec{\Omega}_e \times \vec{R}_0$. Therefore

$$\frac{d^2\vec{R}_0}{dt^2}_i = \vec{\dot{V}}_0^e + \vec{\Omega}_e \times \vec{V}_0^e + \frac{d\vec{\Omega}_e}{dt} \times \vec{R}_0 + \vec{\Omega}_e \times (\vec{\dot{V}}_0^e + \vec{\Omega}_e \times \vec{R}_0)$$

Expanding the right hand-side vectors

$$\frac{d^2\vec{R}_0}{dt^2}_i = \vec{\dot{V}}_0^e + \vec{\Omega}_e \times \vec{V}_0^e + \frac{d\vec{\Omega}_e}{dt} \times \vec{R}_0 + \vec{\Omega}_e \times \vec{\dot{V}}_0^e + \vec{\Omega}_e \times \vec{\Omega}_e \times \vec{R}_0$$

Since the angular velocity of the earth's rotation is constant, and there is no acceleration observed in the inertial frame, $\frac{d\vec{\Omega}_e}{dt}$ is zero. Therefore

$$\begin{aligned} \frac{d^2\vec{R}_0}{dt^2}_i &= \vec{\dot{V}}_0^e + \vec{\Omega}_e \times \vec{V}_0^e + \vec{\Omega}_e \times \vec{\dot{V}}_0^e + \vec{\Omega}_e \times \vec{\Omega}_e \times \vec{R}_0 \\ &= \vec{\dot{V}}_0^e + 2\vec{\Omega}_e \times \vec{V}_0^e + \vec{\Omega}_e \times \vec{\Omega}_e \times \vec{R}_0 \end{aligned} \quad (\text{A.10})$$

Applying moving axis theorem to the third term in Equation A.4:

$$\left(\frac{\vec{dr}_b}{dt}\right)_i = \left(\frac{\vec{dr}_b}{dt}\right)_b + \vec{\omega}_{i,b}^i \times \vec{r}_b \quad (\text{A.11})$$

Since the aircraft has been assumed to be a rigid body, points P and O in the body-axis from Figure A.2 are fixed relative to each other. Therefore the rate of change of the position vector $(\vec{dr}_b/dt)_b$ is zero. Therefore

$$\left(\frac{\vec{dr}_b}{dt}\right)_i = \vec{\omega}_{i,b}^i \times \vec{r}_b \quad (\text{A.12})$$

Both the first and second terms in Equation A.4 relate to a position when observed from an earth-fixed axis system that is rotating about its z_i axis. As a result the angular velocity observed for both components, Ω_e is the angular velocity of the earth-fixed system, this is not the case for the body axis system. Therefore, performing vector differentiation of Equation A.12 with respect to time using the chain rule.

$$\begin{aligned} \left(\frac{d^2\vec{r}_b}{dt^2}\right)_i &= \frac{d\vec{\omega}_{i,b}^i}{dt} \times \vec{r}_b + \vec{\omega}_{i,b}^i \times \frac{d\vec{r}_b}{dt} \\ &= \frac{d\vec{\omega}_{i,b}^i}{dt} \times \vec{r}_b + \vec{\omega}_{i,b}^i \times \vec{\omega}_{i,b}^i \times \vec{r}_b \end{aligned} \quad (\text{A.13})$$

Combining Equations A.8, A.10 and A.13 into the form of Equation A.4

$$\begin{aligned} \left(\frac{d^2\vec{R}_i}{dt^2}\right)_i &= \vec{\Omega}_e \times \vec{\Omega}_e \times \vec{R}_e + \vec{V}_0^e + 2\vec{\Omega}_e \times \vec{V}_0^e + \vec{\Omega}_e \times \vec{\Omega}_e \times \vec{R}_0 \\ &\quad + \frac{d\vec{\omega}_{i,b}^i}{dt} \times \vec{r}_b + \vec{\omega}_{i,b}^i \times \vec{\omega}_{i,b}^i \times \vec{r}_b \end{aligned} \quad (\text{A.14})$$

After rearranging

$$\begin{aligned} \left(\frac{d^2\vec{R}_i}{dt^2}\right)_i &= \vec{V}_0^e + \vec{\Omega}_e \times \vec{\Omega}_e \times (\vec{R}_e + \vec{R}_0) + 2\vec{\Omega}_e \times \vec{V}_0^e \\ &\quad + \frac{d\vec{\omega}_{i,b}^i}{dt} \times \vec{r}_b + \vec{\omega}_{i,b}^i \times \vec{\omega}_{i,b}^i \times \vec{r}_b \end{aligned} \quad (\text{A.15})$$

Where:

- $\vec{V}_0^e = (\vec{a}_0)_e$ is the acceleration of the body-axis with respect to the Earth axis system,
- $\vec{\Omega}_e$ is the angular velocity of the Earth axis with respect to the inertial axis. However the angular velocity components in the Earth axis are given by $\vec{\Omega}_e = \hat{i}_e \Omega_e \cos \lambda - \hat{k}_e \Omega_e \sin \lambda$.
- $\vec{\Omega}_e \times \vec{\Omega}_e \times (\vec{R}_e + \vec{R}_0)$ is the centripetal acceleration exhibited on the body axis due to the angular velocity of the Earth
- $2\vec{\Omega}_e \times \vec{V}_0^e$ is the Coriolis acceleration of the Earth fixed axis system,
- $(d/dt) \vec{\omega}_{i,b}^i$ is the acceleration from the body-axis system due to rate of change of the body angular velocity with respect to time,
- $\vec{\omega}_{i,b}^b \times \vec{\omega}_{i,b}^b \times \vec{r}_b$ is the acceleration of the centre of gravity of the body due to the angular velocity of the moving axis system when referenced by the earth-fixed inertial axis system.

Assuming that the body-axis of the aircraft is coincident with the aircraft centre of gravity, the position vector $\vec{r}_b = 0$. Therefore Equation A.15 becomes

$$\left(\frac{d^2 \vec{R}_i}{dt^2} \right)_i = \vec{\Omega}_e \times \vec{\Omega}_e \times (\vec{R}_e + \vec{R}_0) + (\vec{a}_0)_e + 2\vec{\Omega}_e \times \vec{V}_0^e \quad (\text{A.16})$$

Assuming low altitude flight, the magnitude of position vector $\vec{R}_0$ is significantly smaller than the magnitude of position vector $\vec{R}_e$, ($|\vec{R}_0| \ll |\vec{R}_e|$). Therefore

$$\left(\frac{d^2 \vec{R}_i}{dt^2} \right)_i = \vec{\Omega}_e \times \vec{\Omega}_e \times \vec{R}_e + (\vec{a}_0)_e + 2\vec{\Omega}_e \times \vec{V}_0^e \quad (\text{A.17})$$

After multiplying both sides of Equation A.17 by the partial mass derivative of particle P (δ_m).

$$\delta_m \left(\frac{d^2 \vec{R}_i}{dt^2} \right)_i = \delta_m (\vec{\Omega}_e \times \vec{\Omega}_e \times \vec{R}_e) + \delta_m (\vec{a}_0)_e + 2\delta_m (\vec{\Omega}_e \times \vec{V}_0^e) \quad (\text{A.18})$$

and integrating δ_m over the entire aircraft

$$m \left(\frac{d^2 \vec{R}_i}{dt^2} \right)_i = m [\vec{\Omega}_e \times \vec{\Omega}_e \times \vec{R}_e + (\vec{a}_0)_e + 2\vec{\Omega}_e \times \vec{V}_0^e] \quad (\text{A.19})$$

From Newton's Second Law of Motion, $F = ma$. Therefore

$$F = m \left(\frac{d^2 \vec{R}_i}{dt^2} \right)_i = m [\vec{\Omega}_e \times \vec{\Omega}_e \times \vec{R}_e + (\vec{a}_0)_e + 2\vec{\Omega}_e \times \vec{V}_0^e] \quad (\text{A.20})$$

Expanding the four vector quantities, $\vec{\Omega}_e$, $\vec{R}_e$, $\vec{a}_0$ and $\vec{V}_0$ into their respective cartesian coordinate form with unit vectors $\hat{i}$, $\hat{j}$ and $\hat{k}$ [91].

$$\begin{aligned} \vec{\Omega}_e &= \begin{bmatrix} \Omega \cos(\lambda) & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & -\Omega \sin(\lambda) \end{bmatrix} \begin{bmatrix} \hat{i} \\ \hat{j} \\ \hat{k} \end{bmatrix} \\ \vec{V}_0^e &= \begin{bmatrix} V_{0,x} & 0 & 0 \\ 0 & V_{0,y} & 0 \\ 0 & 0 & V_{0,z} \end{bmatrix} \begin{bmatrix} \hat{i} \\ \hat{j} \\ \hat{k} \end{bmatrix} \\ (\vec{a}_0)_e &= \begin{bmatrix} \dot{V}_{0,x} & 0 & 0 \\ 0 & \dot{V}_{0,y} & 0 \\ 0 & 0 & \dot{V}_{0,z} \end{bmatrix} \begin{bmatrix} \hat{i} \\ \hat{j} \\ \hat{k} \end{bmatrix} \\ \vec{R}_e &= \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & -R_e \end{bmatrix} \begin{bmatrix} \hat{i} \\ \hat{j} \\ \hat{k} \end{bmatrix} \end{aligned}$$

Using the Lagrange triple vector cross product method [147], where

$$\vec{a} \times \vec{b} \times \vec{c} = \vec{b} (\vec{a} \cdot \vec{c}) - \vec{c} (\vec{b} \cdot \vec{a}) \quad (\text{A.21})$$

Therefore the centripetal acceleration, $\vec{\Omega}_e \times \vec{\Omega}_e \times \vec{R}_e$ is defined as,

$$\begin{aligned} \vec{\Omega}_e \times \vec{\Omega}_e \times \vec{R}_e &= \vec{\Omega}_e (\vec{\Omega}_e \cdot \vec{R}_e) - \vec{R}_e (\vec{\Omega}_e \cdot \vec{\Omega}_e) \\ &= (\hat{i}_e \Omega_e \cos \lambda - \hat{k}_e \Omega_e \sin \lambda) (\Omega_e R_e \sin \lambda) - (-\hat{k}_e R_e) \Omega_e^2 \\ &= R_e \Omega_e^2 \cos \lambda (\hat{i} \sin \lambda + \hat{k} \cos \lambda) \end{aligned} \quad (\text{A.22})$$

Using the cross product rule, where

$$u \times v = (u_2 v_3 - u_3 v_2) \hat{i} - (u_1 v_3 - u_3 v_1) \hat{j} + (u_1 v_2 - u_2 v_1) \hat{k} \quad (\text{A.23})$$

let $u = \Omega_e$ and $v = V_0^e$. Therefore

$$\begin{aligned} \Omega_e \times V_0^e &= \hat{i} \Omega \sin \lambda V_{0,y} - \hat{j} (\Omega_e \cos \lambda V_{0,z} + \Omega_e \sin \lambda V_{0,x}) \\ &\quad + \hat{k} \Omega_e \cos \lambda V_{0,y} \end{aligned} \quad (\text{A.24})$$

Subsequently each of the components in $\hat{i}$, $\hat{j}$ and $\hat{k}$ can be summated and multiplied by the mass of the aircraft, deriving the equations of motion in cartesian coordinates.

$$\begin{aligned} F_{x,e} &= m[R_e\Omega_e^2 \cos\lambda \sin\lambda + (a_x)_e + 2\Omega_e V_{0,y} \sin\lambda] \\ F_{y,e} &= m[(a_y)_e - 2\Omega_e(V_{0,z} \cos\lambda + V_{0,z} \sin\lambda)] \\ F_{z,e} &= m[R_e\Omega_e^2 \cos^2\lambda + (a_z)_e + 2\Omega_e V_{0,y} \cos\lambda] \end{aligned} \quad (\text{A.25})$$

Where $\vec{F} = \hat{i}F_{x,e} + \hat{j}F_{y,e} + \hat{k}F_{z,e}$

A.2.2 Coriolis Acceleration

It has been previously mentioned that during the study of flight dynamics it is prevalent to define a non-rotating inertial axis system. This non-rotating inertial axis system allows the Coriolis effect and centripetal acceleration, which is induced onto the body axis with respect to the inertial frame, to be neglected. However, it can be seen from Equation A.20 that the Coriolis effect is proportional to the velocity of the body-axis begin observed from the inertial frame. Therefore neglecting the Coriolis effect for high velocity applications will result in an increased error. We will therefore create an operating scenario which will be similar to dynamic simulations being performed.

