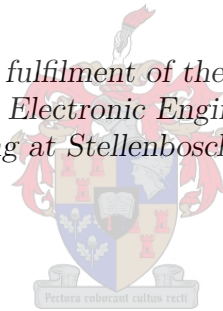


# Optimal Attitude and Flight Vector Recovery for Large Transport Aircraft

by

J.J.K. Engelbrecht

*Thesis presented in partial fulfilment of the requirements for the degree of Master of Science in Electronic Engineering in the Faculty of Engineering at Stellenbosch University*



Department of Electric & Electronic Engineering,  
University of Stellenbosch,  
Private Bag X1, 7602 Matieland, South Africa.

Supervisor: Dr J.A.A. Engelbrecht

December 2017

# Declaration

By submitting this thesis electronically, I declare that the entirety of the work contained therein is my own, original work, that I am the sole author thereof (save to the extent explicitly otherwise stated), that reproduction and publication thereof by Stellenbosch University will not infringe any third party rights and that I have not previously in its entirety or in part submitted it for obtaining any qualification.

Signature: .....J.J.K. Engelbrecht.....

Date: December 2017

Copyright © 2017 Stellenbosch University  
All rights reserved.

# Abstract

Loss of control (LOC) is the largest contributor to commercial jet aircraft fatal accidents worldwide. Aircraft upset conditions are a primary cause leading to LOC situations. Despite flight envelope protection systems, a need exists for an automatic system to assist the pilot in recovering from a flight envelope upset condition.

This thesis presents the design and implementation of an attitude and flight vector recovery system for large transport aircraft. The upset recovery system consists of two major components, namely an optimal trajectory planning component and a practical trajectory execution component. For the optimal trajectory planning, the upset recovery problem is formulated as an optimal control problem and is solved using two different optimal control algorithms, namely dynamic programming (DP) and sequential quadratic programming (SQP). For the trajectory execution, four different control schemes are investigated that use a conventional fly-by-wire flight control system in different configurations to control the aircraft to practically execute the planned optimal trajectory.

The attitude and flight vector recovery system was verified in simulation on the NASA Generic Transport Model (GTM), a wide-envelope aircraft model that is able to model the flight mechanics of large transport aircraft in out-of-envelope conditions. The simulation results show that the trajectory planning component generates kinematically feasible optimal upset recovery trajectories, and that the trajectory execution component successfully controls the aircraft to follow the planned trajectories using a representative flight control system. The SQP trajectory optimisation algorithm proposed in this thesis also improves on the dynamic programming algorithm used in previous research, because it is able to use a more representative model of the aircraft dynamics that includes the inner-loop controller dynamics and the engine lag dynamics.

# Uittreksel

Verlies van beheer (“Loss of control” of LOC) is wêreldwyd die grootste bydraende faktor tot noodlottige ongelukke van kommersiële vliegtuie. Ongewone vlugtoestande is ’n primêre oorsaak van verlies van beheer (LOC) situasies. Ten spyte van vlug-omhullende beveiligingstelsels, bestaan daar steeds ’n behoefte vir ’n outomatiese stelsel om die vlieënier te help om die vlug-omhullende te herstel.

Hierdie tesis beskryf die ontwerp en implementering van ’n oriëntasie en vlugvektor herstel stelsel vir groot kommersiële passassiersvliegtuie aan. Die vlugherstel stelsel bestaan uit twee hoofkomponente, naamlik ’n optimale trajekbeplanning komponent en ’n praktiese trajekuitvoering komponent. Vir die optimale trajekbeplanning word die probleem as ’n optimale beheer probleem geformuleer, en word deur twee verskillende optimale beheer algoritmes opgelos, naamlik dinamiese programmering (“dynamic programming (DP)”) en sekvensiële kwadratiese programmering (“sequential quadratic programming (SQP)”). Vir die trajekuitvoering, word vier verskillende beheerskemas ondersoek. Hierdie beheerskemas maak gebruik van ’n konvensionele “fly-by-wire” vlugbeheerstelsel in verskillende konfigurasies om die vliegtuig te beheer om prakties die beplande optimale trajek uit te voer.

Die outomatiese oriëntasie en vlugvektor herstel stelsel is in simulاسie op die NASA “Generic Transport Model (GTM)” geverifieer. Die GTM ’n wye-omhullende vliegtuigmodel wat in staat is om die vlugmeganika van groot transport vliegtuie in toestande buite die normale vlug-omhullende te modelleer. Die simulاسie resultate wys dat die trajekbeplanning komponent realistiese optimale hersteltrajekte genereer, en dat die trajekuitvoering komponent die vliegtuig suksesvol beheer om die beplande trajekte uit te voer deur gebruik te maak van ’n verteenwoordigende vlugbeheerstelsel. Die SQP trajekoptimalisering algoritme wat in hierdie tesis voorgestel word, verbeter ook op die dinamiese programmering algoritme wat in vorige navorsing gebruik is, omdat dit ’n meer verteenwoordigende model van die vliegtuig dinamika kan gebruik wat die binnelus beheerder dinamika en die enjin naloop dinamika insluit.

# Contents

<b>Declaration</b>	<b>i</b>
<b>Abstract</b>	<b>ii</b>
<b>Uittreksel</b>	<b>iii</b>
<b>Contents</b>	<b>iv</b>
<b>List of Figures</b>	<b>ix</b>
<b>List of Tables</b>	<b>xv</b>
<b>Nomenclature</b>	<b>xvii</b>
<b>Acknowledgements</b>	<b>xxiv</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Background . . . . .	2
1.2 Research Objectives . . . . .	3
1.3 Primary Contributions . . . . .	3
1.4 Definition of Flight Envelope Upset . . . . .	4
1.5 Thesis Outline . . . . .	7
<b>2 The NASA Generic Transport Model</b>	<b>9</b>
2.1 Background . . . . .	9
2.2 Axis Systems and Notation . . . . .	10
2.2.1 Axis Systems . . . . .	10
2.2.2 Aircraft Notation . . . . .	12
2.3 Six Degrees of Freedom Equations of Motion . . . . .	13
2.3.1 Kinetics . . . . .	13
2.3.2 Kinematics . . . . .	14
2.4 Forces and Moments . . . . .	15
2.5 Aerodynamic Model . . . . .	15
2.5.1 Thrust Model . . . . .	16

2.5.2	Gravitational Model . . . . .	17
2.6	Matlab Simulink Model . . . . .	17
2.7	Dynamic Scaling . . . . .	18
2.8	Linearisation of GTM Model . . . . .	19
2.8.1	Calculating the Trim States and Inputs . . . . .	19
2.8.2	Decoupled State Model . . . . .	19
2.8.3	Natural Modes of Motion . . . . .	20
2.9	Conventional Fly-By-Wire Controllers . . . . .	22
<b>3</b>	<b>Conventional Flight Controller Design</b>	<b>23</b>
3.1	Conventional Commercial Aircraft Fly-by-Wire . . . . .	23
3.1.1	General Structure and Objective . . . . .	24
3.1.2	Longitudinal Law . . . . .	24
3.1.3	Lateral Law . . . . .	25
3.1.4	Integrated Controller Architecture for the GTM Aircraft . . . . .	26
3.2	Longitudinal Control System Design . . . . .	27
3.2.1	Normal Acceleration Controller ('DQ Law') . . . . .	28
3.2.2	Flight Path Angle Controller . . . . .	36
3.2.3	Airspeed Controller ('Autothrust') . . . . .	41
3.3	Lateral Control System Design . . . . .	48
3.3.1	Roll and Sideslip Angle Controller ('DPDR Law') . . . . .	48
3.4	Gain-Scheduled Angle of Attack Controller . . . . .	56
3.4.1	Design . . . . .	57
3.4.2	Results . . . . .	59
3.5	Command Tracking for Flight Controllers . . . . .	60
<b>4</b>	<b>Optimal Trajectory Planning</b>	<b>66</b>
4.1	The Upset Recovery Problem . . . . .	67
4.2	Trajectory Optimisation Background . . . . .	67
4.2.1	Numerical Approaches for Optimal Control . . . . .	67
4.2.2	Previous Research . . . . .	70
4.3	Optimisation Problem and System Model Formulation . . . . .	70
4.3.1	Point Mass Translational and Input Dynamics . . . . .	70
4.3.2	Assumptions and Requirements . . . . .	75
4.3.3	Optimal Control Problem Formulation . . . . .	76
4.3.4	Cost Function . . . . .	77
4.3.5	State Constraints . . . . .	77
4.3.6	Input Constraints . . . . .	78
4.3.7	Terminal State Constraints . . . . .	79

4.4	Trajectory Optimisation using Dynamic Programming . . . . .	79
4.4.1	Reduced-Order Model . . . . .	79
4.4.2	Hierarchical Multi-Objective Cost Function . . . . .	80
4.4.3	Dynamic Programming Background . . . . .	80
4.4.4	Dynamic Programming Implimentation . . . . .	84
4.4.5	Dynamic Programming Results . . . . .	89
4.4.6	Results Summary . . . . .	98
4.5	Trajectory Optimisation using SQP . . . . .	98
4.5.1	Non-linear Programming Problem . . . . .	99
4.5.2	General SQP algorithm . . . . .	99
4.5.3	Direct Transcription . . . . .	101
4.5.4	Transcribing the Flight Trajectory Optimisation Problem into an NLP .	105
4.5.5	Scaling . . . . .	110
4.5.6	Obtaining the Objective Gradient and Constraint Jacobian using Auto- matic Differentiation . . . . .	111
4.5.7	Calling the SQP Solver . . . . .	114
4.5.8	Representing the Solution . . . . .	115
4.5.9	Estimating the Error in the Solution . . . . .	117
4.5.10	Using Z-Transform Difference Equations in Direct Transcription Method	117
4.5.11	SQP Trajectory Results . . . . .	118
4.5.12	Results Summary . . . . .	141
4.6	Conclusions . . . . .	142
<b>5</b>	<b>Trajectory Execution</b>	<b>143</b>
5.1	Background and Motivation . . . . .	143
5.2	Overview of Control Schemes . . . . .	146
5.3	Design of Control Schemes . . . . .	148
5.3.1	Input Commands Only Control Scheme . . . . .	148
5.3.2	State Trajectories Only Control Scheme . . . . .	150
5.3.3	Combined State and Input Control Scheme . . . . .	151
5.3.4	Compensated State Trajectories Control Scheme . . . . .	157
5.3.5	Sensing Wind-Axis Bank Angle and Wind-Axis Roll Rate . . . . .	159
5.3.6	Converting Wind-Axis Reference Angles . . . . .	161
5.4	Trajectory Execution Results . . . . .	163
5.4.1	Comparative Control Scheme Results . . . . .	163
5.4.2	Input Commands Only Control Scheme . . . . .	166
5.4.3	State Commands Only Control Scheme . . . . .	176
5.4.4	Combined State and Input Control Scheme . . . . .	181
5.4.5	Compensated State Trajectories Control Scheme . . . . .	189

5.4.6	Observations from Trajectory Execution Using SQP Trajectory Planning	197
5.5	Summary of Results	198
5.6	Conclusions	199
<b>6</b>	<b>Conclusions and Recommendations</b>	<b>200</b>
6.1	Optimal Trajectory Planning	200
6.2	Trajectory Execution	201
6.3	Future Work	203
<b>A</b>	<b>Mathematical Aircraft Modelling</b>	<b>204</b>
A.1	Axis Systems and Notation	204
A.1.1	Axis Systems	204
A.1.2	Aircraft Notation	206
A.2	Six Degrees of Freedom Equations of Motion	208
A.2.1	Kinetics	209
A.2.2	Kinematics	210
A.3	Forces and Moments	211
A.3.1	Aerodynamic Model	212
A.3.2	Thrust Model	214
A.3.3	Gravitational Model	215
A.4	Linearisation and Linear Analysis	215
A.4.1	Linearisation Derivation	216
A.4.2	Linear Analysis of Natural Modes of Motion	218
<b>B</b>	<b>Additional Numerical Results</b>	<b>222</b>
B.1	Error Plots of Numerical Results	222
B.2	Initial SQP Trajectory Results	231
B.2.1	System Dynamics	231
B.2.2	Problem Setup	233
B.2.3	Two-Dimensional case	235
B.2.4	Three-Dimensional Thrust dynamic case	239
B.2.5	Three-Dimensional bank angle case	243
B.2.6	Four-Dimensional bank angle with thrust lag case	246
B.2.7	Results summary	249
<b>C</b>	<b>DQ Elevator Feed-Forward Closed-Loop Constant</b>	<b>250</b>



<i>CONTENTS</i>	<b>viii</b>
<b>D First Iteration of Trajectory Regulation Control</b>	<b>251</b>
D.1 Upset Command Tracking . . . . .	251
D.1.1 Input Reference Only . . . . .	252
D.1.2 State Reference Only with AT off . . . . .	254
D.1.3 State Reference Only with AT on . . . . .	256
D.1.4 State and Input Combined Reference . . . . .	258
D.2 Angle of Attack vs Normal Acceleration (DQ) Controller . . . . .	271
D.2.1 Angle of Attack Controller Model Comparison . . . . .	278
<b>Bibliography</b>	<b>281</b>

# List of Figures

1.1	Quantitative loss-of-control envelopes . . . . .	5
2.1	Image of the NASA Generic Transport Model . . . . .	9
2.2	North-East-Down axis system . . . . .	10
2.3	Body axis system . . . . .	11
2.4	Wind-axis system . . . . .	11
2.5	Standard aircraft notation . . . . .	12
2.6	Simulink model of the NASA GTM (top level) . . . . .	18
3.1	General fly-by-wire controller structure . . . . .	24
3.2	Conventional lateral law architecture . . . . .	25
3.3	Conventional lateral law architecture . . . . .	26
3.4	Conventional flight controller architecture . . . . .	27
3.5	DQ controller architecture . . . . .	28
3.6	Comparison between full-order and reduced-order longitudinal model response for unit elevator step command . . . . .	30
3.7	Comparison of closed-loop AoA controller poles and desired closed-loop DQ controller poles . . . . .	34
3.8	Results of DQ controller closed-loop unit step responses . . . . .	35
3.9	Closed-loop poles after adding DQ controller . . . . .	36
3.10	Flight path angle controller architecture . . . . .	36
3.11	Root locus of the reduced-order plant with flight path angle controller for a chosen gain $K_\gamma$ that meets the chosen specifications as well as the closed-loop step response of reduced-order model with flight path angle controller . . . . .	39
3.12	Flight path angle controller step responses simulated on the full-order linear and non-linear models of the flight path angle dynamics . . . . .	40
3.13	Flight path angle controller step response simulated non-linear model of the flight path angle dynamics at a high bank angle . . . . .	41
3.14	Airspeed controller architecture . . . . .	42
3.15	Pole placement of desired autothrust closed-loop poles . . . . .	45
3.16	Closed-loop autothrust controller response to unit step command . . . . .	46
3.17	Airspeed response with and without anti-windup . . . . .	47

<i>LIST OF FIGURES</i>	<b>x</b>
3.18 DPDR controller architecture . . . . .	48
3.19 Closed-loop poles after full-state feedback using LQR optimisation algorithm . .	52
3.20 Bode plot of full-order and reduced-order RCC . . . . .	52
3.21 Closed-loop roll angle response of DPDR controller with and without RCC . . .	53
3.22 Closed-loop sideslip response of DPDR controller with and without mixing gains	54
3.23 Closed-loop linear and non-linear roll rate and roll angle response to roll rate step input . . . . .	54
3.24 Closed-loop poles after adding the full DPDR controller, including the full-state feedback, the input mixing gains, and the turn co-ordination controller . . . . .	56
3.25 Angle of attack controller architecture . . . . .	56
3.26 Results of linear AoA controller closed-loop unit step responses . . . . .	59
3.27 Closed-loop non-linear model response to a normal acceleration unit step with autothrust controller active . . . . .	61
3.28 Closed-loop non-linear model response to a FPA rate $\dot{\gamma}$ unit step with autothrust controller active . . . . .	62
3.29 Closed-loop non-linear model response to FPA ramp reference command . . . . .	63
3.30 Closed-loop non-linear model response to roll angle ramp reference command . .	64
4.1 Numerical methods for solving optimal control problems . . . . .	68
4.2 Point mass translational dynamics . . . . .	71
4.3 Aerodynamic lift and drag coefficient functions . . . . .	73
4.4 Dynamic programming path cost interpolation . . . . .	83
4.5 Max allowed roll rate as a function of angle of attack . . . . .	90
4.6 Illustrative dynamic programming recovery trajectories . . . . .	93
4.7 Plot of all state recovery trajectories . . . . .	94
4.8 Plot of all control input recovery sequences . . . . .	95
4.9 Plot of all altitude change recovery trajectories . . . . .	95
4.10 Colour map of recoverable and unrecoverable states. Airspeed vs. flight path angle, at a bank angle of 0 degrees. (green = recoverable, red = unrecoverable) .	96
4.11 Colour map of recoverable and unrecoverable states. Flight path angle vs. bank angle, at various airspeeds. (green = recoverable, red = unrecoverable) . . . . .	97
4.12 Collocation conditions for third-order method . . . . .	102
4.13 Euler Collocation . . . . .	104
4.14 Hermite-Simpson Collocation . . . . .	105
4.15 Computational graph representing $f(x_1, x_2) = x_1x_2 + \sin(x_1)$ . . . . .	112
4.16 Jacobian sparsity pattern of constraints . . . . .	113
4.17 SQP vs DP for steep flight path angle descent upset . . . . .	121
4.18 SQP vs DP for steep flight path angle climb upset . . . . .	122
4.19 SQP vs DP for bank angle upset . . . . .	123
4.20 SQP vs DP for underspeed upset . . . . .	124

## LIST OF FIGURES

xi

4.21	SQP vs DP for overspeed upset . . . . .	125
4.22	SQP vs DP for flight path angle steep descent and bank angle upset . . . . .	126
4.23	SQP vs DP for flight path angle steep descent with bank angle in overspeed upset	127
4.24	SQP vs DP for flight path angle steep climb with bank angle in underspeed upset	128
4.25	SQP vs DP for steep flight path angle descent upset . . . . .	133
4.26	SQP vs DP for steep flight path angle climb upset . . . . .	134
4.27	SQP vs DP for bank angle upset . . . . .	135
4.28	SQP vs DP for underspeed upset . . . . .	136
4.29	SQP vs DP for overspeed upset . . . . .	137
4.30	SQP vs DP for flight path angle steep descent and bank angle upset . . . . .	138
4.31	SQP vs DP for flight path angle steep descent with bank angle in overspeed upset	139
4.32	SQP vs DP for flight path angle steep climb with bank angle in underspeed upset	140
5.1	Integrated flight envelope recovery architecture using Lyapunov controller . . . .	144
5.2	Integrated flight envelope recovery architecture using conventional controllers . .	145
5.3	Architecture of ‘input only’ control scheme . . . . .	146
5.4	Architecture of ‘state only’ trajectory control scheme . . . . .	147
5.5	Architecture of ‘combined state and input’ control scheme . . . . .	147
5.6	Architecture of ‘compensated state’ trajectory control scheme . . . . .	148
5.7	Flight controller architecture for ‘input only’ control scheme . . . . .	149
5.8	Flight controller architecture for ‘state only’ control scheme . . . . .	151
5.9	Modified flight controller architecture for ‘combined state and input’ control scheme	153
5.10	Flight controller architecture for ‘compensated state’ scheme . . . . .	158
5.11	Comparison of state trajectories and altitude loss for all four control schemes presented . . . . .	165
5.12	State trajectory response using ‘input only’ control scheme that compares normal acceleration and angle of attack controller variants for a recovery trajectory that uses higher angles of attack . . . . .	167
5.13	Input trajectory response using ‘input only’ control scheme for normal acceleration controller variant for a recovery trajectory that uses higher angles of attack . . .	168
5.14	Input trajectory response using ‘input only’ control scheme for angle of attack controller variant for a recovery trajectory that uses higher angles of attack . . .	169
5.15	State trajectory response using ‘input only’ control scheme and normal acceleration controller variant for a recovery trajectory that uses lower angles of attack .	174
5.16	Input trajectory response using ‘input only’ scheme and normal acceleration controller variant for a recovery trajectory that uses lower angles of attack . . . . .	175
5.17	State trajectory response using ‘state only’ control scheme and normal acceleration controller for initial upset condition. . . . .	179
5.18	Input trajectory response using ‘state only’ control scheme and normal acceleration controller for initial upset condition . . . . .	180

## LIST OF FIGURES

xii

5.19	State trajectory response using ‘combined state and input’ scheme and normal acceleration controller for a recovery trajectory that uses lower angles of attack . . . . .	182
5.20	Input trajectory response using ‘combined state and input’ scheme and normal acceleration controller for a recovery trajectory that uses lower angles of attack . . . . .	183
5.21	Altitude change response using ‘combined state and input’ scheme and normal acceleration controller for a recovery trajectory that uses lower angles of attack . . . . .	184
5.22	State trajectory response using ‘combined state and input’ scheme and normal acceleration controller for a recovery trajectory that uses higher angles of attack . . . . .	186
5.23	Input trajectory response using ‘combined state and input’ scheme and normal acceleration controller for a recovery trajectory that uses higher angles of attack . . . . .	187
5.24	Altitude change response using ‘combined state and input’ scheme and normal acceleration controller for a recovery trajectory that uses higher angles of attack . . . . .	188
5.25	State trajectory response using ‘compensated state’ scheme and normal acceleration controller for a recovery trajectory that uses lower angles of attack . . . . .	190
5.26	Input trajectory response using ‘compensated state’ scheme and normal acceleration controller for a recovery trajectory that uses lower angles of attack . . . . .	191
5.27	Altitude change response using ‘compensated state’ scheme and normal acceleration controller for initial upset condition using low angles of attack . . . . .	192
5.28	State trajectory response using ‘compensated state’ scheme and normal acceleration controller for a recovery trajectory that uses higher angles of attack . . . . .	194
5.29	Input trajectory response using ‘compensated state’ scheme and normal acceleration controller for a recovery trajectory that uses higher angles of attack . . . . .	195
5.30	Altitude change response using ‘compensated state’ scheme and normal acceleration controller for a recovery trajectory that uses higher angles of attack . . . . .	196
A.1	North-East-Down axis system . . . . .	204
A.2	body-axis system . . . . .	205
A.3	wind-axis system . . . . .	206
A.4	Standard aircraft notation . . . . .	207
A.5	Polar velocity co-ordinates . . . . .	208
A.6	Block Diagram Overview of 6DOF EOM . . . . .	208
A.7	Sequence of 3-2-1 Euler angles . . . . .	210
A.8	Decomposition of the total angular rate vector into steady state and oscillatory components . . . . .	213
A.9	Three decomposition schemes used with the Kalviste methods . . . . .	213
A.10	Longitudinal Poles in the S-Plane. . . . .	219
A.11	Lateral Poles in the S-Plane . . . . .	219
A.12	Non-linear versus linear longitudinal system response for small elevator control input doublet . . . . .	220
A.13	Non-linear versus linear lateral system response for small aileron and rudder control input doublet respectively . . . . .	221
B.1	Estimated error in SQP trajectory for steep flight path angle descent upset . . . . .	223

## LIST OF FIGURES

xiii

B.2	Estimated error in SQP trajectory for steep flight path angle climb upset . . . . .	224
B.3	Estimated error in SQP trajectory for bank angle upset . . . . .	225
B.4	Estimated error in SQP trajectory for underspeed upset . . . . .	226
B.5	Estimated error in SQP trajectory for overspeed upset . . . . .	227
B.6	Estimated error in SQP trajectory for flight path angle and bank angle upset . . .	228
B.7	Estimated error in SQP trajectory for FPA, bank angle and overspeed upset . . .	229
B.8	Estimated error in SQP trajectory for FPA, bank angle and underspeed upset . . .	230
B.9	Aerodynamic lift and drag coefficient functions . . . . .	235
B.10	SQP vs dynamic programming solution with variable thrust dynamic programming seed . . . . .	236
B.11	SQP vs dynamic programming solution with constant thrust dynamic programming seed . . . . .	237
B.12	SQP vs dynamic programming solution with a truncated time window and a variable thrust dynamic programming seed . . . . .	238
B.13	SQP vs dynamic programming solution with a truncated time window and a constant thrust dynamic programming seed. The SQP's Thrust state has also been bounded to 30 N . . . . .	239
B.14	SQP vs dynamic programming thrust lag solution with variable thrust dynamic programming seed . . . . .	241
B.15	SQP vs dynamic programming thrust lag solution with constant thrust dynamic programming seed . . . . .	242
B.16	SQP vs dynamic programming underspeed and bank angle recovery solution with variable thrust dynamic programming seed . . . . .	244
B.17	SQP underspeed and bank angle recovery solution with no solution for dynamic programming seed . . . . .	245
B.18	SQP vs dynamic programming four dimensional solution with variable thrust dynamic programming seed . . . . .	247
B.19	SQP vs dynamic programming four dimensional time varying solution with constant thrust dynamic programming seed . . . . .	248
D.1	Closed-loop non-linear response to SQP input commands from upset condition. . .	253
D.2	Closed-loop non-linear response to SQP input commands from upset condition continued. . . . .	254
D.3	Closed-loop non-linear response to SQP state reference only commands from upset condition with AT off. . . . .	255
D.4	Closed-loop non-linear response to SQP state reference only commands from upset condition with AT off continued. . . . .	256
D.5	Closed-loop non-linear response to SQP state reference only commands from upset condition with AT on. . . . .	257
D.6	Closed-loop non-linear response to SQP state reference only commands from upset condition with AT on continued. . . . .	258
D.7	Closed-loop non-linear response for Case 1 combined SQP state and input commands with $\Phi_{lim} = 80^\circ$ . . . . .	260

## LIST OF FIGURES

xiv

D.8	Closed-loop non-linear response for Case 1 combined SQP state and input commands with $\Phi_{lim} = 80^\circ$ continued. . . . .	261
D.9	Closed-loop non-linear response for Case 1 combined SQP state and input commands with $\Phi_{lim} = 40^\circ$ . . . . .	262
D.10	Closed-loop non-linear response for Case 1 combined SQP state and input commands with $\Phi_{lim} = 40^\circ$ continued. . . . .	263
D.11	Closed-loop non-linear response for Case 1 combined SQP state and input commands with $\Phi_{lim} = 180^\circ$ . . . . .	264
D.12	Closed-loop non-linear response for Case 1 combined SQP state and input commands with $\Phi_{lim} = 180^\circ$ continued. . . . .	265
D.13	Closed-loop non-linear response for Case 2 combined SQP state and input commands. . . . .	266
D.14	Closed-loop non-linear response for Case 2 combined SQP state and input commands continued. . . . .	267
D.15	Closed-loop non-linear response for Case 3 combined SQP state and input commands with $\Phi_{lim} = 80^\circ$ . . . . .	268
D.16	Closed-loop non-linear response for Case 3 combined SQP state and input commands with $\Phi_{lim} = 80^\circ$ continued. . . . .	269
D.17	Closed-loop non-linear response for Case 3 combined SQP state and input commands with $\Phi_{lim} = 60^\circ$ . . . . .	270
D.18	Closed-loop non-linear response for Case 3 combined SQP state and input commands with $\Phi_{lim} = 60^\circ$ continued. . . . .	271
D.19	Closed-loop non-linear response to SQP input commands from upset condition. . . . .	273
D.20	Closed-loop non-linear response to SQP input commands from upset condition continued. . . . .	274
D.21	Closed-loop non-linear response to SQP input commands from upset condition. . . . .	275
D.22	Closed-loop non-linear response to SQP input commands from upset condition continued. . . . .	276
D.23	Closed-loop non-linear response to SQP input commands from upset condition. . . . .	277
D.24	Closed-loop non-linear response to SQP input commands from upset condition continued. . . . .	278
D.25	GTM response to trajectory regulation using a first order angle of attack model with angle of attack inner-loop controller . . . . .	279
D.26	GTM response to trajectory regulation using a second order angle of attack model with angle of attack inner-loop controller . . . . .	280

# List of Tables

2.1	Dynamic scaling factors for NASA GTM . . . . .	18
2.2	Calculated trim values . . . . .	19
2.3	Longitudinal characteristics. . . . .	21
2.4	Longitudinal characteristics. . . . .	21
3.1	Longitudinal DQ design specifications . . . . .	33
3.2	Calculated trim values for each AoA condition . . . . .	57
3.3	Gain-scheduled angle of attack closed-loop short-period design specifications . . . . .	59
4.1	Problem parameter values . . . . .	91
4.2	Sequence of value and derivative calculations performed in AD example . . . . .	113
4.3	Problem parameter values . . . . .	120
5.1	Initial upset condition . . . . .	163
5.2	Initial upset condition . . . . .	166
5.3	Initial upset condition . . . . .	172
5.4	Initial upset condition . . . . .	177
5.5	Initial upset condition . . . . .	181
5.6	Initial upset condition . . . . .	185
5.7	Initial upset condition . . . . .	189
5.8	Initial upset condition . . . . .	193
A.1	Equations used in the Hybrid Kalviste method . . . . .	214
A.2	Longitudinal characteristics. . . . .	218
A.3	Longitudinal characteristics. . . . .	220
B.1	Problem parameter values . . . . .	234
B.2	Initial state values and dynamic programming setup for two dimensional case . . . . .	235
B.3	Initial state values and dynamic programming setup for three dimensional thrust lag case . . . . .	240
B.4	Initial state values and dynamic programming setup for three dimensional bank angle case . . . . .	243
B.5	Initial state values and dynamic programming setup for four dimensional case . . . . .	246



<i>LIST OF TABLES</i>	<b>xvi</b>
B.6 Dynamic programming setup for four dimensional time varying case . . . . .	246
D.1 Initial upset condition . . . . .	251
D.2 Initial upset condition . . . . .	272
D.3 Initial upset condition . . . . .	274
D.4 Initial upset condition . . . . .	276

# Nomenclature

## Abbreviations and Acronyms

LOC	Loss of control
6DOF	6-Degrees-of-freedom
EOM	Equations of motion
FBW	Fly-by-wire
AirSTAR	Airborne Subscale Transport Aircraft Research
GTM	Generic Transport Model
NASA	National Aeronautics and Space Administration
UAV	Unmanned Aerial Vehicle
QLC	Quantitative Loss-of-Control Criteria
NLP	Non-linear Programming Problem
BFGS	Broyden-Fletcher-Goldfarb-Shanno
TPBVP	Two-point boundary-value problem
ODE	Ordinary differential equation
SQP	Sequential Quadratic Programming
DP	Dynamic Programming
AoA	Angle of Attack
FPA	Flight Path Angle
CG	Centre of Gravity
RCC	Rudder Coordination Controller
AW	Anti-Windup

## Symbol Conventions

$x$	Scalar
$\mathbf{x}$	Vector
$\mathbf{X}$	Matrix
$x^*$	Optimal value of variable $x$

$\dot{x}$	Derivative of $x$
$\ddot{x}$	Second derivative of $x$
$\frac{\partial f}{\partial x}$	The partial derivative of function, $f$ , with respect to $x$

**Constants**

$g$	Gravitational acceleration
-----	----------------------------

**Aircraft Parameters**

$m$	Mass
$\mathbf{I}_B$	Moment of inertia matrix of the aircraft body
$I_{xx}, I_{yy}, I_{zz}$	Principle moment of inertia of aircraft's body x-axis, y-axis and z-axis
$S$	Wing surface area
$\bar{c}$	Mean aerodynamic chord
$b$	Wing span

**Aircraft Dynamics**

$X, Y, Z$	Coordinates of the force vector in the body-axis (axial, lateral, and normal force)
$L, M, N$	Coordinates of the moment vector in the body-axis (roll, pitch, and yaw moment)
$U, V, W$	Coordinates of the linear velocity vector in the body-axis (axial, lateral, and normal velocity)
$P, Q, R$	Coordinates of the angular velocity vector in the body-axis (roll, pitch, and yaw rates)
$\delta_A, \delta_R, \delta_E$	Aileron, rudder and elevator control surface deflections. A positive deflection is defined by it producing a negative moment.
$\bar{V}, \alpha, \beta$	Airspeed magnitude, angle of attack, and sideslip angle
$N, E, D$	Coordinates of position vector in inertial-axes (north, east and down position)
$\Phi, \Theta, \Psi$	Euler 3-2-1 attitude parameters of the body-axis system with respect to inertial-axis system (roll, pitch, and yaw angle)

**Aerodynamic Model**

$C_X, C_Y, C_Z$	Aerodynamic force coefficients of the aircraft's body x-axis, y-axis and z-axis
$C_l, C_m, C_n$	Aerodynamic moment coefficients of the aircraft's body x-axis, y-axis and z-axis (pitching, rolling and yawing moment)
$\rho$	Air density

$\bar{q}$	Dynamic pressure ( $\bar{q} = \frac{1}{2}\rho\bar{V}^2$ )
$\bar{\Omega}$	Total angular rate vector
$\bar{\Omega}_{osc}$	Oscillatory component of total angular rate vector
$\bar{\Omega}_{xz}$	Projection of total angular rate vector in body xz-plane
$p_b, q_b, r_b$	Body-axis roll, pitch and yaw angular rates
$p_{osc}, q_{osc}, r_{osc}$	Oscillatory component of body-axis roll, pitch and yaw angular rates
$\omega_{ss}$	Steady-state angular rate component along velocity vector
$\hat{p}_{osc}, \hat{q}_{osc}, \hat{r}_{osc}$	Non-dimensional oscillatory component of body-axis roll, pitch and yaw angular rates
$\hat{\omega}_{ss}$	Non-dimensional steady-state angular rate component along velocity vector

### Reduced-order Model Dynamics

$L, D$	Aerodynamic lift and drag forces
$C_L, C_D$	Static aerodynamic lift and drag coefficients
$\gamma$	Flight path angle
$\Phi_W$	Wind-axis bank angle
$\alpha_c, \tau_\alpha$	Angle of attack command and angle of attack input delay time constant
$P_W, P_{Wc}, \tau_P$	Wind-axis roll rate, wind-axis roll rate command and wind-axis roll rate input delay time constant
$T, T_c, \tau$	Thrust force, thrust command and engine thrust lag time constant
$\delta_T, \delta_{Tc}$	Throttle state and throttle command
$\Psi_W$	Wind-axis heading angle
$V_N, V_E, V_D$	North, east and down velocity components in inertial-axis
$n_L$	Normal load factor
$\delta_T, \delta_{Tc}$	Throttle state and throttle command

### Optimal Control

$\mathbf{x}, \mathbf{u}$	State and control input vectors
$\mathbf{x}(t), \mathbf{u}(t)$	Continuous state and input trajectories
$t_f$	Final time
$t_0$	Initial time
$J$	Total cost function
$h()$	Terminal state cost function
$g()$	Transition/path cost function

$\mathbf{f}()$	Non-linear state dynamics vector function
$\boldsymbol{\eta}()$	Constraint function
$h$	Altitude
$\gamma_f$	Terminal flight path angle value
$\Phi_{Wf}$	Terminal wind-axis bank angle value