The F-16 model contains non-linear aerodynamics that are only valid in subsonic to low transonic conditions. We will consider the aircraft to be travelling at Mach Number 0.6 at sea-level conditions. Therefore the position vector of the navigation axis will coincide with the position vector from the fixed Earth reference ($|\vec{R}_o| \ll |\vec{R}_e|$). The speed of sound is approximately 340.3 m/s. Therefore the velocity, V_0 of the aircraft is approximately 204 m/s. The aircraft will be considered to be flying along the equator, with a latitude angle of zero, i.e. $\lambda = 0^\circ$. The equatorial radius of the earth is 6378.137 km [148].

Due to the latitude being along the equator, the centripetal acceleration from Equation A.22 is

$$\begin{aligned} |\vec{\Omega}_e \times \vec{\Omega}_e \times \vec{R}_e| &= R_e\Omega_e^2 \cos(0)(\hat{i} \sin(0) + \hat{k} \cos(0)) \\ &= \hat{k}R_e\Omega_e^2 \end{aligned} \quad (\text{A.26})$$

Where $\vec{\Omega}_e$ is the angular velocity of the Earth's rotation. Since the Earth completes a full rotation approximately once every 24 hours, the angular velocity

can be determined as

$$\Omega_e = \frac{2\pi}{24 \times 3600} = 7.272205 \times 10^{-5} \text{rads}^{-1} \quad (\text{A.27})$$

Therefore the centripetal acceleration due to the angular velocity of the earth is

$$|\vec{\Omega}_e \times \vec{\Omega}_e \times \vec{R}_e| = (7.272205 \times 10^{-5})^2 \times 6378.137 \times 10^3 = 0.0337 \text{ ms}^{-1} \quad (\text{A.28})$$

From Equation A.24 Coriolis effect can be calculated, where $\lambda = 0$. Therefore

$$\begin{aligned} 2\Omega_e \times V_0^e &= 2(\hat{i} \Omega \sin \lambda V_{0,y} - \hat{j}(\Omega_e \cos \lambda V_{0,z} + \Omega_e \sin \lambda V_{0,x}) + \hat{k} \Omega_e \cos \lambda V_{0,y}) \\ &= 2(\hat{i} \Omega \sin(0) V_{0,y} - \hat{j}(\Omega_e \cos(0) V_{0,z} + \Omega_e \sin(0) V_{0,x}) + \hat{k} \Omega_e \cos(0) V_{0,y}) \\ &= -\hat{j}\Omega_e V_{0,z} + \hat{k}\Omega_e V_{0,y} \end{aligned} \quad (\text{A.29})$$

It can be assumed that the aircraft is locations in steady-level flight and the magnitude of the velocity vectors is the flight speed, i.e. $2\vec{\Omega}_e \times \vec{V}_0^e \approx 2\Omega_e V_0$.

Therefore

$$\begin{aligned} \vec{\Omega}_e \times \vec{V}_0^e &\approx 2\Omega_e V_0 \\ &\approx 2 \times 7.272205 \times 10^{-5} \times 204 \text{ ms}^{-1} = 0.02967 \text{ ms}^{-2} \end{aligned} \quad (\text{A.30})$$

Based on the information obtained, it was determined that the Earth's rotation will not provide enough of a magnitude, with, both the Coriolis and centripetal accelerations both being small quantities relative to the expected velocity profile of the aircraft, and thus can be neglected. As a result the equations of motion with respect to a non-rotating inertial frame were formulated.

However since the Coriolis effect is proportional to velocity, should the airspeed increase significantly, it may be required for consideration. However due to the scope of the research, this was not expected to be the case. Therefore Equation A.20 is simplified down into the form of Newton's second law of motion.

$$\begin{aligned} F &= m \left(\frac{d^2 \vec{R}_i}{dt^2} \right)_i = m [\vec{\Omega}_e \times \vec{\Omega}_e \times \vec{R}_e + (\vec{a}_0)_e + 2\vec{\Omega}_e \times \vec{V}_0^e] \\ &= m(\vec{a}_0)_e \end{aligned} \quad (\text{A.31})$$

A.2.3 Translational Motion

During Appendix A.2.1, a velocity and acceleration analysis was conducted. This analysis enabled Newton's second law of motion to be described by a single acceleration vector $(\vec{a}_0)$ with components $\hat{i}$, $\hat{j}$ and $\hat{k}$.

Consider the following axis system as depicted in Figure A.3. An aircraft whose body-axis (x_b, y_b, z_b) lies coincident with the vehicles centre of gravity. The body-axes are observed with respect to a fixed non-rotating inertial frame (x_i, y_i, z_i) .

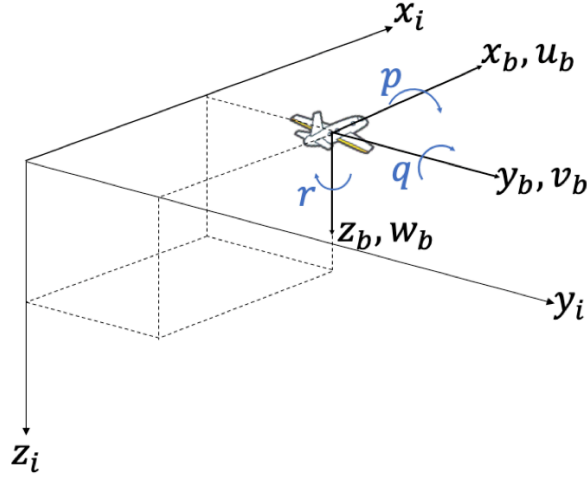


FIGURE A.3 Translational motion schematic between inertial and body-axes

Since Newton's second law is only valid when the acceleration is being observed from an inertial frame, the translational acceleration in the body-axis $\overrightarrow{(a_0)_i}$, can be observed from the inertial frame using moving axis theorem [91]. Therefore

$$F = m(\overrightarrow{a_0})_i = m \left(\frac{d\overrightarrow{V_0}}{dt} \right)_i = m \left(\frac{d\overrightarrow{V_0}}{dt} \right)_b + \overrightarrow{\omega_{i,b}^b} \times \overrightarrow{V_0} \quad (\text{A.32})$$

By expressing the individual components from Equation A.32 in their individual cartesian unit vectors $\hat{i}$, $\hat{j}$ and $\hat{k}$

$$\overrightarrow{\omega_{i,b}^b} = p\hat{i} + q\hat{j} + r\hat{k} \quad (\text{A.33})$$

$$\overrightarrow{V_0} = u\hat{i} + v\hat{j} + w\hat{k} \quad (\text{A.34})$$

$$\left(\frac{d\overrightarrow{V_0}}{dt} \right) = \dot{u}\hat{i} + \dot{v}\hat{j} + \dot{w}\hat{k} \quad (\text{A.35})$$

The vector cross product (Equation A.23) can be used on terms $\overrightarrow{\omega_{i,b}^b}$ and $\overrightarrow{V_0}$

$$\overrightarrow{\omega_{i,b}^b} \times \overrightarrow{V_0} = (qw - rv)\hat{i} - (pw - ru)\hat{j} + (pv - qu)\hat{k} \quad (\text{A.36})$$

Combining Equation A.35 and Equation A.36 into Equation A.32

$$F = (\dot{u} + qw - rv)\hat{i} + (\dot{v} + ru - pw)\hat{j} + (\dot{w} + pv - qu)\hat{k} \quad (\text{A.37})$$

Writing Equation A.37 in the form of Equation A.25

$$\begin{aligned} F_{x,i} &= (\dot{u} + qw - rv) \\ F_{y,i} &= (\dot{v} + ru - pw) \\ F_{z,i} &= (\dot{w} + pv - qu) \end{aligned} \quad (\text{A.38})$$

A.2.4 Angular Motion

Equation A.2 describes the application of Newton's second law to the rate of change of angular momentum of the vehicle to determine its angular acceleration. Therefore, considering the angular momentum vector (H) acting on a rigid vehicle, the net torque (T_{net}) acting on the aircraft's centre of gravity is defined as [18].

$$T_{net} = \frac{d}{dt}(H) \quad (\text{A.39})$$

When considering angular momentum, it is important to remember that angular momentum does not have a definitive value, meaning that to determine the angular momentum of an object or particle, requires an observational point known as the origin. For radial motion, this observation point is often referred to as the radius of curvature, but really is just describing a perpendicular position vector between itself and the objects angular velocity, $\vec{\omega}_{i,b}^i$. When considering a particle of mass δm , the translational momentum can be described as

$$\delta h = (\vec{\omega}_{i,b}^i \times \vec{r}_0) \delta m \quad (\text{A.40})$$

Which, is acting at a distance perpendicular to the observation point, therefore

$$\frac{dH}{dt} = (\vec{r}_b \times \vec{\omega}_{i,b}^i \times \vec{r}_b) \delta m \quad (\text{A.41})$$

Using the triple vector cross product rule (Equation A.21) we can solve for $\vec{r}_b \times \vec{\omega}_b \times \vec{r}_b$. Where $\vec{\omega}_b$ has unit vector components of (p, q, r) and $\vec{r}_b$ has unit vector components of (x, y, z) . Therefore,

$$\begin{aligned} H &= \vec{\omega}_b (\vec{r}_b \cdot \vec{r}_b) - \vec{r}_b (\vec{\omega}_b \cdot \vec{r}_b) \\ H &= (p\hat{i} + q\hat{j} + r\hat{k})(x^2\hat{i} + y^2\hat{j} + z^2\hat{k}) - (x\hat{i} + y\hat{j} + z\hat{k})(px\hat{i} + qy\hat{j} + rz\hat{k}) \end{aligned} \quad (\text{A.42})$$

Expanding and applying into matrix form.

$$H = \begin{bmatrix} p(x^2 + y^2 + z^2) - (px^2 + qxy + rzx) \\ q(x^2 + y^2 + z^2) - (pxy + qy^2 + rzy) \\ r(x^2 + y^2 + z^2) - (pxz + qxy + rz^2) \end{bmatrix} = \begin{bmatrix} p(y^2 + z^2) - (qxy + rzx) \\ q(x^2 + z^2) - (pxy + rzy) \\ r(x^2 + y^2) - (pxz + qxy) \end{bmatrix} \quad (\text{A.43})$$

Multiplying both sides by the particle mass, δ_m ,

$$\delta_m H = \begin{bmatrix} p(y^2 + z^2)\delta_m - (qxy + rzx)\delta_m \\ q(x^2 + z^2)\delta_m - (pxy + rzy)\delta_m \\ r(x^2 + y^2)\delta_m - (pxz + qxy)\delta_m \end{bmatrix} \quad (\text{A.44})$$

Integrating the particle mass δ_m over the entire aircraft,

$$H = \begin{bmatrix} p \int (y^2 + z^2) dm - q \int xy dm + r \int zx dm \\ q \int (x^2 + z^2) dm - p \int xy dm + r \int zy dm \\ r \int (x^2 + y^2) dm - p \int xz dm + q \int xy dm \end{bmatrix} \quad (\text{A.45})$$

The integrals in Equation A.45 can be defined as the moments and products of inertial.

$$\begin{aligned} I_{xx} &= \int (y^2 + z^2) dm & I_{yy} &= \int (x^2 + z^2) dm & I_{zz} &= \int (x^2 + y^2) dm \\ I_{xy} &= \int xy dm & I_{xz} &= \int xz dm & I_{yz} &= \int yz dm \\ I_{yx} &= \int yx dm & I_{zx} &= \int zx dm & I_{zy} &= \int zy dm \end{aligned}$$