### Dynamic Programming

$J^h$	Altitude cost function
$J^{\bar{V}}$	Airspeed cost function
$J^\Phi$	Bank angle cost function
$\mathbf{X}_q$	Quantised state array
$\mathbf{U}_q$	Quantised input array
$\mathbf{X}_{bound}$	Set of admissible state values
$\mathbf{U}_{bound}$	Set of admissible input values
$\mathbf{X}_{final}$	Set of admissible terminal state values
$\mathbf{x}_i$	State value at index value $i$ for current time instant
$\mathbf{x}_j$	State value at index value $j$ for next time instant
$\mathbf{u}_{ij}$	Input value that transitions from current state $\mathbf{x}_i(k)$ to next state $\mathbf{x}_j(k+1)$
$J_{ij}$	Total path cost of moving from current state $\mathbf{x}_i(k)$ via next state $\mathbf{x}_j(k+1)$ to a terminal state
$\Delta J_{ij}$	Transition path cost of moving from current state $\mathbf{x}_i(k)$ to next state $\mathbf{x}_j(k+1)$
$J_i^*$	Optimal total path cost of moving from current state $\mathbf{x}_i(k)$ to a terminal state
$J_j^*$	Optimal total path cost of moving from next state $\mathbf{x}_j(k+1)$ to a terminal state
$\mathbf{u}_{ij}^*$	Optimal input that will transition system from current state $\mathbf{x}_i(k)$ to a next state $\mathbf{x}_j(k+1)$
$\Delta t$	Sampling time
$N$	Number of time steps
$n$	Number of quantised state combinations
$\mathbf{J}_{n \times N}^*$	Data table that stores optimal cost $J_{i,k}^*$ for state $\mathbf{x}_i$ at time index $k$
$\mathbf{U}_{n \times N}^*$	Data table that stores optimal input $\mathbf{u}_{i,k}^*$ for state $\mathbf{x}_i$ at time index $k$
$\mathbf{j}_{n \times N}^*$	Data table that stores optimal state transition index $j_{i,k}^*$ for state $\mathbf{x}_i$ at time index $k$

$\bar{\mathbf{V}}_q$	Quantised airspeed state array
$\Gamma_q$	Quantised flight path angle state array
$\Phi_{\mathbf{W}q}$	Quantised wind-axis bank angle state array

**Direct Transcription**

$F()$	NLP cost function
$\mathbf{z}$	NLP decision variable vector
$\mathbf{c}()$	NLP equality and inequality constraint function
$\tilde{F}()$	Normalised NLP cost function
$\tilde{\mathbf{x}}, \tilde{\mathbf{u}}$	Normalised state and control input vectors
$\tilde{\mathbf{z}}$	Normalised NLP decision variable vector
$\tilde{\mathbf{c}}()$	Normalised NLP equality and inequality constraint function
$\mathbf{H}$	Hessian of Lagrangian
$\mathcal{L}()$	Lagrangian function
$\boldsymbol{\lambda}$	Lagrange multiplier vector
$\mathbf{H}$	Hessian of Lagrangian
$\mathbf{d}$	SQP search direction
$\mathbf{x}_k, \mathbf{u}_k$	State and input values for knot point at time instant $k$
$K$	Number of intervals
$N$	Number of collocation points
$t_K$	Time value of final knot point
$t_s$	Sample time of each interval
$\zeta_k$	Full defect constraint vector for knot point $k$
$\zeta_k^h, \zeta_k^p$	Hermite and Simpson defect constraint vector for knot point $k$
$s_k$	Slack variable at knot point $k$
$\mathbf{V}_x, \mathbf{V}_u$	State and input variable scaling matrices
$\mathbf{W}_f, \mathbf{W}_g$	Defect and path constraint scaling matrices
$\mathbf{r}_x, \mathbf{r}_u$	State and input variable bias/shift vectors
$\mathbf{g}$	Path constraints
$\mathbf{J}\mathbf{f}()$	Jacobian of vector function $\mathbf{f}$
$\epsilon(t)$	Error function of state dynamics
$\xi_k$	Error estimate value in state values for interval starting at knot point $k$

**Linear Analysis**

$\Delta \mathbf{x}, \Delta \mathbf{u}$	State and Input perturbations from trim values
$\omega_n$	Natural frequency
$\zeta$	Damping ratio

### Controller Design and Regulation

$a_z$	Normal acceleration in body z-axis
$\mathbf{Q}, \mathbf{R}$	State and input LQR weighting matrices
$H(s)$	Transfer function in frequency domain
$a_{W_x}$	Acceleration in wind x-axis (Axial acceleration)
$a_{W_y}$	Acceleration in wind y-axis (Lateral acceleration)
$a_{W_z}$	Acceleration in wind z-axis (Normal acceleration)
$\tau_\gamma$	Flight path angle controller time constant

### Subscripts

$B$	Body-axis
$S$	Stability-axis
$W$	Wind-axis
$I$	Inertial-axis
$0$	Value for initial time
$f$	Value for final time
$i$	Index of quantised array at current time index $k$
$j$	Index of quantised array at next time index $k + 1$
$i, k$	Index of two-dimensional data table, indexing a value for the $i$ 'th state combination at the $k$ 'th time instant
$q$	Quantisation
$k$	Knot point number
$c$	Command signal
$ref$	Controller reference signal (Absolute command with trim included)
<i>Static</i>	Static aerodynamic coefficient component
<i>Initial</i>	Initial state value
<i>lat</i>	Lateral
<i>long</i>	Longitudinal
<i>gravity</i>	Component of gravity acceleration in variable direction
<i>specific</i>	Specific acceleration in variable direction

## NOMENCLATURE

xxiii

$AT$	Autothrust
$r$	Reduced-order
$RCC$	Rudder Coordination Controller
$ss$	Steady-state
$T$	Trim
$u$	upper bound of continuous variable
$l$	lower bound of continuous variable
$U$	upper bound of decision variable
$L$	lower bound of decision variable
max	Maximum value
min	Minimum value

**Superscripts**

$T$	Thrust
$A$	Aerodynamic
$G$	Gravitational
$(k)$	Iteration number



# Acknowledgements

I would like to thank the following people and parties that helped bring this thesis to fruition:

- My amazing supervisor Dr. Japie Engelbrecht (no relation) for his support, guidance and wisdom throughout this project. You are a great role model who helped me become a better engineer by inspiring greater levels of critical thinking, while still enjoying the work and having a good laugh every now and then. I specifically appreciate the effort you put into conveying insights and all the detailed feedback that you gave on my thesis. It was a great pleasure having you as a supervisor.
- Everyone involved at the Electronic Systems Laboratory (ESL). It was an honour doing my postgraduate studies in such an inspiring and pleasant environment, with such talented people.
- My best friends and fellow students (and ‘office buddies’) at the ESL, Izaak van Zyl and Christiaan Müller, who joined me in the challenge of doing studies here. I would not trade the experience of true camaraderie with them during my postgraduate studies for anything.
- My parents, Kotzé and Marlien Engelbrecht, for their undying love and support and who led me on this aspiring path. A specific thank you to my mother for helping me proofread my thesis.
- A special person whom I hold dear to my heart, Connery Maddick, for their emotional support during the highs and lows throughout this two year endeavour. You helped me become a more fulfilled person and motivated me to see this project through to the end and continue to aspire to greater things still.

# Chapter 1

## Introduction

This thesis presents the design and implementation of an attitude and flight vector recovery system for large transport aircraft. The upset recovery system consists of two major components, namely an optimal trajectory *planning* component and a practical trajectory *execution* component. For the optimal trajectory planning, two optimal control algorithms are investigated, namely dynamic programming (DP) and sequential quadratic programming (SQP). For the trajectory execution, four different control schemes are investigated that use a conventional fly-by-wire flight control system in different configurations to control the aircraft to practically execute the planned optimal trajectory.

In the first part of this thesis, the problem is formulated as an optimal control problem that uses a reduced-order model of the aircraft's dynamics to determine simultaneous bank angle, flight path angle, and airspeed recovery trajectories from an initial flight upset condition. The objective of the optimal control problem is to find the optimal sequence of control inputs to recover the aircraft back to nominal wings level flight with minimum altitude loss, while adhering to the aerodynamic and structural constraints of the aircraft. These constraints translate to state constraints, such as minimum and maximum allowed airspeed, as well as input constraints, such as admissible angle of attack and load factor. The optimal control problem is solved using two separate numerical methods. The first numerical method uses a dynamic programming algorithm that only uses the point mass translational dynamics of the aircraft and omits the input lag dynamics, so that the problem remains tractable for the method, as originally proposed by [1]. This thesis then proposes an alternative numerical method that can overcome the limitations of the dynamic programming method. The proposed method uses a trajectory optimisation technique, known as direct transcription, to transcribe the optimal control problem into a non-linear programming problem (NLP), that is then solved using a gradient-based NLP solver called SQP. The SQP method allows the use of a more representative, higher-order model that includes the input lag dynamics. The aspects of these two methods and their solutions are compared and discussed, such as analogous functionality and recovery strategy.

The second major part of this thesis is the design of conventional aircraft fly-by-wire inner and middle-loop controllers, and the exploration of control schemes that use these flight controllers to execute the planned optimal recovery trajectories. In order to use conventional aircraft controllers to perform trajectory execution and obtain the best performance, four different inner and middle-loop controller configuration schemes are investigated. The control schemes integrate the trajectory solutions from the numerical SQP routine as a guidance law/trajectory planner, and the schemes are then also used to verify the optimal recovery trajectories of the SQP on a full aircraft model in simulation. The aircraft model used is the NASA Generic Transport Model (GTM), a full non-linear, wide-envelope aircraft simulation model that is

able to model the flight mechanics of large transport aircraft in out-of-envelope conditions.

## 1.1 Background

Loss of control (LOC) is the largest contributor to commercial jet aircraft fatal accidents worldwide. [2]. In-flight LOC can be described as motion that is outside the normal operating flight envelope, not predictably altered by pilot control inputs and characterized by the inability to maintain heading, altitude and wings-level flight. A primary cause leading to in-flight LOC situations are aircraft upset conditions. There are numerous other factors that contribute to LOC situations including vehicle damage, inappropriate crew response and external hazards and disturbances, such as adverse weather conditions [2].

Flight upset occurs when an aircraft exits its safe flight envelope. The flight envelope of an aircraft is the domain of flight conditions in which the aircraft can safely be operated without exceeding its aerodynamic and structural limits [1]. The flight envelope is typically represented by state constraints such as airspeed, attitude or angle of attack. Most commercial aircraft have flight envelope protection systems that ensure that the aircraft remains within its operational flight envelope. Extensive literature exists on automatic flight envelope protection, which deals with keeping the aircraft within a safe set of states (envelope).

The main responsibility of preventing the aircraft from leaving the flight envelope, or recovering the aircraft back into the safe flight envelope after it is exited, falls on the pilot. However, increased pilot workload raises the risk of the pilot becoming disorientated, which can lead to inappropriate crew response that worsens the situation. Inappropriate crew responses can also be attributed to limitations in crew training related to LOC situations due to model limitations for full stall and other upset conditions, and subjecting the real aircraft to upset conditions is considered too dangerous. A need therefore exists for an automatic system to assist the pilot in recovering from a flight envelope upset condition.

Passenger aircraft control is still an active field of research, as there is room for improvement on current systems. Aircraft loss of control (LOC) remains an issue, because conventional passenger aircraft controllers are designed for nominal flight. That said, aircraft controllers and autopilots have become very powerful to the point of being able to control the aircraft from take-off all the way to landing at its destination. The problem of aircraft upset recovery, of which trajectory optimisation is a subproblem, is actively being researched. Both offline and online implementations are being explored. The information gained from trajectory solutions for upset recovery that is generated by numerical routines can give pilots valuable insight to help improve pilot training and aviation safety as a whole.

A comprehensive research and development approach to the LOC problem is given by NASA in [2], and highlights that current upset and LOC research is largely concentrated into two broad categories, known as upset prevention and upset recovery. As part of these active research categories, Kwatney et al [3] used constrained control and safe set control theory, that uses bifurcation to analyse how aircraft behave near LOC situations such as a stall. Bifurcation analysis deals with stability analysis of different plant equilibriums or trim conditions and the transition between them. A large part of upset research deals with preventing the aircraft from leaving the safe set (or safe state space) and how to return it into this set. Detecting when an aircraft is in danger of leaving the safe set will also play an important role in these systems. The literature sources [4; 5; 6; 7] present research implementations such as linear quadratic regulator (LQR) state feedback, non-linear regulators and adaptive control for upset prevention and recovery [8].

## 1.2 Research Objectives

This thesis aims to fulfil the following comprehensive list of research objectives/contributions:

1. To gain an understanding of general optimal control theory, trajectory optimisation involving the use of dynamic programming and sequential quadratic programming, numerical method performance evaluation methods, aircraft dynamics and conventional commercial aircraft fly-by-wire flight control laws.
2. Design and implement an optimal control algorithm for attitude and flight vector recovery for large transport aircraft using a similar dynamic programming algorithm as originally proposed by Engelbrecht [1], which will act as a baseline method.
3. Design and implement an alternative optimal control algorithm for attitude and flight vector recovery for large transport aircraft using sequential quadratic programming (SQP) that can overcome the model limitations of the DP method.
4. To critically compare the aspects and performance of the two implemented numerical methods (DP and SQP) for upset recovery.
5. To model, design, implement and verify inner-loop and middle-loop controllers similar to those used in a conventional fly-by-wire flight control architecture for large transport aircraft using the subscale NASA generic transport model (GTM) simulation model.
6. To implement the SQP optimal control algorithm as an outer-loop guidance law that uses the designed inner-loop and middle-loop flight controllers in trajectory execution control schemes to perform upset recovery on the full non-linear NASA GTM simulation model.
7. To investigate different control scheme configurations using conventional middle-loop and inner-loop flight controllers for trajectory execution on a full non-linear aircraft simulation model.

It is assumed that the aircraft is fully functional, i.e. not damaged in any way such as sensor or actuator failures and that the aircraft finds itself in a known attitude and flight vector envelope upset condition. It is further assumed that the aircraft's angular rates and aerodynamic envelope is already recovered and within the inner-loop flight controllers' authority. Upset detection and fault tolerant control was not the purpose of this thesis' research, as extensive literature already exists on these topics.

The optimal control algorithms must determine the optimal state trajectory and optimal sequence of input commands to simultaneously recover the aircraft flight path angle, airspeed and bank angle, while keeping the aircraft within its aerodynamic envelope, i.e. within the inner-loop controllers' authority, and its structural integrity envelope.

## 1.3 Primary Contributions

The research presented in this thesis builds on previous research performed by Engelbrecht (the supervisor for this Master's degree project) for his doctoral dissertation [1]. The first major contribution of this thesis is to propose a new approach to trajectory planning that uses sequential quadratic programming to solve the optimal upset recovery problem as an alternative to the dynamic programming approach used by Engelbrecht [1]. The second major contribution of this thesis is the detailed development and verification of four different control architectures for executing the planned upset recovery trajectories using conventional flight control systems.

A research decision was made to use a flight control system similar to the fly-by-wire flight control systems used on modern commercial passenger aircraft. This decision was motivated by the fact that the research project was performed in collaboration with Airbus, and that their experts were interested in the feasibility of applying the optimal attitude and flight vector recovery to a flight control architecture similar to those used on their aircraft.

## 1.4 Definition of Flight Envelope Upset

The Airplane Upset Recovery Training Aid [9] defines aircraft upset broadly as an airplane in flight unintentionally exceeding the parameters normally experienced in line operations or training. The training aid provides some quantitative upset definitions and manual procedures to recover from commercial aircraft upset. An upset is when an aircraft unexpectedly exits the flight envelope and is approaching unsafe conditions. Furthermore, before the aircraft enters a fully defined upset condition, the pilot should start recovering by stabilising the aircraft back to safe flight path parameters, since being in a flight upset can lead to a loss of control situation.

Qualitatively, a flight envelope upset is associated with high angular rates (usually in the case of a spin), high angles of attack and sideslip angles, unusual pitch and bank angles, airspeeds that are either too low or too high, and high normal load factors. The Boeing Company and the NASA Langley Research Center jointly developed a quantitative set of metrics for defining LOC through a NASA-funded partnership under the Aviation Safety and Security Program. These metrics are collectively known as the Quantitative Loss-of-Control Criteria (QLC) and consists of five subenvelopes [10]. These quantitative definitions are used by the commercial aviation industry to define upsets such as in the Recovery Training Aid. Figure 1.1 illustrates the five envelopes of the QLC.

The Adverse Aerodynamic Envelope is concerned with the angle of attack  $\alpha$  versus the sideslip angle  $\beta$  of the aircraft. The angle of attack is the angle between the aircraft's nose and its velocity vector in the longitudinal plane, while sideslip is the angle between the aircraft's nose and the velocity vector in the lateral plane. The normalised angle of attack and sideslip bounds of the envelope are the minimum and maximum angle of attack and sideslip that can be expected during normal aircraft operation. This envelope indicates stall when the angle of attack limits are exceeded and indicates sideslip-induced roll produced by large sideslip angles.

The Unusual Attitude Envelope is concerned with pitch angle versus bank angle and its bounds are defined by what is considered an 'unusual attitude' in the commercial aviation industry. These bounds are a nose high and nose low pitch angle attitude of positive 20 degrees and negative 10 degrees respectively, and a bank angle attitude of positive or negative 45 degrees. Thus, the aircraft should normally remain within a pitch angle attitude of between positive 20 degrees and negative 10 degrees and remain within a bank angle attitude of lower than 45 degrees

The Structural Integrity Envelope is concerned with normal load factor versus airspeed, and its bounds are defined by the structural design requirements of the aircraft set by the Federal Aviation Regulations. The envelope bounds for normal load factor are -1 g's to +2.5 g's for a flaps-up condition and 0 g's to +2 g's for a flaps-down condition. This envelope indicates accelerated stall, overspeed, and structural overload.

The Dynamic Pitch Control Envelope is concerned with the dynamic pitch attitude versus the pitch authority of the aircraft. The dynamic pitch attitude is the sum of the current pitch attitude and the expected change in pitch in one second. The bounds of this envelope represent the ability to recover the pitch attitude before it has a chance to exceed the Unusual Attitude envelope.

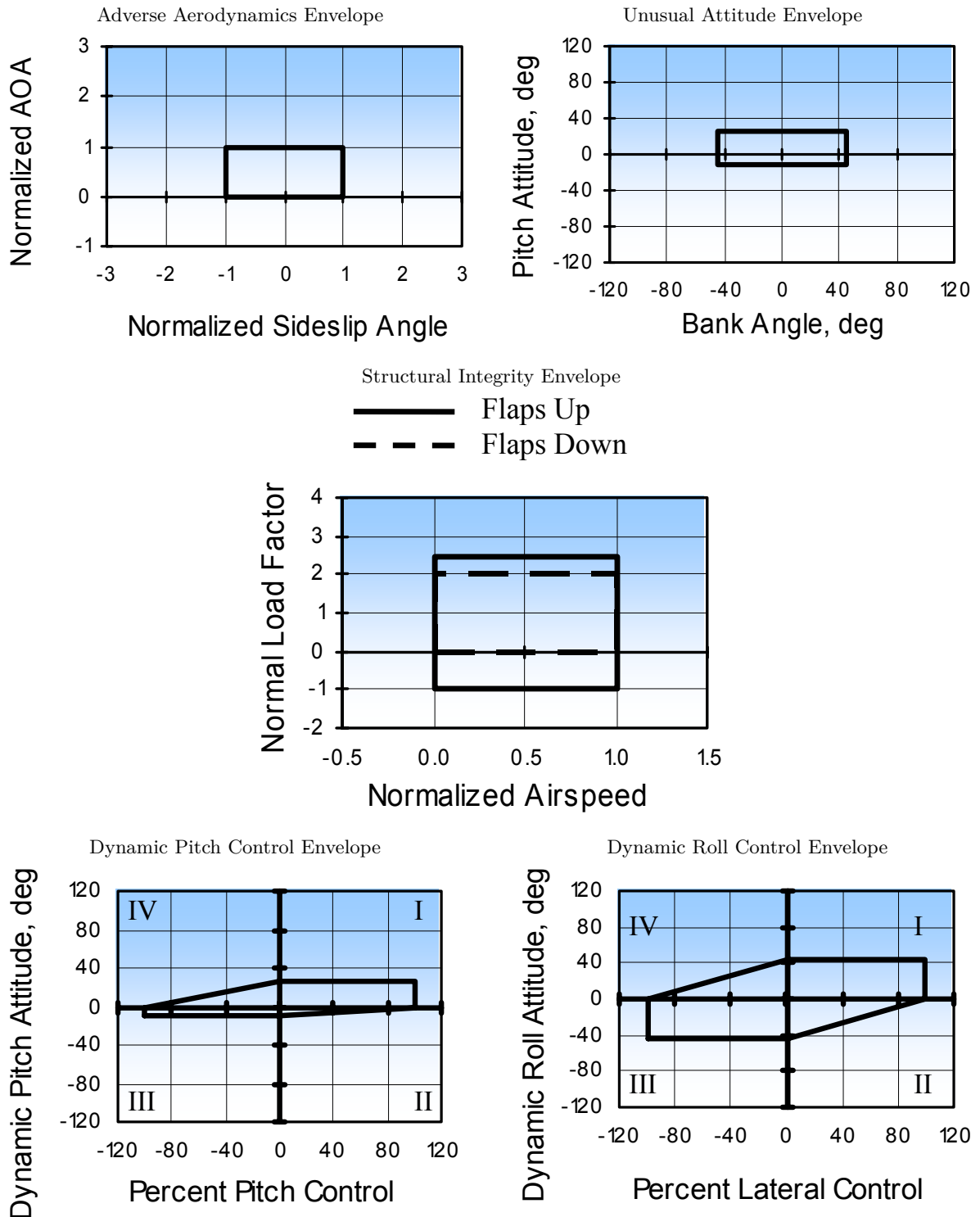


Figure 1.1: Quantitative loss-of-control envelopes defined by Wilborn et al [10]

The Dynamic Roll Control Envelope is concerned with the dynamic roll attitude versus the roll authority of the aircraft. The dynamic roll attitude is the sum of the current roll attitude and the expected change in roll in one second. The bounds of this envelope represent the ability to recover the roll attitude before it has a chance to exceed the Unusual Attitude envelope.

With the definition of flight envelope upset in place, there are certain typical types of upsets in aviation. The FAA Airplane Flying Handbook [11] provides definitions and explanations of these types of upsets, and a brief summary of the major types of upsets are given here with additional reference material from [1].

**Stall** - A stall is when the angle of attack exceeds the critical angle of attack value. The critical angle of attack is the angle of attack value at which maximum lift is generated by the aircraft. Lift generally increases roughly linearly with angle of attack. Beyond the critical angle of attack the lift generated starts to decrease again with increase in angle of attack. It is important to note that a stall is only a function of angle of attack, and not other states such as airspeed or orientation relative to the Earth.

**Spin** - A spin is a type of stall where the aircraft is fully stalled while being in a yawed/sideslip state that results in the aircraft following a downwards corkscrew path. As the aircraft rotates around a vertical axis, the outboard wing is less stalled than the inboard wing, which creates a rolling, yawing, and pitching motion and the aircraft is basically descending due to gravity.

**Unusual Attitude** - Large pitch angles of either a nose up attitude or nose down attitude is considered a flight upset. For large pitch angles it can signify that either the angle of attack or flight path angle is out of envelope. The pitch angle is the angle between the nose of the aircraft and the horizontal plane, while the flight path angle is the angle between the velocity vector of the aircraft and the horizontal plane. If the flight path angle is level, a large pitch angle will indicate that the aircraft is exceeding the stall angle of attack. If the flight path angle is not level, but the angle of attack is within the aerodynamic envelope, a large positive or negative pitch angle indicates that the aircraft is in a steep ascending or descending flight path angle respectively. These are also upset conditions, as these situations can lead to an underspeed or overspeed condition. A steep ascending flight path angle might lead to underspeed if not enough thrust can be produced to counter gravity. A steep descending flight path angle causes gravity to accelerate the aircraft, putting it at risk of eventual overspeed. An unusual attitude also includes high or inverted bank angles (bank angles higher than 90 degrees) where the lift vector is not orientated in a direction to effectively counteract the force of gravity. It can be seen that the aircraft can be in an unusual attitude upset without being in an aerodynamic envelope upset.

**Underspeed** - An underspeed condition is when the aircraft is below its stall airspeed. A stall is only a function of angle of attack, but the slower an aircraft flies, the more angle of attack is needed to produce sufficient lift to counter gravity. Thus the stall airspeed is an airspeed at which the critical angle of attack or higher is needed to produce enough lift.

**Overspeed** - An overspeed condition is considered a state where the aircraft is above its maximum allowed operational airspeed. At airspeeds above this value, excessive structural vibrations might occur that put the aircraft at risk of exceeding its structural integrity envelope.

The standard procedure suggested in the flight manuals is to recover from a stall or spin condition first, and then recover from an unusual attitude. In other words, the high angular rates associated with a spin should be recovered first and the angle of attack reduced to below stall angles of attack, before attempting to recover the other sub-envelopes such as the aerodynamic envelope and usual attitude envelope.

Engelbrecht [1] re-characterised the flight upset envelopes of the QLC slightly to connect them more directly to the dynamics of the aircraft. Flight envelope upset is then characterised by Engelbrecht as consisting of four major sub-envelopes:

- **Aerodynamic Envelope** - normalised angle of attack vs. sideslip angle:  
The Aerodynamic Envelope's boundaries are defined by the maximum and minimum angle of attack and sideslip angles that are expected to be used during normal aircraft operations. Exceeding these limits puts the aircraft at risk of stalling or entering a spin upset.
- **Attitude Envelope** - bank angle vs. pitch angle:  
The Attitude Envelope's boundaries are defined by the aviation industry's definition of an unusual attitude, which are nose up pitch angles greater than 20 degrees, nose down pitch angles of greater than 10 degrees and bank angles beyond  $\pm 45$  degrees.
- **Flight Vector Envelope** - Flight path angle vs airspeed:  
The Flight Vector Envelope's boundaries are defined by expected flight path angles and airspeeds during normal aircraft operation. Large flight path angles imply that the aircraft is ascending steeply and risks entering an underspeed condition. Very negative flight path angles indicate that the aircraft is descending steeply and risks entering an overspeed condition.
- **Structural Integrity Envelope** - normal load factor vs. normalised airspeed:  
The Structural Integrity Envelope's boundaries are defined by the maximum airspeed and normal load factor values allowed for normal aircraft operation to prevent the structural integrity of the aircraft from being compromised. Exceeding these values can lead to critical structural failure or injury to crew and passengers. The envelope's normal load factor limits are -1 g's to +2.5 g's for a flaps-up condition and 0 g's to +2 g's for a flaps-down condition.

When any of these envelopes are exceeded, the aircraft is considered to be in flight envelope upset. To successfully recover from upset, the aircraft must recover all sub-envelopes while adhering to the structural integrity constraint envelope. Engelbrecht [1] proposed a recovery strategy and system that would recover the sub-envelopes successively, by first recovering from high angular rates typical of a stall/spin situations, then recovering the aerodynamic envelope followed by recovering the attitude and flight vector envelopes simultaneously, while staying within the structural integrity envelope at all times.

This thesis focusses on the attitude and flight vector recovery phase as part of a complete recovery and assumes that the angular rates and aerodynamic envelope have already been recovered. Furthermore, it is also assumed that the aircraft is fully functional.

## 1.5 Thesis Outline

Chapter 1 provides the background of the thesis topic, with a discussion of flight envelope upset. With the established scope, the research objectives of the thesis is then stated.

Chapter 2 gives some background on the research vehicle used, namely the NASA Generic Transport Model and establishes the aircraft notation used in this work. Finally the chapter also provides a brief linear analysis of the GTM's aircraft dynamics.

Chapter 3 presents the detailed design process and results of conventional fly-by-wire controllers used on large transport aircraft that were designed using the linear dynamics presented in Chapter 2



Chapter 4 focusses on the two trajectory planning methods used to solve the problem of attitude and flight vector recovery, namely dynamic programming and sequential quadratic programming. A short literature discussion on numerical methods used to solve optimal control problems is given followed by the problem formulation and the reduced-order aircraft dynamic model. A detailed discussion on the implementation and results of both the dynamic programming method and sequential quadratic programming method and how they compare are presented.

Chapter 5 presents the design, implementation and results of control schemes that allow a full aircraft model to execute a planned recovery trajectory provided by an optimal control algorithm acting as a guidance law, such as the numerical methods presented in Chapter 4, using the designed flight controllers of Chapter 3.

Chapter 6 gives the final conclusions on both the trajectory planning methods used for attitude and flight vector recovery and the trajectory control schemes that were implemented, with a brief discussion on possible future work.

## Chapter 2

# The NASA Generic Transport Model

The aircraft model used for this research project is the NASA Generic Transport Model (GTM). The GTM is a subscale model of a generic commercial twin engine jet transport that was created by NASA Langley Research Center. The GTM is a 5.5% dynamically scaled unmanned aerial vehicle (UAV) developed by NASA for out-of-envelope flight tests of large transport aircraft and a full non-linear simulation model of the GTM was made available for research use.

This chapter gives some background on the NASA GTM platform, a description of mathematical aircraft modelling with details on the wide-envelope aerodynamic model implemented on the GTM simulation model, and a brief overview of the high level Simulink model of the GTM.

### 2.1 Background

The GTM, shown in Figure 2.1, is a 5.5% dynamically scaled, twin-turbine, swept-wing aircraft that is part of the Airborne Subscale Transport Aircraft Research (AirSTAR) at NASA Langley Research Center. Its purpose is to provide experimental flight test research for modelling large transport aircraft in adverse or upset conditions [12].



Figure 2.1: Image of the NASA Generic Transport Model

The aircraft is remotely piloted and is equipped with extensive research equipment for data gathering and experimental tests. The GTM's aerodynamic model is built from wind tunnel tests conducted at Langley Research Center. These tests consisted of both static and dynamic wind tunnel tests that included high wind angles (angle of attack and sideslip angle) and high angular rates often experienced in loss-of-control accidents. The resulting model was then validated using free-spin testing [13]. A full non-linear flight dynamics simulation model of the GTM was created in Simulink that includes the wide-envelope aerodynamic model, and was made available by NASA for aircraft upset research use.

## 2.2 Axis Systems and Notation

This section establishes the axis systems and notation that will be used in the modelling of the aircraft flight dynamics. The material presented here is adapted from [14], with additional reference material from [1] and [15], and is reproduced for the convenience of the reader. More in-depth detail is given in Appendix A.

### 2.2.1 Axis Systems

Three standard axis systems are commonly used when modelling aircraft dynamics in conventional aerospace applications: the inertial-, body- and wind-axis systems.

#### Inertial-Axis System

To apply Newton's equations of motion we require an inertial axis system. The standard North-East-Down (NED) axis system, illustrated in Figure 2.2, is often used for this as an inertial reference frame. The NED axis system assumes a flat, non-rotating Earth. For short flight distances, and for the upset recovery trajectories considered in this work, this adequately approximates an inertial reference frame.

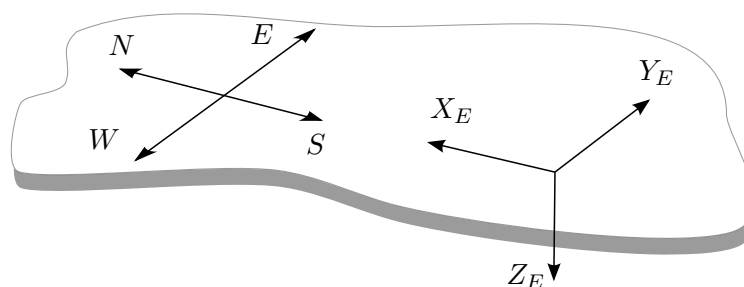


Figure 2.2: North-East-Down axis system

The x-axis points in the North direction, the y-axis points in the East direction, and the z-axis points in the Down direction. This forms a complete right-handed, orthogonal axis system.

### Body-Axis System

The body-axis system, illustrated in Figure 2.3, is fixed to the aircraft body with its origin coinciding with the aircraft's centre of mass. The x-axis lies in the plain of symmetry, pointing through the nose along the zero lift line of the wing. The y-axis lies perpendicular to the plane of symmetry pointing in the direction of the starboard wing. The z-axis points downwards through the underside of the aircraft and completes the right-handed orthogonal axis system.

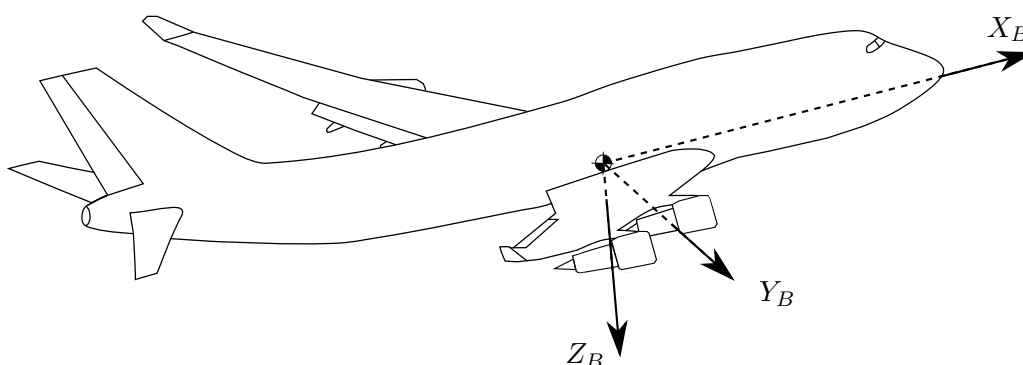


Figure 2.3: Body axis system (Adapted from [16])

### Wind-Axis System

The wind-axis, illustrated in Figure 2.4, can be seen as a version of the body-axis system where the x-axis  $X_W$  points in the direction of the velocity vector of the aircraft. For nominal flight, the z-axis of the wind-axis  $Z_W$  still points roughly downwards through the underside of the aircraft (and remains in the symmetry plane of the aircraft body-axis system) and the y-axis of the wind-axis  $Y_W$  points roughly in the direction of the right-hand (starboard) wing.

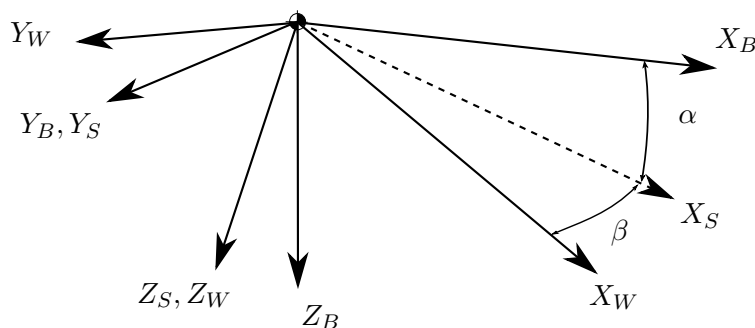


Figure 2.4: Wind-axis system

The angles  $\alpha$  and  $\beta$  are the angle of attack and sideslip angles respectively. The terms  $X_S$ ,  $Y_S$  and  $Z_S$  are the x-axis, y-axis and z-axis of the stability-axis system respectively. The stability axis is obtained by rotating the body-axis around the body y-axis by the angle of attack  $\alpha$ .

The wind axis is turn is obtained by rotating the stability axis around the stability z-axis by the sideslip angle  $\beta$ .

### 2.2.2 Aircraft Notation

The following aircraft notation, illustrated in Figure 2.5, is used in the body-axis system for the aircraft model

- $X, Y, Z$ : Co-ordinates of the force vector in the body-axis (axial, lateral, and normal force)
- $L, M, N$ : Co-ordinates of the moment vector in the body-axis (roll, pitch, and yaw moment)
- $U, V, W$ : Co-ordinates of the linear velocity vector in the body-axis (axial, lateral, and normal velocity)
- $P, Q, R$ : Co-ordinates of the angular velocity vector in the body-axis (roll, pitch, and yaw rates)
- $\delta_A, \delta_R, \delta_E$ : Aileron, rudder and elevator control surface deflections. A positive deflection is defined by it producing a negative moment.