Where, $I_{xy} = I_{yx}$, $I_{xz} = I_{zx}$ and $I_{yz} = I_{zy}$. In practice the moments and products of inertia would change as the aircraft uses fuel or releases payloads. However it was assumed that the aircraft was operating under a constant mass, the moments and products do not change during dynamic simulation. Substituting the moments and products of inertia into Equation A.45,

$$H = \begin{bmatrix} pI_{xx} - qI_{xy} + rI_{xz} \\ qI_{yy} - pI_{xy} + rI_{zy} \\ rI_{zz} - pI_{xz} + qI_{yz} \end{bmatrix} = \begin{bmatrix} H_x \\ H_y \\ H_z \end{bmatrix} \quad (\text{A.46})$$

Since the body-axis system is coincident with the aircraft's centre of gravity, for which, the x - axis is located along the aircraft's line of symmetry, $I_{xy} = I_{yx} = 0$ and $I_{yz} = I_{zy} = 0$. Therefore, Equation A.46 can be written as

$$\begin{aligned} H_x &= pI_{xx} + rI_{xz} \\ H_y &= qI_{yy} \\ H_z &= rI_{zz} - pI_{xz} \end{aligned} \quad (\text{A.47})$$

Having established a set of equations which govern the angular momentum vector of the aircraft, differentiation of the vector $\vec{H}$ from Equation A.2 needs to be performed. To perform vector differentiation angular momentum will be written as a function of its unit vectors $1_H = (\hat{i}, \hat{j}, \hat{k})$ [92]. Therefore,

$$M = \frac{dH}{dt} = \frac{d}{dt}(1_H \cdot H) \quad (\text{A.48})$$

Differentiation using the chain rule, $(d/dt) = (du/dt)v + u(dv/dt)$

$$\frac{dH}{dt} = \frac{d1_H}{dt}H + \frac{dH}{dt}1_H \quad (\text{A.49})$$

Where $(d1_H/dt)H$ is the rate of change of momentum due to the angular momentum. Therefore

$$\frac{d1_H}{dt}H = \vec{w}_b \times \vec{H} \quad (\text{A.50})$$

Combining Equations A.49 and A.50

$$\frac{dH}{dt} = \frac{dH}{dt}1_H + \vec{w}_b \times \vec{H} \quad (\text{A.51})$$

Using the vector cross product rule, (Equation A.23), resolve $\vec{w}_b \times \vec{H}$, where $\vec{w}_b = (p, q, r)$ and $\vec{H} = (H_x, H_y, H_z)$

$$\begin{aligned} \vec{w}_b \times \vec{H} &= (qH_z - rH_y)\hat{i} - (pH_z - rH_x)\hat{j} + (pH_y - qH_x)\hat{k} \\ &= (qH_z - rH_y)\hat{i}(rH_x - pH_z)\hat{j}(pH_y - qH_x)\hat{k} \end{aligned} \quad (\text{A.52})$$

Using the the dot notation,

$$\frac{dH}{dt}1_H = \frac{d}{dt}H = \begin{bmatrix} \frac{d}{dt}H_x \\ \frac{d}{dt}H_y \\ \frac{d}{dt}H_z \end{bmatrix} = \begin{bmatrix} \dot{H}_x \\ \dot{H}_y \\ \dot{H}_z \end{bmatrix} \quad (\text{A.53})$$

Therefore from Equation A.47

$$\begin{aligned} \dot{H}_x &= \dot{p}I_{xx} + \dot{r}I_{xz} \\ \dot{H}_y &= \dot{q}I_{yy} \\ \dot{H}_z &= \dot{r}I_{zz} - \dot{p}I_{xz} \end{aligned} \quad (\text{A.54})$$

Combining Equations A.52 and A.54

$$\begin{aligned}\Sigma M = \frac{dH}{dt} &= \begin{bmatrix} \dot{H}_x + (qH_z - rH_y) \\ \dot{H}_y + (rH_x - pH_z) \\ \dot{H}_z + (pH_y - qH_x) \end{bmatrix} \\ &= \begin{bmatrix} \dot{p}I_{xx} + \dot{r}I_{xz} + q(rI_{zz} - pI_{xz}) - r(qI_{yy}) \\ \dot{q}I_{yy} + r(pI_{xx} + rI_{xz}) - p(rI_{zz} - pI_{xz}) \\ \dot{r}I_{zz} - \dot{p}I_{xz} + p(qI_{yy}) - q(pI_{xx} + rI_{xz}) \end{bmatrix}\end{aligned}\quad (\text{A.55})$$

The vector for the summation of moments has components $(\bar{L}, M, N)$. So expanding Equation A.55

$$\begin{aligned}\bar{L} &= \dot{p}I_{xx} + \dot{r}I_{xz} + qrI_{zz} - pqI_{xz} - rqI_{yy} \\ &= \dot{p}I_{xx} + \dot{r}I_{xz} + qr(I_{zz} - I_{yy}) - pqI_{xz}\end{aligned}\quad (\text{A.56})$$

$$\begin{aligned}M &= \dot{q}I_{yy} + prI_{xx} + r^2I_{xz} - prI_{zz} - p^2I_{xz} \\ &= \dot{q}I_{yy} + I_{xz}(r^2 - p^2) + pr(I_{xx} - I_{zz})\end{aligned}\quad (\text{A.57})$$

$$\begin{aligned}N &= \dot{r}I_{zz} - \dot{p}I_{xz} + pqI_{yy} - pqI_{xx} + qrI_{xz} \\ &= \dot{r}I_{zz} - \dot{p}I_{xz} + pq(I_{yy} - I_{xx}) + qrI_{xz}\end{aligned}\quad (\text{A.58})$$

A.2.5 Forces and Moments

Having established a series of six degrees of freedom equations representing a nonlinear, rigid and symmetrical aircraft with constant and uniform mass distribution, we now need to define the forces and moments acting on the aircraft. Cook theorises that the disturbing forces and moments can be categorised as aerodynamic effects, gravitational effects, movement of the aerodynamic control surfaces, effects due to power generation and effects due to atmospheric disturbances Cook. For the scope of this research, the forces applied to the equations of motion will focus on aerodynamics effects (η_a), gravitational effects (η_g), movements of control surfaces (η_c) and power generation (η_p). Where η specifies the axis in which the force or moment is acting. It also important to note that the data obtained for the aircraft does not quantify a moment or force that is induced due to the movement of a control surface, it only quantifies the aerodynamic effect of that movement. Therefore from Equations A.38, A.56, A.57,

A.58,

$$\begin{aligned}
F_{x,i} &= x_a + x_g + x_p = (\dot{u} + qw - rv) \\
F_{y,i} &= y_a + y_g + y_p = (\dot{v} + ru - pw) \\
F_{z,i} &= z_a + z_g + z_p = \dot{w} + pv - qu \\
\bar{L} &= L_a + L_g + L_p = \dot{p}I_{xx} + \dot{r}I_{xz} + qr(I_{zz} - I_{yy}) - pqI_{xz} \\
M &= M_a + M_g + M_p = \dot{q}I_{yy} + I_{xz}(r^2 - p^2) + pr(I_{xx} - I_{zz}) \\
N &= N_a + N_g + N_p = \dot{r}I_{zz} - \dot{p}I_{xz} + pq(I_{yy} - I_{xx}) + qrI_{xz}
\end{aligned} \tag{A.59}$$

The gravitational forces are exhibited parallel to the z_b axis when $\theta = 0^\circ$, and therefore in the Euler axis translation (Equation 3.7) transformation about z_b can be neglected by setting ψ to zero. Therefore

$$\begin{bmatrix} F_{g,x} \\ F_{g,y} \\ F_{g,z} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & \sin\phi \\ 0 & -\sin\phi & \cos\phi \end{bmatrix} \begin{bmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ mg \end{bmatrix} \tag{A.60}$$

$$\begin{bmatrix} F_{g,x} \\ F_{g,y} \\ F_{g,z} \end{bmatrix} = \begin{bmatrix} \cos\theta & 0 & -\sin\theta \\ \sin\phi \sin\theta & \cos\phi & \sin\phi \cos\theta \\ \cos\phi \sin\theta & -\sin\phi & \cos\theta \cos\phi \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix} = mg \begin{bmatrix} -\sin\theta \\ \sin\phi \cos\theta \\ \cos\theta \cos\phi \end{bmatrix} \tag{A.61}$$

Modelling the thrust developed from an engine into the equations of motion can be simplified by theorising only a linear quantity parallel with the x axis. As the body-axis system is fixed and rigid, as the aircraft rotates about an axis, the x, y, z are offset by that angular movement. With this configuration thrust will solely exist in the x axis. Therefore

$$\begin{bmatrix} F_{T,x} \\ F_{T,y} \\ F_{T,z} \end{bmatrix} = \begin{bmatrix} T \\ 0 \\ 0 \end{bmatrix} \tag{A.62}$$

However for a high speed fighter aircraft the operating envelope may allow for high angle of attack and high rate of angle of attack manoeuvres where conditions could cause the aerodynamics to stall. This allows the exhaust gases to be utilised through thrust vectoring to deflect and redirect flow inducing a moment onto the aircraft. This will allow the aircraft to produce control using thrust generated [20]. For simplicity, it will be presumed that the nozzle is concentric and

symmetric about the x axis, and induces a constant angular momentum of 216.6 Nm. Therefore it can be assumed in the Euler interpretation that only velocities are present as components of yaw and pitch, thus Equation 3.4 and Equation 3.5 will be incorporated into the analysis. A schematic can be seen in Figure A.4, where $\psi = \delta_{yTv}$ and $\theta = \delta_{pTv}$.

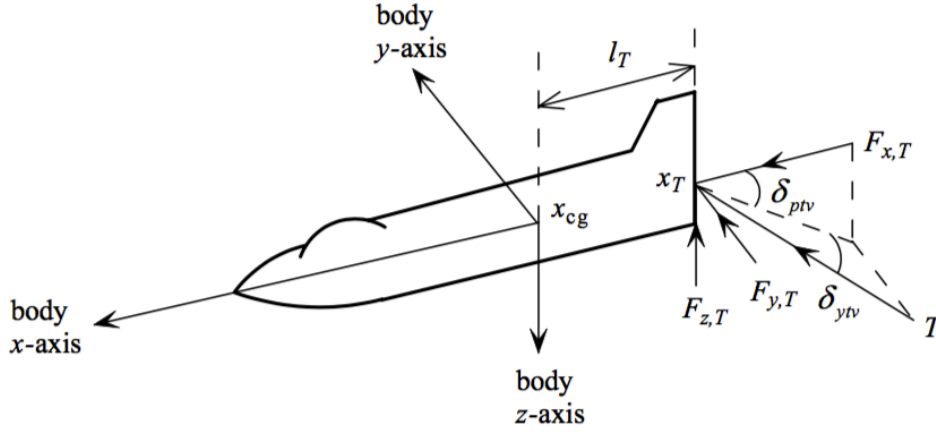


FIGURE A.4 Thrust vectoring schematic for the F-16 aircraft [20]

$$\begin{bmatrix} F_{T,x} \\ F_{T,y} \\ F_{T,z} \end{bmatrix} = \begin{bmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{bmatrix} \begin{bmatrix} \cos\psi & \sin\psi & 0 \\ -\sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} T \\ 0 \\ 0 \end{bmatrix} \quad (\text{A.63})$$

$$\begin{bmatrix} F_{T,x} \\ F_{T,y} \\ F_{T,z} \end{bmatrix} = \begin{bmatrix} \cos\theta \cos\psi & \cos\theta \sin\psi & -\sin\psi \\ -\sin\psi & \cos\psi & 0 \\ \sin\theta \cos\psi & \sin\theta \sin\psi & \cos\theta \end{bmatrix} \begin{bmatrix} T \\ 0 \\ 0 \end{bmatrix} = T \begin{bmatrix} \cos\theta \cos\psi \\ -\sin\psi \\ \sin\theta \cos\psi \end{bmatrix} \quad (\text{A.64})$$