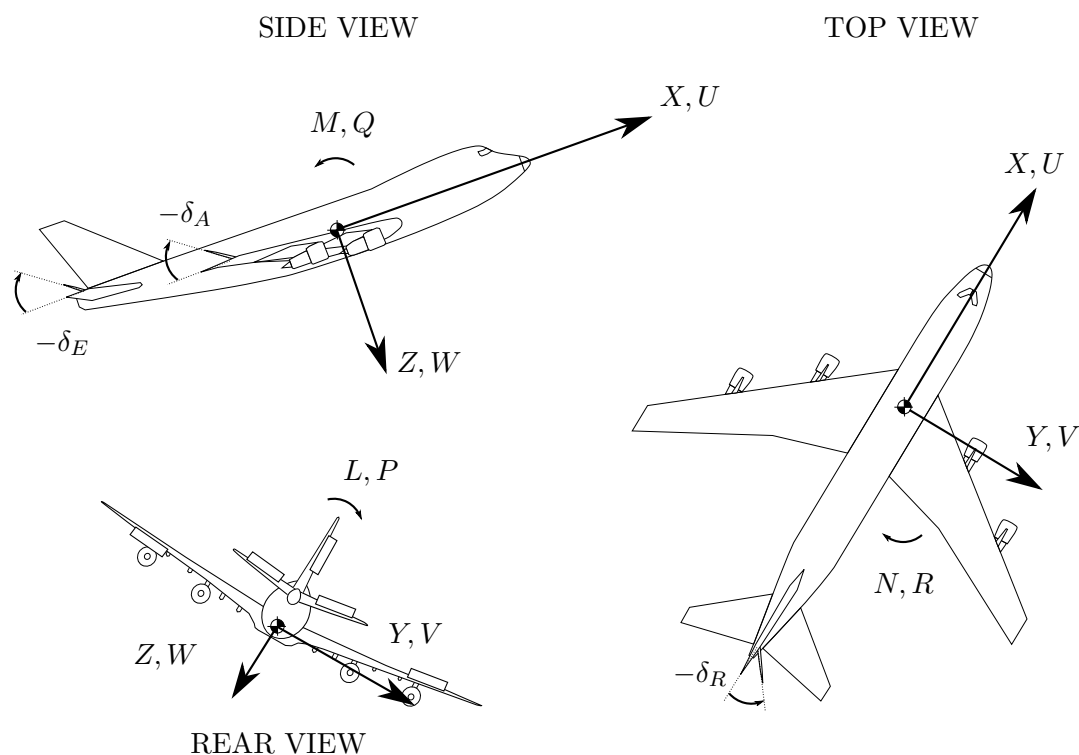


Figure 2.5: Standard aircraft notation

It is often useful to express the velocity variables in polar co-ordinates, as a velocity magnitude and two angles. The magnitude of the aircraft's velocity vector, i.e. the airspeed magnitude  $\bar{V}$ , angle of attack  $\alpha$  and sideslip angle  $\beta$  are important aerodynamic variables that affect the aircraft's dynamics. The angle of attack can be seen as the angle between the aircraft's nose and the velocity vector in the longitudinal plane, and the sideslip angle can be seen as the angle between the nose of the aircraft and the velocity vector in the lateral plane. The relationship of

these aerodynamic variables to the aircraft's body-axis velocity co-ordinates can be expressed as,

$$\bar{V} = \sqrt{U^2 + V^2 + W^2} \quad (2.2.1)$$

$$\alpha = \arctan \frac{W}{U} \quad (2.2.2)$$

$$\beta = \arcsin \frac{V}{\bar{V}} \quad (2.2.3)$$

The inverse relationship is,

$$U = \bar{V} \cos \alpha \cos \beta \quad (2.2.4)$$

$$V = \bar{V} \sin \beta \quad (2.2.5)$$

$$W = \bar{V} \sin \alpha \cos \beta \quad (2.2.6)$$

## 2.3 Six Degrees of Freedom Equations of Motion

We can model the aircraft dynamics quite well using a six-degrees-of-freedom rigid body for control system design purposes. This section presents a brief description on the six degrees of freedom equations of motion model. An in-depth review of the mathematical aircraft modelling is given in Appendix A.

### 2.3.1 Kinetics

Kinetic equations of motion relate the forces and moments acting on an aircraft to the kinematic state of the aircraft, i.e. its position, velocity and acceleration [14]. The relationship can be modelled using Newton's laws of motion in the body-axis of the aircraft. The resulting kinetic equations of motion as derived in [17] are given as,

$$\mathbf{F}_B = m \left( \frac{d\mathbf{V}_B}{dt} + \boldsymbol{\omega}_B \times \mathbf{V}_B \right) \quad (2.3.1)$$

$$\mathbf{M}_B = \frac{d}{dt} (\mathbf{I}_B \boldsymbol{\omega}_B) + \boldsymbol{\omega}_B \times \mathbf{I}_B \boldsymbol{\omega}_B \quad (2.3.2)$$

where  $\mathbf{F}_B$  and  $\mathbf{M}_B$  are the force and moment vectors acting on the aircraft, and  $\mathbf{V}_B$  and  $\boldsymbol{\omega}_B$  are the velocity and angular rate vectors of the aircraft. All these vectors are co-ordinated in the body axis. The term  $m$  is the mass of the aircraft, and  $\mathbf{I}_B$  is the moment of inertia matrix of the aircraft.

These kinetic vector equations can be displayed in scalar form as a set of six non-linear, coupled differential equations that consists of 3 translational and 3 rotational degrees of freedom equations given in Appendix A section §A.2.1.

### 2.3.2 Kinematics

The kinematic equations of motion relate the motion variables (such as linear velocity, angular rate, position and attitude) to each other over time, without reference to the forces and moments [14]. The position and attitude of the aircraft are represented by the following variables,

- $N, E, D$ : Co-ordinates of the position vector in inertial axes (north, east and down position)
- $\Phi, \Theta, \Psi$ : The Euler 3-2-1 attitude parameters of the body-axis system relative to the inertial-axis system (roll, pitch, and yaw angle)

It is necessary to describe the orientation of the body-axis system with respect to the inertial-axis, as the dynamics thus far have been derived in the body-axis.

#### Attitude Representation

The Euler angles are typically used to represent the attitude of the body-axis system relative to the inertial-axis system. The Euler angles use three angles, namely the roll, pitch and yaw angles, of which the order is important. The Euler 3-2-1 sequence starts with the two axis systems aligned and then moves the body-axis system through the following set of ordered rotations,

1. Yaw the body-axis system by rotating it about its z-axis through the yaw angle  $\Psi$
2. Pitch the resulting first intermediate axis system about its y-axis through the pitch angle  $\Theta$
3. Roll the resulting second intermediate axis system about its x-axis through the roll angle  $\Phi$

The co-ordinates of a vector in the inertial-axis system may be transformed to co-ordinates in the body-axis system using the direction cosine matrix, denoted **DCM**, which is a function of the Euler angles as follows,

$$\mathbf{DCM}_{I \rightarrow B} = \begin{bmatrix} C_{\Psi}C_{\Theta} & S_{\Psi}C_{\Theta} & -S_{\Theta} \\ C_{\Psi}S_{\Theta}S_{\Phi} - S_{\Psi}C_{\Phi} & S_{\Psi}S_{\Theta}S_{\Phi} + C_{\Psi}C_{\Phi} & C_{\Theta}S_{\Phi} \\ C_{\Psi}S_{\Theta}C_{\Phi} + S_{\Psi}S_{\Phi} & S_{\Psi}S_{\Theta}C_{\Phi} - C_{\Psi}S_{\Phi} & C_{\Theta}C_{\Phi} \end{bmatrix}, S_{(\ )}=\sin(\ ), C_{(\ )}=\cos(\ ) \quad (2.3.3)$$

Conversely, the co-ordinates of a vector in the body-axis system may be transformed to co-ordinates in the inertial-axis system by using the inverse direction cosine matrix. The DCM is an orthogonal matrix, therefore matrix inverse is simply its transpose,

$$\mathbf{DCM}_{B \rightarrow I} = \mathbf{DCM}_{I \rightarrow B}^{-1} = \mathbf{DCM}_{I \rightarrow B}^T \quad (2.3.4)$$

### Attitude Dynamics

The attitude dynamics equation describing how the body angular rates  $P$ ,  $Q$ , and  $R$  relate to the time rates of changes of the Euler angles is given by,

$$\begin{bmatrix} \dot{\Phi} \\ \dot{\Theta} \\ \dot{\Psi} \end{bmatrix} = \begin{bmatrix} 1 & \sin \Phi \tan \Theta & \cos \Phi \tan \Theta \\ 0 & \cos \Phi & -\sin \Phi \\ 0 & \sin \Phi \sec \Theta & \cos \Phi \sec \Theta \end{bmatrix} \begin{bmatrix} P \\ Q \\ R \end{bmatrix}, \quad |\Theta| \neq \frac{\pi}{2} \quad (2.3.5)$$

Note that a mathematical singularity exists in the dynamics equation when the pitch angle is equal to positive or negative 90 degrees. In the upset recovery problem we will constrain the problem in order to avoid pitch angle attitudes of positive or negative 90 degrees.

### Position Dynamics

The time rate of change of the aircraft position is related to the aircraft velocity co-ordinated in body axes using the inverse DCM through the following kinematic equation,

$$\begin{bmatrix} \dot{N} \\ \dot{E} \\ \dot{D} \end{bmatrix} = \mathbf{DCM}_{B \rightarrow I} \begin{bmatrix} U \\ V \\ W \end{bmatrix} \quad (2.3.6)$$

## 2.4 Forces and Moments

For the conventional aircraft considered in this thesis, there are three main categories that contribute to the forces and moments that act on the aircraft, namely the aerodynamic, thrust and gravitational forces and moments. The force and moment equations can thus be expressed as the sum of the components of the three categories,

$$\mathbf{F} = \mathbf{F}^A + \mathbf{F}^T + \mathbf{F}^G \quad (2.4.1)$$

$$\mathbf{M} = \mathbf{M}^A + \mathbf{M}^T + \mathbf{M}^G \quad (2.4.2)$$

where the superscripts  $A$ ,  $T$ , and  $G$  denote aerodynamic, thrust, and gravitational components respectively. The models for the aerodynamics, thrust, and gravitational forces will be discussed in more details in the following sections.

## 2.5 Aerodynamic Model

The aerodynamic forces and moments introduce most of the uncertainty into the aircraft model and are by far the most complex to model [14]. For subsonic flight the aerodynamic forces and moments are proportional to the dynamic pressure experienced by the aircraft, denoted  $\bar{q}$ ,

$$\bar{q} = \frac{1}{2} \rho \bar{V}^2 \quad (2.5.1)$$



where  $\rho$  is the air density. The aerodynamic force and moment co-ordinates are expanded as follows,

$$X^A = \bar{q}SC_X \quad (2.5.2)$$

$$Y^A = \bar{q}SC_Y \quad (2.5.3)$$

$$Z^A = \bar{q}SC_Z \quad (2.5.4)$$

$$L^A = \bar{q}SbC_l \quad (2.5.5)$$

$$M^A = \bar{q}S\bar{c}C_m \quad (2.5.6)$$

$$N^A = \bar{q}SbC_n \quad (2.5.7)$$

where  $S$  is the wing area,  $b$  is the wingspan,  $\bar{c}$  is the mean aerodynamic chord and  $C(\cdot)$  are the non-dimensional aerodynamic force and moment coefficients. The non-dimensional aerodynamic coefficients capture the aerodynamic properties of the shape of the aircraft, independently of its size.

### Wide-Envelope Aerodynamic Model

The aerodynamic model of the GTM is represented using non-dimensional aerodynamic force and moment coefficients and span a wide aerodynamic envelope. The coefficients are characterised using a summation of a baseline static term and incremental terms for control surface deflections and dynamic effects from angular rates [13],

$$C_i = C_{i,Static}(\alpha, \beta) + \Delta C_{i,\delta}(\alpha, \beta, \delta_A, \delta_E, \delta_R) + \Delta C_{i,\hat{q}_{osc}}(\alpha, \hat{q}_{osc}) + \Delta C_{i,\hat{\omega}_{ss}}(\alpha, \beta, \hat{\omega}_{ss}) \quad (2.5.8)$$

$$C_j = C_{j,Static}(\alpha, \beta) + \Delta C_{j,\delta}(\alpha, \beta, \delta_A, \delta_E, \delta_R) + \Delta C_{j,\hat{p}_{osc}}(\alpha, \hat{p}_{osc}) + \Delta C_{j,\hat{r}_{osc}}(\alpha, \hat{r}_{osc}) + \Delta C_{j,\hat{\omega}_{ss}}(\alpha, \beta, \hat{\omega}_{ss}) \quad (2.5.9)$$

where  $i = X, Z, m$  and  $j = Y, l, n$ . The baseline static coefficients  $C_{Static}$  are only a function of the angle of attack  $\alpha$  and sideslip angle  $\beta$ . The incremental dynamic coefficients are made up of two types of terms. A rotary balance data term  $\Delta C_{\hat{\omega}_{ss}}$  associated with a steady-state angular rate, and forced oscillation data terms  $\Delta C_{\hat{p}_{osc}}$ ,  $\Delta C_{\hat{q}_{osc}}$  and  $\Delta C_{\hat{r}_{osc}}$  associated with oscillatory angular rates. The incremental control surface coefficients  $\Delta C_{\delta}$  model effects from control surface deflections  $\delta_A$ ,  $\delta_E$ , and  $\delta_R$ . More detail on how the rotary balance data and forced oscillation data is implemented in the aerodynamic model is given in Appendix A section §A.3.1.

#### 2.5.1 Thrust Model

The thrust forces and moments are produced by the engines of the aircraft. Here we assume a standard engine configuration for large transport aircraft with twin underwing-mounted engines, one mounted below each wing, such as in the case of the NASA GTM model used in this research project. The thrust forces and moments are functions of the throttle settings of the engines  $\delta_T$ , as well as the air density and the airspeed and can be expressed as,

$$\mathbf{F}^T = \mathbf{f}^T(\delta_T, \rho, \bar{V}) \quad (2.5.10)$$

$$\mathbf{M}^T = \mathbf{m}^T(\delta_T, \rho, \bar{V}) \quad (2.5.11)$$

where  $\mathbf{F}^T$  and  $\mathbf{M}^T$  are the thrust force and moment vectors, and  $\mathbf{f}^T$  and  $\mathbf{m}^T$  are general multivariable non-linear functions that are determined by the characteristics of the specific engines used on the aircraft.

The thrust force vector produced by the engines lies primarily along the positive body x-axis. Due to the alignment of the engines relative to the aircraft body, there may be small components in the body y-axis and body z-axis directions. Typical commercial aircraft have their engines pointed slightly upwards and inwards towards the fuselage. If the engines produce equal thrust as in most cases, the y-axis thrust components should cancel out, but not the z-axis components, resulting in a small thrust component expected in the body z-axis [1].

The thrust moment vector exists due to the engine thrust vector not acting through the aircraft's centre of mass and due to the gyroscoping torques from the angular momentums of the two engines. In the case of aircraft with underwing-mounted engines, the thrust vector acts through a point below the centre of mass, which produces a dominant nose-up pitching moment proportional to the total engine thrust. The rolling and yawing moments due to the thrust vectors not acting through the centre of mass, tend to oppose each other and cancel out, due to the symmetry of the aircraft and the fact that the left and right engines are normally operated to produce equal thrust. The angular momentums of the engines also typically to oppose each other and cancel out, since the engines are designed to rotate in opposite directions and are usually operated at equal engine speeds [1].

### 2.5.2 Gravitational Model

In a flat earth NED axis system, the gravitational acceleration vector is adequately modelled as providing a force equivalent to the aircraft's mass in the down direction, that does not vary with latitude and longitude. The corresponding gravitational force co-ordinate vector in inertial axes is thus,

$$\mathbf{F}_I^G = \begin{bmatrix} 0 \\ 0 \\ mg \end{bmatrix} \quad (2.5.12)$$

where standard gravitational acceleration  $g$  is used. The gravitational forces co-ordinated into the body-axis are functions of the attitude of the body-axis system relative to the inertial-axis system and is obtained using the DCM transformation matrix,

$$\mathbf{F}^G = \mathbf{DCM}_{I \rightarrow B} \mathbf{F}_I^G = \begin{bmatrix} -\sin \Theta \\ \cos \Theta \sin \Phi \\ \cos \Theta \cos \Phi \end{bmatrix} mg \quad (2.5.13)$$

Finally, because in a uniform gravitational field the centre of gravity coincides with the centre of mass, the gravitational force produces no moment on the aircraft. Thus,

$$\mathbf{M}^G = \mathbf{0} \quad (2.5.14)$$

## 2.6 Matlab Simulink Model

The NASA GTM Simulink model was made available for research use by NASA and it is the simulation model used for controller algorithm validation in this thesis. A screenshot of the NASA GTM Simulink model (top level) is shown in Figure 2.6.

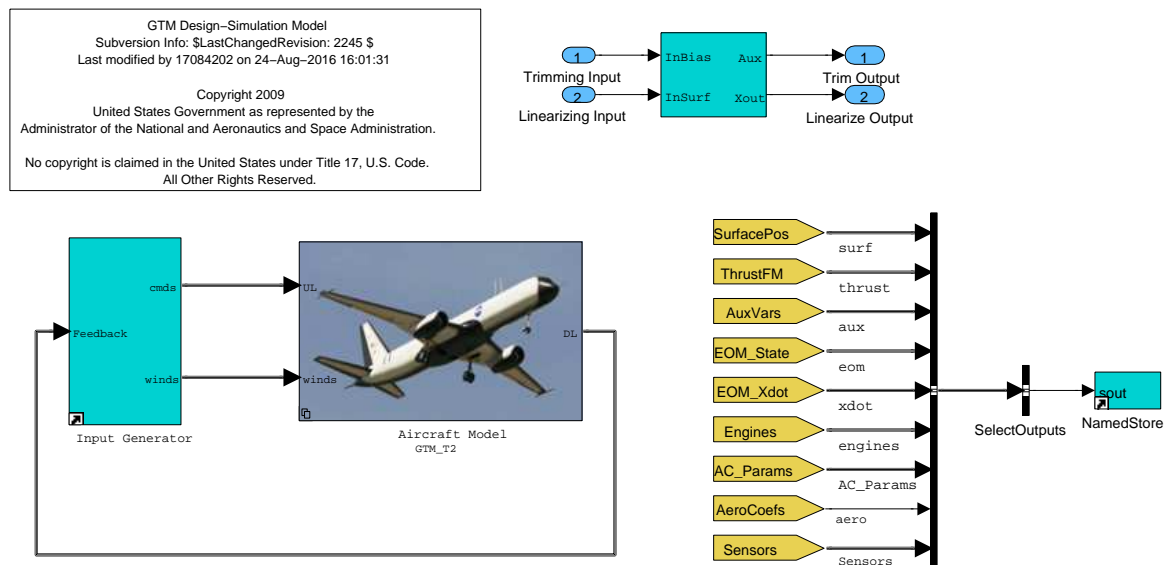


Figure 2.6: Simulink model of the NASA GTM (top level)

The Aircraft model block contains the wide-envelope aerodynamic model provided by NASA and the GTM’s 6DOF EOM model along with other non-linear models such as the thrust model, actuator models and sensor models. The Input Generator block is where the user can implement custom flight control algorithms.

The aerodynamic forces and moments are modelled using non-linear aerodynamic coefficients that are implemented as multi-dimensional lookup tables, consisting of static and dynamic effects. The engine model includes engine alignment, gyroscopic torques and engine dynamics in the form of throttle input to RPM and RPM to thrust non-linear curves. Sensor models include inertial sensors, GPS and for air data such as angle of attack.

## 2.7 Dynamic Scaling

The NASA GTM is a 5.5% dynamically scaled aircraft model and results from the simulation model should be seen with this scaling in perspective. The time scales might seem short, dynamic rates high and linear velocities low compared to what is expected of a large transport aircraft. Table 2.1 can be used to scale the results from the GTM simulations to a full size large transport aircraft for a more realistic view [18].

Table 2.1: Dynamic scaling factors for NASA GTM (Adapted from [1])

Scale Factor		NASA GTM ( $k_{scf} = 1/0.055$ )
Linear dimension	$k_{scf}$	18.18
Linear velocity	$\sqrt{k_{scf}}$	4.264
Linear acceleration	1	1
Angular displacement	1	1
Angular velocity	$1/\sqrt{k_{scf}}$	0.2345
Time	$\sqrt{k_{scf}}$	4.264

## 2.8 Linearisation of GTM Model

Conventional flight controllers must be designed in order to investigate how they can be used to perform trajectory execution. Since the flight controllers are mainly linear controllers, the aircraft dynamics must first be linearised in an appropriate manner before they are designed. The GTM model was trimmed and linearised for a straight and level flight condition and then a brief linear analysis of the system's dynamics are given before the flight controllers are designed in the next chapter.

### 2.8.1 Calculating the Trim States and Inputs

A trim condition is a steady state condition where the aircraft is in a state of equilibrium, meaning all forces and moments acting on the aircraft sum to zero. The most common trim point for aircraft is for straight and level flight. Calculating the trim condition involves solving the non-linear dynamic equations simultaneously using some form of a non-linear root finding algorithm.

The GTM was trimmed using the trim function that is included with its Simulink model. It uses a non-linear constrained minimisation routine on the full 6-degrees-of-freedom Simulink model to calculate the trim variables for a given state condition, knowing that all state derivatives of the Simulink model at the trim condition must equal zero. The selected trim condition was a flight path angle of 0 degrees and an angle of attack of 3 degrees, with all other trim variables free. The result of the trim calculation script is listed in Table 2.2.

Table 2.2: Calculated trim values

State	Value	Units
$\bar{V}_T$	92	kn
$\gamma_T$	0	degrees
$\alpha_T$	3	degrees
$\beta_T$	0	degrees
$\Phi_T$	0	degrees
$T_T$	22.7	%
$\delta_{eT}$	2.66	degrees
$\delta_{aT}$	-0.0045	degrees
$\delta_{rT}$	0.01	degrees

### 2.8.2 Decoupled State Model

The decoupled four state longitudinal and four state lateral linear dynamics were obtained for the chosen trim condition using the provided linearise function that is included with the GTM model. The detail of the mathematical linearisation and derivation process of the linear decoupled dynamics is given in Appendix A section §A.4.1.

The longitudinal linear dynamics modelled given by its state space representation,

$$\Delta \dot{\mathbf{x}}_{long} = \mathbf{A}_{long} \Delta \mathbf{x}_{long} + \mathbf{B}_{long} \Delta \mathbf{u}_{long} \quad (2.8.1)$$

where the longitudinal state and input matrices are given by,

$$\mathbf{A}_{long} = \begin{bmatrix} \frac{\partial \dot{U}}{\partial U} & \bar{V}_T \frac{\partial \dot{U}}{\partial \alpha} & \frac{\partial \dot{U}}{\partial Q} & \frac{\partial \dot{U}}{\partial \Theta} \\ \frac{1}{\bar{V}_T} \frac{\partial \dot{\alpha}}{\partial U} & \frac{\partial \dot{\alpha}}{\partial \alpha} & \frac{1}{\bar{V}_T} \frac{\partial \dot{\alpha}}{\partial Q} & \frac{1}{\bar{V}_T} \frac{\partial \dot{\alpha}}{\partial \Theta} \\ \frac{\partial \dot{Q}}{\partial U} & \bar{V}_T \frac{\partial \dot{Q}}{\partial \alpha} & \frac{\partial \dot{Q}}{\partial Q} & \frac{\partial \dot{Q}}{\partial \Theta} \\ \frac{\partial \dot{\Theta}}{\partial U} & \bar{V}_T \frac{\partial \dot{\Theta}}{\partial \alpha} & \frac{\partial \dot{\Theta}}{\partial Q} & \frac{\partial \dot{\Theta}}{\partial \Theta} \end{bmatrix}, \quad \mathbf{B}_{long} = \begin{bmatrix} \frac{\partial \dot{U}}{\partial \delta_e} & \frac{\partial \dot{U}}{\partial T} \\ \frac{1}{\bar{V}_T} \frac{\partial \dot{\alpha}}{\partial \delta_e} & \frac{1}{\bar{V}_T} \frac{\partial \dot{\alpha}}{\partial T} \\ \frac{\partial \dot{Q}}{\partial \delta_e} & \frac{\partial \dot{Q}}{\partial T} \\ \frac{\partial \dot{\Theta}}{\partial \delta_e} & \frac{\partial \dot{\Theta}}{\partial T} \end{bmatrix} \quad (2.8.2)$$

and the longitudinal states and input vectors consist of perturbation variables from the trim condition,

$$\Delta \mathbf{x}_{long} = [\bar{v} \quad \alpha \quad q \quad \theta]^\top, \quad \Delta \mathbf{u}_{long} = [\delta_e \quad \Delta T]^\top \quad (2.8.3)$$

The lateral linear dynamics are modelled by its state space representation,

$$\Delta \dot{\mathbf{x}}_{lat} = \mathbf{A}_{lat} \Delta \mathbf{x}_{lat} + \mathbf{B}_{lat} \Delta \mathbf{u}_{lat} \quad (2.8.4)$$

where the lateral state and input matrices consist of perturbation variables from the trim condition,

$$\mathbf{A}_{lat} = \begin{bmatrix} \frac{\partial \dot{V}}{\partial V} & \frac{1}{\bar{V}_T} \frac{\partial \dot{V}}{\partial P} & \frac{1}{\bar{V}_T} \frac{\partial \dot{V}}{\partial R} & \frac{1}{\bar{V}_T} \frac{\partial \dot{V}}{\partial \Phi} \\ \bar{V}_T \frac{\partial \dot{P}}{\partial V} & \frac{\partial \dot{P}}{\partial P} & \frac{\partial \dot{P}}{\partial R} & \frac{\partial \dot{P}}{\partial \Phi} \\ \bar{V}_T \frac{\partial \dot{R}}{\partial V} & \frac{\partial \dot{R}}{\partial P} & \frac{\partial \dot{R}}{\partial R} & \frac{\partial \dot{R}}{\partial \Phi} \\ \bar{V}_T \frac{\partial \dot{\Phi}}{\partial V} & \frac{\partial \dot{\Phi}}{\partial P} & \frac{\partial \dot{\Phi}}{\partial R} & \frac{\partial \dot{\Phi}}{\partial \Phi} \end{bmatrix}, \quad \mathbf{B}_{lat} = \begin{bmatrix} \frac{1}{\bar{V}_T} \frac{\partial \dot{V}}{\partial \delta_a} & \frac{1}{\bar{V}_T} \frac{\partial \dot{V}}{\partial \delta_r} \\ \frac{\partial \dot{P}}{\partial \delta_a} & \frac{\partial \dot{P}}{\partial \delta_r} \\ \frac{\partial \dot{R}}{\partial \delta_a} & \frac{\partial \dot{R}}{\partial \delta_r} \\ \frac{\partial \dot{\Phi}}{\partial \delta_a} & \frac{\partial \dot{\Phi}}{\partial \delta_r} \end{bmatrix} \quad (2.8.5)$$

and the lateral states and input vectors are,

$$\Delta \mathbf{x}_{lat} = [\beta \quad p \quad r \quad \phi]^\top, \quad \Delta \mathbf{u}_{lat} = [\delta_a \quad \delta_r]^\top \quad (2.8.6)$$

### 2.8.3 Natural Modes of Motion

A brief analysis of the GTM's natural modes of motion are provided in this section with references from course notes [14].

The dynamic response of the linearised aircraft dynamics is governed by the system's poles, which are the eigenvalues of the system's state matrix  $\mathbf{A}$ . The poles of the longitudinal dynamics and lateral dynamics are the eigenvalues of the respective linear systems'  $\mathbf{A}_{long}$  and  $\mathbf{A}_{lat}$  state matrices. A summary of the natural modes of motion for the decoupled linear systems is provided in this section.

Table 2.3: Longitudinal characteristics.

Characteristic	Short-Period Mode	Phugoid Mode
$\omega_n$ (rad/s)	8.09	0.247
$\zeta$	0.46	0.076

### Longitudinal Modes of Motion

The longitudinal system poles consist of two complex pole pairs. The high frequency pole pair is referred to as the short-period mode and the low frequency pair is referred to as the phugoid mode. The natural frequency and damping characteristics of the longitudinal modes of motion are listed in Table 2.3.

The short-period mode describes the aircraft's tendency to realign itself with the velocity vector when disturbed longitudinally. The phugoid mode is a largely a slow kinematic mode of motion and describes the exchange of potential and kinetic energy when the aircraft is disturbed from trimmed flight [14].

### Lateral Modes of Motion

The lateral system poles consist of two real poles and a complex pole pair. These modes of motion are commonly referred to (from highest to lowest natural frequency) as the roll, dutch roll and spiral modes. The natural frequency and damping characteristics of the lateral modes of motion are listed in Table 2.4.

Table 2.4: Longitudinal characteristics.

Characteristic	Roll Mode	Dutch Roll Mode	Spiral Mode
$\omega_n$ (rad/s)	7.56	7.1	0.053
$\zeta$	1	0.153	1

The fast roll mode describes the roll rate dynamics of an aircraft. When an aircraft experiences a roll moment disturbance, the roll rate will initially start to grow but will quickly be damped by the wing's natural roll damping to a constant roll rate. The fast nature of this mode usually makes aircraft appear to always operate at a constant roll rate. The dutch roll mode and is usually very poorly damped. Its similar to the short-period mode, as it describes the tendency of the aircraft to align itself with the oncoming airflow when disturbed laterally. The spiral mode is usually quite slow and it is also very common for the mode to be slightly unstable. Like the phugoid mode, the spiral mode is a largely kinematic mode of motion and describes the tendency of the aircraft to restore itself to wings level flight or diverge from wings level flight when laterally disturbed [14].

The pole plots of the NASA GTM's natural modes of motion are provided in Appendix A section §A.4.2. The linearised model was validated by comparing its responses to the responses of the full non-linear model. The decoupled linear models corresponds quite well with the non-linear simulation model and the linear models can be used in the conventional linear aircraft controller design process in the next chapter. The plots of the validation experiments can be found in Appendix A section §A.4.2.

## 2.9 Conventional Fly-By-Wire Controllers

One of the requirements for the attitude and flight vector recovery system is that it should use flight controllers that are representative of conventional fly-by-wire controllers typically used on commercial airliners. Since the NASA GTM simulation model does not include fly-by-wire controllers, representative inner-loop and middle-loop controllers had to be designed specifically for the aircraft and then added to the existing simulation model in the Input Generator block. A simulation model of an Airbus A330 was used as reference to derive the specifications for the design of the GTM-specific fly-by-wire controllers. The following inner-loop and middle-loop controllers were designed and implemented for the NASA GTM:

Inner-loop controllers:

- Normal acceleration controller (DQ controller)
- Roll and sideslip controller (DPDR controller using roll rate reference)

Middle-loop controllers:

- Flight path angle controller (FPA controller)
- Airspeed controller (Autothrust controller)
- Roll and sideslip controller (DPDR controller using bank angle reference)

The DQ controller damps the short-period mode and controls the normal acceleration of the aircraft. The DQ inner-loop controller is commanded by the middle-loop flight path angle controller to control the flight path angle of the aircraft. The middle-loop autothrust controller uses the engine throttle input to control the airspeed of the aircraft. The DPDR controller controls the bank angle and sideslip of the aircraft as a middle-loop controller. Alternatively, the DPDR controller can control the roll rate directly instead of the bank angle and then acts as an inner-loop controller.

The next chapter will give an overview of the fly-by-wire flight control architecture, and will present the detailed design and verification of the individual controllers.

## Chapter 3

# Conventional Flight Controller Design

This chapter presents the design of the flight controllers for the NASA GTM aircraft that are representative of conventional fly-by-wire controllers typically used on commercial airliners. The flight controller dynamics are included in the aircraft model that is used to *plan* the optimal recovery trajectories (see Chapter 4) and the flight controllers themselves may be used in different configurations to control the aircraft to *execute* the planned trajectories (see Chapter 5). Since the NASA GTM is not supplied with fly-by-wire controllers included, some representative inner-loop and middle-loop controllers had to be designed specifically for the aircraft and then added to the existing simulation model.

The design of the conventional fly-by-wire flight controllers for the NASA GTM aircraft will be presented as follows: First some background is given on the typical fly-by-wire control architecture used on Airbus aircraft. For the purposes of attitude and flight vector recovery, only inner-loop normal law and middle-loop attitude hold controllers are designed. The specifications for the flight controllers were selected to provide closed-loop responses similar to those of a typical commercial airliner. A high-fidelity simulation model of a commercial airliner that was provided by a third party was used as the reference model for the closed-loop specifications. The design of the conventional longitudinal and lateral controllers is discussed and the closed-loop step responses are simulated using both the linearised aircraft dynamics and the full non-linear GTM aircraft. For the purposes of upset recovery, the conventional outer-loop guidance laws, such as the cross-track controller and the altitude controller, are not implemented. Instead, it is assumed that the reference commands for the middle-loop controllers shall be provided by the upset recovery algorithm that acts as a guidance law. Finally, the command tracking performances of the various inner-loop and middle-loop flight controllers are tested for both step references and ramp references, using the full GTM aircraft model with the flight controllers implemented. These tests provide an early indication of the flight control system's ability to track time-varying reference trajectories.

### 3.1 Conventional Commercial Aircraft Fly-by-Wire

This section gives an overview of fly-by-wire (FBW) implemented on commercial Airbus aircraft. This thesis uses controller architecture designs established from previous research in a thesis done by [16] which was part of a research collaboration with Airbus. In this section a brief recall of an overview on Airbus FBW control laws and systems by Trollip [16] is given, which references from a publication in the 1994 International Journal of Control by Favre [19].



The term fly-by-wire is derived from the control system architecture used on modern transport aircraft. The architecture uses electronic wires to transmit inputs which are converted from physical pilot inputs to a Flight Control System (FCS). The FCS then determines the command signals that need to be given to the aircraft's physical control surface actuators. Automatic commands can also be given to the FCS to perform actions without pilot inputs, which can help stabilise the aircraft and prevent unsafe operation.

### 3.1.1 General Structure and Objective

Favre [19] states that the main objective of integrated flight controls is to improve the natural response of the aircraft. The general fly-by-wire (FBW) control law is illustrated in Figure 3.1. The use of onboard flight computers gives easy access to sensor data enabling simple control objectives to be used [19].

Longitudinal control is achieved with load factor objectives while lateral control is achieved by roll rate, sideslip and bank angle objectives. The FBW autopilot can be seen as an inner-loop controller, while the pilot acts as an outer-loop guidance law that manages the vertical load factor, roll rate, sideslip and bank angle objectives, i.e. commands. Significant redundancy is typically implemented in practice by using multiple full authority flight computers. All flight computers are active simultaneously and if one fails, nominal operation and safety is still achieved. A mechanical backup is also implemented for certain control surfaces [19].

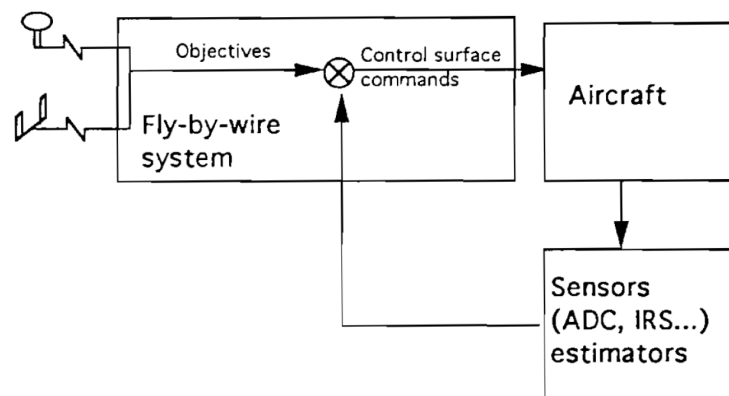


Figure 3.1: General fly-by-wire controller structure (Taken from [19])

Each flight computer's longitudinal and lateral law has different control law configurations. There are three different modes, namely the Normal mode, Alternative mode and Direct mode. The Normal mode has flight envelope protection systems enabled (such as an angle of attack limiter), where the Alternative mode has them disabled. The Direct mode is for severe failure states where the control surface actuators must be controlled directly using rudimentary feedback [20]. In this thesis, we will focus on the Normal mode configuration of the control laws, referred to as the Normal laws. The operation of the Normal laws are highly dependent on full sensor data availability, as the control laws use various states of the aircraft (which are typically estimated by onboard sensors) in their feedback laws.