If the aircraft has been modelled with respect to a wind-axis moving axis system, we can perform wind axes transformation, where $\alpha_T = \theta_T$, $\beta_T = -\psi_T$. Therefore

$$\begin{bmatrix} F_{T,x} \\ F_{T,y} \\ F_{T,z} \end{bmatrix} = T \begin{bmatrix} \cos\theta \cos\psi \\ -\sin\psi \\ \sin\theta \cos\psi \end{bmatrix} = T \begin{bmatrix} \cos\alpha_T \cos\beta_T \\ \sin\beta_T \\ -\sin\alpha_T \cos\beta_T \end{bmatrix} \quad (\text{A.65})$$

The aerodynamic forces are extremely complex, and for a non-linear computation require their own section of discussion within the thesis. We will therefore denote each of the aerodynamic forces acting in each of the axis as a being

a function of an aerodynamic force vector C_A , dynamic pressure ($\bar{q}$) and plan form area S .

$$\begin{bmatrix} F_{A,x} \\ F_{A,y} \\ F_{A,z} \end{bmatrix} = \bar{q}S \begin{bmatrix} C_{A,x} \\ C_{A,y} \\ C_{A,z} \end{bmatrix} \quad (\text{A.66})$$

Compiling the forces acting on the airframe

$$\begin{aligned} F_{x,i} &= \bar{q}SC_{A,x} - mg\sin\theta + T = m(\dot{u} + qw - rv) \\ F_{y,i} &= \bar{q}SC_{A,y} + mg\sin\phi\cos\theta = m(\dot{v} + ru - pw) \\ F_{z,i} &= \bar{q}SC_{A,z} + mg\cos\theta\sin\phi = m\dot{w} + pv - qu \end{aligned} \quad (\text{A.67})$$

And resolving for the acceleration, and noting that due to the right-handed orthogonal axis system,

$$\begin{aligned} \dot{u} &= rv - qw - g\sin\theta + \frac{1}{m}(\bar{q}SC_{A,x} + T) \\ \dot{v} &= pw - ru - g\sin\phi\cos\theta + \frac{1}{m}(\bar{q}SC_{A,y}) \\ \dot{w} &= qu - pv - g\cos\theta\sin\phi + \frac{1}{m}(\bar{q}SC_{A,z}) \end{aligned} \quad (\text{A.68})$$

Since the forces have now been categorised we can determine the moments induced by each of the forces. Throughout Appendix A it was assumed that the body-axis system, which moves with the aircraft, was coincident with the center of gravity for the aircraft. Under this assumption the distance that the gravity is acting from the center of gravity, l_g is zero, and thus:

$$\begin{bmatrix} L_g \\ M_g \\ N_g \end{bmatrix} = \begin{bmatrix} F_{g,x}l_g \\ F_{g,y}l_g \\ F_{g,z}l_g \end{bmatrix} = l_g g \begin{bmatrix} -\sin\theta \\ \sin\phi \cos\theta \\ \cos\theta \cos\phi \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \quad (\text{A.69})$$

Although I have determined the force components to model a rudimentary thrust vectoring configuration, which is only operating in the longitudinal axis to for high-angle of attack manoeuvres, I have not implemented this into Equations A.68. This significantly reduces the calculations for the moments induced due to power generation, as the force acting on the body through power generation is restricted to only one axis. Using the free-body diagrams in Figure A.5, the

force components about each moving axis can be collated. Therefore

$$\begin{bmatrix} L_T \\ M_T \\ N_T \end{bmatrix} = \begin{bmatrix} l_{T,zx}F_{T,y} + l_{T,yx}F_{T,z} \\ l_{T,zy}F_{T,x} + l_{T,xy}F_{T,z} \\ l_{T,yz}F_{T,x} - l_{T,xz}F_{T,y} \end{bmatrix} \quad (\text{A.70})$$

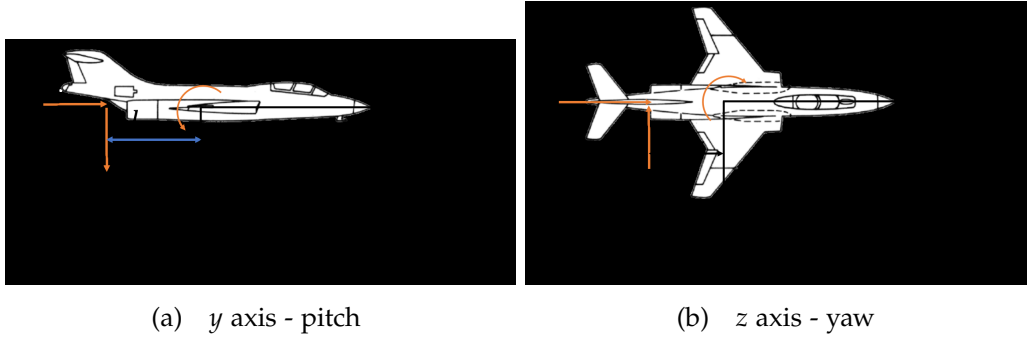


FIGURE A.5 Free-body diagram for moments induced through thrust vectoring (Images Modified from: [149])

where

- $l_{T,zx} = 0$ Distance $F_{T,y}$ acts from the c_g relative to the rotation of the x axis
- $l_{T,yx} = 0$ Distance $F_{T,z}$ acts from the c_g relative to the rotation of the x axis
- $l_{T,zy} = 0$ Distance $F_{T,x}$ acts from the c_g relative to the rotation of the y axis
- $l_{T,xy} \neq 0$ Distance $F_{T,z}$ acts from the c_g relative to the rotation of the y axis
- $l_{T,yz} = 0$ Distance $F_{T,x}$ acts from the c_g relative to the rotation of the z axis
- $l_{T,xz} \neq 0$ Distance $F_{T,y}$ acts from the c_g relative to the rotation of the z axis

However it was previously determined that $F_{T,y}$ and $F_{T,z}$ are zero, and since $F_{T,x}$ acts parallel and coincident to the x axis, the distance $l_{T,xz}$, $l_{T,xy}$ are zero, and therefore

$$\begin{bmatrix} L_T \\ M_T \\ N_T \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \quad (\text{A.71})$$

As discussed previously the aerodynamic forces are complex and can be attributed to numerous functions, still to be discussed. It is assumed that the moments induced due to aerodynamic forces in the x and z axes will be moments

about the wingspan of the aircraft (b). Whereas the moment induced about the y axis will occur at the quarter chord for each wing ($\bar{c}$). Therefore:

$$\begin{bmatrix} L_{a,x} \\ M_{a,y} \\ N_{a,z} \end{bmatrix} = \bar{q}S \begin{bmatrix} bC_{a,L} \\ \bar{c}C_{a,M} \\ bC_{a,N} \end{bmatrix} \quad (\text{A.72})$$

Collating the moments

$$\begin{aligned} \bar{L} &= \bar{q}SbC_{A,L} \\ M &= \bar{q}S\bar{c}C_{A,M} + h_e r \\ N &= \bar{q}SbC_{A,N} + h_e q \end{aligned} \quad (\text{A.73})$$

Therefore from Equation A.59

$$\begin{aligned} \bar{L} &= \bar{q}SbC_{A,L} \\ &= \dot{p}I_{xx} + \dot{r}I_{xz} + qr(I_{zz} - I_{yy}) - pqI_{xz} \end{aligned} \quad (\text{A.74})$$

$$\begin{aligned} M &= \bar{q}S\bar{c}C_{A,M} + h_e r \\ &= \dot{q}I_{yy} + I_{xz}(r^2 - p^2) + pr(I_{xx} - I_{zz}) \end{aligned} \quad (\text{A.75})$$

$$\begin{aligned} N &= \bar{q}SbC_{A,N} + h_e q \\ &= \dot{r}I_{zz} - \dot{p}I_{xz} + pq(I_{yy} - I_{xx}) + qrI_{xz} \end{aligned} \quad (\text{A.76})$$

Resolving for the accelerations, p , q and r

$$\dot{p} = \frac{I_y - I_z}{I_x}qr + \frac{I_{xz}}{I_x}(\dot{r} + pq) + \frac{\bar{q}Sb}{I_x}C_{A,L} \quad (\text{A.77})$$

$$\dot{q} = \frac{I_z - I_x}{I_y}pr + \frac{I_{xz}}{I_y}(r^2 - p^2) + \frac{\bar{q}S\bar{c}}{I_y}C_{A,M} - h_e r \quad (\text{A.78})$$

$$\dot{r} = \frac{I_x - I_y}{I_z}pq + \frac{I_{xz}}{I_z}(\dot{p} - qr) + \frac{\bar{q}Sb}{I_z}C_{A,N} + h_e q \quad (\text{A.79})$$

Note that both $\dot{p}$ and $\dot{r}$ are dynamically coupled, meaning that each term is present as component of the latter. This is one of the reasons why many studies perform linearisation of the equations of motion, particularly in analytical and control studies. These internal acceleration components can be solved by attaining a solution to a set of simultaneous equations. For simplicity moments will be denoted as $\bar{L}$, M and N throughout the final derivation. Therefore Equations

A.77, A.78 and A.79 becomes

$$\dot{p} = \frac{I_y - I_z}{I_x}qr + \frac{I_{xz}}{I_x}(\dot{r} + pq) + \frac{1}{I_x}\bar{L} \quad (\text{A.80})$$

$$\dot{q} = \frac{I_z - I_x}{I_y}pr + \frac{I_{xz}}{I_y}(r^2 - p^2) + \frac{1}{I_y}M - h_er \quad (\text{A.81})$$

$$\dot{r} = \frac{I_x - I_y}{I_z}pq + \frac{I_{xz}}{I_z}(\dot{p} - qr) + \frac{1}{I_z}N + h_eq \quad (\text{A.82})$$

To resolve the angular accelerations about the x axis ($\dot{p}$), Equations A.80 and A.82 are compiled into a set of simultaneous equations, both of which are equal to $\dot{r}$. The equations can then be solved, allowing a solution to be obtained for $\dot{p}$. Therefore, after rearranging Equation A.80 to make $\dot{r}$ the subject,

$$\dot{r} = \frac{I_x}{I_{xz}}\dot{p} - \frac{I_y - I_z}{I_{xz}}qr - pq + \frac{1}{I_{xz}}\bar{L} \quad (\text{A.83})$$

Combining Equations A.83 and A.82, and expanding the brackets

$$\begin{aligned} \frac{I_x}{I_{xz}}\dot{p} - \frac{I_y - I_z}{I_{xz}}qr - pq + \frac{1}{I_{xz}}\bar{L} &= \frac{I_x - I_y}{I_z}pq + \frac{I_{xz}}{I_z}(\dot{p} - qr) + \frac{1}{I_z}\bar{N} + h_eq \\ \frac{I_x}{I_{xz}}\dot{p} - \frac{I_y - I_z}{I_{xz}}qr - pq + \frac{1}{I_{xz}}\bar{L} &= \frac{I_x - I_y}{I_z}pq + \frac{I_{xz}}{I_z}\dot{p} - \frac{I_{xz}}{I_z}qr + \frac{1}{I_z}\bar{M} + h_eq \end{aligned} \quad (\text{A.84})$$

Rearranging Equation A.84 so both $\dot{p}$ terms are present on the left hand side of the equation

$$\dot{p}\frac{I_x}{I_{xz}} - \dot{p}\frac{I_{xz}}{I_z} = pq\frac{I_x - I_y}{I_z} + pq + qr\frac{I_y - I_z}{I_{xz}} - qr\frac{I_{xz}}{I_z} + \frac{1}{I_{xz}}\bar{L} + \frac{1}{I_z}\bar{N} + h_eq$$