### 3.1.2 Longitudinal Law

The longitudinal Normal law implements a vertical load factor controller and its architecture is illustrated in Figure 3.2. The controller allows the pilot to control the vertical load factor of

the aircraft, which is a measure of the force experienced by the aircraft in the direction of its lift vector. The load factor objective is obtained by translating the pilot's stick force into a vertical load factor command. The longitudinal controller uses pitch rate and load factor feedback with integral control and controls the short-period mode of the aircraft. The integrator is used for short term stability and precision tracking. Typically a load factor limiter is also incorporated to ensure that the aircraft remains within the structural integrity envelope, i.e. does not experience excessive stress (or load) on the aircraft structure.

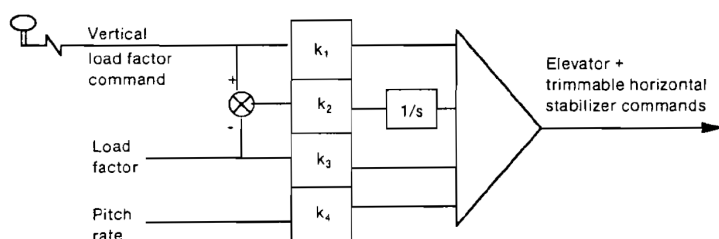


Figure 3.2: Conventional longitudinal law architecture (Taken from [19])

The controlled modes are selected so that they resemble the natural aircraft modes of motion to minimise control surface activity and that the response is similar to the Direct mode laws. By using this command law the longitudinal static stability of the aircraft is close to neutral while the phugoid mode becomes a highly damped mode [19]. A neutrally static aircraft means that if the aircraft is disturbed, the aircraft will tend to remain in the disturbed state. In other words, a neutrally static aircraft will not diverge further away from the disturbed state and will not tend to go back towards its original undisturbed state. Neutral longitudinal stability implies that the aircraft's pitching moment is independent of its angle of attack state. The Normal law incorporates angle of attack protection at low speeds (to prevent stall), high speed protection and pitch rate protection. A general law is typically implemented in practice which uses gains scheduling, i.e. tabulation of control gains for different flight conditions such as airspeed, centre of gravity location and high lift configurations.

### 3.1.3 Lateral Law

The lateral Normal law controls the bank angle and sideslip of the aircraft, and its architecture is illustrated in Figure 3.3. The controller allows the pilot to control the roll rate of the aircraft by integrating the roll rate command into a bank angle command objective. The roll rate objective is obtained by translating sidestick inputs into roll rate commands. The sideslip objective is obtained by translating the pilot's rudder pedal inputs into a sideslip command. Additionally, the rudder pedals also cause a bank angle command. This combination of commands allows the controller to imitate a natural aircraft response which is similar to what a pilot would expect from an aircraft without fly-by-wire. The sideslip is minimised by the controller when a bank angle is commanded using automatic turn co-ordination to reduce pilot workload.

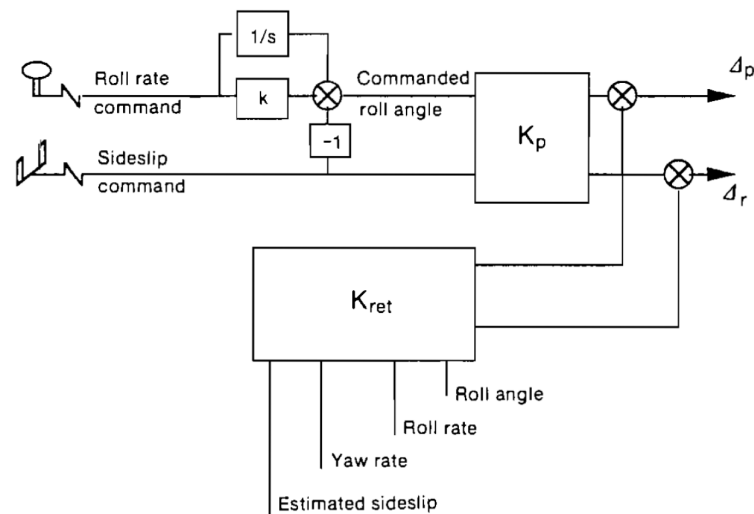


Figure 3.3: Conventional lateral law architecture (Taken from [19])

The lateral controller uses full lateral state feedback of the estimated sideslip, roll rate, yaw rate and bank angle. Steady state tracking of the commanded bank angle and sideslip is achieved using the feed-forward mixing gain matrix  $K_p$ . The lateral law augments the aircraft stability by increasing dutch roll damping beyond 0.6 without exciting the lateral modes. This leaves the roll mode unchanged and increases the spiral mode stability. Similarly to the longitudinal law, the lateral control law is designed so that the controlled modes resemble the natural modes of the aircraft to minimise control surface activity and to provide a response similar to the Direct Law [19].

### 3.1.4 Integrated Controller Architecture for the GTM Aircraft

The conventional flight controller architecture that was implemented on the GTM is illustrated in Figure 3.4. A simulation model of an Airbus A330 was used as reference for the designed Normal longitudinal and lateral laws in this thesis. The Airbus longitudinal controller is called the DQ law and is a normal (vertical) acceleration controller. Normal acceleration is simply divided by the gravity constant to obtain load factor and is thus practically the same as a normal load factor. The Airbus lateral controller is called the DPDR law and controls the roll rate, bank angle and sideslip.

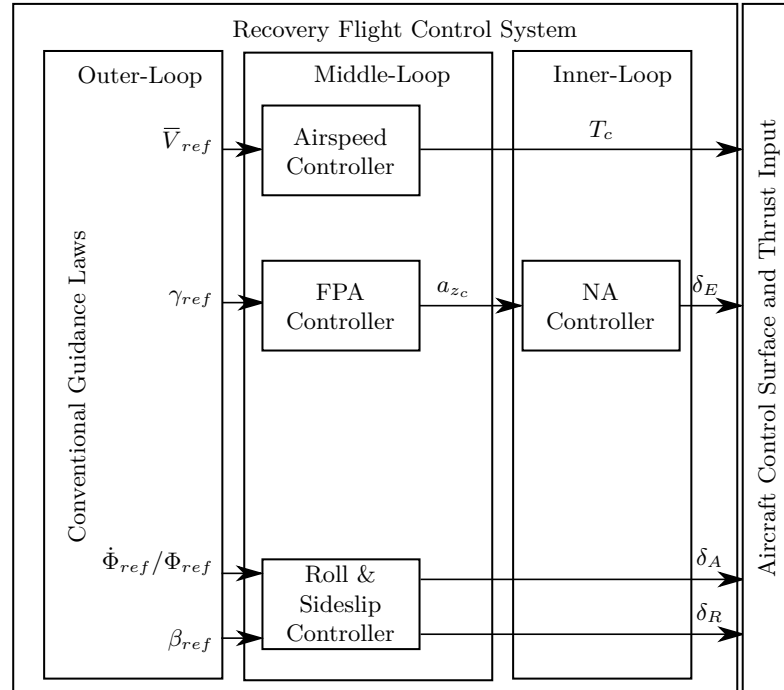


Figure 3.4: Conventional flight controller architecture

Outer-loop guidance laws are typically implemented on commercial aircraft that use the inner-loop fly-by-wire controllers to control the flight path of the aircraft. Conventional guidance laws believed to be representative of those used on commercial large transport aircraft are presented by Trollip in his thesis [16]. The conventional longitudinal guidance law implemented by Trollip uses an altitude hold controller with the objective of maintaining a reference altitude. He stated that the altitude hold controller could use either a flight path angle middle-loop controller or a climb rate middle-loop controller. The flight path angle controller is used to control the altitude during take-off and landing, while the climb rate controller is used during normal cruising conditions. For lateral guidance a cross track controller was implemented by Trollip which guides the aircraft to track a given ground track[16].

For this project the full conventional guidance laws were not implemented and only the flight path angle middle-loop controller is used to control the longitudinal attitude of the aircraft. Instead, the reference commands for the middle-loop controllers are provided by the upset recovery trajectory planner which assumes the role of the outer-loop guidance controllers.

Section §3.2 presents the design of the longitudinal control system, which consists of an inner-loop normal acceleration controller (‘DQ controller’), a middle-loop flight path angle controller, and a middle-loop airspeed controller (‘autothrust’). Section §3.3 presents the design of the lateral control system, which consists of a combined roll rate, bank angle and sideslip angle controller (‘the DPDR controller’). The DPDR controller can be used both as an inner-loop roll rate controller and as a middle-loop bank angle controller.

## 3.2 Longitudinal Control System Design

This section describes the design and verification of the longitudinal controllers, namely the normal acceleration controller (‘DQ controller’), the flight path angle controller, and the airspeed controller (‘autothrust controller’). The DQ controller acts as both a pitch rate damper

and a normal acceleration controller, and essentially damps the short period mode of the aircraft while also controlling the measured normal acceleration to track a commanded normal acceleration. The flight path angle controller and airspeed controller were designed separately, but because of the strong coupling between the axial and vertical dynamics of the aircraft, the flight path angle controller and the airspeed controller were tested together in simulation.

### 3.2.1 Normal Acceleration Controller ('DQ Law')

The inner-loop normal acceleration controller ('DQ controller') performs pitch rate damping and normal acceleration control. A block diagram of the DQ controller architecture is shown in Figure 3.5. The controller uses feedback from a pitch rate sensor and a normal acceleration sensor to actuate the elevator deflection. The control strategy is full state feedback with integral control. The control design is performed using a reduced-order model of the longitudinal dynamics of the aircraft.

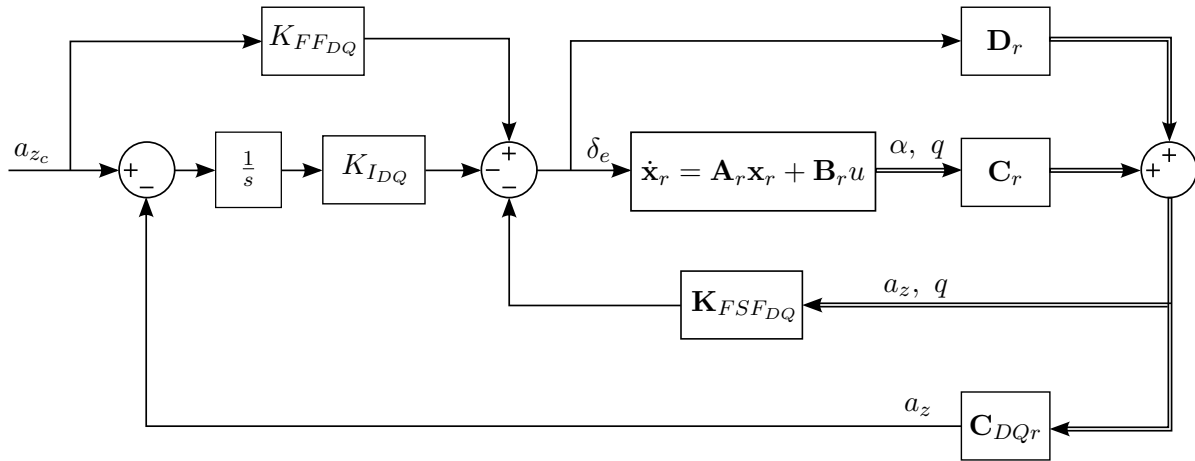


Figure 3.5: DQ controller architecture

### Linearising Normal Acceleration

Since we wish to perform normal acceleration control, we need a linearised output equation that relates the normal acceleration output variable to longitudinal state variables. We therefore start with the full-order non-linear equation that relates the normal acceleration to all four longitudinal states (airspeed, angle of attack, pitch rate, and pitch angle) and then derive the linearised output equation that can be used in the design of the normal acceleration controller. The non-linear equation that describes the normal acceleration  $a_z$  experienced in the body-axis is given by,

$$a_z = \left[ \bar{q} S C_Z + mg \cos \Theta \cos \Phi \right] \frac{1}{m} \quad (3.2.1)$$

where  $\bar{q}$  is the dynamic pressure and  $C_Z$  is the non-dimensional body z-axis aerodynamic coefficient which is a non-linear function of the states and inputs of the aircraft that can be expressed as,

$$C_Z = \mathcal{F}(\mathbf{x}, u) \quad (3.2.2)$$

with,

$$\mathbf{x} = \left[ \bar{V} \quad \alpha \quad Q \quad \Theta \right]^T \quad (3.2.3)$$

$$u = \delta_E \quad (3.2.4)$$

We only consider  $C_Z$  to be a function of the longitudinal states and inputs, since we are using the linear decoupled longitudinal model of the aircraft in the design of the DQ controller. For a level flight trim condition, the steady-state flight path angle and bank angle are zero  $\gamma = 0$ ,  $\Phi = 0$ , and the non-dimensional aerodynamic coefficient in the body z-axis  $C_Z$  is related to the wind-axis aerodynamic coefficients by,

$$C_Z = -C_D \sin \alpha - C_L \cos \alpha \quad (3.2.5)$$

where  $C_L$  is the non-dimensional aerodynamic lift coefficient and  $C_D$  is the non-dimensional aerodynamic drag coefficient. For a level flight trim condition it can be assumed that the angle of attack  $\alpha$  is sufficiently small and that the lift is an order magnitude greater than the drag. Thus,

$$C_Z \approx -C_L \quad (3.2.6)$$

Linearising the normal acceleration about trim we have,

$$a_z = a_{zT} + \Delta a_z \quad (3.2.7)$$

and for the system to be in equilibrium at trim the trim normal acceleration must equal zero  $a_{zT} = 0$ . The linear approximation for the normal acceleration as a function of the states and inputs is then given by,

$$\Delta a_z \approx \mathbf{C}_{a_z} \Delta \mathbf{x} + D_{a_z} \Delta u \quad (3.2.8)$$

where,

$$\mathbf{C}_{a_z} = \left[ \frac{\partial a_z}{\partial \bar{V}} \quad \frac{\partial a_z}{\partial \alpha} \quad \frac{\partial a_z}{\partial Q} \quad \frac{\partial a_z}{\partial \Theta} \right]_T \quad (3.2.9a)$$

$$D_{a_z} = \left. \frac{\partial a_z}{\partial \delta_E} \right|_T \quad (3.2.9b)$$

The partial derivative terms in the vector  $\mathbf{C}_{a_z}$  and of the term  $D_{a_z}$  are calculated and evaluated at the trim condition.

### Reduced-Order Model

The design of the DQ controller is performed using a reduced-order model of the full fourth-order model of the aircraft's longitudinal dynamics. The reduced-order model includes only the short-period mode fast rotational dynamics present in the angle of attack  $\alpha$  and the pitch rate  $q$ , and consider the airspeed and flight path angle to be slowly-varying parameters and

therefore assumed to be constants in the model. This simplifies the design process since only the short-period poles have to be considered. The reduced-order longitudinal state space model is given by,

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

$$\begin{bmatrix} \dot{\alpha} \\ \dot{q} \end{bmatrix} = \begin{bmatrix} \frac{\partial \dot{\alpha}}{\partial \alpha} & \frac{\partial \dot{\alpha}}{\partial Q} \\ \frac{\partial \dot{q}}{\partial \alpha} & \frac{\partial \dot{q}}{\partial Q} \end{bmatrix} \begin{bmatrix} \alpha \\ q \end{bmatrix} + \begin{bmatrix} \frac{\partial \dot{\alpha}}{\partial \delta_e} \\ \frac{\partial \dot{q}}{\partial \delta_e} \end{bmatrix} \delta_e \quad (3.2.10)$$

Figure 3.6 shows the simulated responses of both the full-order and the reduced-order models to the same elevator step command (1 degree step). These responses verify that the reduced-order model is a good approximation of the full model over the time scales of the short-period dynamics.

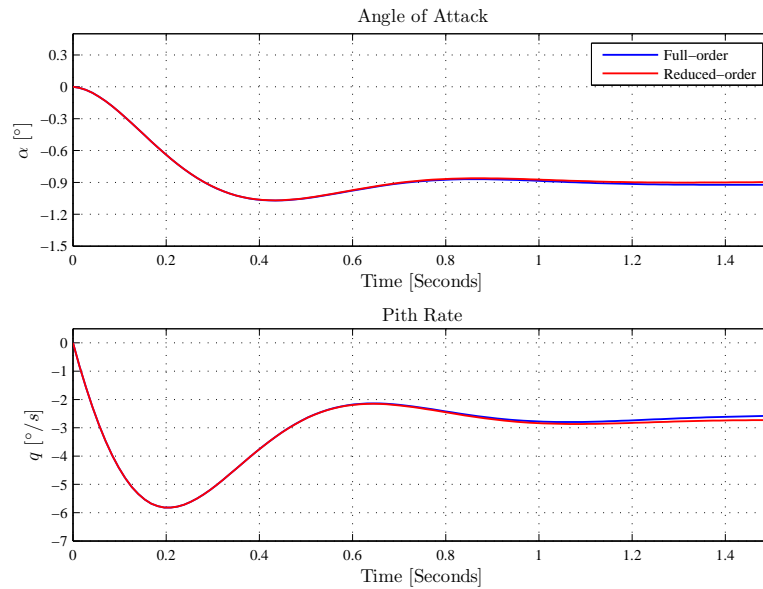


Figure 3.6: Comparison between full-order and reduced-order longitudinal model response for unit elevator step command

The linear normal acceleration output equation for this reduced-order model can then be expressed as,

$$a_z = \mathbf{C}_{r_{a_z}} \mathbf{x}_r + D_{a_z} u \quad (3.2.11)$$

where,

$$\mathbf{C}_{r_{a_z}} = \left[ \frac{\partial a_z}{\partial \alpha} \quad \frac{\partial a_z}{\partial Q} \right]_T \quad (3.2.12a)$$

$$D_{a_z} = \left. \frac{\partial a_z}{\partial \delta_E} \right|_T \quad (3.2.12b)$$

The partial derivative terms in the reduced-order output vector  $\mathbf{C}_{r_{a_z}}$  are obtained from the corresponding terms in the full-order linear normal acceleration output vector of Equation 3.2.9a.

### Design

The DQ controller that regulates the normal acceleration will be designed using the reduced-order model of the longitudinal dynamics. The plant for the controller is a transformed form of the reduced-order model of Equations 3.2.10 and 3.2.11 with the angle of attack state replaced with a normal acceleration state. The normal acceleration dynamics that serves as the plant for the DQ controller is therefore represented by the following state space model,

$$\begin{aligned}\dot{\mathbf{x}}_{DQr} &= \mathbf{A}_{DQr}\mathbf{x}_{DQr} + \mathbf{B}_{DQr}u \\ a_z &= \mathbf{C}_{DQr}\mathbf{x}_{DQr} + D_{DQr}u\end{aligned}\tag{3.2.13}$$

where,

$$\mathbf{x}_{DQr} = [a_z \quad q]^\top\tag{3.2.14a}$$

$$u = \delta_e\tag{3.2.14b}$$

$$\mathbf{A}_{DQr} = \mathbf{C}_r\mathbf{A}_r\mathbf{C}_r^{-1}, \quad \mathbf{C}_r = \begin{bmatrix} \mathbf{C}_{r_{a_z}} \\ 0 \quad 1 \end{bmatrix}\tag{3.2.14c}$$

$$\mathbf{B}_{DQr} = \mathbf{C}_r\mathbf{B}_r\tag{3.2.14d}$$

$$\mathbf{C}_{DQr} = [1 \quad 0], \quad D_{DQr} = 0\tag{3.2.14e}$$

The input feed-forward term  $D_{a_z}$  of Equation 3.2.12b is omitted in the reduced-order model as it introduces an undesirable non-minimum phase response which can complicate the design process. The direct contribution of the elevator input to the normal acceleration output of the aircraft is typically significantly smaller compared to the contribution of the states. The term  $\mathbf{D}_r = [D_{a_z} \quad 0]$  is therefore still included in Figure 3.5 for the sake of comprehensiveness. A full-state feedback controller with integral control will be designed for the reduced-order plant model of Equation 3.2.13. Augmenting the system with an integrator eliminates the steady state error in normal acceleration output, but produces slow integrator dynamics that result in a long settling time in the closed-loop response. A pole-zero cancellation technique is used to remove the integrator dynamics by introducing a feed-forward term  $K_{FFDQ}$  from the reference input to the plant input. The feed-forward term effectively introduces a zero that is then placed on the closed-loop pole that originates from the open-loop integrator.

The integrator dynamics can be written as,

$$\begin{aligned}\dot{x}_{IDQ} &= a_{z_c} - a_z \\ &= a_{z_c} - (\mathbf{C}_{DQr}\mathbf{x}_{DQr})\end{aligned}\tag{3.2.15}$$



where  $x_{IDQ}$  is the time integral of the normal acceleration error,

$$x_{IDQ} = \int (a_{z_c} - a_z) dt \quad (3.2.16)$$

The state space model of the reduced-order longitudinal dynamics augmented with the integrator state for the DQ controller is then given by,

$$\begin{bmatrix} \dot{\mathbf{x}}_{DQr} \\ \dot{x}_{IDQ} \end{bmatrix} = \begin{bmatrix} \mathbf{A}_{DQr} & \mathbf{0}_{2 \times 1} \\ -\mathbf{C}_{DQr} & 0 \end{bmatrix} \begin{bmatrix} \mathbf{x}_{DQr} \\ x_{IDQ} \end{bmatrix} + \begin{bmatrix} \mathbf{B}_{DQr} \\ 0 \end{bmatrix} \delta_e + \begin{bmatrix} \mathbf{0}_{2 \times 1} \\ 1 \end{bmatrix} a_{z_c} \quad (3.2.17)$$

with the full-state feedback control law given by,

$$\begin{aligned} u = \delta_e &= -\mathbf{K}_{DQr} \begin{bmatrix} \mathbf{x}_{DQr} \\ x_{IDQ} \end{bmatrix} + K_{FFDQ} a_{z_c} \\ &= -\begin{bmatrix} \mathbf{K}_{FSFDQ} & K_{IDQ} \end{bmatrix} \begin{bmatrix} \mathbf{x}_{DQr} \\ x_{IDQ} \end{bmatrix} + K_{FFDQ} a_{z_c} \end{aligned} \quad (3.2.18)$$

where  $\mathbf{K}_{FSFDQ} = [k_{a_z} \quad k_q]$ . The DQ control law gain  $\mathbf{K}_{DQr}$  is calculated using a pole placement design technique. The feed-forward control gain  $K_{FFDQ}$  is then calculated to place the resulting zero on the closed-loop pole that originates from the open-loop integrator, to eliminate the integrator dynamics. Since the gain  $K_{IDQ}$  as well as the location of the closed-loop pole originating from the integrator will be known after the pole placement design, the feed-forward gain can be calculated with,

$$K_{FFDQ} = \frac{K_{IDQ}}{p_i} \quad (3.2.19)$$

where  $p_i$  is the location of the closed-loop pole originating from the open-loop integrator pole, resulting from the pole placement design.

## Specifications

In order to perform the pole placement design, we first need to select specifications for the desired closed-loop response of the normal acceleration controller ('DQ controller'), which are then translated into desired closed-loop poles. Typically the closed-loop poles are chosen so that the closed-loop response of the DQ controller has the same damping and natural frequency as the aircraft's own short-period mode, so that the aircraft will behave naturally as expected by the pilots. The GTM model includes a baseline controller library with an angle of attack (AoA) inner-loop controller that performs short-period damping and angle of attack regulation. The baseline angle of attack controller is also a state feedback with integral controller and its closed-loop short-period specification is used as a starting point and guideline for the DQ controller specification.

Table 3.1 lists the natural open-loop short period dynamics of the GTM along with the angle of attack controller and the chosen DQ controller closed-loop design specifications. The closed-loop poles of the angle of attack controller were calculated using the provided gains of the controller, and calculating the closed-loop reduced-order state space model using the gains, and then calculating the eigenvalues of the closed-loop model's state matrix. The natural frequency of the desired closed-loop short-period poles of the DQ controller is close to that of

the natural short-period poles (at around 8 rad/s), which is slightly slower than the closed-loop short-period poles of the angle of attack controller (at around 10 rad/s). The desired real pole of the closed-loop integrator pole of the DQ controller is also slightly faster than the angle of attack controller's closed-loop integrator pole (3.5 rad/s compared to 2.7 rad/s). A high damping ratio of 0.9 was chosen for the closed-loop short-period poles of the DQ controller, which is close to that of the high damping ratio of 0.8 of the angle of attack controller.

During the design it was found that placing the poles of the DQ controller near the pole locations of the angle of attack controller as in Table 3.1, which is at a natural frequency higher than the natural short-period frequency, caused an undesirable underdamped response. The step response exhibited high frequency oscillations of roughly 5 Hz before settling. It was found that these undamped oscillations could be attributed to the excitement of unmodelled non-linear actuator dynamics. Thus it was decided to keep the DQ controller's closed-loop short period poles near the natural frequency of the natural short-period dynamics and increase the speed of the DQ controller's closed-loop integrator pole, so that the closed-loop step response (without the feed-forward term) still matches the angle of attack controller step response. The short-period poles had a significant enough effect on the closed-loop transient response of the DQ controller and angle of attack controller, so that the slower closed-loop integrator pole did not entirely dominate the response. The location of the DQ controller's closed-loop integrator pole was therefore iteratively chosen so that the closed-loop step response constituted by the DQ controller's three pole system matched that of the step response of the three pole closed-loop system of the angle of attack controller, i.e. has the same response time constant. Figure 3.7 shows the pole placement design of the DQ controller in the s-plane.

Table 3.1: Longitudinal DQ design specifications

Longitudinal System	Mode	Poles	$\omega_n$	$\zeta$
Open-loop	Short-Period	$-3.744 \pm 7.174i$	8.09	0.46
AoA Closed-Loop	Short-Period	$-8.259 \pm 5.719i$	10	0.82
AoA Integrator		-2.7061		
DQ Design Closed-Loop	Short-Period	$-7.2900 \pm 3.5307i$	8.1	0.9
DQ Integrator Placement		-3.5		

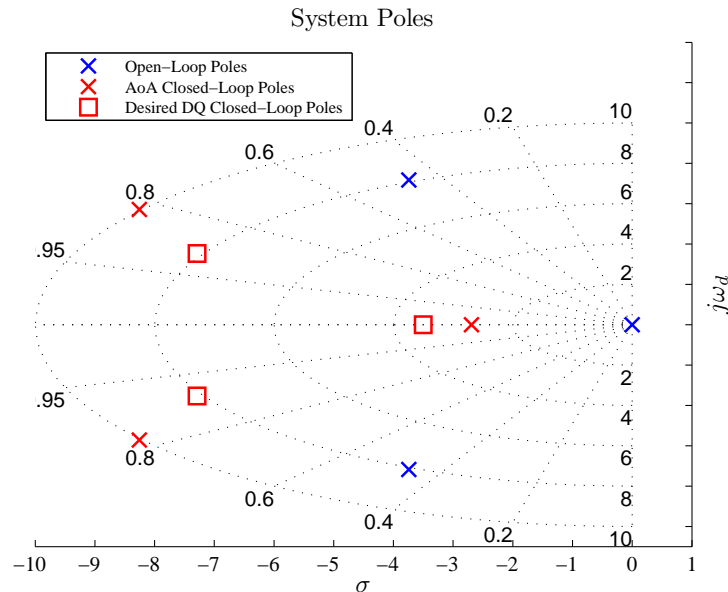
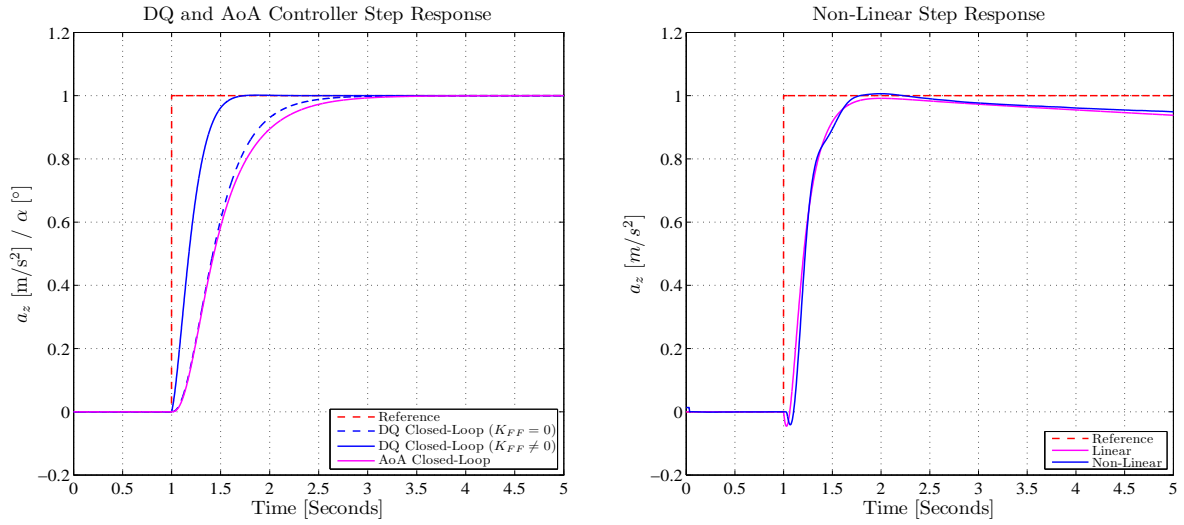


Figure 3.7: Comparison of closed-loop AoA controller poles and desired closed-loop DQ controller poles

## Results

Figure 3.8 shows the closed-loop step responses of the normal acceleration controller ('DQ controller') simulated using both the linearised plant model and the non-linear plant model. Figure 3.8(a) shows the response of the DQ controller to a unit step reference command with and without the feed-forward gain using the reduced-order linear model, and compares it to the response of the angle of attack controller using the reduced-order linear model. The DQ controller's transient response (without feed-forward) matches well with the angle of attack controller's response. The response of the DQ controller using the reduced-order model exhibits zero steady state error. Figure 3.8(b) shows a comparison between the DQ controller's unit step response simulated using the full-order linear model and simulated using the non-linear model. The transient response of both the DQ controller simulated using the non-linear model and the DQ controller simulated using the full-order linear model matches the transient response of the DQ controller (with feed-forward) using the reduced-order model well. The response of the DQ controller using the non-linear model exhibits the same time constant and damping as the response of the DQ controller using the full-order model, and thus satisfies the design specifications. The slow divergence is a result of the uncontrolled phugoid dynamics and is expected, because no flight path angle controller has been implemented yet at this stage in the design.

The full-order linear controller dynamics included the elevator actuator lag dynamics, modelled as a first-order lag with a bandwidth of 5 Hz. The slight dip in the full-order linear and non-linear initial response is due to the non-minimum phase response that can be attributed to input feed-forward term  $D_{a_z}$  that was included in the full-order linear model.



(a) Closed-loop step response of DQ controller and AoA controller using reduced-order plant

(b) Closed-loop response of DQ controller using full-order linear model vs DQ controller using non-linear model

Figure 3.8: Results of DQ controller closed-loop unit step responses

### Closed-Loop Model

The closed-loop model that encapsulates the full-order longitudinal dynamics with the DQ law added, can be expressed in state space form as,

$$\dot{\mathbf{x}}_{DQ} = \mathbf{A}_{DQ}\mathbf{x}_{DQ} + \mathbf{B}_{DQ}\mathbf{u}_{DQ}$$

$$\begin{bmatrix} \dot{\mathbf{x}}_{long} \\ \dot{x}_{IDQ} \end{bmatrix} = \begin{bmatrix} \mathbf{A}_{long} - \mathbf{B}_{\delta_e} \mathbf{N} \mathbf{K}_{DQ} \mathbf{C}_{DQ} & -\mathbf{N} \mathbf{K}_{IDQ} \mathbf{B}_{\delta_e} \\ -\mathbf{C}_{a_z} + \mathbf{N} \mathbf{K}_{DQ} \mathbf{C}_{DQ} D_{a_z} & \mathbf{N} \mathbf{K}_{FFDQ} D_{a_z} \end{bmatrix} \begin{bmatrix} \mathbf{x}_{long} \\ x_{IDQ} \end{bmatrix} + \begin{bmatrix} \mathbf{N} \mathbf{K}_{FFDQ} \mathbf{B}_{\delta_e} & \mathbf{B}_{\Delta T} \\ 1 - \mathbf{N} \mathbf{K}_{FFDQ} D_{a_z} & 0 \end{bmatrix} \begin{bmatrix} a_{zc} \\ \Delta T \end{bmatrix}$$

$$\mathbf{C}_{DQ} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ & \mathbf{C}_{a_z} & & \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \mathbf{B}_{\delta_e} = \begin{bmatrix} \frac{\partial \dot{V}}{\partial \delta_e} \\ \frac{\partial \dot{\alpha}}{\partial \delta_e} \\ \frac{\partial \dot{Q}}{\partial \delta_e} \\ \frac{\partial \dot{\epsilon}}{\partial \delta_e} \end{bmatrix} \quad \mathbf{B}_{\Delta T} = \begin{bmatrix} \frac{\partial \dot{V}}{\partial T} \\ \frac{\partial \dot{\alpha}}{\partial T} \\ \frac{\partial \dot{Q}}{\partial T} \\ \frac{\partial \dot{\epsilon}}{\partial T} \end{bmatrix} \quad (3.2.20)$$

$$\mathbf{K}_{DQ} = [0 \quad k_{a_z} \quad k_q \quad 0]$$

$$N = \frac{1}{1 + \mathbf{K}_{DQ} \mathbf{D}_{DQ}}, \quad \mathbf{D}_{DQ} = [0 \quad D_{a_z} \quad 0 \quad 0]$$

where  $N$  is an auxiliary variable, and is derived in Appendix C.

The closed-loop model will serve as the open-loop plant for the next controller loop, which will be the flight path angle controller. Figure 3.9 shows the new closed-loop poles after state

feedback with integral control and also shows the effect of the elevator feed-forward term  $D_{a_z}$  on the closed-loop poles. It can be seen from the figure that the closed-loop integrator pole has shifted to the right by adding the feed-forward term, meaning that the zero added using the feed-forward gain  $K_{FFDQ}$  no longer cancels out the integrator pole exactly. The gain  $K_{FFDQ}$  was adjusted accordingly to place the zero at the corrected location at  $s = -3$ . The phugoid poles have also shifted to become real with one pole becoming unstable. However, this is not a reason for concern, since the DQ control law is only responsible for controlling the normal acceleration dynamics of the aircraft, and is not responsible for controlling the flight path angle or the airspeed (the phugoid dynamics). The next layer of feedback control loops, namely the airspeed controller and the flight path angle controller, will stabilise the phugoid mode dynamics.

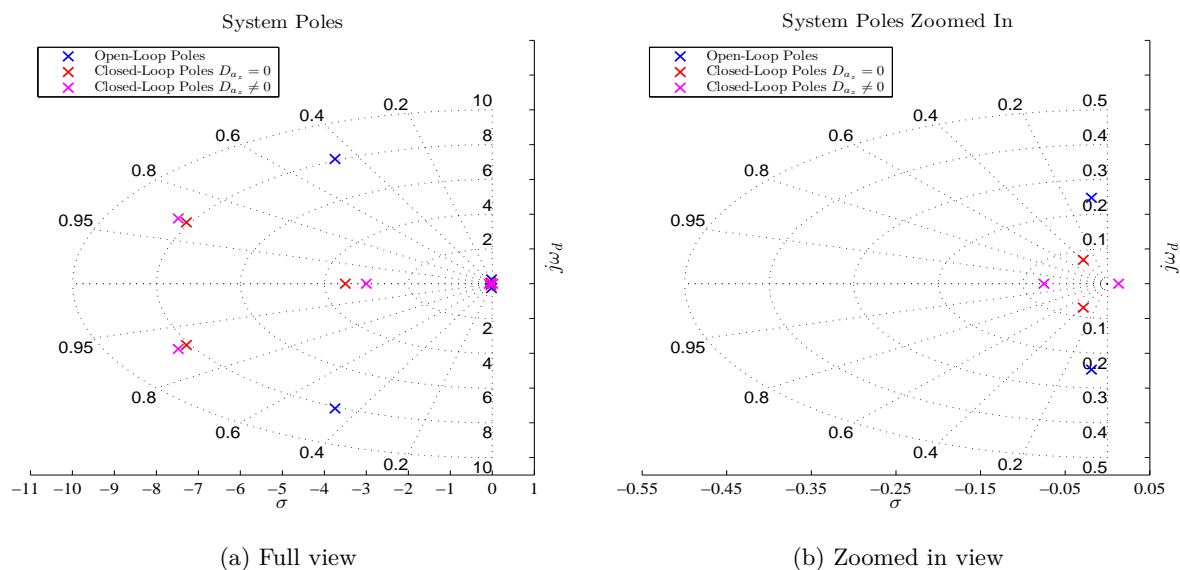


Figure 3.9: Closed-loop poles after adding DQ controller

### 3.2.2 Flight Path Angle Controller

With the inner-loop normal acceleration controller ('DQ controller') in place, the flight path angle controller can be designed. The middle-loop flight path angle controller controls the flight path angle of the aircraft to follow a commanded reference, and uses the normal acceleration controller as an inner-loop controller. A block diagram of the flight path angle controller architecture is shown in Figure 3.10. The controller uses the calculated flight path angle as the feedback signal and supplies normal acceleration references to the inner-loop normal acceleration controller. The control strategy is simple proportional feedback, and the control design is performed using the normal acceleration controller closed-loop model as the plant.

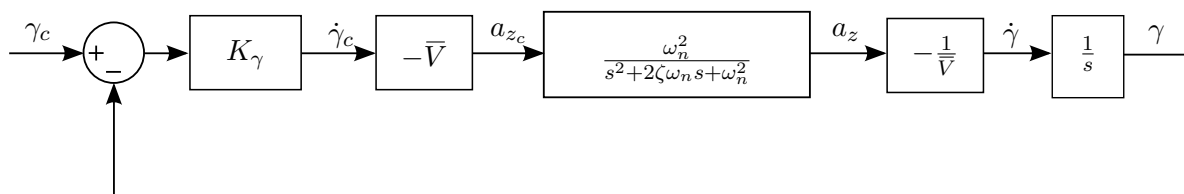


Figure 3.10: Flight path angle controller architecture

## Design

For the design of the flight path angle controller the concept of time scale separation is used to abstract the dynamics of the inner-loop normal acceleration controller from the phugoid dynamics. The dynamics of the normal acceleration controller is approximated with a reduced second-order system which is essentially the closed-loop short-period dynamics, which will form part of the plant model of the flight path angle controller design. The second-order transfer function equation in Figure 3.10 represents the reduced-order dynamics of the normal acceleration controller ('DQ controller'). The flight path angle controller design is evaluated on the full-order longitudinal plant model (with the airspeed controller) to verify that the full-order system has a stable response.

The total normal acceleration  $a_z$  of the aircraft is the sum of the specific normal acceleration  $a_{z_{specific}}$  and the normal component of the gravitational acceleration  $a_{z_{gravity}}$ . Thus, consider that the normal acceleration output of the normal acceleration controller can be expressed as,

$$a_z = a_{z_{specific}} + a_{z_{gravity}} \quad (3.2.21)$$

with,

$$a_{z_{specific}} = \frac{qSC_Z}{m}, \quad a_{z_{gravity}} = g \cos \Theta \cos \Phi \quad (3.2.22)$$

and can be related to flight path angle rate with,

$$-\bar{V}\dot{\gamma} = a_{z_{specific}} \cos \Phi + a_{\gamma_{gravity}}, \quad a_{\gamma_{gravity}} = g \cos \Theta \quad (3.2.23)$$

Thus, from the above equation, the flight path angle controller can be seen as a type 1 system, because we can control the flight path angle rate using the output of the normal acceleration controller, which is naturally integrated to flight path angle. The plant for the flight path angle controller is then simply the second-order closed-loop normal acceleration controller dynamics and an integrator. It is assumed that the closed-loop dynamics of this first-order response will be chosen to be an order of magnitude faster than the phugoid dynamics.

Note that the normal acceleration controller produces an acceleration that is normal to the wings, but the rate of change of the flight path angle  $\dot{\gamma}$  is determined only by the vertical component of normal acceleration. This means that the bank angle has an effect on the flight path angle rate, and the relationship between the normal acceleration and its vertical component is the cosine of the bank angle. When the bank angle is zero, then the vertical component of normal acceleration equals the normal acceleration, and when the bank angle is 90 degrees, then the vertical component of normal acceleration is zero.

The flight path angle command law is chosen to give a flight path angle rate command  $\dot{\gamma}_c$  that is proportional to the error in the flight path angle state and the commanded flight path angle (at the wings level trim condition of zero flight path angle rate),

$$\dot{\gamma}_c = K_\gamma \gamma_{error} = K_\gamma (\gamma_c - \gamma) \quad (3.2.24)$$

The flight path angle rate command  $\dot{\gamma}_c$  can then be converted into a total normal acceleration command  $a_{z_c}$  for the inner-loop normal acceleration controller that consists of a specific acceleration command  $a_{z_{specific}}$  and the normal component of the gravitational acceleration  $a_{z_{gravity}}$ ,

$$a_{z_c} = a_{z_{c_{specific}}} + a_{z_{gravity}} \quad (3.2.25)$$

with,

$$\begin{aligned} a_{z_{c_{specific}}} &= \frac{-\bar{V}\dot{\gamma}_c - a_{\gamma_{gravity}}}{\cos \Phi} \\ &= \frac{-\bar{V}K_\gamma(\gamma_c - \gamma) - a_{\gamma_{gravity}}}{\cos \Phi} \end{aligned} \quad (3.2.26)$$

The flight path angle controller is designed to produce a dominantly first-order closed-loop response. The design therefore involves selecting the desired time constant for the closed-loop response, translating the time constant specification into a desired real closed-loop pole, and then calculating the proportional feedback gain  $K_\gamma$  that moves the dominant closed-loop pole of the system to the desired location.

Using the airspeed state  $\bar{V}$  in the control law instead of the constant trim airspeed, has the advantage of making the controller insensitive to change in airspeed resulting in a more robust controller. Another advantage is that airspeed and normal acceleration can be directly measured by the aircraft's sensors. The bank angle  $\Phi$  is also used in the control law to compensate for the reduced lift component in the vertical plane when the aircraft is banked. The terms  $\bar{V}$  and  $\cos \Phi$  are essentially time varying parameters that have the effect of 'cancelling out' the non-linear effects of airspeed and bank angle in the flight path angle response, ensuring a linear flight path angle response. Thus we can apply a linear control design and expect the response to be consistent at different airspeeds and bank angles.

### Specifications

The specifications for the flight path angle controller were derived using the closed-loop flight path angle response of a simulated A330 aircraft model and by consulting specifications derived by Trollip for his Masters research [16]. These specifications were then adapted by taking the GTM's scaling factor into account. The desired specifications for the full scale model are: an overshoot of less than 5%, a 2% settling time of approximately 15 seconds and a steady-state tracking error of zero for a constant reference input. Taking the GTM's 5.5% scaling factor into account, the response rate specifications are related by,

$$scaled\ model\ rates \approx (full-scale\ model\ rates) \times \sqrt{k_{scf}}, \quad k_{scf} = 0.055 \quad (3.2.27)$$

where  $k_{scf}$  is the GTM's scaling factor. Thus the dynamically scaled specifications for the GTM's flight path angle response are: an overshoot of less than 5% and a 2% settling time of approximately 3.5 seconds. The feedback gain  $K_\gamma$  was calculated using a root locus design technique to achieve the desired closed-loop specifications. These specifications translate to a desired dominant first-order closed-loop pole at  $s = -1.143$  using the following calculation,

$$t_s = \frac{4}{\sigma} = 3.5 \quad (3.2.28)$$

$$\sigma = \frac{4}{3.5} = 1.143 \quad (3.2.29)$$

$$s = -\sigma = -1.143 \quad (3.2.30)$$

## Results

Figure 3.11(a) shows the root locus plot of the reduced-order plant with the flight path angle controller for a gain  $K_\gamma$  that sufficiently achieves the desired specifications. Figure 3.11(b) shows the closed-loop flight path angle step response using the reduced-order model. The response exhibits zero steady-state error with no overshoot. The 2% settling time of the response is 3.7 seconds, just over the desired 3.5 seconds. This is likely due to the slight effect of the normal acceleration controller closed-loop poles on the dominant response.

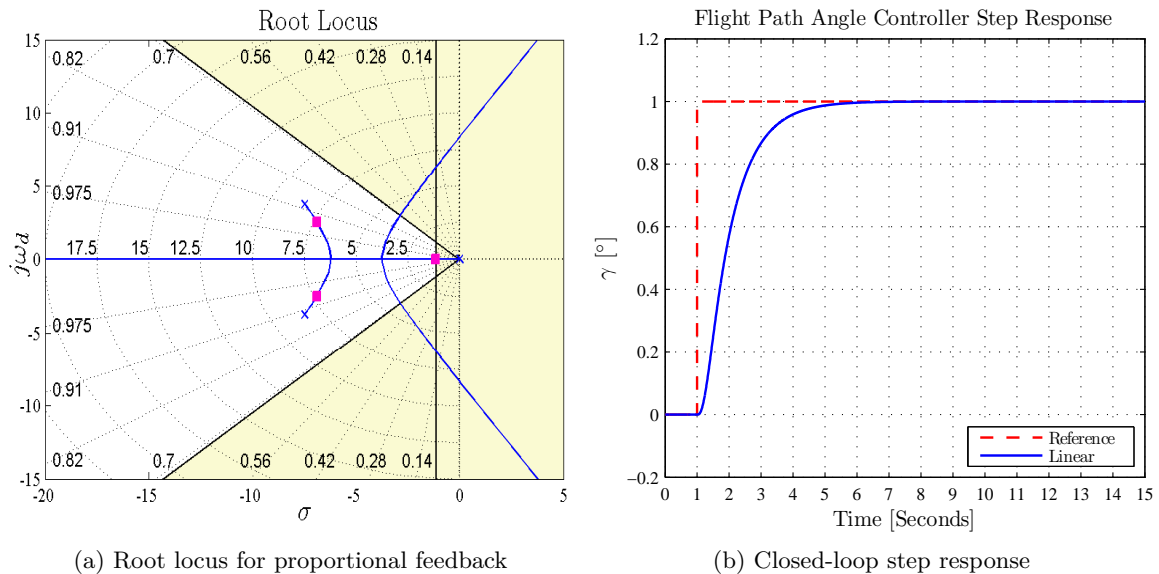


Figure 3.11: Root locus of the reduced-order plant with flight path angle controller for a chosen gain  $K_\gamma$  that meets the chosen specifications as well as the closed-loop step response of reduced-order model with flight path angle controller

The reduced-order design was verified by simulating the full-order linear and non-linear system model with the flight path angle controller added. The full-order linear model that includes the closed-loop normal acceleration controller's dynamics was used when testing the controller design. Figure 3.12 shows the closed-loop flight path angle response to a unit flight path angle step command for both the full-order linear model and the non-linear model. Due to the strong coupling between the flight path angle response and airspeed response of the aircraft, the flight path angle controller was tested in parallel with the airspeed controller for both the linear and non-linear case to regulate a constant airspeed during the response. The closed-loop flight path angle response using the non-linear model matches the closed-loop response using the full-order linear model fairly well. The step response using the non-linear model shows a slight overshoot, but is still well within the 5% specification. Both responses have a 2% settling time close to 3 seconds, which is slightly faster than the specification, and both responses exhibit zero steady-state error.



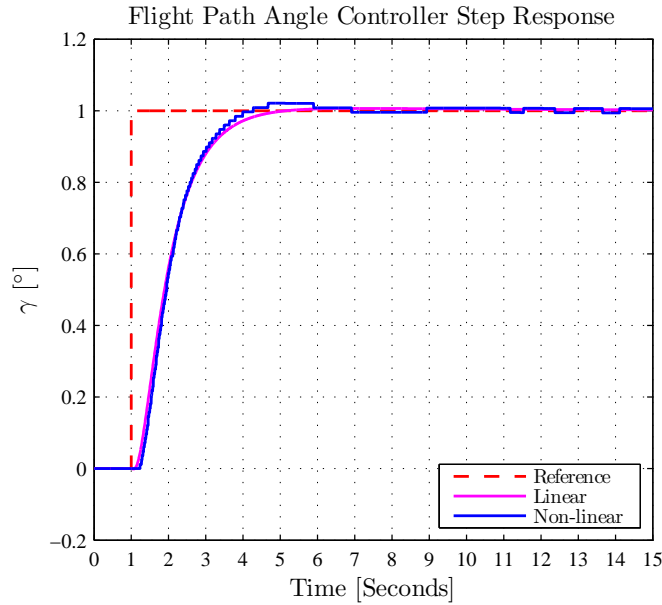


Figure 3.12: Flight path angle controller step responses simulated on the full-order linear and non-linear models of the flight path angle dynamics

### Practical Implementation

After both the flight path angle controller and bank angle and sideslip controller were implemented on the non-linear aircraft simulation model, a simulation was done to verify that the flight path angle response is stable while the aircraft is at high bank angles. (Note that the design of the bank angle and sideslip controller is presented later in this chapter.) The flight path angle response was simulated on the non-linear system model at a high bank angle of 60 degrees ( $\Phi = 60^\circ$ ). Figure 3.13(a) shows the flight path angle response to a unit flight path angle step command that was given at  $t = 25$  seconds. The step command was given after initialisation transient responses have vanished. Figure 3.13(b) shows the bank angle and sideslip angle response. The bank angle and sideslip angle controller is commanded to maintain a bank angle of 60 degrees and to regulate the sideslip angle to zero.

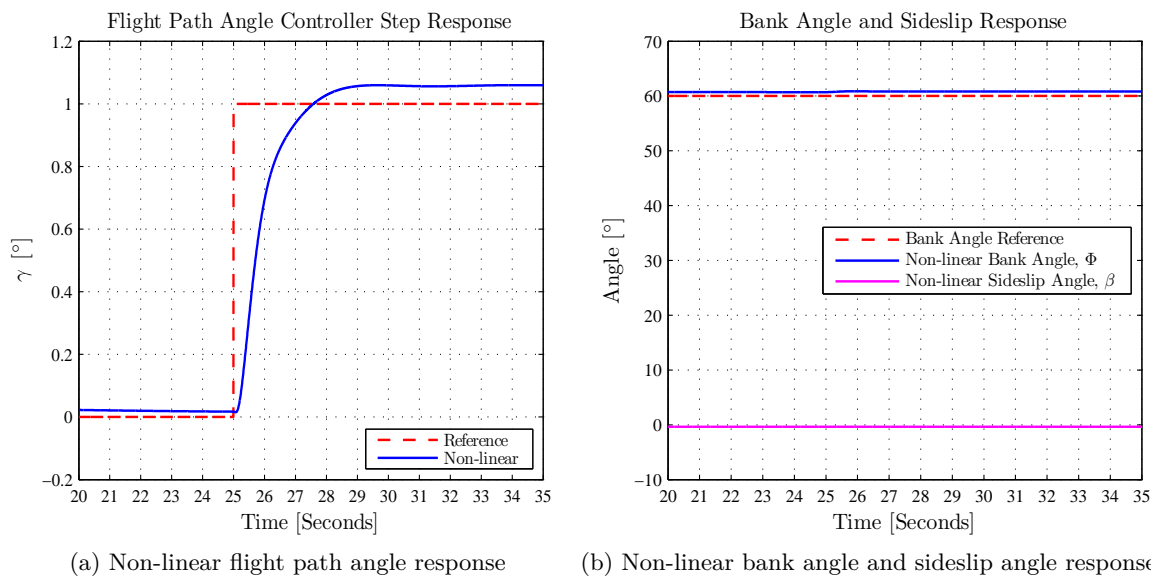


Figure 3.13: Flight path angle controller step response simulated non-linear model of the flight path angle dynamics at a high bank angle

The simulation result shows that the closed-loop flight path angle response is stable, while the bank angle is regulated fairly close to 60 degrees. The sideslip angle is also successfully regulated to zero degrees. The flight path angle response exhibits no overshoot with a 2% settling time of roughly 3.5 seconds, but shows a steady-state error of roughly 5%. Thus the non-linear transient response at high bank angles agrees well with non-linear transient response of Figure 3.11, and still satisfies the overshoot and settling time specifications. The steady-state error specification of zero is not satisfied at in this high bank angle case. However, this is to be expected if there is any error in the trim normal acceleration calculation that counters gravity. This error in the normal acceleration command will manifest as a disturbance signal for the flight path angle controller. The flight path angle controller will not be able to completely reject this disturbance signal, since the flight path angle controller is only a proportional controller and does not have an integrator.

### 3.2.3 Airspeed Controller (‘Autothrust’)

With the inner-loop normal acceleration controller (‘DQ controller’) in place, the aircraft requires an airspeed controller along with a flight path angle controller to maintain a reference airspeed during a longitudinal manoeuvre. In this section an autothrust controller is designed to maintain a desired airspeed and avoid stalling while holding the commanded flight path angle. A block diagram of the airspeed controller architecture is shown in Figure 3.14. The controller uses the measured airspeed and axial acceleration as feedback signals and actuates the engine thrust. The control strategy is full state feedback with integral control, and the control design is performed using a reduced-order model of the airspeed dynamics, that includes the thrust lag dynamics, as the plant.

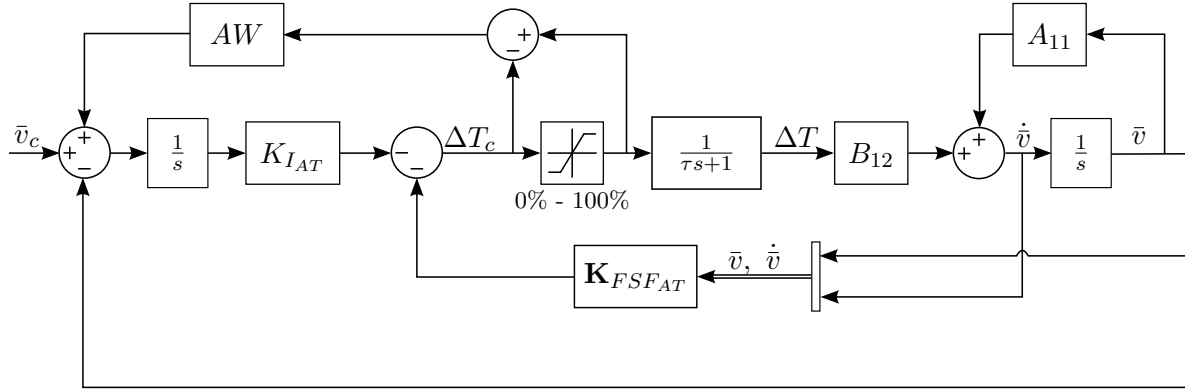


Figure 3.14: Airspeed controller architecture

### Reduced-Order Model

The plant dynamics for the airspeed controller is a reduced-order two state system of the full-order longitudinal dynamics. The full-order longitudinal model is reproduced here:

The longitudinal linear dynamics are given by its state space representation,

$$\dot{\mathbf{x}}_{long} = \mathbf{A}_{long}\mathbf{x}_{long} + \mathbf{B}_{long}\mathbf{u}_{long} \quad (3.2.31)$$

where the full longitudinal state and input matrices are given by,

$$\mathbf{A}_{long} = \begin{bmatrix} \frac{\partial \dot{U}}{\partial U} & \bar{V}_T \frac{\partial \dot{U}}{\partial \alpha} & \frac{\partial \dot{U}}{\partial Q} & \frac{\partial \dot{U}}{\partial \Theta} \\ \frac{1}{\bar{V}_T} \frac{\partial \dot{\alpha}}{\partial U} & \frac{\partial \dot{\alpha}}{\partial \alpha} & \frac{1}{\bar{V}_T} \frac{\partial \dot{\alpha}}{\partial Q} & \frac{1}{\bar{V}_T} \frac{\partial \dot{\alpha}}{\partial \Theta} \\ \frac{\partial \dot{Q}}{\partial U} & \bar{V}_T \frac{\partial \dot{Q}}{\partial \alpha} & \frac{\partial \dot{Q}}{\partial Q} & \frac{\partial \dot{Q}}{\partial \Theta} \\ \frac{\partial \dot{\Theta}}{\partial U} & \bar{V}_T \frac{\partial \dot{\Theta}}{\partial \alpha} & \frac{\partial \dot{\Theta}}{\partial Q} & \frac{\partial \dot{\Theta}}{\partial \Theta} \end{bmatrix}, \quad \mathbf{B}_{long} = \begin{bmatrix} \frac{\partial \dot{U}}{\partial \delta_e} & \frac{\partial \dot{U}}{\partial T} \\ \frac{1}{\bar{V}_T} \frac{\partial \dot{\alpha}}{\partial \delta_e} & \frac{1}{\bar{V}_T} \frac{\partial \dot{\alpha}}{\partial T} \\ \frac{\partial \dot{Q}}{\partial \delta_e} & \frac{\partial \dot{Q}}{\partial T} \\ \frac{\partial \dot{\Theta}}{\partial \delta_e} & \frac{\partial \dot{\Theta}}{\partial T} \end{bmatrix} \quad (3.2.32)$$

and the full longitudinal state and input vectors are,

$$\mathbf{x}_{long} = [\bar{v} \quad \alpha \quad q \quad \theta]^T, \quad \mathbf{u}_{long} = [\delta_e \quad \Delta T]^T \quad (3.2.33)$$

The reduced-order model includes the airspeed and wind-axis axial acceleration as states. The full-order linear longitudinal state space model of the aircraft dynamics omits the thrust lag dynamics. To take the thrust lag of the engines into account, it is modelled as a first-order lag state. For the reduced-order model, at trimmed flight, it is assumed that the wind-axis axial acceleration is equal to the total acceleration,

$$\dot{\bar{v}} = A_{11}\bar{v} + B_{12}\Delta T, \quad A_{11} = \frac{\partial \dot{U}}{\partial U}, \quad B_{12} = \frac{\partial \dot{U}}{\partial T} \quad (3.2.34)$$

where the terms  $A_{11}$  and  $B_{12}$  are obtained from the corresponding terms in the full state space matrices  $\mathbf{A}_{long}$  and  $\mathbf{B}_{long}$  respectively, and  $\Delta T$  is the throttle state and the thrust lag dynamics are described by,

$$\Delta \dot{T} = -\frac{1}{\tau}\Delta T + \frac{1}{\tau}\Delta T_c \quad (3.2.35)$$

where  $\tau$  is the thrust lag time constant and  $\Delta T_c$  is the throttle command input. The throttle state is directly related to the thrust state of the aircraft in the linear model, so the thrust lag can be modelled by imposing a lag on the throttle state. The throttle state and throttle command input are percentage values which are limited to between 0% - 100% and the term  $B_{12}$  converts the throttle percentage value into wind-axis axial acceleration. The term  $A_{11}\bar{v}$  represents the change in aerodynamic drag due to the airspeed deviation from the nominal airspeed, and including this term in the reduced-order state space model would translate into a very slow open-loop pole. However, instead of modelling the aerodynamic drag as part of the plant dynamics, the term  $A_{11}\bar{v}$  is omitted in the design and is instead treated as an external disturbance signal when testing the linear design. The reduced-order dynamics of the plant model can be represented in state space form as,

$$\begin{aligned}\dot{\mathbf{x}}_{ATr} &= \mathbf{A}_{ATr}\mathbf{x}_{ATr} + \mathbf{B}_{ATr}u \\ \begin{bmatrix} \dot{\bar{v}} \\ \ddot{\bar{v}} \end{bmatrix} &= \begin{bmatrix} 0 & 1 \\ 0 & -\frac{1}{\tau} \end{bmatrix} \begin{bmatrix} \bar{v} \\ \dot{\bar{v}} \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{B_{12}}{\tau} \end{bmatrix} \Delta T_c\end{aligned}\quad (3.2.36)$$

### Design

The design of the autothrust involves a pole placement design technique and uses full-state feedback with integral control. The design takes advantage of an acceleration-based controller by feeding back the wind-axis axial acceleration as a state. The wind-axis axial acceleration state estimate is obtained by converting the components of the body-axis acceleration measurements into equivalent wind-axis acceleration components, and then using the calculated wind-axis axial acceleration component as the state. Integral control is added so that the controller can follow an airspeed reference command with zero steady state error. An anti-windup component is added to the controller to reduce integral windup, which occurs because of the non-linear saturation limits of the throttle percentage. The controller's integrator dynamics can be written as,

$$\begin{aligned}\dot{x}_{IAT} &= \bar{v}_c - \bar{v} \\ &= \bar{v}_c - (\mathbf{C}_{ATr}\mathbf{x}_{ATr})\end{aligned}\quad (3.2.37)$$

where

$$x_{IAT} = \int (\bar{v}_c - \bar{v}) \, dt \quad (3.2.38)$$

$$\mathbf{C}_{ATr} = [1 \quad 0] \quad (3.2.39)$$

The state space form of the reduced-order airspeed dynamics augmented with the integrator state for the autothrust controller is then given by,

$$\begin{bmatrix} \dot{\mathbf{x}}_{ATr} \\ \dot{x}_{IAT} \end{bmatrix} = \begin{bmatrix} \mathbf{A}_{ATr} & \mathbf{0}_{2 \times 1} \\ -\mathbf{C}_{ATr} & 0 \end{bmatrix} \begin{bmatrix} \mathbf{x}_{ATr} \\ x_{IAT} \end{bmatrix} + \begin{bmatrix} \mathbf{B}_{ATr} \\ 0 \end{bmatrix} \Delta T_c + \begin{bmatrix} \mathbf{0}_{2 \times 1} \\ 1 \end{bmatrix} \bar{v}_{Ref} \quad (3.2.40)$$

with the full-state feedback control law given by,

$$\begin{aligned}
u = \Delta T_c &= -\mathbf{K}_{AT} \begin{bmatrix} \mathbf{x}_{ATr} \\ x_{IAT} \end{bmatrix} \\
&= -[\mathbf{K}_{FSFAT} \quad K_{IAT}] \begin{bmatrix} \mathbf{x}_{ATr} \\ x_{IAT} \end{bmatrix}
\end{aligned} \tag{3.2.41}$$

where  $\mathbf{K}_{FSFAT} = [k_{\dot{v}} \quad k_{\ddot{v}}]$ . The autothrust control law gain  $\mathbf{K}_{AT}$  is calculated using a pole placement technique as presented in the next subsections.

### Specifications

The airspeed response should be close to overdamped with the damping ratio  $\zeta$  close to  $\zeta = 1$  and the response time should be as fast as possible with the limiting factor being the thrust input lag time constant. The airspeed response should also have a steady-state error of zero. This means that the natural frequency of the dominant poles should be as high as possible, but still much lower than the frequency of the thrust lag pole. The location of one of the closed-loop poles was chosen so that it would not be far away from the open-loop pole of the thrust lag dynamics. This is so that the controller does not unnecessarily exert effort to change the thrust lag dynamics, so that the thrust lag dynamics of the closed-loop system will remain the same,

$$p_{I/tau} = p_{\tau} = -\frac{1}{\tau}, \quad \tau = 3 \text{ seconds} \tag{3.2.42}$$

where the thrust lag time constant  $\tau$  was determined from the non-linear GTM simulation model's thrust step response. A throttle step command was given to the GTM and the time constant was obtained by measuring the time the GTM's thrust step response took to reach 63 % of the final steady-state thrust value.

The specifications for the closed-loop dominant pole pair was chosen to have a damping coefficient of  $\zeta = 0.9$  and a natural frequency of  $\omega_n = 0.25$ , which translates into a 2 % settling time of roughly 18 seconds for a second-order system. The specifications were determined by using the specification derived by Trollip [16] in his Masters research as a guide, where Trollip used the closed-loop airspeed response of a simulated A330 aircraft model as a guide to design an autothrust controller. The natural frequency specification was determined iteratively to strike a balance between response time and control effort. Choosing a closed-loop natural frequency that was too high or a higher damping ratio would result in the control effort saturating too quickly and the closed-loop pole pair would not show a dominant second-order response when placed too close to the closed-loop thrust lag pole. Figure 3.15 shows the system open-loop poles and the desired closed-loop pole placement in the s-plane for the autothrust controller.

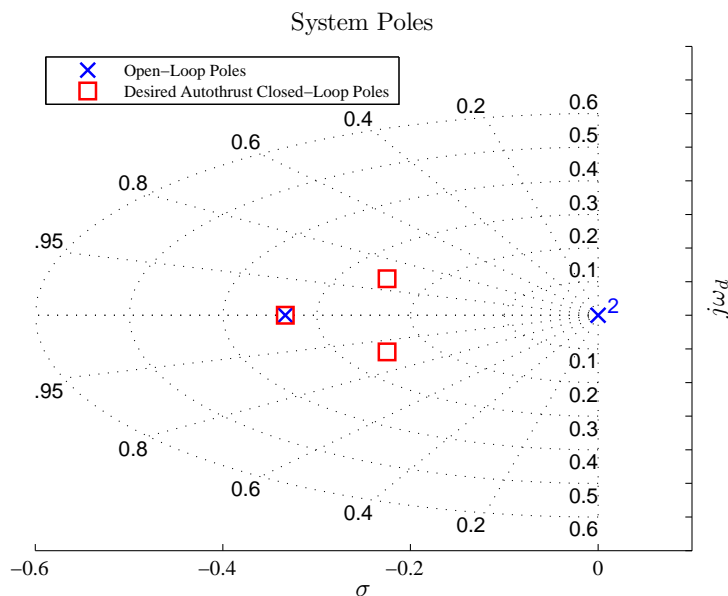


Figure 3.15: Pole placement of desired autothrust closed-loop poles

## Results

Figure 3.16 shows a comparison between the simulated autothrust controller response using the full-order linear and non-linear plant to a unit step command from trim airspeed. The autothrust controller was tested with the flight path angle controller in parallel in both the linear and non-linear case. The flight path angle controller must be present, otherwise the flight path angle would not remain constant, and the acceleration due to gravity would appear as an increasing disturbance signal for the airspeed controller.

The response results show that the non-linear model response matches the linear model (that includes the airspeed disturbance term  $A_{11}\bar{v}$ ) response well. The non-linear and linear model responses both exhibit a response that is well damped with no overshoot and zero steady-state error. The response of the linear model that omits the airspeed disturbance term  $A_{11}\bar{v}$  has a settling time of 21.5 seconds, which is slightly slower than the designed for settling time of 18 seconds. This is due to the thrust lag pole's effect on the dominant 2nd order system poles of the reduced-order model. The non-linear and linear model (with airspeed disturbance term  $A_{11}\bar{v}$ ) response both have a settling time of 31.5 seconds, which is slower than the linear response that omits the airspeed disturbance term  $A_{11}\bar{v}$ . This is to be expected, as the dynamics of the airspeed term  $A_{11}\bar{v}$  effectively adds a slow dominant real pole, which is visible as a slower response in the linear model that includes this term, and this effect is also reflected in the non-linear response.

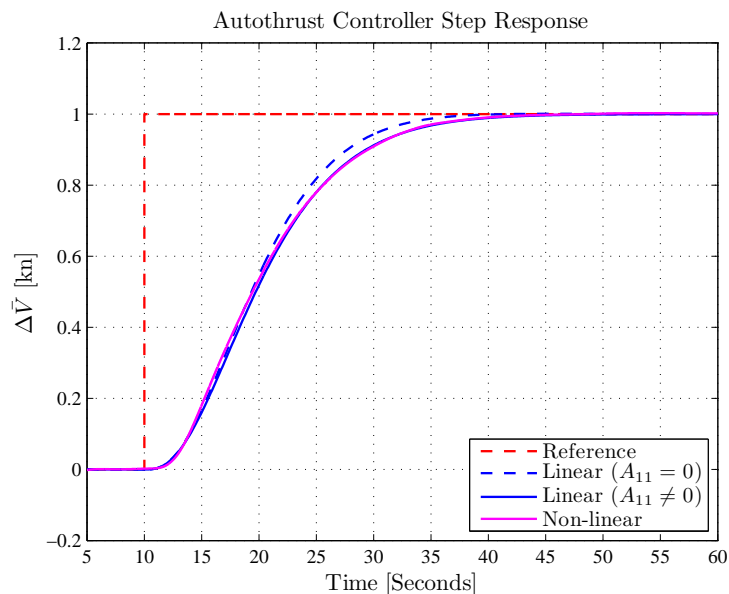


Figure 3.16: Closed-loop autothrust controller response to unit step command

The throttle input is physically limited to between 0% and 100%. This introduces a non-linear saturation region on the control input. In the case of the non-linear model, it was noticed that for larger airspeed step commands, the commanded throttle signal from the controller in Figure 3.17(b) reaches a peak value far outside the range of the realised throttle input and operates in a non-linear saturation region where the control signal has no effect on the system output. This condition is known as integrator windup and causes a large overshoot as seen in Figure 3.17(a), because the integrator error state must first ‘wind down’ in the controller to within the throttle limits. This issue can be solved by implementing an anti-windup scheme known as back-calculation, which involves subtracting the difference between the commanded signal and realised signal after saturation from the integrator error state via a gain  $AW$ . Figure 3.17(c) shows the airspeed response for the same airspeed step command, but with anti-windup implemented. It can be noted that the airspeed changes linearly while the throttle command is saturated, and this means that the time constant cannot be determined from these responses that use larger input step commands that would saturate the throttle command.

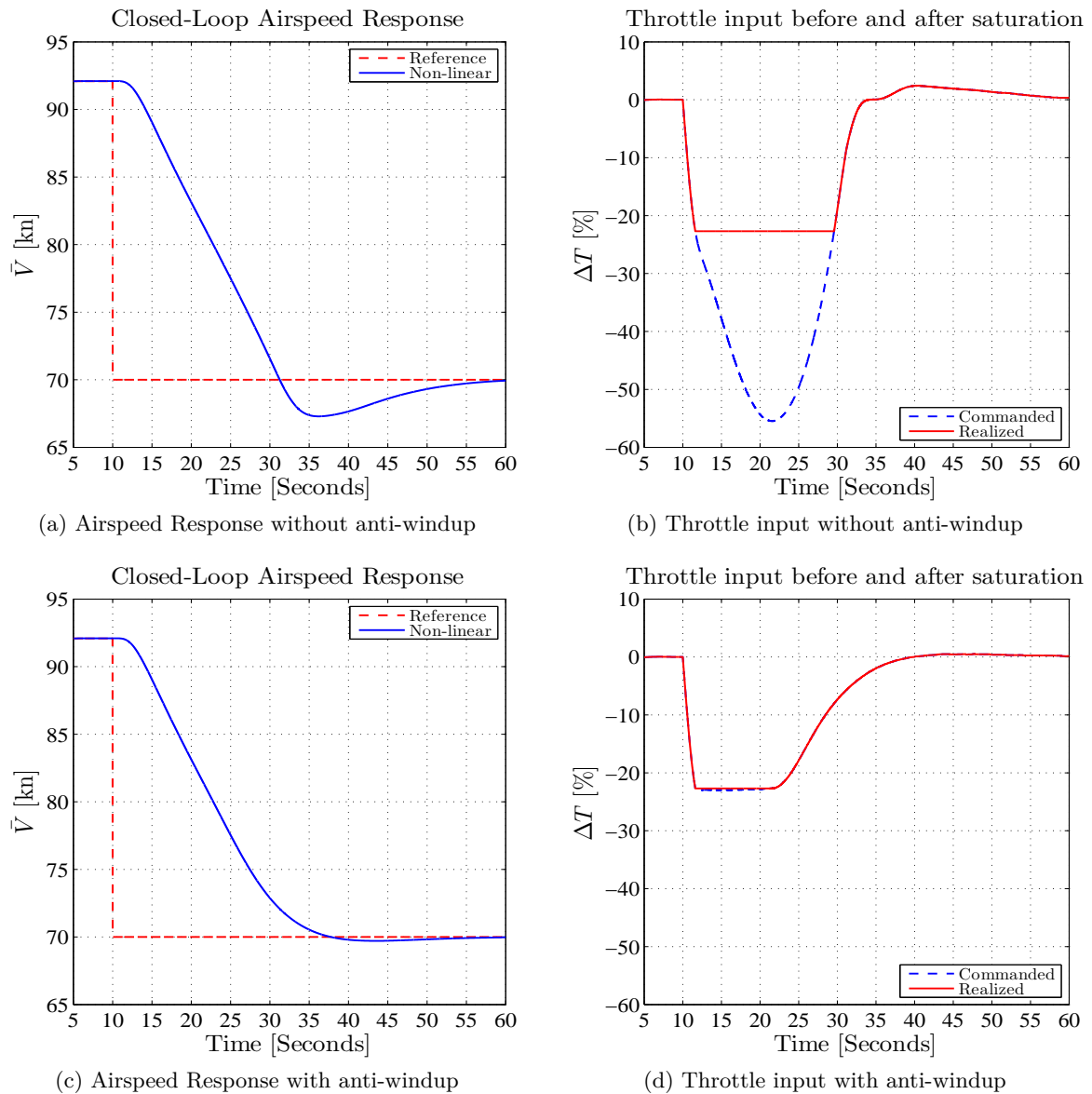


Figure 3.17: Closed-loop step response for an airspeed step with corresponding throttle input  $\Delta T$  before and after saturation for cases (a)-(b) without anti-windup and (c)-(d) with anti-windup

The throttle command in Figure 3.17(d) returns to the linear region and recovers from the saturation much faster than before anti-windup, resulting in reduced overshoot in the airspeed response.