Factorising both sides

$$\dot{p}\left(\frac{I_x}{I_{xz}} - \frac{I_{xz}}{I_z}\right) = pq\left(\frac{I_x - I_y}{I_z} + 1\right) + qr\left(\frac{I_y - I_z}{I_{xz}} - \frac{I_{xz}}{I_z}\right) + \frac{1}{I_{xz}}\bar{L} + \frac{1}{I_z}\bar{N} + h_eq$$

Developing a common denominator for $\left(\frac{I_x}{I_{xz}} - \frac{I_{xz}}{I_z}\right)$ and $\left(\frac{I_y - I_z}{I_{xz}} - \frac{I_{xz}}{I_z}\right)$

$$\dot{p}\left(\frac{I_x I_z}{I_{xz} I_z} - \frac{I_{xz} I_{xz}}{I_{xz} I_z}\right) = pq\left(\frac{I_x - I_y}{I_z} + 1\right) + qr\left(\frac{I_z(I_y - I_z)}{I_{xz} I_z} - \frac{I_{xz}^2}{I_{xz} I_z}\right) + \frac{1}{I_{xz}}\bar{L} + \frac{1}{I_z}\bar{N} + h_eq$$

$$\dot{p}\left(\frac{I_x I_z - I_{xz}^2}{I_{xz} I_z}\right) = pq\left(\frac{I_x - I_y}{I_z} + 1\right) + qr\left(\frac{I_z(I_y - I_z) - I_{xz}^2}{I_{xz} I_z}\right) + \frac{1}{I_{xz}}\bar{L} + \frac{1}{I_z}\bar{N} + h_eq$$

Dividing both sides by $\left(\frac{I_x I_z - I_{xz}^2}{I_{xz} I_z}\right)$

$$\begin{aligned} \dot{p} &= pq\left(\frac{I_{xz} I_z (I_x - I_y)}{I_z (I_x I_z - I_{xz}^2)} + \frac{I_{xz} I_z}{I_x I_z - I_{xz}^2}\right) + qr\left(\frac{I_{xz} I_z (I_z (I_y - I_z) - I_{xz}^2)}{(I_x I_z - I_{xz}^2) I_{xz} I_z}\right) + \bar{L} \frac{I_{xz} I_z}{I_{xz} (I_x I_z - I_{xz}^2)} \\ &\quad + \bar{N} \frac{I_{xz} I_z}{I_z (I_x I_z - I_{xz}^2)} + h_eq \frac{I_{xz} I_z}{I_x I_z - I_{xz}^2} \end{aligned}$$

Simplifying

$$\dot{p} = pq \left(\frac{I_{xz}(I_x - I_y + I_z)}{I_x I_z - I_{xz}^2} \right) + qr \left(\frac{I_z(I_y - I_z) - I_{xz}^2}{I_x I_z - I_{xz}^2} \right) + \bar{L} \frac{I_z}{I_x I_z - I_{xz}^2} + \bar{N} \frac{I_{xz}}{I_x I_z - I_{xz}^2} + h_e q \frac{I_{xz} I_z}{I_x I_z - I_{xz}^2}$$

Factorising the right-hand side by $\frac{1}{I_x I_z - I_{xz}^2}$

$$\dot{p} = \frac{1}{I_x I_z - I_{xz}^2} \left\{ qr(I_z(I_y - I_z) - I_{xz}^2) + pq(I_{xz}(I_z + I_x - I_y)) + I_z \bar{L} + I_{xz} \bar{N} + I_{xz} I_z h_e q \right\} \quad (\text{A.85})$$

To resolve the angular accelerations about the z axis ($\dot{r}$), Equations A.80 and A.82 are compiled into a set of simultaneous equations, both of which are equal to $\dot{p}$. The equations can then be solved, allowing a solution to be obtained for $\dot{r}$. Therefore, after rearranging Equation A.82 to make $\dot{r}$ the subject

$$\dot{p} = \dot{r} \frac{I_z}{I_{xz}} - pq \frac{I_x - I_y}{I_{xz}} + qr - \frac{1}{I_{xz}} \bar{N} - \frac{I_z}{I_{xz}} h_e q \quad (\text{A.86})$$

Therefore combining Equation A.86 and A.80

$$\begin{aligned} \frac{I_y - I_z}{I_x} qr + \frac{I_{xz}}{I_x} (\dot{r} + pq) + \frac{1}{I_x} \bar{L} &= \dot{r} \frac{I_z}{I_{xz}} - pq \frac{I_x - I_y}{I_{xz}} + qr - \frac{1}{I_{xz}} \bar{N} - \frac{I_z}{I_{xz}} h_e q \\ \frac{I_y - I_z}{I_x} qr + \frac{I_{xz}}{I_x} \dot{r} + \frac{I_{xz}}{I_x} pq + \frac{1}{I_x} \bar{L} &= \frac{I_z}{I_{xz}} \dot{r} - \frac{I_x - I_y}{I_{xz}} pq + qr - \frac{1}{I_{xz}} \bar{N} - \frac{I_z}{I_{xz}} h_e q \end{aligned} \quad (\text{A.87})$$

Rearranging Equation A.87 so only $\dot{r}$ is on the left hand side.

$$\frac{I_z}{I_{xz}} \dot{r} - \frac{I_{xz}}{I_x} \dot{r} = \frac{I_y - I_z}{I_x} qr + \frac{I_{xz}}{I_x} pq + \frac{1}{I_x} \bar{L} + \frac{I_x - I_y}{I_{xz}} pq - qr + \frac{1}{I_{xz}} \bar{N} + \frac{I_z}{I_{xz}} h_e q$$

Factorising both sides

$$\dot{r} \left(\frac{I_z}{I_{xz}} - \frac{I_{xz}}{I_x} \right) = qr \left(\frac{I_y - I_z}{I_x} - 1 \right) + pq \left(\frac{I_{xz}}{I_x} - \frac{I_x - I_y}{I_{xz}} \right) + \frac{1}{I_x} \bar{L} + \frac{1}{I_{xz}} \bar{N} + \frac{I_z}{I_{xz}} h_e q$$

Developing a common denominator for $\left(\frac{I_z}{I_{xz}} - \frac{I_{xz}}{I_x} \right)$ and $\left(\frac{I_{xz}}{I_x} - \frac{I_x - I_y}{I_{xz}} \right)$

$$\dot{r} \left(\frac{I_x I_z}{I_x I_{xz}} - \frac{I_{xz}^2}{I_x I_{xz}} \right) = qr \left(\frac{I_y - I_z}{I_x} - 1 \right) + pq \left(\frac{I_{xz}^2}{I_{xz} I_x} + \frac{I_x(I_x - I_y)}{I_{xz} I_x} \right) + \frac{1}{I_x} \bar{L} + \frac{1}{I_{xz}} \bar{N} + \frac{I_z}{I_{xz}} h_e q$$

$$\dot{r} \left(\frac{I_x I_z - I_{xz}^2}{I_x I_{xz}} \right) = qr \left(\frac{I_y - I_z}{I_x} - 1 \right) + pq \left(\frac{I_{xz}^2 + I_x^2 - I_x I_y}{I_{xz} I_x} \right) + \frac{1}{I_x} \bar{L} + \frac{1}{I_{xz}} \bar{N} + \frac{I_z}{I_{xz}} h_e q$$

Dividing both sides by $(\frac{I_x I_z - I_{xz}^2}{I_z I_{xz}})$

$$\begin{aligned} \dot{r} = qr \left(\frac{I_x I_{xz} (I_y - I_z)}{I_x I_z - I_{xz}^2 (I_x)} - \frac{I_x I_{xz}}{I_x I_z - I_{xz}^2} \right) + pq \left(\frac{I_x I_{xz} (I_{xz}^2 + I_x (I_x - I_y))}{I_x I_z - I_{xz}^2 (I_{xz} I_x)} \right) \\ + \frac{I_x I_{xz}}{I_x I_z - I_{xz}^2 (I_x)} \bar{L} + \frac{I_x I_{xz}}{I_x I_z - I_{xz}^2 (I_x)} \bar{N} + \frac{I_x I_z I_{xz}}{I_x I_z - I_{xz}^2 (I_{xz})} h_e q \end{aligned} \quad (\text{A.88})$$

Simplifying

$$\begin{aligned} \dot{r} = qr \left(\frac{I_{xz} (I_y - I_x - I_z)}{I_x I_z - I_{xz}^2} \right) + pq \left(\frac{I_{xz}^2 + I_x^2 - I_x I_y}{I_x I_z - I_{xz}^2} \right) + \frac{I_{xz}}{I_x I_z - I_{xz}^2} \bar{L} \\ + \frac{I_x}{I_x I_z - I_{xz}^2} \bar{N} + \frac{I_x I_z}{I_x I_z - I_{xz}^2} h_e q \end{aligned} \quad (\text{A.89})$$

Factorising the right-hand side by $\frac{1}{I_x I_z - I_{xz}^2}$

$$\begin{aligned} \dot{r} = \frac{1}{I_x I_z - I_{xz}^2} \left\{ [I_{xz} (I_y - I_x - I_z)] qr + \right. \\ \left. [I_{xz}^2 + I_x^2 - I_x I_y] pq + I_{xz} \bar{L} + I_x \bar{N} + I_x I_z h_e q \right\} \end{aligned} \quad (\text{A.90})$$

Compiling the moments of inertial into I variables, the equations for angular acceleration can be written as

$$\begin{aligned} I_1 &= I_z (I_y - I_z) - I_{xz}^2, & I_2 &= I_{xz} (I_z + I_x - I_y), & I_3 &= I_z, \\ I_4 &= I_{xz}, & I_5 &= I_{xz} I_z, & I_6 &= (I_z - I_x) / I_y, \\ I_7 &= I_{xz} / I_y, & I_8 &= 1 / I_y, & I_9 &= I_{xz} (I_y - I_x - I_z), \\ I_{10} &= I_{xz}^2 + I_x^2 - I_x I_y, & I_{11} &= I_{xz}, & I_{12} &= I_x, \\ I_{13} &= I_x I_z \end{aligned}$$

Therefore Equations A.85, A.81 and A.90 can be written as

$$\begin{aligned} \dot{p} &= qr I_1 \lambda + pq I_2 \lambda + \bar{L} I_3 \lambda + N I_4 \lambda + h_e q I_5 \lambda \\ \dot{q} &= pr I_6 + I_7 (r^2 - p^2) + I_8 M - h_e r \\ \dot{r} &= qr I_9 \lambda + pq I_{10} \lambda + \bar{L} I_{11} \lambda + N I_{12} \lambda + h_e q I_{13} \lambda \end{aligned} \quad (\text{A.91})$$

Where

$$\lambda = \frac{1}{I_x I_z - I_{xz}^2}$$

A.3 Engine Model Data

TABLE A.1 F-16 Turbofan and afterburner thrust data (N) [16]

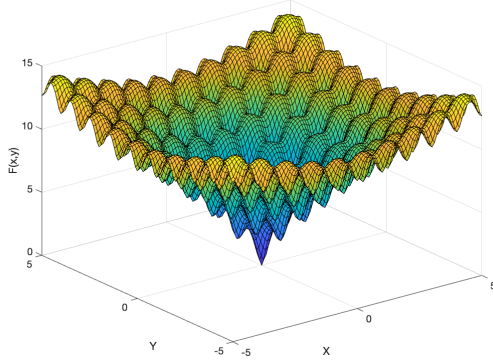
Mode	Mach	Altitude (m)					
		0	3048	6096	9144	12192	15240
Tmin	0.2	2824	1890	3069	4492	5916	7562
	0.4	267	111	1535	3358	5026	6783
	0.6	-4537	-3158	-1334	1557	4048	6069
	0.8	-12010	-8451	-5782	-1099	2669	4893
	1.0	-16013	-6227	-2647	-1521	-890	3114
Tmil	0.2	56041	40699	28080	17970	10987	6227
	0.4	56089	41420	29401	19082	11565	6939
	0.6	56223	43764	31536	20728	12632	7384
	0.8	55111	45263	34472	23663	14456	8585
	1.0	51953	43804	35806	27113	16902	10275
Tmax	0.2	95276	69834	49929	35273	19727	11565
	0.4	100970	74993	54488	35269	22240	12610
	0.6	107820	84112	61204	41300	25354	14300
	0.8	115959	93742	71057	49400	30512	17570
	1.0	128485	103727	81398	59977	38440	22494

Appendix B

Optimisation Test Functions

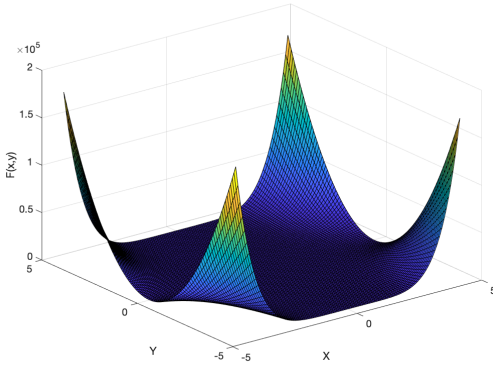
In Appendix B a series of optimisation test functions are described. These optimisation test functions are used during Chapter 4 for measuring the performance of the bat algorithm and genetic algorithm. Each test function is described mathematically as a three-dimensional surface, with two inputs corresponding to a single output.