### 3.3 Lateral Control System Design

This section describes the design and verification of the lateral controllers, namely the roll rate controller, the bank angle controller, and the sideslip angle controller (which were shown previously in Figure 3.4). All three these controller functions are encapsulated in a single controller called the DPDR controller.

#### 3.3.1 Roll and Sideslip Angle Controller ('DPDR Law')

The DPDR controller is a single controller that can be used as either an inner-loop roll rate controller or as a middle-loop bank angle controller. In both modes of operation, it also operates as an inner-loop sideslip angle controller. The DPDR controller provides rudder turn co-ordination as well as a natural response to commanded sideslip. A block diagram of the DPDR controller architecture is shown in Figure 3.18.

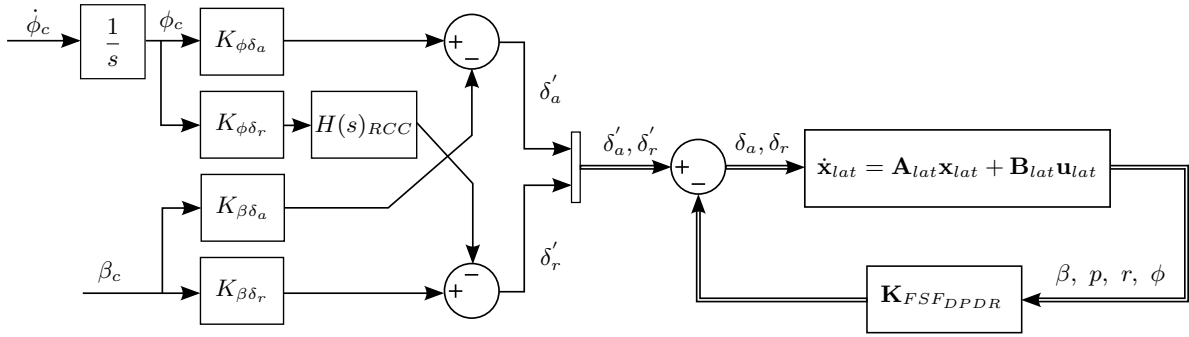


Figure 3.18: DPDR controller architecture

The controller uses feedback from all four lateral states (sideslip angle, roll rate, yaw rate, and bank angle) and actuates both the ailerons and the rudder. The control strategy is full state feedback, and the control design is performed using the full-order lateral dynamics of the aircraft. Input mixing gains are added to introduce the bank angle command and sideslip angle command references. Finally, a rudder co-ordination computer (RCC) is added to provide automatic turn co-ordination control. A similar design process as that used by Trollip [16] is followed.

#### Design

Figure 3.18 shows the architecture of the DPDR controller where the plant for the controller is the lateral dynamics,

$$\dot{\mathbf{x}}_{lat} = \mathbf{A}_{lat}\mathbf{x}_{lat} + \mathbf{B}_{lat}\mathbf{u}_{lat} \quad (3.3.1)$$

where,

$$\mathbf{B}_{lat} = \begin{bmatrix} \mathbf{B}_{\delta_a} & \mathbf{B}_{\delta_r} \end{bmatrix}, \quad \mathbf{u}_{lat} = \begin{bmatrix} \delta_a \\ \delta_r \end{bmatrix} \quad (3.3.2)$$

and the output matrix is,

$$\mathbf{C}_{DPDR} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.3.3)$$

Full-state feedback is used to damp the dutch roll mode and place the lateral poles to achieve the desired closed-loop response. A rudder turn co-ordination controller is used, represented by the transfer function  $H(s)_{rcc}$ , to calculate the correct amount of rudder input needed to produce a sideslip angle to counter the adverse yaw effect of the aircraft when issuing a roll command, and thereby achieving perfect turn co-ordination. The DP law has no turn co-ordination, but the cross forward mixing gain  $K_{\beta\delta_a}$  is used to produce a natural roll response for a commanded sideslip, by allowing a slight roll angle when commanding sideslip.

The full-state feedback control law is given by,

$$\mathbf{u}_{lat} = \begin{bmatrix} \delta'_a \\ \delta'_r \end{bmatrix} - \mathbf{K}_{FSFDPDR} \mathbf{x}_{lat} \quad (3.3.4)$$

The optimal feedback gain  $\mathbf{K}_{FSFDPDR}$  is calculated using a LQR optimisation algorithm that minimises the cost function,

$$J = \int_0^{\infty} \left( \mathbf{x}_{lat}^T \mathbf{Q} \mathbf{x}_{lat} + \mathbf{u}_{lat}^T \mathbf{R} \mathbf{u}_{lat} \right) dt \quad (3.3.5)$$

where the state weighting matrix  $\mathbf{Q}$  and the control weighting matrix  $\mathbf{R}$  are both chosen by the designer to achieve the desired response. As a starting point for the design, the weighting matrices are chosen using Bryson's rule [21], which recommends choosing diagonal matrices with diagonal values,

$$\mathbf{Q}_{ii} = \frac{1}{\text{max acceptable value of } \mathbf{x}_i^2}, \quad i \in (1, 2, 3, 4) \quad (3.3.6a)$$

$$\mathbf{R}_{jj} = \frac{1}{\text{max acceptable value of } \mathbf{u}_j^2}, \quad j \in (1, 2, 3, 4) \quad (3.3.6b)$$

These weightings are then iteratively adjusted to produce the desired response by balancing state response time and control effort.

The input mixing gains are added after full-state feedback is implemented on the lateral dynamics. The augmented control law is then,

$$\mathbf{u}_{lat} = \begin{bmatrix} K_{\phi\delta_a} & K_{\beta\delta_a} \\ K_{\phi\delta_r} H(s)_{RCC} & K_{\beta\delta_r} \end{bmatrix} \begin{bmatrix} \phi_c \\ \beta_c \end{bmatrix} - \mathbf{K}_{FSFDPDR} \mathbf{x}_{lat} \quad (3.3.7)$$

where the forward gains  $K_{\phi\delta_a}$  and  $K_{\beta\delta_r}$  are used to give a unity DC gain from the commands to the outputs respectively to track the reference commands with near zero steady-state error. The forward cross mixing gain  $K_{\phi\delta_r}$  is used to tune the RCC response and  $K_{\beta\delta_a}$  is used to produce a natural roll response for a commanded sideslip.

The control law can be augmented further to enable the controller to follow a reference roll rate command  $\dot{\phi}_c$  by augmenting the roll angle reference with the integral of the roll rate command,

$$\phi_c = \int_0^{\infty} \dot{\phi}_c dt \quad (3.3.8)$$

To use the DPDR controller as an inner-loop roll rate controller, a roll rate reference is supplied to the input of the integrator  $\dot{\phi}_c$ . To use the DPDR controller as a middle-loop bank angle controller, the integrator is bypassed and the bank angle reference is supplied directly to the bank angle input  $\phi_c$ . In both cases the sideslip angle reference is supplied to the sideslip angle input  $\beta_c$ .

### Turn Co-ordination Controller

Conventional flight controllers for large transport aircraft usually implement some form of turn co-ordination control when performing lateral manoeuvres, such as heading changes. When the aircraft banks to turn, it experiences an adverse yaw effect in the form of induced sideslip, which causes unwanted lateral acceleration. A co-ordinated turn is defined as zero lateral acceleration of the aircraft centre of gravity (CG). Turn co-ordination is desirable for passenger comfort and minimises undesirable aerodynamic loading of the structure [22]. Several methods of automatic turn co-ordination exists (see [23]), and for the purpose of the DPDR controller, a method using a component referred to as the rudder co-ordination computer (RCC), will be used.

The RCC component is added to the DR law (see Figure 3.18) that computes the amount of rudder required to produce the necessary sideslip angle for a given amount aileron to counter the adverse yaw effect of the aircraft when banking. The transfer function for the RCC is derived as follows as in [23]. The total sideslip experienced by the aircraft is equal to the sideslip resulting in the rudder and aileron, given by,

$$\beta_{total} = H(s)_{[\delta'_r \rightarrow \beta]} \delta'_r + H(s)_{[\delta'_a \rightarrow \beta]} \delta'_a \quad (3.3.9)$$

where,

$$H(s)_{[\delta'_r \rightarrow \beta]} = \frac{\beta(s)}{\delta'_r(s)} \quad (3.3.10a)$$

$$H(s)_{[\delta'_a \rightarrow \beta]} = \frac{\beta(s)}{\delta'_a(s)} \quad (3.3.10b)$$

But if the co-ordination is perfect, the total sideslip is zero. Then from Equation 3.3.9 we have,

$$\frac{\delta'_r}{\delta'_a} = \frac{-H(s)_{[\delta'_a \rightarrow \beta]}}{H(s)_{[\delta'_r \rightarrow \beta]}} \quad (3.3.11)$$

Assuming that for zero sideslip a zero sideslip reference  $\beta_c = 0$  is applied to the controllers in Figure 3.18 and the only non-zero input is the roll angle reference  $\phi_c \neq 0$ , then from Figure 3.18 we see that,

$$\delta'_r = -K_{\phi\delta_r}\phi_c H(s)_{RCC} \quad (3.3.12a)$$

$$\delta'_a = K_{\phi\delta_a}\phi_c \quad (3.3.12b)$$

where the negative term arises from the fact that, for the sideslip produced by a positive aileron, a negative rudder is required. Dividing Equation 3.3.12a by Equation 3.3.12b we obtain,

$$\frac{\delta'_r}{\delta'_a} = \frac{-K_{\phi\delta_r}\phi_c H(s)_{RCC}}{K_{\phi\delta_a}\phi_c} \approx -H(s)_{RCC} \quad (3.3.13)$$

assuming that any actuator lag for the rudder and aileron are the same and that the feed-forward gains are more or less the same. The transfer function for the RCC can then be obtained by obtaining the transfer functions  $H(s)_{[\delta'_a \rightarrow \beta]}$  and  $H(s)_{[\delta'_r \rightarrow \beta]}$  and using the equation,

$$H(s)_{RCC} = \frac{H(s)_{[\delta'_a \rightarrow \beta]}}{H(s)_{[\delta'_r \rightarrow \beta]}} \quad (3.3.14)$$

### Specifications

The specifications for the DPDR controller were obtained by using the closed-loop responses of a simulated A330 aircraft model as a reference and by consulting similar specifications derived by Trollip for his Masters research [16]. In this way, the suggested DPDR controller specifications for a full scale aircraft were determined, and were then dynamically scaled for the NSA GTM aircraft. For the full-scale aircraft a peak time of less than 10 seconds and roll angle overshoot of less than 5% is desired. For the GTM these specifications translate to a peak time of roughly 2.3 seconds and the same overshoot. The steady-state error does not have to be zero and a specification of a steady-state error of less than 2% was chosen. Using turn co-ordination, the sideslip needs to be regulated close to zero.

The sideslip response specifications were also obtained using the closed-loop response of a simulated A330 aircraft model as a reference. The overshoot should be less than 20 %, but where the A330 for the test condition used as in [16] had a steady-state error of 50 % for the sideslip response, it was decided to achieve a steady -state error that is closer to zero for the sideslip response. Thus it was chosen that the steady-state error should be less than 2 % for a constant reference. Lastly, an additional specification for the sideslip response is to achieve a natural roll response for a commanded sideslip. This means that the steady-state ratio for a bank angle induced by a sideslip angle command should be,

$$\frac{\phi_{ss}}{\beta_{ss}} = -5 \quad (3.3.15)$$

Figure 3.19 shows the closed-loop poles of the lateral dynamics after full-state feedback using LQR optimisation was added. The dominant pole was placed at a location with a time constant of,

$$\tau = 0.206 \text{ s} \quad (3.3.16)$$

resulting in a 2% settling time of approximately,

$$t_{s2\%} = 3.9\tau \approx 0.8 \text{ s} \quad (3.3.17)$$

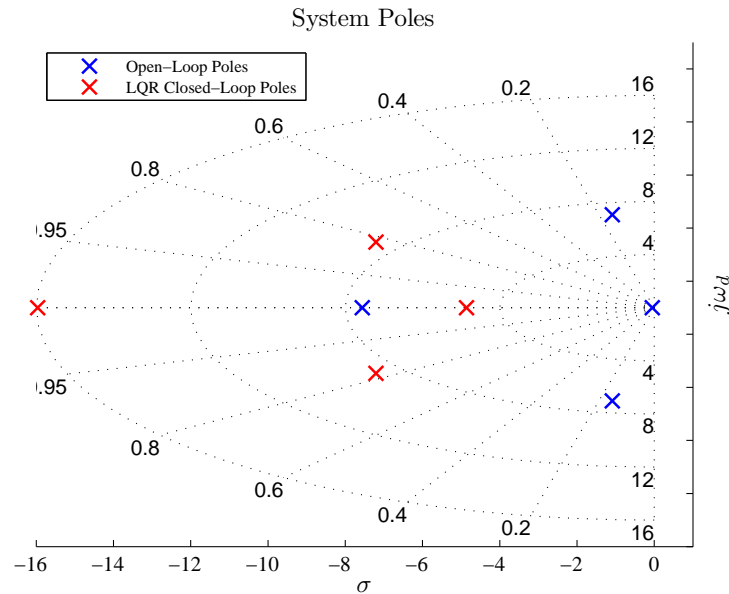


Figure 3.19: Closed-loop poles after full-state feedback using LQR optimisation algorithm

The RCC transfer function was obtained using Equation 3.3.14 and resulted in a 7<sup>th</sup> order transfer function, which is relatively complex. However, as suggested in [23], this transfer function can be adequately approximated using a first-order transfer function on the frequency range over which the control law operates (4 - 10 rad/s). The full-order and reduced-order RCC bode plots are shown in Figure 3.20. It can be seen that the reduced-order RCC bode plot is a good approximation of the full-order RCC bode plot in the frequency range over which the controller operates.

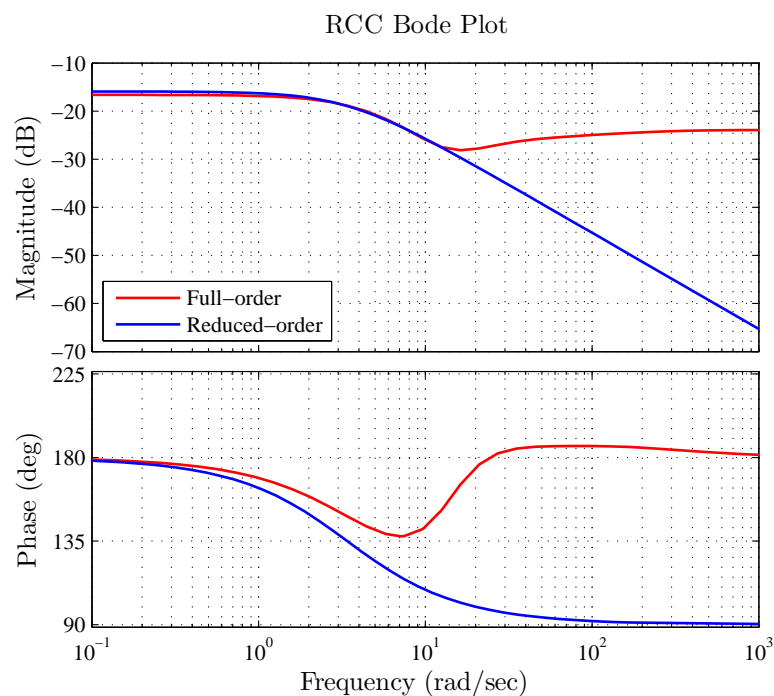
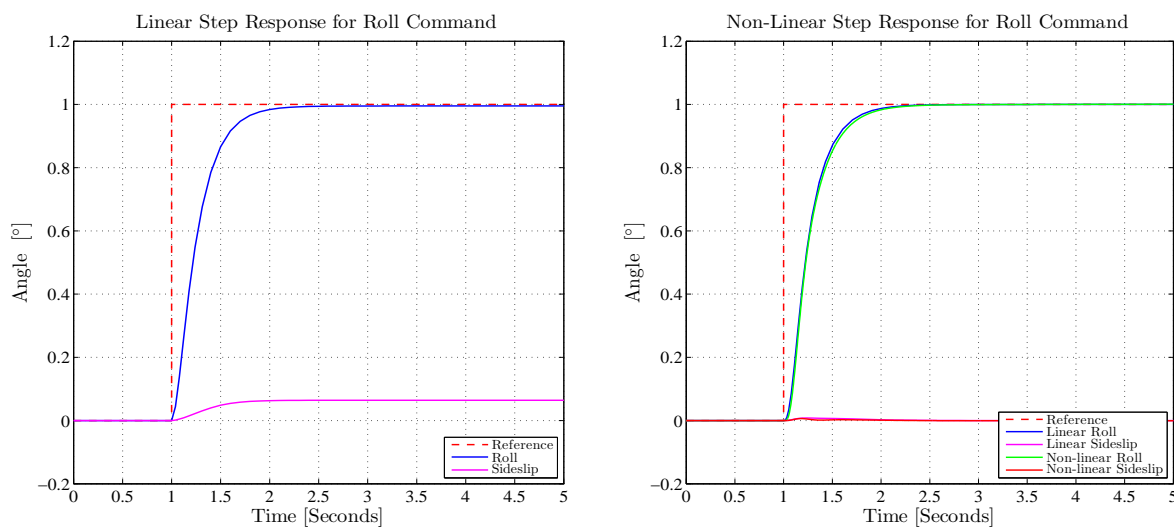


Figure 3.20: Bode plot of full-order and reduced-order RCC

## Results

Figure 3.21 shows the closed-loop roll angle response to a unit commanded roll angle before turn co-ordination (Figure 3.21(a)) and after the RCC was added and the mixing gain  $K_{\phi\delta_r}$  was tuned (Figure 3.21(b)). The non-linear model response matches very well with the linear model response. The non-linear model's roll angle response achieves the desired specifications, with an overshoot less than 5%, a peak time of 1 second (roughly 1 second faster than the requirement) and a steady-state error less than 2%. The RCC successfully minimises the sideslip angle in both the linear and non-linear case, with a sideslip steady-state error of less than 2%.

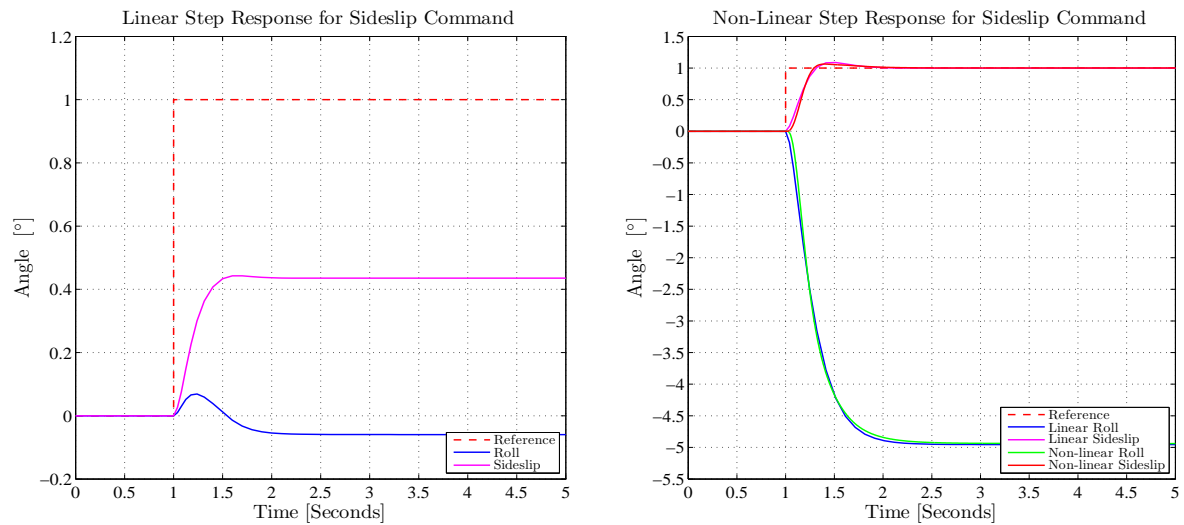


(a) Linear response for roll command without RCC

(b) Linear vs non-linear with RCC

Figure 3.21: Closed-loop roll angle response of DPDR controller with and without RCC

Figure 3.22 shows the closed-loop sideslip response to a unit commanded sideslip angle without the mixing gains used (Figure 3.22(a)) and after the mixing gain  $K_{\beta\delta_r}$  and  $K_{\beta\delta_a}$  were added (Figure 3.22(b)). The non-linear model response matches very well with the linear model response and achieves the desired design specifications. The non-linear sideslip response has an overshoot less than the 20% and exhibits a steady-state error of less than 2%. Both the linear and non-linear models' responses achieve a natural roll response by allowing a roll angle while commanding a sideslip angle, with a steady-state ratio of -5 and thus achieves the natural roll design specification.

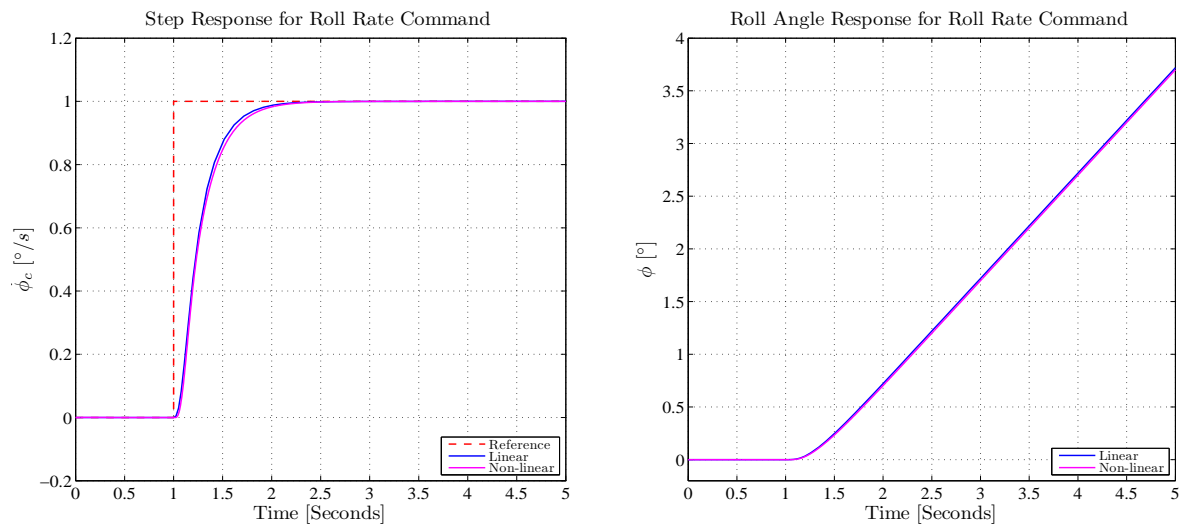


(a) Linear response for sideslip command without mixing gains

(b) Linear vs non-linear response for sideslip command with mixing gains

Figure 3.22: Closed-loop sideslip response of DPDR controller with and without mixing gains

To verify that the DPDR controller can track a roll rate reference, a roll rate step command is given to the roll rate reference input of the controller. Figure 3.23 shows the roll rate and bank angle response to a unit roll rate step command  $\dot{\phi}_c$  for both the linear and non-linear case. The non-linear model's response matches the linear model's response well, with the same response characteristics as the roll angle response. The roll rate response exhibits a steady-state error of less than 2% with a peak time of 1 second. The tracked roll rate step command of 1 degree per second in Figure 3.23(a) produces the expected roll angle ramp response of 1 degree per second in Figure 3.23(b).



(a) Roll rate step response

(b) Roll angle response

Figure 3.23: Closed-loop linear and non-linear roll rate and roll angle response to roll rate step input

### Closed-loop Model

The state space representation of the closed-loop lateral dynamics is obtained by adding the full-state feedback control law to the open-loop lateral dynamics (from Equation 3.3.1),

$$\dot{\mathbf{x}}_{lat} = \mathbf{A}_{latCL}\mathbf{x} - \mathbf{B}_{lat} \begin{bmatrix} \delta'_a \\ \delta'_r \end{bmatrix} \quad (3.3.18)$$

where  $\mathbf{A}_{latCL} = [\mathbf{A}_{lat} - \mathbf{BK}_{DPDR_{FSF}}]$ . The first-order RCC transfer function is added to the system by converting the transfer function into state-space form,

$$\begin{aligned} \dot{x}_{RCC} &= A_{RCC}x_{RCC} + B_{RCC}K_{\phi\delta_r}\phi_c \\ \dot{y}_{RCC} &= C_{RCC}x_{RCC} + D_{RCC}K_{\phi\delta_r}\phi_c \end{aligned} \quad (3.3.19)$$

and then augmenting it into the full-order closed-loop state space representation,

$$\begin{bmatrix} \dot{\mathbf{x}}_{lat} \\ \dot{x}_{RCC} \end{bmatrix} = \begin{bmatrix} \mathbf{A}_{latCL} & \mathbf{0}_{4 \times 1} \\ \mathbf{0}_{1 \times 4} & A_{RCC} \end{bmatrix} \begin{bmatrix} \mathbf{x}_{lat} \\ x_{RCC} \end{bmatrix} + \begin{bmatrix} \mathbf{B}_{\delta_a} & \mathbf{B}_{\delta_r} \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \delta'_a \\ \delta'_r \end{bmatrix} + \begin{bmatrix} \mathbf{0}_{4 \times 1} & \mathbf{0}_{4 \times 1} \\ B_{RCC}K_{\phi\delta_r} & 0 \end{bmatrix} \begin{bmatrix} \phi_c \\ \beta_c \end{bmatrix} \quad (3.3.20)$$

The feed-forward mixing gains are then added to the model with the following substitutions to obtain the complete closed-loop model,

$$\delta'_a = K_{\phi\delta_r}\phi_c - K_{\beta\delta_a}\beta_c \quad (3.3.21a)$$

$$\delta'_r = K_{\beta\delta_r}\beta_c - C_{RCC}x_{RCC} - D_{RCC}K_{\phi\delta_r}\phi_c \quad (3.3.21b)$$

The full-order lateral dynamics for the DPDR closed-loop model can then be expressed in state space form as,

$$\dot{\mathbf{x}}_{DPDR} = \mathbf{A}_{DPDR}\mathbf{x}_{DPDR} + \mathbf{B}_{DPDR}\mathbf{u}_{DPDR}$$

$$\begin{aligned} \begin{bmatrix} \dot{\mathbf{x}}_{lat} \\ \dot{x}_{RCC} \end{bmatrix} &= \begin{bmatrix} \mathbf{A}_{latCL} & -\mathbf{B}_{\delta_r}C_{RCC} \\ \mathbf{0}_{1 \times 4} & A_{RCC} \end{bmatrix} \begin{bmatrix} \mathbf{x}_{lat} \\ x_{RCC} \end{bmatrix} \\ &+ \begin{bmatrix} \mathbf{B}_{\delta_a}K_{\phi\delta_a} - \mathbf{B}_{\delta_r}D_{RCC}K_{\phi\delta_r} & \mathbf{B}_{\delta_r}K_{\beta\delta_r} - \mathbf{B}_{\delta_a}K_{\beta\delta_a} \\ B_{RCC}K_{\phi\delta_r} & 0 \end{bmatrix} \begin{bmatrix} \phi_c \\ \beta_c \end{bmatrix} \end{aligned} \quad (3.3.22)$$

Figure 3.24 shows the poles of the full-order DPDR closed-loop model with the added RCC pole at  $s = -3.4$ .



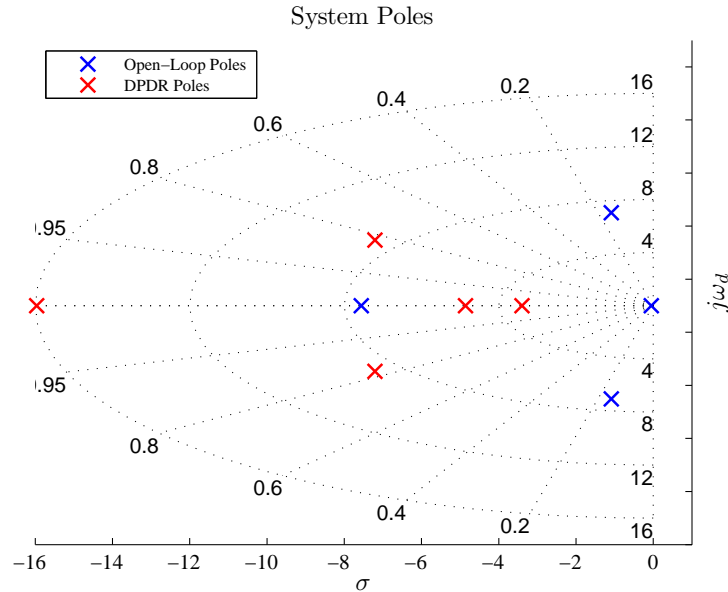


Figure 3.24: Closed-loop poles after adding the full DPDR controller, including the full-state feedback, the input mixing gains, and the turn co-ordination controller

### 3.4 Gain-Scheduled Angle of Attack Controller

The angle of attack controller provides pitch rate damping (just like the normal acceleration controller) but controls the angle of attack instead of the normal acceleration. A block diagram of the gain-scheduled angle of attack controller architecture is shown in Figure 3.25. The controller uses feedback from a pitch rate sensor and an angle of attack sensor to actuate the elevator deflection. The control strategy is full state feedback with integral control, and the controller design is performed using the reduced-order model of the angle of attack dynamics (Equation 3.2.10). The architectural difference between the baseline controller and the new controller, is that the new angle of attack controller includes a feed-forward term from the angle of attack reference to the elevator command to speed up the response. Also, instead of using fixed gains, the controller gains are scheduled as a function of dynamic pressure  $\bar{q}$ .

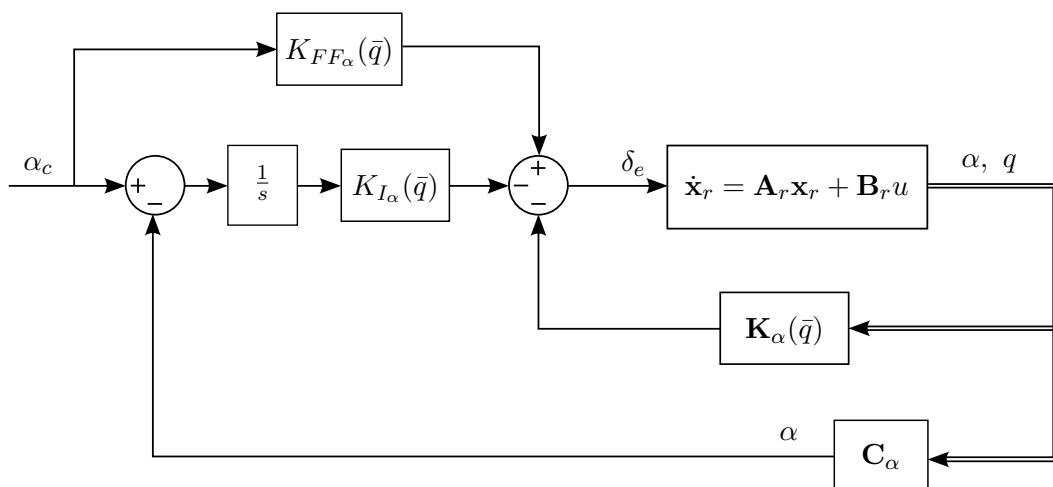


Figure 3.25: Angle of attack controller architecture

### 3.4.1 Design

The same reduced-order longitudinal model of the state dynamics of Equation 3.2.10, that was used in the normal acceleration controller design, is used as the plant for the design of the angle of attack controller. For the gain scheduling design, the dynamics are linearised for several trim points. For each trim point a linear quadratic regulator with integral control (LQI) algorithm is used for the design. The trim points were selected by choosing fixed angle of attack values for level flight conditions and calculating the trim airspeed and trim elevator input. Table 3.2 shows the six trim points chosen for angle of attack and the calculated trim variables for those conditions.

Table 3.2: Calculated trim values for each AoA condition

Trim condition	AoA[deg] $\alpha_T$	Airspeed[kn] $\bar{V}_T$	Elevvator[deg] $\delta_{E_T}$
Condition 1	15	20	-8.75
Condition 2	15	35	-8.75
Condition 3	15	51.23	-4.6
Condition 4	10	56	-1.72
Condition 5	5.5	72.2	0.56
Condition 6	1	133.46	4.37

Trim conditions 3 to 5 are calculated using the normal trim function provided by the GTM model that trims the full aircraft model to an equilibrium condition. However, below an airspeed value of 50 knots, the angle of attack required to fully trim the aircraft becomes too large, exceeding the aerodynamic envelope. For conditions 1 and 2 the angle of attack is capped at a maximum value of 15 degrees and lower airspeed conditions are chosen. Only the trim elevator is calculated using an alternative method, discussed as follows.

For this controller we are only interested in controlling the short-period dynamics and we are not concerned with the equilibrium of the slow kinematic mode, i.e. the translation of the point mass of the aircraft. Thus we simplify the trim calculation by only requiring the aircraft's rotational motion to be in equilibrium, and not requiring the aircraft's translational motion to be in equilibrium. Stated differently, the trim calculation will only require the pitching moments *about* the aircraft centre of mass to be balanced, and will not also require the forces acting *on* the aircraft centre of mass to be in balance. Following this approach we calculate the trim elevator deflection needed to make the total pitching moment zero. This simplified trim equation is given by,

$$\bar{q}_T S \bar{c} C_{m_T} = 0 \quad (3.4.1)$$

which reduces to finding the elevator input that will lead to a total pitch coefficient  $C_{m_T}$  of zero,

$$\begin{aligned} 0 &= C_{m,Static}(\alpha_T) + C_{m,\delta_E}(\delta_{E_T}) \\ \delta_{E_T} &= C_{m,\delta_E}^{-1}(-C_{m,Static}(\alpha_T)) \end{aligned} \quad (3.4.2)$$

For conditions 1 and 2 the reduced-order longitudinal model cannot be obtained using the GTM's linearisation script, because we cannot use the standard trim script for these conditions.

Instead, the state space terms are derived from the simplified analytical expression of the partial derivative terms,

$$\mathbf{A}_r = \begin{bmatrix} 0 & 1 \\ \frac{\bar{q}_T S \bar{c}}{I_{yy}} C_{m_\alpha} & \frac{\bar{q}_T S \bar{c}}{I_{yy}} \frac{\bar{c}}{2\bar{V}_T} C_{m_Q} \end{bmatrix}, \quad \mathbf{B}_r = \begin{bmatrix} 0 \\ \frac{\bar{q}_T S \bar{c}}{I_{yy}} C_{m_{\delta_E}} \end{bmatrix} \quad (3.4.3)$$

where  $I_{yy}$  is the principle moment of inertia around the body y-axis. The terms of the form,

$$C_{AB} = \frac{\partial C_A}{\partial B'}, \quad B' = nB \quad (3.4.4)$$

are the stability and control derivatives, where the normalising coefficient  $n$  for wind angles and control deflections is 1 and the pitch rate is  $n = \frac{\bar{c}}{2\bar{V}}$ . The terms in Equation 3.4.3 were derived by Peddle in [17] with the additional assumption that the angle of attack rate is equal to the pitch rate  $\dot{\alpha} = q$ . This means that we assume the only coefficients affecting the short-period dynamics are the pitch stability derivatives  $C_{m_Q}$ , and not the lift stability derivatives  $C_{L_Q}$ .

With the second-order linear state space model obtained for each trim condition, a LQI design is done for each of them. The LQI method is an optimal state feedback LQR design where the plant is augmented with an integrator state. The LQI algorithm is used to obtain the feedback gain  $K_\alpha$  and integrator gain  $K_{I_\alpha}$ . For each case the feed-forward gain  $K_{FF_\alpha}$  is determined that will cancel out the integrator pole to produce the same response as the normal acceleration controller. This is done in the same way as with the normal acceleration controller design,

$$K_{FF_\alpha} = \frac{K_{I_\alpha}}{p_{i_\alpha}} \quad (3.4.5)$$

where  $p_{i_\alpha}$  is the closed-loop pole location of the integrator state.

The gains associated with each trim airspeed case are then linearly interpolated (scheduled) as a function of dynamic pressure  $\bar{q}$ , which is a function of the airspeed. Thus each of the dynamic pressure breakpoints is calculated from each trim airspeed case,

$$\bar{q}_{T_i} = \frac{1}{2} \rho \bar{V}_{T_i}^2, \quad i \in (1, \dots, 6) \quad (3.4.6)$$

The specification for the angle of attack controller is that it should have roughly the same response as the designed normal acceleration controller, with similar damping characteristics. This required the feed-forward term to be added to cancel out the integrator dynamics. Some of the angle of attack controller's trim conditions could not achieve the exact same closed-loop characteristics as the normal acceleration controller's response, because in some cases the open-loop poles of the trim conditions were far away from the closed-loop poles of the normal acceleration controller. This means that moving the poles far away from their natural open-loop response causes increased control effort to be required, which in some cases led to unwanted closed-loop oscillations in the response. Thus, the trade-off was made to have the closed-loop angle of attack response of each of the trim conditions be as close to the response of the normal acceleration controller, yet still remain close enough to the open-loop dynamics of the respective trim condition to avoid excessive control effort. The open-loop and closed-loop pole locations of the reduced-order model representing the short-period mode of the angle of attack controller's trim conditions are given in Table 3.3.

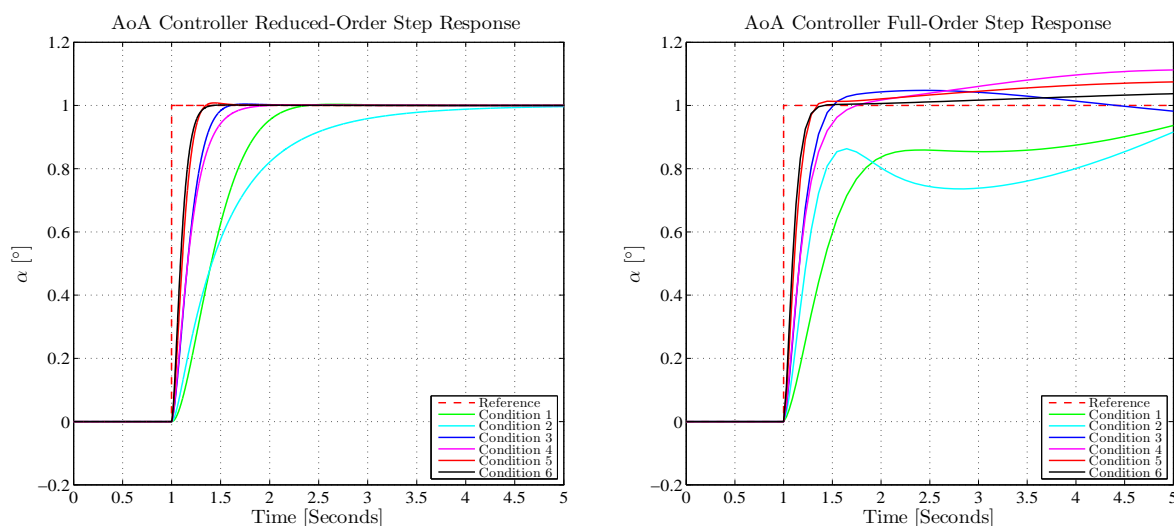
Table 3.3: Gain-scheduled angle of attack closed-loop short-period design specifications

Trim condition	Open-loop $\omega_n$	Open-loop $\zeta$	Closed-loop $\omega_n$	Closed-loop $\zeta$
Condition 1	2.7	0.12	3.9	0.87
Condition 2	4.84	0.12	6.4	0.82
Condition 3	7.27	0.14	9.1	0.9
Condition 4	3.75	0.46	16.4/6.47	1
Condition 5	6.36	0.44	13.5	0.85
Condition 6	11.6	0.49	17.5	0.95

The LQI weights were adjusted to give a damping ratio of between 0.8 and 1 for all the conditions. The natural frequency of the short-period mode becomes very slow at low airspeeds and high at higher airspeeds. Condition 4 has the closed-loop short-period poles as two real poles. The closed-loop poles of the conditions were kept close to the natural frequency of the open-loop pole so that it is close to the natural mode of the aircraft and to prevent large gains that cause instability. The integrator poles all have a frequency of roughly 1.3 rad/s.

### 3.4.2 Results

Figure 3.26 shows the step responses for each trim condition using the reduced-order and full linear models. Conditions 1 and 2 are at very low airspeeds close to stall and show noticeably slower closed-loop responses, where the natural responses are relatively slow.



(a) Closed-loop step response of reduced-order AoA controller for 6 trim conditions

(b) Closed-loop step response of full linear AoA controller for 6 trim conditions

Figure 3.26: Results of linear AoA controller closed-loop unit step responses

The gain scheduled design improved the baseline angle of attack controller's response at low airspeeds. The original problematic responses of the baseline angle of attack controller can be seen in Appendix D during verification of the initial trajectory control schemes. The baseline angle of attack controller showed large undesirable oscillations in the angle of attack output at

low speeds, and was unable to track the planned angle of attack trajectory as a result. The gain-scheduled angle of attack controller will be revisited in Chapter 5 when it is used in the trajectory execution control schemes to execute the planned upset recovery trajectories on the full non-linear aircraft model. It will be seen that the gain-scheduled angle of attack controller achieves good angle of attack tracking (Figures 5.12 and 5.14) for a large range of airspeeds (40 - 100 knots).

### 3.5 Command Tracking for Flight Controllers

The conventional flight controllers are intended to be used in trajectory control schemes of the next chapter. To validate the performance of the controllers for this use, they should be tested with various command signals on the full non-linear simulation model of the NASA GTM. In this section the conventional flight controllers that were designed in the previous sections are tested with various artificial step and ramp commands given to either the middle-loop or inner-loop controllers to test the integrated inner-loop and middle-loop control system's command tracking performance. The inner-loop controller's will typically receive step references, while the middle-loop controllers will typically receive ramp references.

Figure 3.27 shows the normal acceleration and flight path angle response to a normal acceleration step command with the airspeed controller active. If the airspeed controller is not active in parallel, the changing airspeed would act as a changing disturbance signal on the flight path angle response in steady-state, i.e. phugoid dynamics would be present in steady-state. The tracked positive normal acceleration step in Figure 3.27(a) corresponds to a negative flight path angle rate in Figure 3.27(b) as expected, because a positive normal acceleration means a positive acceleration in the down direction. The normal acceleration step of  $1 \text{ m/s}^2$  in magnitude should result in a flight path angle rate equal to the normal acceleration divided by the constant airspeed value, and the flight path angle response in Figure 3.27(b) exhibits the expected flight path angle rate of roughly  $0.021 \text{ rad/s} = 1.2 \text{ deg/s}$  at an airspeed value of  $47.37 \text{ m/s}$ . The flight path angle response also shows that the flight path angle rate becomes constant at  $t = 1$  second when the normal acceleration step is given, as is evident by the flight path angle response resembling a ramp response.

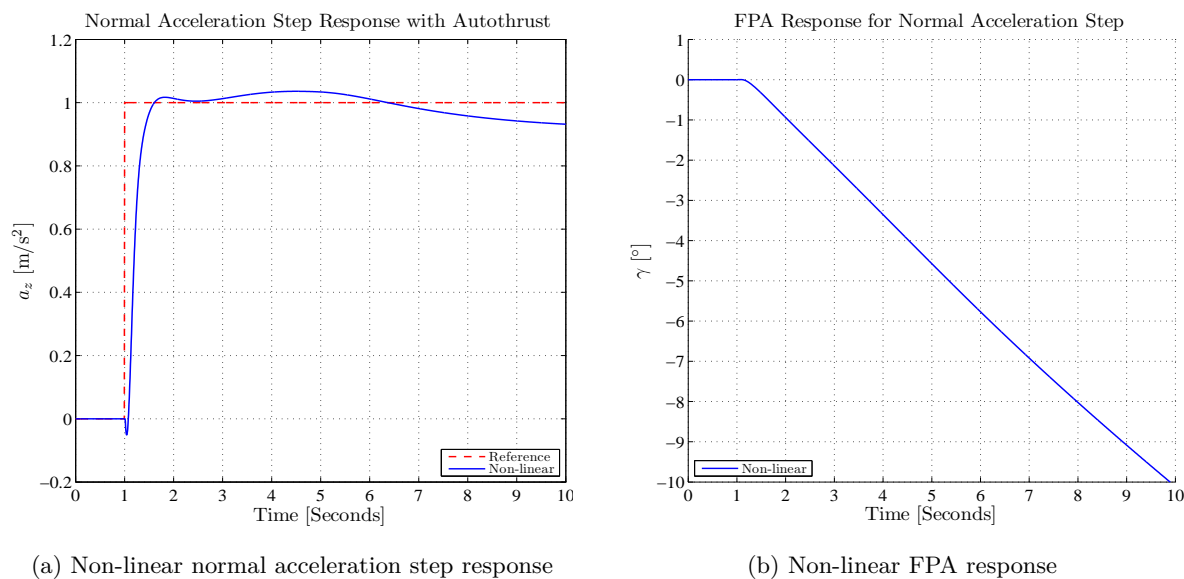
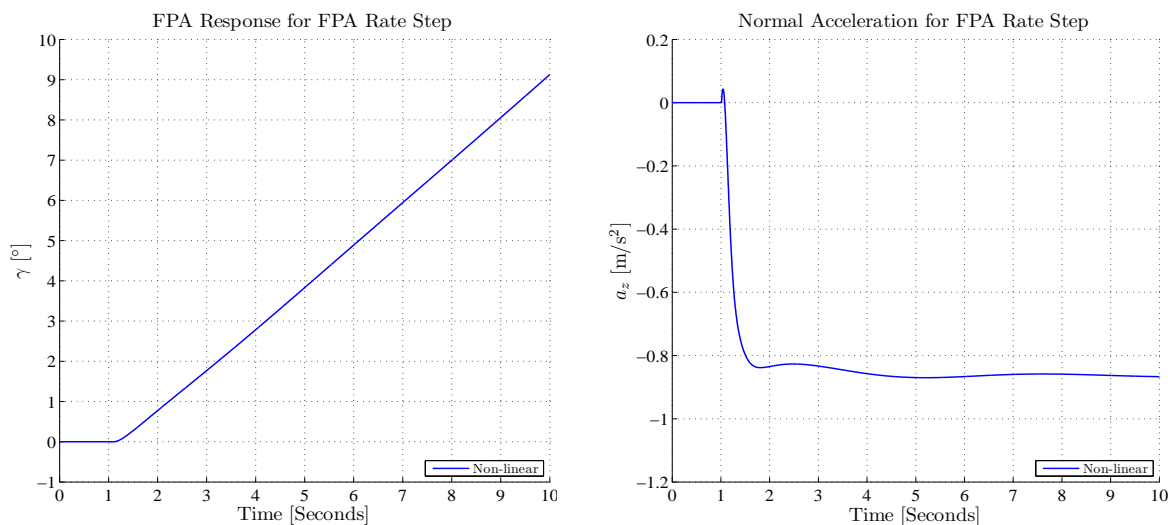


Figure 3.27: Closed-loop non-linear model response to a normal acceleration unit step with autothrust controller active

The flight path angle controller commands a flight path angle rate  $\dot{\gamma}_c$  which is proportional to the flight path angle error. This flight path angle rate command is then converted to an equivalent normal acceleration reference for the inner-loop normal acceleration controller. In order to verify that the commanded flight path angle rate  $\dot{\gamma}_c$  results in the correct flight path angle rate response, the term  $\dot{\gamma}_c$  in Equation 3.2.25 is replaced by a step command, which will be converted into a reference command for the normal acceleration controller, and the resulting flight path angle response is then simulated. Figure 3.28 shows the flight path angle and normal acceleration response to a unit flight path angle rate step command  $\dot{\gamma}_c = 1$  at  $t = 1$  second with the autothrust controller active.

Figure 3.28(a) shows that the flight path angle rate  $\dot{\gamma}_c$  changes at a constant rate of 1 deg/s after  $t = 1$  seconds. This confirms that the flight path angle rate command results in the correct flight path angle rate. Figure 3.28(b) shows that the flight path angle rate command results in the correct normal acceleration output. The normal acceleration response is negative, which is expected, as a negative normal acceleration should result in a positive flight path angle rate. The normal acceleration value observed in the Figure 3.28(b), divided by the nominal airspeed  $\bar{V}$  does in fact result in the expected flight path angle rate of 1 deg/s in Figure 3.28(a). Thus the response verifies that the flight path angle rate command  $\dot{\gamma}_c$  is correctly converted into a normal acceleration command for the normal acceleration controller, and that the normal acceleration output results in the expected flight path angle rate.



(a) Non-linear model normal acceleration response

(b) Non-linear model FPA acceleration response

Figure 3.28: Closed-loop non-linear model response to a FPA rate  $\dot{\gamma}$  unit step with autothrust controller active

During recovery manoeuvres it is expected that the middle-loop flight path angle controller will receive reference commands that are similar to ramp commands. Thus the flight path angle controller's ability to follow a ramp reference is investigated in simulation. The middle-loop flight path angle controller is given a flight path angle ramp command as reference, and the flight path angle controller in turn issues a normal acceleration command to the inner-loop normal acceleration controller. Figure 3.29 shows the flight path angle, normal acceleration, airspeed and bank angle response to a flight path angle ramp command from an initial flight path angle condition of  $\gamma = -40^\circ$  to  $\gamma = 0^\circ$ , while issuing a constant trim airspeed reference to the airspeed controller. As discussed previously, the airspeed should be regulated to a constant value in order to prevent the airspeed to act as a disturbance signal onto the normal acceleration, that would affect the flight path angle response in steady-state.

Figure 3.29(a) shows that the flight path angle signal follows the ramp reference with some lag. This is expected, as the middle-loop flight path angle controller is a type 1 system that can only follow ramp reference with a finite steady-state error. After  $t = 4$  seconds the ramp command transitions into a constant zero flight path angle command that is then tracked with zero steady-state error. Figure 3.29(b) shows the corresponding normal acceleration signal that results in the flight path angle response. Figure 3.29(c) shows that the airspeed initially increases while the flight path angle is negative. This is due to a component of gravity that accelerates the aircraft when the velocity vector is pointed downwards. Once the flight path angle returns to level flight, the airspeed controller regulates the airspeed back to the trim value. Figure 3.29(d) shows that the bank angle is regulated to remain zero.

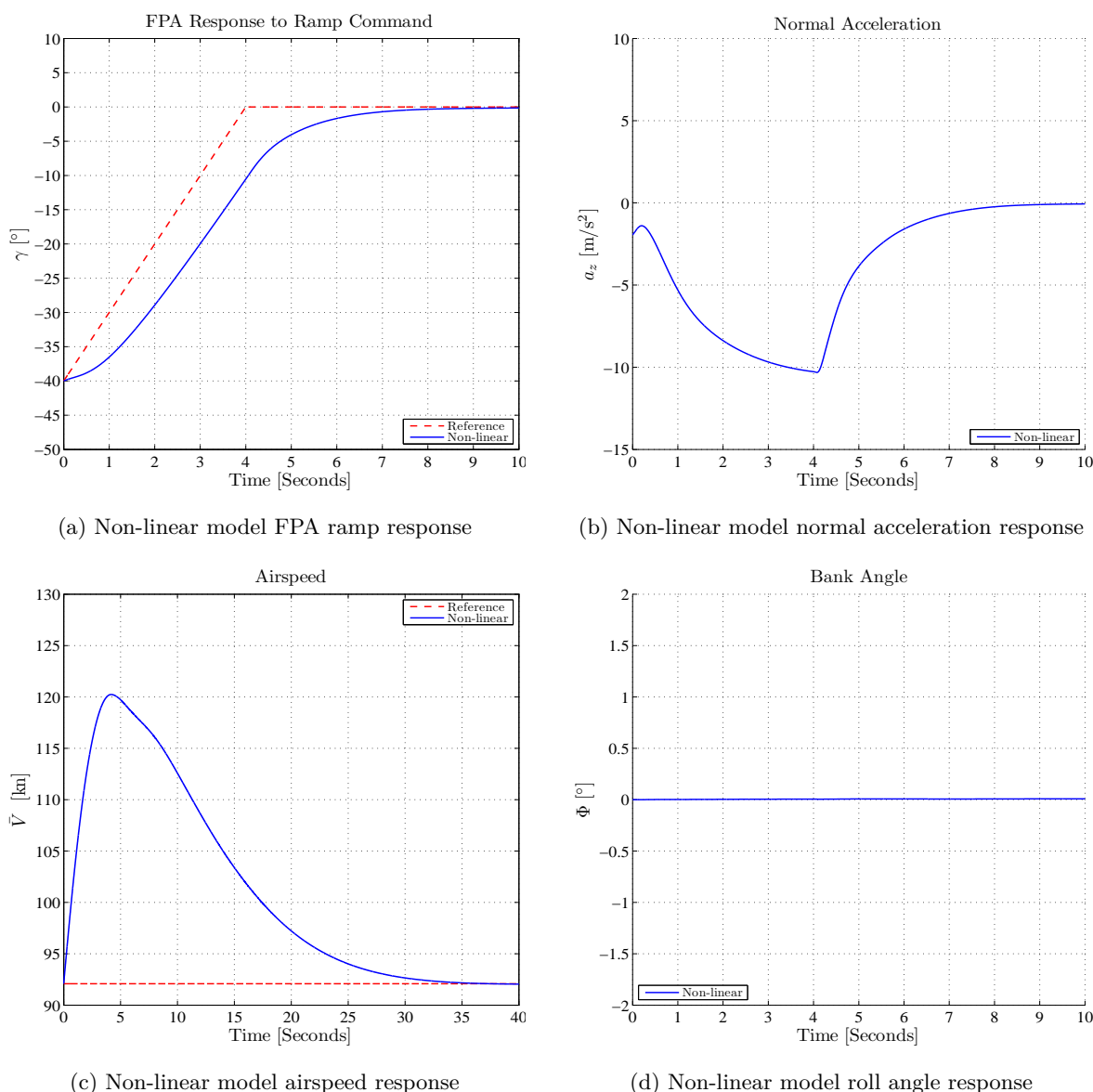


Figure 3.29: Closed-loop non-linear model response to FPA ramp reference command

During recovery manoeuvres it is expected that the middle-loop bank angle controller will receive reference commands that are similar to ramp commands. Thus the bank angle controller's ability to follow a ramp reference is investigated in simulation. The bank angle controller is given a ramp command as reference, while the middle-loop flight path angle is commanded to regulate the flight path angle to zero. Figure 3.30 shows the flight path angle, normal acceleration, airspeed and bank angle response to a bank angle ramp command from an initial bank angle condition of  $\Phi = 150^\circ$  to  $\Phi = 0^\circ$  while issuing a constant trim airspeed reference.

Figure 3.30(d) shows that the bank angle response tracks the bank angle reference command, but the bank angle response slightly lags the bank angle command. This is expected, as the middle-loop bank angle controller uses feedback to correct the error between the bank angle and the reference command, and must first measure an error signal in the state in order to act on the error by issuing a feedback command. Figure 3.30(a) shows that the flight path angle initially 'dips' and becomes negative before being recovered back to zero. The flight path angle controller is designed to only issue non-zero specific acceleration commands when the



bank angle is below 70 degrees, and only let acceleration due to gravity effect the flight path angle when the bank angle is higher than 70 degrees. This is to prevent equivalent negative normal load factor commands from being given at high or inverted bank angles. At  $t = 2.5$  seconds the bank angle transitions below 70 degrees, and the flight path angle controller starts issuing non-zero normal specific acceleration commands in order to regulate the flight path angle back to zero degrees. Figure 3.30(b) shows the normal acceleration response. The step signal ‘dip’ at  $t = 2.5$  is when the bank angle transitions below 70 degrees and the flight path angle controller starts issuing non-zero normal specific acceleration commands which result in a large negative normal acceleration response to regulate the flight path angle. Figure 3.30(c) shows the airspeed response, which initially increases due to the negative flight path angle due to gravity as previously discussed. However, at around  $t = 2.5$  seconds the airspeed falls below the trim airspeed. This is due to the large aerodynamic drag produced by the large negative normal acceleration command. A large negative normal acceleration responses would typically produce large angles of attack that produces more drag, which in turn would cause a decrease in airspeed.

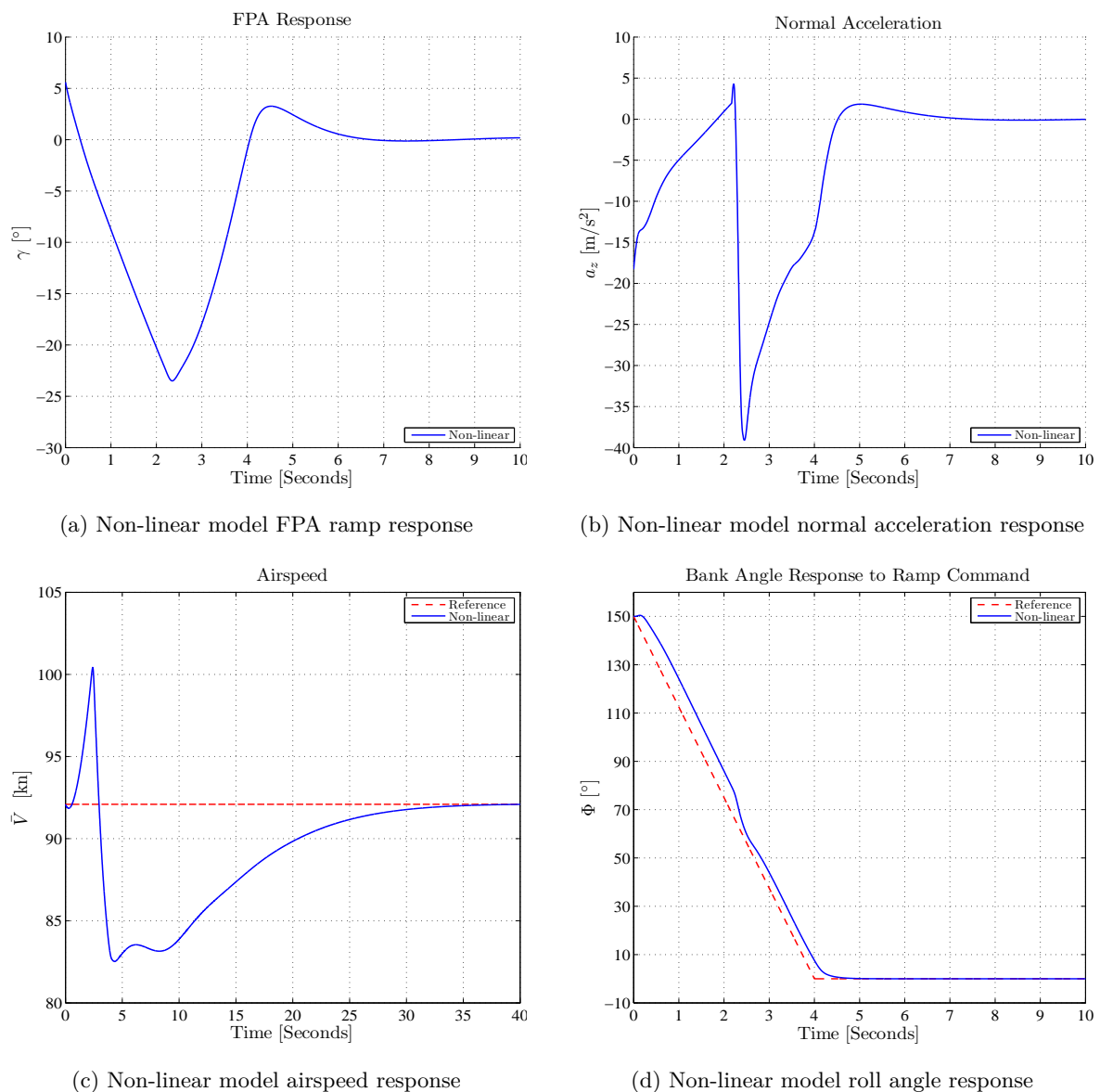


Figure 3.30: Closed-loop non-linear model response to roll angle ramp reference command

The controllers perform as expected where the flight path angle controller and bank angle controller follow their respective ramp references with a lagging steady-state error due to the controller being type 1 systems. In the case of the bank angle ramp reference, the flight path angle dips initially, but is regulated back to level flight as the bank angle transitions below 70 degrees.

With the conventional inner- and middle-loop flight controllers designed and successfully tested with time-varying reference signals, they can be used for the trajectory execution control schemes of Chapter 5. The closed-loop models of the inner-loop controllers can also be used in the formulation of the trajectory optimisation problem for the trajectory planning methods presented in the next chapter.

## Chapter 4

# Optimal Trajectory Planning

The problem of attitude and flight vector recovery can be divided into two major sub-problems: trajectory *planning* and trajectory *execution*. First the optimal recovery trajectory must be planned, and then the planned recovery trajectory must be executed using the existing flight control system of the aircraft. This chapter will focus on the first sub-problem of trajectory *planning*, while Chapter 5 will focus on the second sub-problem of trajectory *execution*. This chapter presents two trajectory planning methods for optimal attitude and flight vector recovery, namely dynamic programming (DP) and sequential quadratic programming (SQP).

The dynamic programming (DP) trajectory planner was developed by Engelbrecht (the supervisor of this research project) for his PhD dissertation [1]. The recovery problem was formulated as an optimal control problem and then solved using a dynamic programming algorithm to find the optimal state trajectories and the optimal sequence of control inputs to recover the attitude, flight path angle, and airspeed of the aircraft from upset conditions with minimal altitude loss. The aircraft dynamics were simplified to the slower point mass translational dynamics of the aircraft, while the fast rotational dynamics were abstracted through time-scale separation. The reduced-order model of the aircraft dynamics made the optimal control problem tractable to be solved with dynamic programming. (Dynamic programming suffers from the “curse of dimensionality” which limits it to using lower-dimensional models of the aircraft dynamics.)

The sequential quadratic programming (SQP) trajectory planner was proposed and developed by Engelbrecht (the author) for this Master’s degree research project. The SQP planner uses direct transcription to transcribe the optimal control problem into a non-linear programming problem, which is then solved using a constrained optimisation algorithm called sequential quadratic programming. The SQP algorithm does not suffer from the “curse of dimensionality” and therefore allows higher-dimensional models to be used for the trajectory planning. The SQP planner therefore uses a higher-dimensional model of the aircraft dynamics that includes the fast rotational dynamics (as represented by the inner-loop angle of attack controller and roll rate controller) as well as the engine lag dynamics.

The chapter is organised as follows: first some background on numerical optimisation methods is provided and an overview of the methods that were considered is given. The recovery problem is then formulated as an optimal control problem, and the reduced-order models of the aircraft dynamics that are used for the trajectory planning are presented. The dynamic programming (DP) approach developed by Engelbrecht (the supervisor) is presented first, to provide context and to serve as the baseline against which the SQP approach will be compared. The SQP approach developed by Engelbrecht (the author) is then presented, and its performance is compared to that of the DP approach. Finally, the advantages and disadvantages of the DP trajectory planner and the SQP trajectory planner are discussed.

The research presented in this chapter has been published by the author in Engelbrecht and Engelbrecht [8].

## 4.1 The Upset Recovery Problem

The objective of the optimal attitude and flight vector recovery is to recover the bank angle to wings level, the flight path angle to level flight, and the airspeed to an acceptable range, with the minimum of altitude loss, while adhering to the constraints of the aircraft dynamics and structural limitations [1].

## 4.2 Trajectory Optimisation Background

This section provides background on different numerical methods used for trajectory optimisation, and then provides some cases in literature of previous research done on the upset recovery problem.

### 4.2.1 Numerical Approaches for Optimal Control

There is a vast amount of numerical methods for solving the trajectory optimisation problem and considering all of them is beyond the scope of this thesis. The material presented here is mainly based on numerical methods for optimal control surveys by Betts and Rao [24; 25] and trajectory optimisation tutorial papers by Kelly [26; 27].

The field of numerical optimisation has seen considerable progress in the past few decades. Some of the first applications of numerical trajectory optimisation methods could be seen in the late 60's early 70's. These applications used numerical techniques called variational calculus for aerospace missions, while numerical gradient based methods were still in their infancy. Today, powerful modern global numerical optimising routines exist which can handle tens of thousands of variables. A large contributor to the advancements in the field of numerical methods, is the increase of computational power and the amount of fast memory available. Methods that were previously thought to be intractable for the computers of the day, have become viable methods in the age of modern computing.

With the usefulness of numerical methods increasing in almost every field of engineering, an increasing amount of applications for these methods are being found. This thesis explores the application of numerical methods on the optimisation of aircraft trajectories, particularly upset recovery trajectories. Previous traditional uses of numerical methods in the aerospace field were mainly confined to space vehicle missions. The mission trajectory solutions could be computed offline, pre-mission, and then loaded onto the vehicle to execute in sequence. Alternatively, the space vehicle is in a parking orbit and there is relatively ample time to compute updated trajectories offline and resend them to the vehicle.

Numerical methods for optimal control problems can be divided into trajectory optimisation methods, which deals with open-loop solutions, and so-called policy optimisation methods, which deals with closed-loop solutions. Figure 4.1 shows a diagram of the broad categories of numerical methods used for solving optimal control problems. Policy optimisation methods considers the entire state space and gives a solution as a function of the state. Dynamic programming is one of the main examples of a policy optimisation method. The main shortcoming of conventional dynamic programming algorithms is the so-called “curse of dimensionality” and the computational burden of the dynamic programming algorithm scales exponentially with the dimension of the problem. Some heuristic methods, such as genetic algorithms, can also

be seen as policy optimisation methods. However, genetic algorithms do not give closed-loop solutions. Heuristic methods such as genetic algorithms are typically slow compared to gradient based methods, and more suited towards problems with very non-linear behaviour.

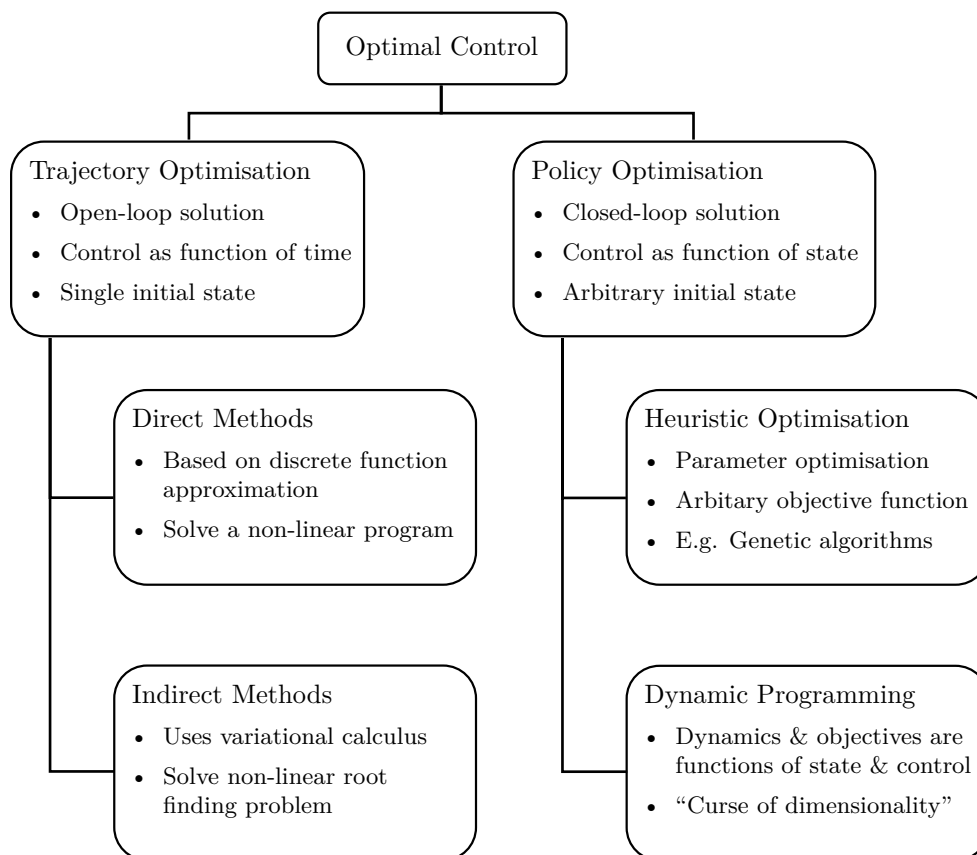


Figure 4.1: Numerical methods for solving optimal control problems

Open-loop trajectory optimisation methods provide solutions that are a function of time and are typically suited for problems with high-dimensions. Open-loop trajectory optimisation methods can be part of one of two broad categories depending on their formulation, namely direct methods and indirect methods. Trajectory optimisation problems are formulated to be solved with parameter optimisation methods known as non-linear programming problem (NLP) solvers which are all essentially gradient-based Newton methods.

With indirect methods, variational calculus [28] is used to explicitly determine the first-order optimality/necessary conditions of the *original* optimal control problem, that involve helper variables called adjoint variables. Variational calculus is a method concerned with finding functions that optimise a functional (a function of functions). These conditions formulated with variational calculus are then discretised and lead to a two-point boundary-value problem (TPBVP) which involves numerically solving a system of non-linear algebraic equations to determine candidate optimal trajectories called extremals [25]. This is a continuous solution and usually employs an ordinary differential equation solver (ODE), which commonly uses a root-finding method that is driven by a NLP routine. Deriving the optimality conditions analytically for complicated problems can be challenging and initialising an indirect method can be problematic, as the user must guess the initial value of the adjoint variables, which often has no physical meaning.

Direct methods are based on discrete function approximation where the state and/or controls of the optimal control problem is first parameterised/discretised in some manner and the problem is transcribed into a non-linear programming problem which is then solved using a NLP solver such as sequential quadratic programming (SQP). The optimal control problem is then solved by solving the *new* first order optimality conditions for this new NLP and implicitly solves the optimality conditions of the original control problem. By solving the NLP the new optimality conditions are satisfied/solved automatically and not analytically as in the indirect method.