TABLE B.1 Optimisation Functions

Ackley Function

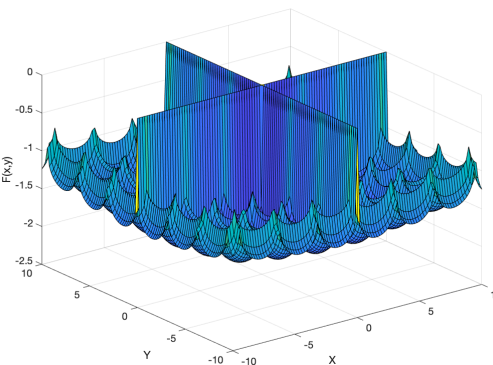
$$f(x,y) = -20 \exp \left[-0.2 \sqrt{0.5(x^2 + y^2)} \right] \\ - \exp \left[0.5(\cos 2\pi x + \cos 2\pi y) \right] \\ + \exp(1) + 20$$

Range : $-5 \leq x, y \leq 5$

Beale Function

$$f(x,y) = (1.5 - x + xy)^2 \\ + (2.25 - x + xy^2)^2 \\ + (2.625 - x + xy^3)^2$$

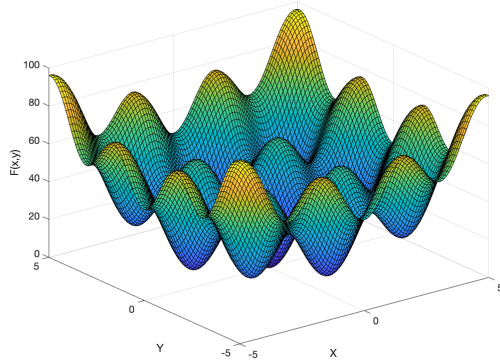
Range : $-4.5 \leq x, y \leq 4.5$

Cross-In-Tray Function

$$f(x,y) = -0.0001 \left[\sin x \sin y \right. \\ \left. \exp \left(\left| 100 - \frac{\sqrt{x^2 + y^2}}{\pi} \right| \right) + 1 \right]^{0.1}$$

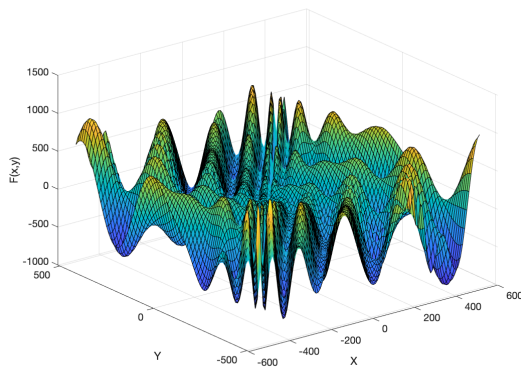
Range : $-10 \leq x, y \leq 10$

TABLE B.2 Optimisation Functions Continued

Egg Crate Function


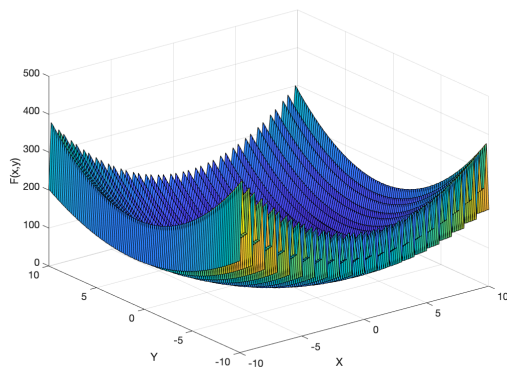
$$f(x,y) = x^2 + y^2 + 25[\sin^2(x) + \sin^2(y)]$$

$$\text{Range : } -5 \leq x, y \leq 5$$

Egg Holder Function


$$f(x,y) = -(y + 45) \sin \sqrt{\left| \frac{x}{2} + (y + 47) \right|} \\ - x \sin \sqrt{|x - (y + 47)|}$$

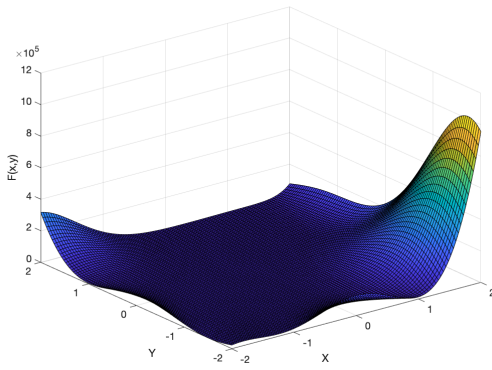
$$\text{Range : } -512 \leq x, y \leq 512$$

Levi Function


$$f(x,y) = \sin^2 3\pi x + (x - 1)^2 \\ \times (1 + \sin^2 3\pi y) + (y - 1)^2 \\ \times (1 + \sin^2 2\pi y)$$

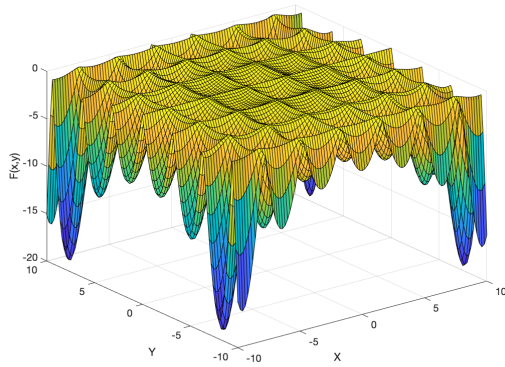
$$\text{Range : } -10 \leq x, y \leq 10$$

TABLE B.3 Optimisation Functions Continued

Goldstein-Price Function

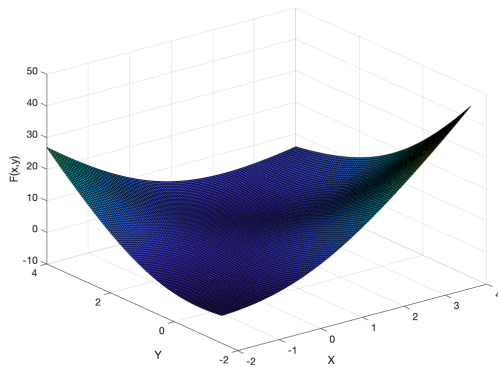
$$f(x,y) = [1 + (x + y + 1)^2 \times (19 - 14x + 3x^2 - 14y6xy + 3y^2)] \times [30 + (2x - 3y)^2 \times (18 - 32x + 12x^2 + 48y - 36xy + 27xy^2)]$$

Range : $-2 \leq x, y \leq 2$

Holder Table Function

$$f(x,y) = - \left| \sin x \cos y \exp \left(\left| 1 - \frac{\sqrt{x^2 + y^2}}{\pi} \right| \right) \right|$$

Range : $-10 \leq x, y \leq 10$

McCormick Function

$$f(x,y) = \sin(x + y) + (x - y)^2 - 1.5x + 2.5y + 1$$

Range : $-1.5 \leq x \leq 4$
 $-3.0 \leq y \leq 4$

Appendix C

Autopilot Controller Gains

In Appendix C, the controller gains for the all of the controllers optimised in Chapter 4 are tabulated. The displayed gains correspond to the angular rate controllers and the airspeed controller, allowing rotational and translational motion to be controlled.

C.1 Pitch Rate Controller Gains

C.1.1 High Precision Gain Surface

Controller gains for the pitch rate controllers high precision gain surface optimised in Section 4.4.2.

TABLE C.1 Pitch rate controllers high precision surface optimised gains.

Altitude (m)	Airspeed (m/s)								
	40	60	80	100	150	200	250	300	340
	Proportional Gains (k_p)								
0	6.68	5.48	4.47	4.07	3.51	3.42	5.22	9.52	11.13
2,000	7.12								
6,000		8.19							
8,000			6.41						
10,000				12.57	4.53	4.30	5.38	9.82	11.96
	Integral Gains (k_i)								
0	21.90	21.10	20.99	23.80	25.41	26.03	26.42	29.56	37.75
2,000	20.54								
6,000		28.73							
8,000			19.21						
10,000				15.22	23.76	28.12	27.19	29.94	40.81
	Derivative Gains (k_d)								
0	0.00	0.02	0.00	0.08	0.09	0.07	0.06	0.10	0.07
2,000	0.00								
6,000		0.00							
8,000			0.00						
10,000				0.00	0.06	0.10	0.03	0.07	0.17

C.1.2 Primary Positive Gain Surface

Controller gains for the pitch rate controllers primary positive gain surface optimised in Section 4.4.3.

TABLE C.2 Pitch rate controllers high precision surface optimised gains.

Altitude (m)	Airspeed (m/s)								
	40	60	80	100	150	200	250	300	340
	Proportional Gains (k_p)								
0	65.57	65.44	65.04	65.99	66.81	68.88	67.15	67.52	67.94
2,000	66.14	67.49	67.93	69.01	68.73	69.43	69.93	70.28	70.68
4,000		68.79	68.86	70.40	70.15	72.01	73.64	75.37	76.31
6,000		68.84	70.17	70.09	72.38	73.88	77.18	77.40	77.14
8,000			69.95	70.95	71.24	73.75	75.15	75.88	76.10
10,000				71.31	71.69	72.81	73.51	74.29	75.18
	Integral Gains (k_i)								
0	2.31	1.49	1.12	0.76	0.74	2.03	2.70	2.87	3.31
2,000	4.67	1.92	1.25	0.03	0.81	0.20	0.76	0.00	0.84
4,000		3.69	3.27	1.98	0.03	0.27	0.45	0.00	0.51
6,000		5.46	3.91	2.32	0.57	0.27	0.63	0.00	0.41
8,000			2.77	0.93	0.00	0.00	0.23	0.73	0.88
10,000				0.63	0.00	1.16	0.00	0.17	0.00
	Derivative Gains (k_d)								
0	7.76	8.75	8.50	7.81	6.73	5.44	4.51	3.76	3.04
2,000	7.42	8.22	9.77	8.91	7.41	5.98	4.88	4.24	3.74
4,000		7.58	7.39	9.34	7.29	6.71	5.74	4.11	4.14
6,000		5.00	6.96	8.00	6.80	6.23	5.54	5.06	4.55
8,000			6.29	7.04	7.56	5.98	5.51	5.11	4.71
10,000				6.23	7.06	6.36	5.04	5.04	5.01

C.1.3 Primary Negative Gain Surface

Controller gains for the pitch rate controllers primary positive gain surface optimised in Section 4.4.3.

TABLE C.3 Pitch rate controllers primary negative optimised gain surface.

Altitude (m)	Airspeed (m/s)								
	40	60	80	100	150	200	250	300	340
	Proportional Gains (k_p)								
0	23.19	67.29	46.29	79.11	91.22	95.17	109.04	111.48	93.89
2,000	29.51	68.16	48.00	79.28	91.56	96.35	100.79	112.31	96.08
4,000		69.26	49.49	80.17	94.46	96.67	110.83	114.03	95.08
6,000		66.96	50.75	80.44	95.99	99.96	111.63	115.85	96.85
8,000			53.21	82.37	97.41	101.42	112.25	116.47	98.02
10,000				82.67	97.91	101.77	115.12	117.63	99.73
	Integral Gains (k_i)								
0	0.96	0.35	0.00	0.71	1.42	1.61	4.33	2.38	0.06
2,000	0.00	2.40	0.01	0.13	1.03	0.77	1.32	3.37	1.19
4,000		2.94	0.04	0.02	0.02	1.77	2.68	3.29	0.19
6,000		0.08	4.20	0.01	0.01	0.54	0.00	0.22	1.96
8,000			3.97	0.52	0.09	0.44	3.00	0.17	1.01
10,000				6.19	0.44	0.84	1.88	0.00	0.35
	Derivative Gains (k_d)								
0	4.45	8.12	5.93	10.60	9.83	8.69	7.98	6.28	4.44
2,000	3.09	10.24	8.47	11.05	9.74	8.90	7.86	7.23	5.02
4,000		9.22	7.92	10.56	9.79	9.26	8.90	7.77	6.31
6,000		11.03	5.70	11.66	9.47	8.13	8.39	7.53	5.97
8,000			5.70	9.81	9.74	8.56	8.76	7.67	6.35
10,000				8.56	9.55	8.87	9.06	9.15	7.17

C.1.4 Neutral Surface

Controller gains for the pitch rate controllers neutral gain surface optimised in Section 4.4.3.

TABLE C.4 Pitch rate controllers primary negative optimised gain surface.

Altitude (m)	Airspeed (m/s)								
	40	60	80	100	150	200	250	300	340
	Proportional Gains (k_p)								
0	58.97	18.67	21.14	10.77	12.28	4.62	6.76	5.55	4.44
2,000	38.27								
6,000		38.27							
8,000			57.41						
10,000				36.57	12.45	7.61	7.88	6.75	6.14
	Integral Gains (k_i)								
0	0.00	0.17	0.34	0.00	0.00	0.00	0.00	0.00	0.00
2,000	0.00								
6,000		0.00							
8,000			0.00						
10,000				0.00	0.37	0.14	0.00	0.00	0.00
	Derivative Gains (k_d)								
0	4.45	8.12	5.93	10.60	9.83	8.69	7.98	6.28	4.44
0	14.62	3.51	2.90	1.45	1.01	0.46	0.43	0.22	0.17
2,000	8.87								
6,000		10.16							
8,000			12.04						
10,000				12.00	1.54	1.40	1.04	0.50	0.50

C.2 Roll Rate Controller Gains

C.2.1 High Precision Gain Surface

Controller gains for the roll rate controllers high precision gain surface optimised in Section 4.4.2.

TABLE C.5 Pitch rate controllers high precision surface optimised gains.

Altitude (m)	Airspeed (m/s)								
	40	60	80	100	150	200	250	300	340
	Proportional Gains (k_p)								
0	0.00	6.67	8.49	9.69	9.13	9.07	11.03	10.53	8.68
2,000	0.00								
6,000		12.54							
8,000			7.61						
10,000				7.98	6.68	7.80	10.27	10.07	8.43
	Integral Gains (k_i)								
0	0	17.32	20.78	22.02	17.86	13.66	14.74	9.61	12.53
2,000	0								
6,000		36.34							
8,000			20.66						
10,000				21.3	15.7	14.63	17.81	10.68	12.88
	Derivative Gains (k_d)								
0	0	0.03	0.08	0.08	0.07	0.06	0.06	0.05	0.05
2,000	0								
6,000		0.004							
8,000			0.01						
10,000				0.04	0.07	0.08	0.08	0.02	0.06

C.2.2 Primary Gain Surface

Controller gains for the roll rate controllers primary gain surface optimised in Section 4.4.4.

TABLE C.6 Roll rate controllers primary gain surface optimised gains

Altitude (m)	Airspeed (m/s)								
	40	60	80	100	150	200	250	300	340
	Proportional Gains (k_p)								
0	126.8	164.9	105.8	141.1	128.5	137.5	113.3	118.4	123.0
2,000	129.7	165.7	108.0	142.9	129.3	138.9	114.5	118.6	124.0
4,000	129.7	167.5	107.7	144.7	131.2	139.5	113.4	118.2	125.4
6,000	129.7	170.5	109.8	147.5	132.0	142.6	114.1	118.2	128.0
8,000	129.7	170.5	112.1	149.9	135.4	144.1	115.0	119.3	128.0
10,000	129.7	170.5	112.1	151.6	135.4	145.4	115.7	119.8	128.3
	Integral Gains (k_i)								
0	6.72	0.97	0.52	1.19	0.29	0.69	0.37	0.46	0.07
2,000	8.28	1.43	0.45	0.44	0.34	0.68	0.24	0.37	0
4,000	8.28	3.96	0.78	0	0.37	0.37	0.18	0.19	0.17
6,000	8.28	8.67	0.61	0.1	0.18	0.22	0.16	0.08	0.14
8,000	8.28	8.67	2.03	0.3	0.15	0.31	0.19	0.17	0.14
10,000	8.28	8.67	2.03	1.04	0	0.21	0.14	0.15	0.05
	Derivative Gains (k_d)								
0	7.07	9.68	6.33	6.02	1.82	1.81	3.14	4.03	3.22
2,000	8.42	9.87	4.53	7.52	1.38	2.91	1.47	3.59	3.95
4,000	8.42	12.48	4.96	8.4	3.79	2.79	2.01	3.71	4.64
6,000	8.42	10.61	6.35	8.39	5.35	2.38	2.65	3.86	5.12
8,000	8.42	10.61	8.59	6.84	8.82	5.41	3.31	3.99	5.12
10,000	8.42	10.61	8.59	8.26	9.93	5.42	3.55	4.03	5.45

C.2.3 Neutral Gain Surface

Controller gains for the roll rate controllers neutral gain surface optimised in Section 4.4.4.

TABLE C.7 Roll rate controllers neutral gain surface optimised gains.

Altitude (m)	Airspeed (m/s)								
	40	60	80	100	150	200	250	300	340
	Proportional Gains (k_p)								
0	43.00	20.53	20.8	32.54	27.83	33.04	29.87	32.41	37.11
2,000	44.66								
6,000		23.15							
8,000			23.18						
10,000				36.86	26.17	33.78	29.69	35.16	39.75
	Integral Gains (k_i)								
0	0	0	0.01	0.06	0.16	0.23	0.11	0	0.08
2,000	0.06								
6,000		0							
8,000			0.04						
10,000				0.05	0	0.005	0.03	0.41	0.52
	Derivative Gains (k_d)								
0	7.22	3.52	1.9	3.66	3.43	2.25	1.64	1.61	1.45
2,000	9.16								
6,000		8.52							
8,000			2.86						
10,000				5.45	2.35	3.99	3.36	3.46	3.5

C.3 Yaw Rate Controller Gains

C.3.1 High Precision Gain Surface

Controller gains for the yaw rate controllers high precision gain surface optimised in Section 4.4.2.

TABLE C.8 Yaw rate controllers high precision surface optimised gains.

Altitude (m)	Airspeed (m/s)								
	40	60	80	100	150	200	250	300	340
	Proportional Gains (k_p)								
0	96.83	97.62	96.8	94.84	93.64	92.14	92.63	94.46	95.74
2,000	98.83								
6,000		98.83							
8,000			98.83						
10,000				98.83	100.4	99.53	97.11	97.09	96.07
	Integral Gains (k_i)								
0	0.89	0.66	1.05	0.54	0.18	1.41	1.68	1.05	1.11
2,000	0.00								
6,000		0.00							
8,000			0.00						
10,000				0.00	0.12	0.01	0.00	0.00	0.00
	Derivative Gains (k_d)								
0	4.50	4.09	3.56	2.83	2.02	1.75	1.26	1.10	0.98
2,000	4.31								
6,000		4.31							
8,000			4.31						
10,000				4.31	3.38	2.70	2.16	1.83	1.61

C.3.2 Primary Gain Surface

Controller gains for the yaw rate controllers primary gain surface optimised in Section 4.4.5.

TABLE C.9 Yaw rate controllers primary gain surface optimised gains

Altitude (m)	Airspeed (m/s)								
	40	60	80	100	150	200	250	300	340
	Proportional Gains (k_p)								
0	154.9	330.6	160.1	119.7	155.3	160.6	173.0	151.2	142.1
2,000	160.8	329.6	167.3	137.7	155.1	164.2	170.1	152.5	145.4
4,000	160.8	325.5	168.6	139.0	157.2	166.8	172.3	152.5	146.9
6,000	160.8	329.7	175.2	162.7	159.5	169.1	175.4	157.2	148.6
8,000	160.8	329.7	177.9	163.9	163.6	169.7	176.4	158.6	150.9
10,000	160.8	329.7	177.9	164.0	165.8	169.1	177.6	160.0	148.4
	Integral Gains (k_i)								
0	0.001	3.95	8.01	7.48	9.92	5.80	12.17	7.87	5.43
2,000	0.001	2.42	6.81	7.24	9.54	12.93	13.93	13.90	11.45
4,000	0.001	1.14	5.61	6.61	8.57	10.37	14.16	16.52	16.36
6,000	0.001	0.65	3.08	5.58	6.70	10.74	12.76	16.86	17.63
8,000	0.001	0.65	1.56	4.30	6.13	12.85	10.74	16.21	19.19
10,000	0.001	0.65	1.56	2.12	8.36	7.09	12.60	10.52	18.47
	Derivative Gains (k_d)								
0	17.23	28.31	13.94	10.53	9.53	7.78	6.90	5.41	4.43
2,000	17.83	29.93	14.22	12.26	10.77	8.70	7.68	5.82	4.63
4,000	17.83	33.01	14.49	12.29	11.88	10.13	8.36	6.54	5.20
6,000	17.83	33.13	16.26	13.95	13.38	11.35	10.23	7.88	5.91
8,000	17.83	33.13	17.28	14.29	15.14	12.47	10.87	7.84	7.25
10,000	17.83	33.13	17.28	15.33	15.78	13.78	12.49	9.22	8.01

Appendix D

Classification Network - Dynamic Waypoint Testing

D.1 Mission 1

Mission 1 accounts for the first mission profile for the dynamic and cooperative waypoint flight. The formation for Mission 1 is shown below in Figure D.1.

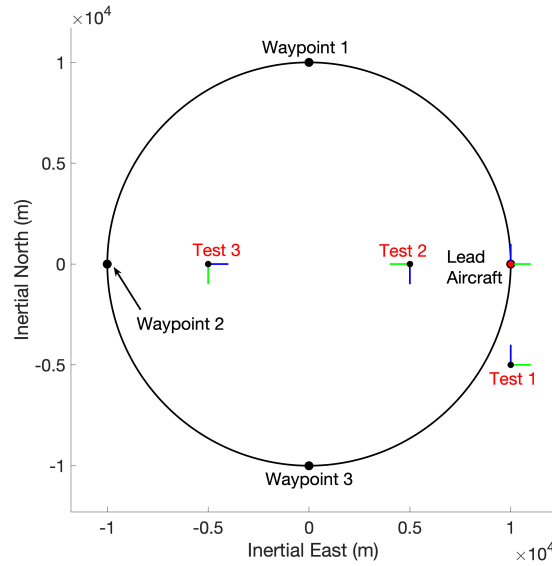


FIGURE D.1 Deep neural network (classification model) dynamic waypoint target Mission 1

D.1.1 Test 1

During Mission 1: Test 1, the autonomous aircraft was initialised with a position vector of $(-5000, 10000, 0)$ and with a heading angle of 0° with respect to the inertial frame.

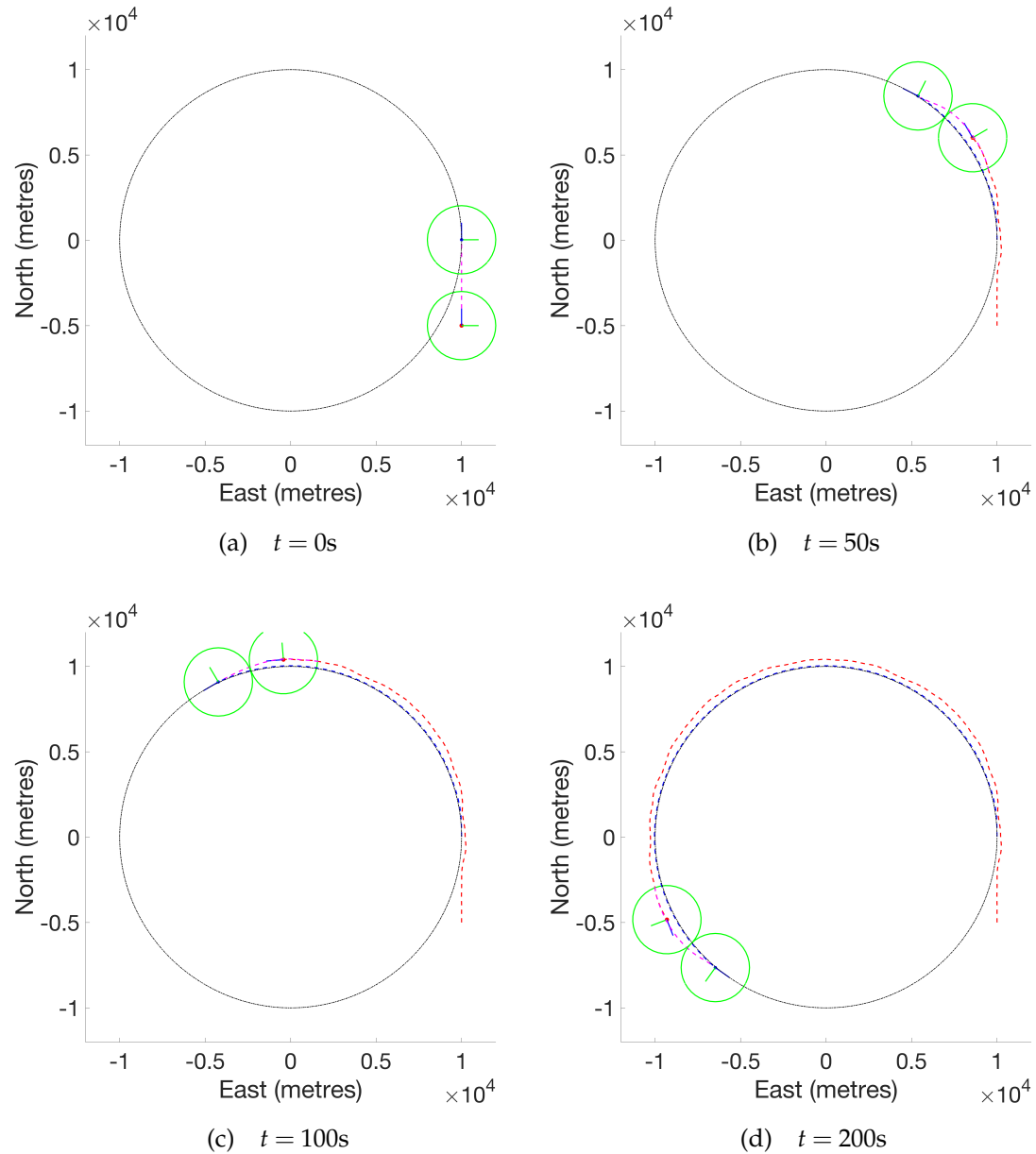


FIGURE D.2 Mission 1: Test 1 - Time history

D.1.2 Test 2

During Mission 1: Test 1, the autonomous aircraft was initialised with a position vector of $(0, 5000, 0)$ and with a heading angle of 180° with respect to the inertial frame.

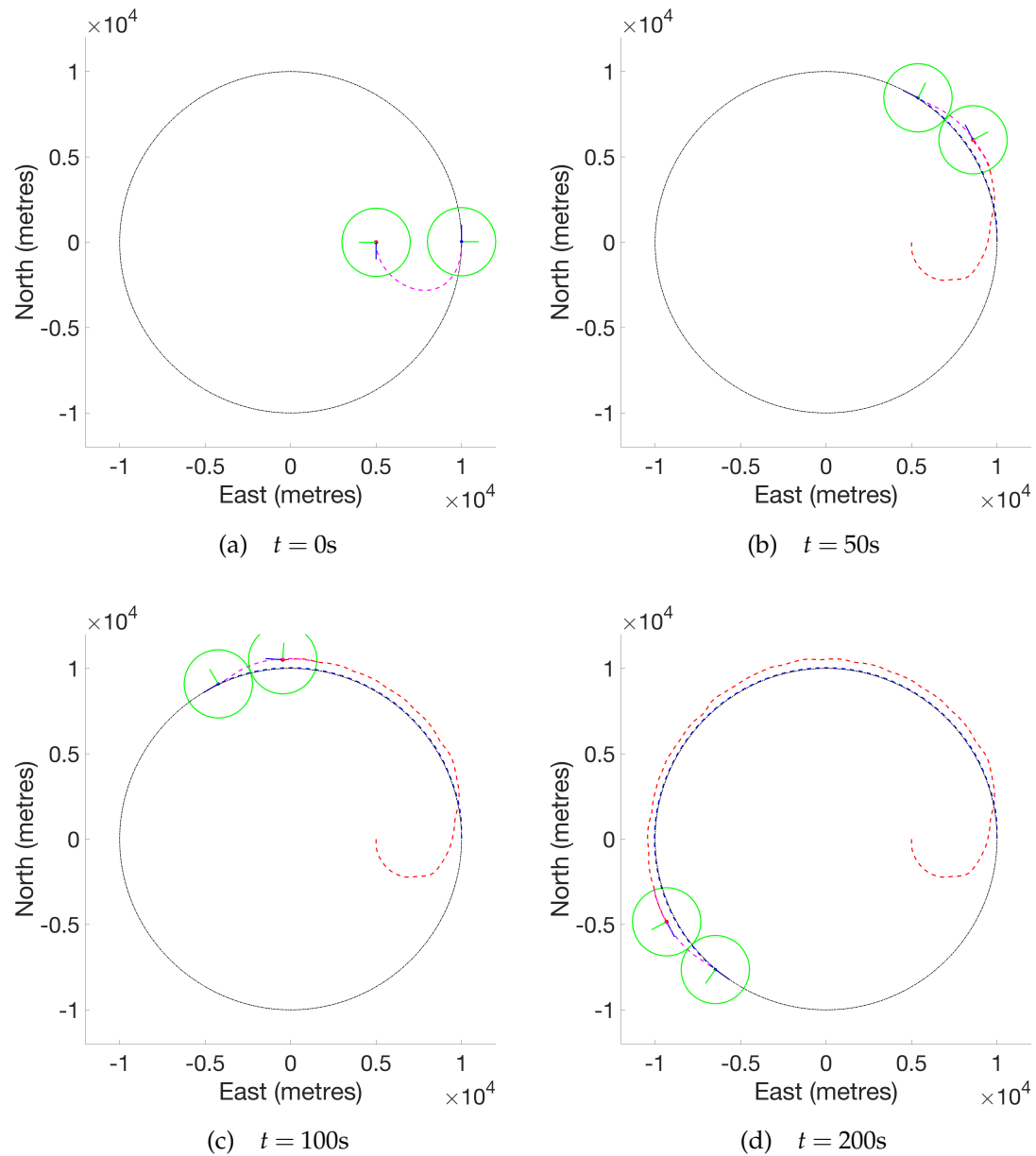


FIGURE D.3 Mission 1: Test 2 - Time history

D.1.3 Test 3

During Mission 1: Test 3, the autonomous aircraft was initialised with a position vector of $(0, -5000, 0)$ and with a heading angle of 90° with respect to the inertial frame.

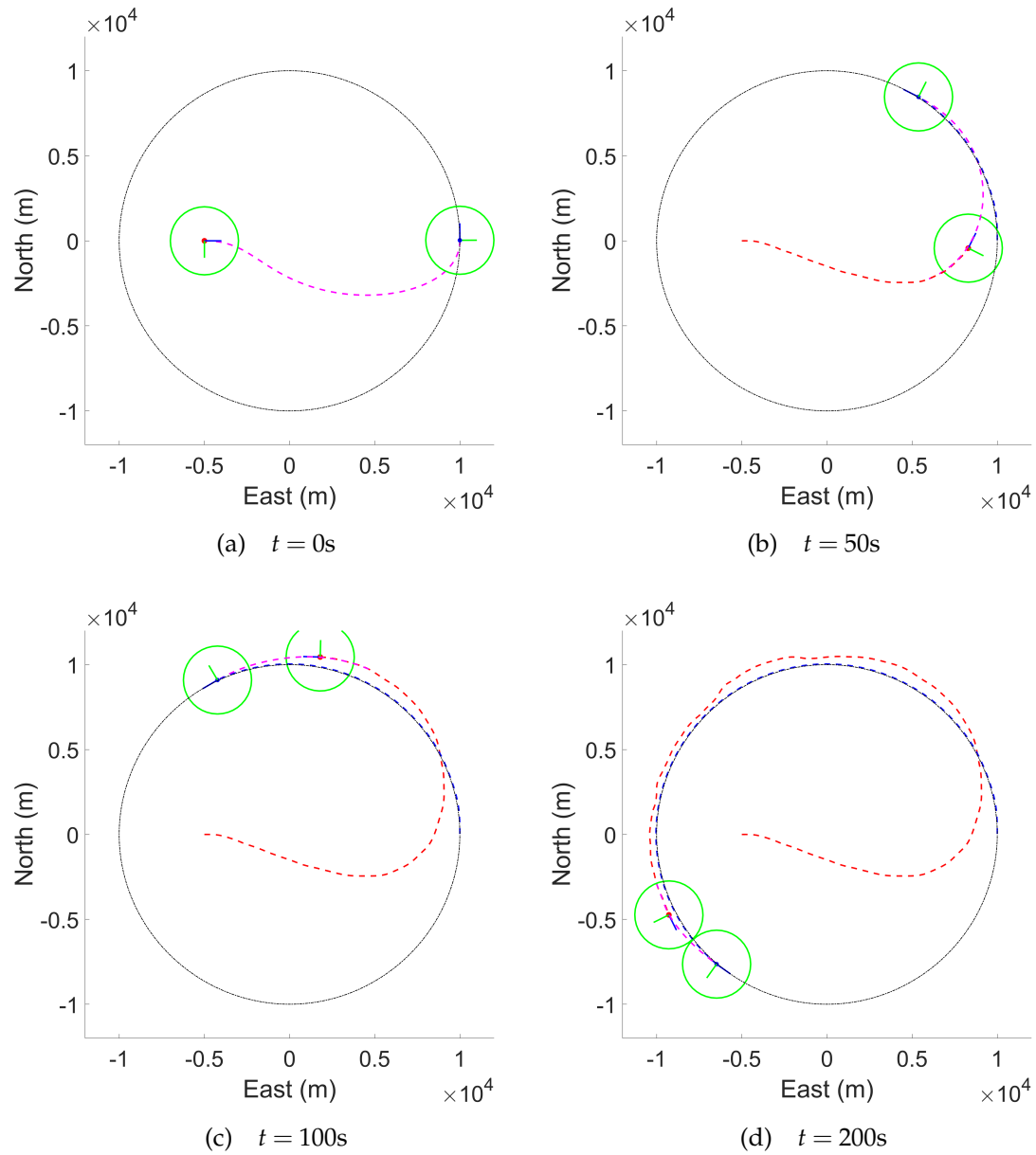


FIGURE D.4 Mission 1: Test 3 - Time history