Two main types of direct methods exist, namely shooting methods and direct transcription methods. Shooting methods are a common direct method that only parameterise the control and initial state. Shooting methods simulate the state dynamics forward in time by solving the system's differential equations sequentially in time using numerical time marching routines [25]. Direct shooting methods are more suited for problems with few path constraints such as space flight missions as it is difficult to implement path constraints with these methods. Direct transcriptions methods, also called simultaneous methods, are very powerful methods that have gained significant popularity in practice. Transcription methods parameterise both the state and control variables using a method known as collocation to simultaneously solve the optimisation problem and dynamics of the system. Direct transcription methods have become popular for solving complex optimal control problems such as various forms of robot locomotion.

In light of dynamic programming's shortcomings, this project explores some other implementation using numerical methods, focussing on open-loop trajectory optimisation methods, since they are suited for problems with high-dimensions. Both indirect and direct open-loop methods were considered.

For the flight trajectory optimisation problem, unique difficulties arise with this indirect formulation. Variational calculus requires that the first-order optimality conditions of the problem must be derived analytically, and must be re-derived for each new dimensionality added. This results in a lengthy process which must be repeated for every new case. More importantly, the appearance of discrete aerodynamic coefficient lookup tables in the problem made it unsuited to be solved with variational calculus. To solve the optimality conditions analytically the functions involved must be continuous and have closed-form equations. Thus having discrete lookup tables severely complicates the derivation process.

The direct formulation has proven to be a more intuitive and versatile way of formulating the upset recovery problem that is the subject of this thesis, and can easily be extended to higher dimensionalities. Gradient based numerical methods have become very powerful and have proven an effective method to solving direct formulated problems. Modern gradient-based methods, such as sequential quadratic programming (SQP), have the advantage of incredibly fast convergence to the solution if initialised correctly, and can accommodate non-linear equality and inequality constraints. Although these gradient-based methods are known to be very sensitive to initial guesses, matured methods such as the SQP are designed to work with poor initial guesses. These methods alone are also local optimisation routines as they can converge to a local minima that is highly dependent on the initial guess to the solution.

As for choosing a specific NLP solver, modern gradient-based constrained NLP solvers are very similar as all are some variation of a Newton step method, with differences in how the constraints are handled and in the methods of approximation. The sequential quadratic programming (SQP) routine was chosen as it is a common method used in practice due to speed, accuracy and reliability above other methods and the routine was readily available in the Matlab 2010 version that worked with the supplied GTM simulation model.

### 4.2.2 Previous Research

Two cases that specifically used numerical methods for attitude and flight vector recovery of a large transport aircraft were found in literature:

An optimal attitude and flight vector recovery approach was proposed by Engelbrecht [1] in his PhD thesis. The recovery problem was formulated as an optimal control problem and then solved using a dynamic programming algorithm to find the optimal state trajectories and the optimal sequence of control inputs. The aircraft model used was the Generic Transport Model (GTM) developed by NASA. The aircraft dynamics was simplified to a reduced-order model that describes only the slower point mass translational dynamics of the aircraft, while the fast rotational dynamics were abstracted through time scale separation. This reduced-order model of the aircraft dynamics enabled the optimal control problem to remain tractable to be solved by the dynamic programming method [1]. Engelbrecht suggested that other approaches to solving the optimal control problem, such as the calculus of variations approach, the sequential quadratic programming approach, and the sampling-based path planning approach, could be investigated.

An example of previous work that specifically used a NLP to solve an aircraft attitude and flight vector recovery problem was presented by Sparks and Moerder [29] of NASA's Langley Research Center. They employed trajectory optimisation using a non-linear programming solver SNOPT to solve a simulated upset scenario taken from a real-world case which included control surface failures [8]. The aircraft model used was a 6-degrees-of-freedom simulation model of a 737-100 research aircraft. The method was based on a second-order midpoint collocation scheme and also used a Fortran based automatic differentiation method ADIFOR 2.1 to obtain the gradient information of the problem.

## 4.3 Optimisation Problem and System Model Formulation

Given the initial aircraft state, the trajectory planner must generate the optimal reference trajectories for airspeed, flight path angle, and wind-axis bank angle, and the optimal sequence of angle of attack, wind-axis roll rate, and thrust commands to recover the aircraft to straight and level flight with an acceptable final airspeed while minimising the peak altitude lost during the recovery manoeuvre [1].

The problem of flight trajectory optimisation is posed as an optimal control problem and the general mathematical formulation of the optimal control problem is given in this section. The specific dynamics of the problem, which uses a reduced-order model of the aircraft dynamics, is then given along with the problem's objective function. Thereon follows the state termination conditions and the associated state and control constraints of the system. The next section then details methods used to solve this formulated optimal control problem.

### 4.3.1 Point Mass Translational and Input Dynamics

The system state dynamics for the problem of flight trajectory optimisation are given by the reduced-order non-linear differential equations that describe the point mass translational dynamics of the aircraft (formulated in Engelbrecht's PhD dissertation [1]). In this research project this reduced-order model is augmented to include the aircraft's fast rotational dynamics abstracted by first-order and second order input responses and thrust dynamics modelled as a first-order input response. This section presents the original reduced-order model and then presents the augmented reduced-order model.

### Original Reduced-Order Model

In Engelbrecht's PhD dissertation, the aircraft dynamics were simplified to only the point mass translational dynamics, and the fast rotational dynamics were neglected, to make the problem tractable to be solved with a dynamic programming approach. This reduced-order model is formulated as follows:

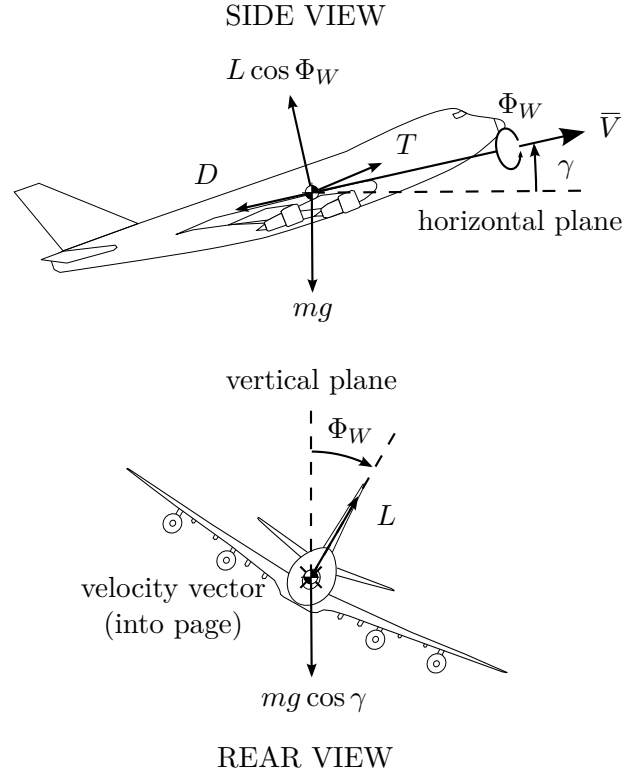


Figure 4.2: Point mass translational dynamics (Adapted from [1])

Figure 4.2 shows a visual representation of the point mass translational dynamics and the forces that dictate it, where  $L$  and  $D$  are the aerodynamic lift and drag forces respectively which are functions of the aerodynamic coefficients, and  $T$  is the engine thrust force. From Figure 4.2 the reduced-order dynamics can be derived,

$$\dot{\bar{V}} = \frac{1}{m} [T - D - mg \sin(\gamma)] \quad (4.3.1)$$

$$\dot{\gamma} = \frac{1}{\bar{V}m} [L \cos(\Phi_W) - mg \cos(\gamma)] \quad (4.3.2)$$

$$\dot{\Phi}_W = P_W \quad (4.3.3)$$

where  $\bar{V}$  is the magnitude of the velocity vector in the wind-axis direction,  $\gamma$  is the flight path angle (the angle between the aircraft velocity vector and the horizontal plane),  $\alpha$  is the angle of attack,  $T$  is the thrust force provided by the engines,  $\Phi_W$  is the wind-axis bank angle and  $P_W$  is the wind-axis roll rate. Note that for this model it is assumed that the thrust force vector acts in a direction parallel to the velocity vector of the aircraft, where normally the thrust vector acts in a direction that is closer to being parallel to direction of the nose of the aircraft. This



assumption is made because we assume that the aircraft is within its aerodynamic envelope, meaning at relatively low angles of attack where the nose is close to the velocity vector.

The aerodynamic lift and drag forces can be modelled with the following simplified equations,

$$L = \frac{1}{2} \rho \bar{V}^2 S C_{L,Static}(\alpha, \beta) \quad (4.3.4)$$

$$D = \frac{1}{2} \rho \bar{V}^2 S C_{D,Static}(\alpha, \beta) \quad (4.3.5)$$

where  $S$  is the wing surface area, and  $C_{L,Static}$  and  $C_{D,Static}$  are the dimensionless static aerodynamic lift and drag coefficients that are a function of the angle of attack (AoA)  $\alpha$  and sideslip  $\beta$ . The lift and drag coefficient functions are obtained from the GTM model in the form of lookup tables and only the static coefficient terms are used from these tables, hence the coefficient functions only being a function of  $\alpha$  and  $\beta$  to simplify the problem, arguing that the effect of the incremental aerodynamic terms constituted by the fast rotational states is significantly smaller than the contribution of the static aerodynamic terms constituted by the wind angles, i.e. angle of attack. Furthermore it is assumed that there are inner-loop flight controllers that can regulate the sideslip angle of the aircraft to zero, i.e.,  $\beta = 0^\circ$ . Thus the problem is only considered in the longitudinal axis. The GTM's aerodynamic coefficients are given in the body-axis system so the lift and drag coefficients are calculated using a longitudinal wind-axis rotation matrix,

$$\begin{bmatrix} C_{D,Static}(\alpha) \\ C_{L,Static}(\alpha) \end{bmatrix} = \begin{bmatrix} -\cos(\alpha) & -\sin(\alpha) \\ \sin(\alpha) & -\cos(\alpha) \end{bmatrix} \begin{bmatrix} C_{X,Static}(\alpha, \beta = 0) \\ C_{Z,Static}(\alpha, \beta = 0) \end{bmatrix} \quad (4.3.6)$$

The resulting static lift and drag curves of the GTM are shown in Figure 4.3.

Thus when only the translational dynamics are modelled, the state vector is,

$$\mathbf{x} = [\bar{V} \quad \gamma \quad \Phi_W]^\top \quad (4.3.7)$$

and the input vector for this reduced-order model is,

$$\mathbf{u} = [\alpha \quad T \quad P_W]^\top \quad (4.3.8)$$

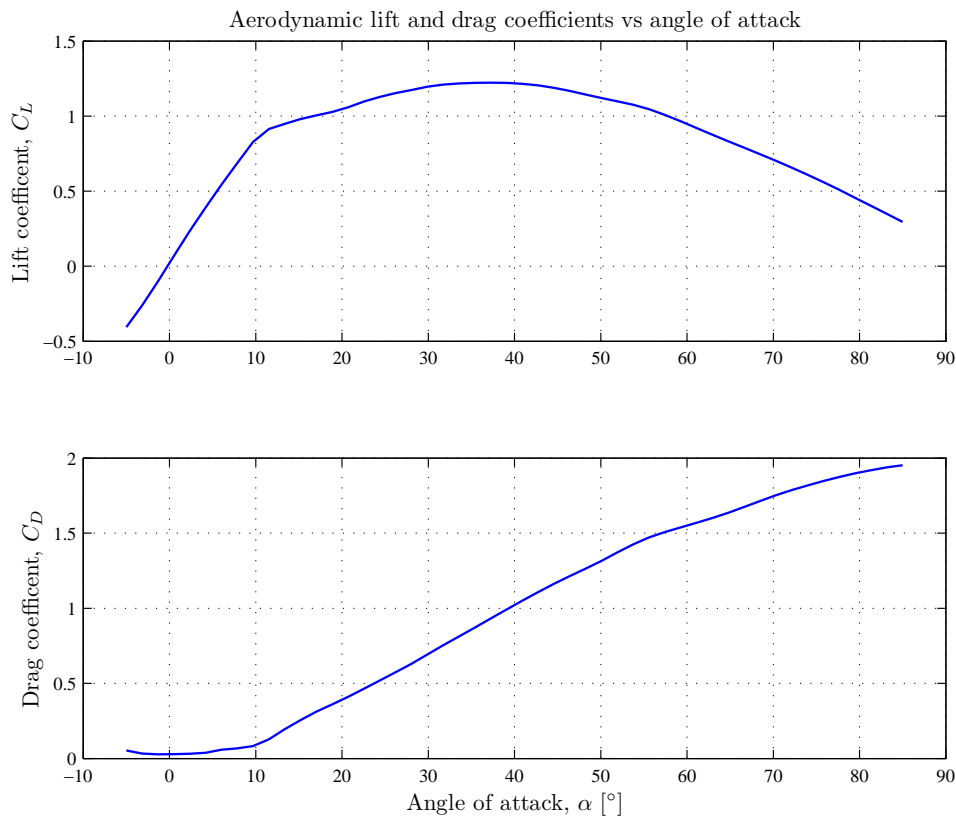


Figure 4.3: Aerodynamic lift and drag coefficient functions

### Augmented Reduced-Order Model

The reduced-order model used by Engelbrecht’s dynamic programming approach assumed that the angle of attack command and roll rate command given to the inner-loop controllers were followed immediately with no transient response. Engelbrecht’s dynamic programming approach compensated for the fact that the inner-loop controller dynamics were neglected by using a one second sampling period for updating the angle of attack and roll rate references. (The argument was that the time constants of the inner-loop controllers are sufficiently shorter than the one second sampling period, so that it would seem as if the references are tracked ‘immediately’ from the viewpoint of the dynamic programming algorithm.)

In this research project, we would like to take the inner-loop controller dynamics and the engine dynamics into account explicitly when we perform the trajectory planning. The SQP approach (presented later in section §4.5), which can accommodate higher-order models, gives us the ability to add these input lag dynamics. Thus in this research project, we augment the point mass translational dynamics model by adding the closed-loop responses of the angle of attack controller and the roll rate controller to the model, and also by adding the engine thrust response to the model.

The first-order input response models approximating the fast rotational dynamics of the aircraft consist of the angle of attack input response and the roll rate input response. These two response models are derived from the closed-loop transient response of the inner-loop flight controllers that were implemented on the GTM in the previous chapter. The longitudinal inner-loop controller controls the angle of attack or normal acceleration of the aircraft and has a fast well-damped closed-loop response, which can be modelled using either a first-order transfer function or a second-order transfer function. The lateral controller can control the roll rate

of the aircraft and the closed-loop response is close to the natural roll rate response of the aircraft, which is dominantly a first-order response. Thus the roll rate input dynamics is well approximated with a first-order model.

The augmented reduced-order model dynamics for the upset recovery problem is then formulated by adding the closed-loop models of the inner-loop controllers to the reduced-order model of Equations 4.3.1 to 4.3.3,

$$\dot{V} = \frac{1}{m} \left[ T - \frac{1}{2} \rho \bar{V}^2 S C_D(\alpha) - mg \sin(\gamma) \right] \quad (4.3.9)$$

$$\dot{\gamma} = \frac{1}{\bar{V}m} \left[ \frac{1}{2} \rho \bar{V}^2 S C_L(\alpha) \cos(\Phi_W) - mg \cos(\gamma) \right] \quad (4.3.10)$$

$$\dot{\Phi}_W = P_W \quad (4.3.11)$$

$$\dot{\alpha} = -\frac{1}{\tau_\alpha} \alpha + \frac{1}{\tau_\alpha} \alpha_c \quad (4.3.12)$$

$$\dot{P}_W = -\frac{1}{\tau_P} P_W + \frac{1}{\tau_P} P_{Wc} \quad (4.3.13)$$

$$\dot{T} = -\frac{1}{\tau} T + \frac{1}{\tau} T_c \quad (4.3.14)$$

where the shorthand notation  $C_{L,Static}(\alpha) = C_L(\alpha)$  and  $C_{D,Static}(\alpha) = C_D(\alpha)$  for the aerodynamic coefficients have been used. The term  $T_c$  is the thrust command,  $P_{Wc}$  is the wind-axis roll rate command and  $\alpha_c$  is the angle of attack command. As with the original reduced-order model, the thrust vector is modelled as acting parallel to the velocity vector of the aircraft, and not parallel to the nose direction

The angle of attack and roll rate input dynamics are abstracted by first-order response models of the inner-loop controllers, with a time constant of  $\tau_\alpha$  and  $\tau_P$  for the angle of attack response and roll rate response respectively. The longitudinal inner-loop controller explicitly controls normal acceleration, but implicitly controls angle of attack, as normal acceleration is a function of angle of attack. Thus it is safe to assume that the angle of attack input response is the same as the normal acceleration response. The time constant of the longitudinal inner-loop controller's response is roughly 0.2 seconds and the time constant of the roll rate controller's response is roughly 0.3 seconds.

A turbo fan jet engine can be adequately modelled with a first-order exponential response to simulate the lag between the thrust command and actual thrust output. The thrust lag time constant is denoted by  $\tau$ .

The angle of attack input dynamics could also be represented by a second-order model, effectively adding the angle of attack rate as a state,

$$\begin{bmatrix} \dot{\alpha} \\ \ddot{\alpha} \end{bmatrix} = \mathbf{A}_\alpha \begin{bmatrix} \alpha \\ \dot{\alpha} \end{bmatrix} + \mathbf{B}_\alpha \alpha_c \quad (4.3.15)$$

where the state space matrices  $\mathbf{A}_\alpha$  and  $\mathbf{B}_\alpha$  are obtained from the reduced-order system representing the closed-loop normal acceleration controller response. The closed-loop normal acceleration controller dynamics is essentially the closed-loop dynamics of the short-period mode of the aircraft, which also equivalently represents the angle of attack dynamics. Initially a first-order model was used for the angle of attack dynamics, but the executed trajectories differed

too much from the planned trajectories. It was then decided to use a second-order model instead, which gave better trajectory execution results. When using the first-order angle of attack dynamics, the system state vector is then,

$$\mathbf{x} = [\bar{V} \quad \gamma \quad \Phi_W \quad \alpha \quad P_W \quad T]^\top \quad (4.3.16)$$

or if the second-order angle of attack dynamics are used the system state vector is,

$$\mathbf{x} = [\bar{V} \quad \gamma \quad \Phi_W \quad \alpha \quad \dot{\alpha} \quad P_W \quad T]^\top \quad (4.3.17)$$

and the input vector for both cases is,

$$\mathbf{u} = [\alpha_c \quad T_c \quad P_{Wc}]^\top \quad (4.3.18)$$

### 4.3.2 Assumptions and Requirements

In order to formulate the problem of flight trajectory optimisation, some requirements are set for the trajectory solutions so that they are valid recovery solutions for an upset condition. Furthermore some assumptions can be made to make the problem more tractable. For a trajectory to be a viable solution to an upset condition, the trajectory must fulfil certain criteria,

- The recovery trajectory must be so that the structural integrity envelope of the aircraft is not exceeded at any time during the manoeuvre. This can be enforced by setting limits on the load factor (or equivalently normal acceleration) and airspeed at any given time.
- The aircraft must be within what is considered a safe speed limit to be considered as successfully recovered.
- The aircraft must also recover to a wings level flight attitude to be considered as successfully recovered.

The assumptions that have been made in the problem are,

- Only the static aerodynamic coefficients are used in the dynamics of the system, as the effect of the incremental aerodynamic terms constituted by the fast rotational states is expected to be significantly smaller than the contribution of the static aerodynamic terms constituted by the wind angles.
- It is assumed that inner-loop controllers are in place to regulate the aircraft's faster dynamics.
- It is assumed that during the recovery trajectories that the sideslip angle of the aircraft remains zero by means of a sideslip controller.

### 4.3.3 Optimal Control Problem Formulation

A trajectory optimisation problem can be viewed as an optimal control problem. In an optimal control problem, one is interested in obtaining the optimal state path, or trajectory, of a system leading to a desired final state condition, as well as the optimal or minimum control effort sequence that drives the dynamics of a system along this path.

The problem of attitude and flight vector recovery is formulated as an optimal control problem. The general optimal control problem can be stated as follows as by Rao [25]. Determine the  $n_x$  states (equivalently, the trajectory or path),  $\mathbf{x}(t) \in \mathbb{R}^{n_x}$ , the  $n_u$  controls  $\mathbf{u}(t) \in \mathbb{R}^{n_u}$ , the initial time,  $t_0 \in \mathbb{R}$ , and the terminal time,  $t_f \in \mathbb{R}$  (where  $t \in [t_0, t_f]$  is the independent variable) that optimises the performance index,

$$J = h(\mathbf{x}(t_f), t_f) + \int_{t_0}^{t_f} g(\mathbf{x}(t), \mathbf{u}(t), t) dt \quad (4.3.19)$$

subject to the dynamic state equations (or dynamic constraints),

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t), t) \quad (4.3.20)$$

the initial state and input constraint,

$$\boldsymbol{\eta}_{0l} \leq \boldsymbol{\eta}_0(\mathbf{x}(t_0), \mathbf{u}(t_0), t_0) \leq \boldsymbol{\eta}_{0u} \quad (4.3.21)$$

the terminal state and input constraint,

$$\boldsymbol{\eta}_{fl} \leq \boldsymbol{\eta}_f(\mathbf{x}(t_f), \mathbf{u}(t_f), t_f) \leq \boldsymbol{\eta}_{fu} \quad (4.3.22)$$

and the state and input space constraints,

$$\mathbf{x}_l \leq \mathbf{x} \leq \mathbf{x}_u \quad (4.3.23)$$

$$\mathbf{u}_l \leq \mathbf{u} \leq \mathbf{u}_u \quad (4.3.24)$$

In this formulation,  $J$  is the performance index (also called the objective function),  $g()$  is the state transition cost function,  $h()$  is the terminal state cost function,  $\mathbf{f}()$  represents the system dynamics as a non-linear time-varying function,  $\boldsymbol{\eta}_0$  represents a general initial value function (that is dependant on the initial state, initial input and initial time values) and  $\boldsymbol{\eta}_f$  represents a general terminal value function (that is dependant on the final state, final input and final time values). The terms  $\boldsymbol{\eta}_{0l}$  and  $\boldsymbol{\eta}_{0u}$  represent the lower and upper constraint bound values respectively imposed on the initial value function,  $\boldsymbol{\eta}_{fl}$  and  $\boldsymbol{\eta}_{fu}$  represent the lower and upper constraint bound values respectively imposed on the final value function,  $\mathbf{x}_l$  and  $\mathbf{x}_u$  represent the lower and upper constraint bound values respectively imposed on the state values of the system, and  $\mathbf{u}_l$  and  $\mathbf{u}_u$  represent the lower and upper constraint bound values respectively imposed on the input values of the system.

The objective function is a performance measure of how optimal the solution to the problem is, and usually consists of a cost function that is minimised by the solution. The dynamic constraints represent the constraints imposed on the system to ensure the solution obeys the dynamics of the system. The initial state constraints ensure that the solution starts from a specific set of initial system conditions. The terminal state constraints ensure that the solution terminates in a set of final system conditions. The state and input space constraints ensures that the solution does not exceed a certain state value bound or input value bound for the entire trajectory.

The next subsections mathematically formulate the problem of attitude and flight vector recovery into an optimal control problem.

### 4.3.4 Cost Function

When recovering an aircraft from upset, the primary goal is to recover the aircraft with minimum altitude loss. For the flight trajectory optimisation problem the cost function of the formulated optimal control problem is then chosen to be,

$$\begin{aligned} J^h &= \int_{t_0}^{t_f} \max(-\dot{h}, 0) dt \\ &= \int_{t_0}^{t_f} \max(-\bar{V} \sin(\gamma), 0) dt \end{aligned} \quad (4.3.25)$$

Equation 4.3.25 represents the altitude loss of the aircraft which we are trying to minimise in a given time window, with  $t_0$  being the initial time and  $t_f$  being the final time of the trajectory. We do not gain any negative cost for gaining altitude. The system should not be allowed to reduce the overall cost of the recovery by gaining altitude in the time span of the trajectory, effectively only minimising the total integral of climb rate [1].

### 4.3.5 State Constraints

State space constraints are imposed on the system in the recovery problem and represent the physical limitations of the aircraft, as well as the acceptable state ranges the flight controllers are allowed to operate in. Bounds are placed on the airspeed to prevent the aircraft from exceeding the structural integrity envelope. An upper bound is placed on the flight path angle to prevent stall and a lower bound is placed on the flight path angle to prevent mathematical singularities in the aircraft dynamics. Bounds are placed on the bank angle so that the angle remains within the range of one rotation. The thrust state is physically limited to the maximum and minimum thrust that the aircraft's engines can produce. The angle of attack is limited so that it remains within the aerodynamic envelope and remains smaller than the stall angle of attack. Additionally, the angle of attack is limited so that the state remains relatively close to the trim condition of the linear flight controllers. The roll rate is limited since passenger aircraft are usually limited to lower roll rates due to safety factors.

The system's state space constraints are defined by the following set of admissible states,

$$\bar{V} \in [\bar{V}_l, \bar{V}_u] \quad (4.3.26)$$

$$\gamma \in [\gamma_l, \gamma_u] \quad (4.3.27)$$

$$T \in [T_l, T_u] \quad (4.3.28)$$

$$\Phi_W \in [\Phi_{Wl}, \Phi_{Wu}] \quad (4.3.29)$$

$$\alpha \in [\alpha_l, \alpha_u] \quad (4.3.30)$$

$$P_W \in [P_{Wl}, P_{Wu}] \quad (4.3.31)$$

The admissible airspeed range is larger than the accepted airspeed range to include underspeed and overspeed conditions. The minimum admissible airspeed  $\bar{V}_l$  is the lowest airspeed the aircraft is expected to recover from and may be below the stall speed. The maximum admissible airspeed  $\bar{V}_u$  is the maximum airspeed allowed by the structural integrity envelope. The terms  $\gamma_l$  and  $\gamma_u$  are the minimum and maximum allowed flight path angle values respectively,  $T_l$  and

$T_u$  are the minimum and maximum allowed thrust values respectively,  $\Phi_{Wl}$  and  $\Phi_{Wu}$  are the minimum and maximum allowed bank angle values respectively,  $\alpha_l$  and  $\alpha_u$  are the minimum and maximum allowed angle of attack values respectively, and  $P_{Wl}$  and  $P_{Wu}$  are the minimum and maximum allowed wind-axis roll rate values respectively.

### 4.3.6 Input Constraints

Input constraints are imposed on the system in the recovery problem that represent physical command limits or to prevent excessively large commands from being given to the flight controllers. The angle of attack command can be limited to prevent commands from being given that would lead to exceeding the aerodynamic envelope. The thrust command is limited due to the physical thrust output limit of the engine. The wind-axis roll rate command can be limited to prevent commands from being given that would lead to exceeding the roll rate constraints of the system. The system's input space constraints are defined by the following set of admissible input,

$$\alpha_c \in [\alpha_{c_l}, \alpha_{c_u}] \quad (4.3.32)$$

$$T_c \in [T_{c_l}, T_{c_u}] \quad (4.3.33)$$

$$P_{W_c} \in [P_{W_{cl}}, P_{W_{cu}}] \quad (4.3.34)$$

where  $\alpha_{c_l}$  and  $\alpha_{c_u}$  are the minimum and maximum limits on the angle of attack command respectively,  $T_{c_l}$  and  $T_{c_u}$  are the minimum and maximum limits on the thrust command respectively, and  $P_{W_{cl}}$  and  $P_{W_{cu}}$  are the minimum and maximum limits on the wind-axis roll rate command respectively.

A load factor limit is also imposed on the system in the recovery problem. The load factor can be thought of as the apparent force experienced by the passengers onboard as well as the force experienced by the plane's structure. The load factor should be kept within safe limits to ensure the aircraft does not compromise its structural integrity or cause physical injury to the passengers. For nominal wings level flight this force is equivalent to gravity alone and the load factor equals positive 1 g.

The limit on load factor can be seen as a state dependent input constraint, given by the non-linear relationship,

$$n_{L_{\min}} \leq \frac{\frac{1}{2}\rho\bar{V}^2 S C_L(\alpha)}{mg} \leq n_{L_{\max}} \quad (4.3.35)$$

The values  $n_{L_{\min}}$  and  $n_{L_{\max}}$  are the minimum and maximum normal load factor allowed during the performed recovery as defined by the structural integrity envelope.

Additionally a roll rate envelope is introduced in the form of a coupled input constraint that limits the maximum roll rate as a function of angle of attack,

$$P_W \leq P_{W_{\max}}(\alpha) \quad (4.3.36)$$

One rationale behind the envelope is to limit the physical wing loading force experienced by the wings when rolling. However, the reduced-order model only regards the point mass translational dynamics, so in this case the roll rate envelope helps limit the load factor experienced by the aircraft due to the induced angle of incidence produced by the wings when rolling.

### 4.3.7 Terminal State Constraints

The recovery problem requires that the aircraft must be safely recovered back to wings level flight which is defined as a state where the aircraft has a flight path angle of zero and a bank angle of zero. It is also required that the aircraft is within a safe airspeed limit as a requirement for successful recovery. A safe airspeed is an airspeed used in normal aircraft operation in cruising conditions, and is usually far from the stall airspeed.

The state termination conditions for wings level flight at a safe cruising airspeed are defined by the following set of admissible final states,

$$\bar{V}(t_f) \in [\bar{V}_{fl}, \bar{V}_{fu}] \quad (4.3.37)$$

$$\gamma(t_f) = \gamma_f \quad (4.3.38)$$

$$\Phi_W(t_f) = \Phi_{Wf} \quad (4.3.39)$$

where  $\bar{V}_{fl}$  and  $\bar{V}_{fu}$  are the minimum and maximum acceptable final airspeed allowed respectively,  $\gamma_f$  is the terminal flight path angle constraint, and  $\Phi_{Wf}$  is the terminal bank angle constraint.

With the optimal control problem formulated, it can now be solved using the two proposed numerical methods in the following sections.

## 4.4 Trajectory Optimisation using Dynamic Programming

Dynamic programming is a multi-stage decision process that uses the principle of optimality to form an optimal control law. Dynamic programming can be an attractive numerical method, since it results in a discrete closed-loop solution. Unlike most other trajectory optimisation methods, dynamic programming gives the optimal state and control sequences from all initial states. The solution is in the form of a lookup table, however the generation of the table is computationally heavy and must be done offline, and for large dimensional problems the computational cost and memory requirements become intractable. This issue is known as “the curse of dimensionality”. Fortunately, the solutions can be retrieved online by simply indexing the pre-computed table.

The following subsections provides the background on the dynamic programming approach and also gives the specific implementation of the method to the problem of flight trajectory optimisation. Finally, a subsection presents and discusses example recovery trajectories solutions generated by the dynamic programming method and also presents a comprehensive set of all recovery trajectories, along with a map showing all recoverable and unrecoverable states.

### 4.4.1 Reduced-Order Model

Due to dynamic programming’s limitations, i.e. the “the curse of dimensionality”, only the translational motion of the aircraft as a point mass that is under the influence of aerodynamic, engine, and gravitational forces, represented by Equations 4.3.9 to 4.3.11, is considered in this implementation. The faster rotational dynamics and thrust dynamics of Equations 4.3.12 to 4.3.14 are omitted under the rationale that they can be abstracted away through time scale



separation [1],

$$\dot{\bar{V}} = \frac{1}{m} \left[ T - \frac{1}{2} \rho \bar{V}^2 SC_D(\alpha) - mg \sin(\gamma) \right] \quad (4.4.1)$$

$$\dot{\gamma} = \frac{1}{\bar{V}m} \left[ \frac{1}{2} \rho \bar{V}^2 SC_L(\alpha) \cos(\Phi_W) - mg \cos(\gamma) \right] \quad (4.4.2)$$

$$\dot{\Phi}_W = P_W \quad (4.4.3)$$

#### 4.4.2 Hierarchical Multi-Objective Cost Function

One of the advantages of the multi-stage numerical approach of dynamic programming is that a hierarchical cost function can be used to strictly prioritise different cost functions. A novel hierarchical cost function as proposed by [1] is implemented, where the primary cost function is the total altitude loss of the trajectory,

$$J^h = \int_{t_0}^{t_f} \max(-\bar{V} \sin(\gamma), 0) dt \quad (4.4.4)$$

This cost function is defined so that only a negative climb rate increases the cost. This is so that the optimisation is not allowed to ‘make up’ for lost altitude by gaining positive altitude back. Only integrating negative climb rates makes the cost function represent the maximum altitude lost during the recovery. The second cost function is the maximum airspeed of the whole trajectory, defined by,

$$J^{\bar{V}} = \max(\bar{V}(t)), t \in [t_0 t_f] \quad (4.4.5)$$

The tertiary cost the time integral of the absolute value of the bank angle, represented by,

$$J^\Phi = \int_{t_0}^{t_f} \|\Phi_W(t)\| dt \quad (4.4.6)$$

This hierarchical cost function is minimised by first only minimising  $J^h$  without considering any of the other cost functions. This prioritizes the altitude loss cost above the secondary and tertiary costs. If no better minimum altitude cost can be achieved, the optimisation looks to minimise the secondary cost  $J^{\bar{V}}$  without considering the tertiary cost, and if no improvement can be made on the secondary cost, the tertiary cost function  $J^\Phi$  is minimised. This allows the optimisation to minimise the altitude loss cost function without compromise, which is considered the most important above all other costs.

This hierarchical multi-objective minimisation technique is an advantage to the dynamic programming method, as we do not have to resort to a weighted multi-objective cost function that includes all of the other costs functions to be minimised. No Pareto-optimal front is created as with a traditional weighted multi-objective function that inherently compromises certain cost functions with weights.

#### 4.4.3 Dynamic Programming Background

This section provides some background on the general dynamic programming approach to solving optimal control problems. The main source of the dynamic programming theory presented

is the textbook *Optimal Control Theory* by Kirk [28] and with some reference to Engelbrecht's work [1].

Dynamic programming is a numerical algorithm that models the optimal control problem as a multi-stage decision process and uses the principle of optimality to solve for the optimal state and input trajectories [1]. The principle of optimality states that whatever the initial state and initial decisions are, the remaining decisions (or subsection of decisions) from an intermediate state, that results from the initial optimal decisions, must also be an optimal trajectory from this intermediate state.

In order to solve the optimal control problem using dynamic programming, the problem must first be discretised in time, and the states and inputs quantised to create a finite set of decisions. The dynamic programming algorithm starts at the terminal states, i.e. the states at the final time instant, and works backwards in time through intermediate states until it finds a valid optimal path from the initial state to a terminal state. The algorithm first assigns a terminal cost to each terminal state and then successively steps back in time, one time step at a time. At each time instant, it iterates through all combinations of discrete states, and for each discrete state determines the admissible control decision that will transition the system from that state to a next state that has an optimal path to a terminal state (or is a terminal state) with minimum cost. The total cost is the sum of the transition cost from moving from the previous state to some next state and the optimal path cost to get to the terminal state from the next state. For each state combination at each time instant with a valid optimal path, the optimal cost to the terminal state is stored along with the control input to transition to the next state in time. The algorithm continues to step back in time until all states have an optimal path to the end, or the maximum time steps have been reached.

### Quantised State and Input Array

The continuous state space and control inputs is quantised into a finite set of admissible states and inputs bounded by the state and input space constraints of 4.3.23. We create arrays  $\mathbf{X}_q$  and  $\mathbf{U}_q$  of  $n$  quantised state values and  $m$  input values that span the admissible state and input range respectively,

$$\mathbf{X}_q = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_i, \dots, \mathbf{x}_n\} \quad (4.4.7)$$

$$\mathbf{X}_q \in \mathbf{X}_{bound} \quad (4.4.8)$$

$$\mathbf{U}_q = \{\mathbf{u}_1, \mathbf{u}_2, \dots, \mathbf{u}_i, \dots, \mathbf{u}_m\} \quad (4.4.9)$$

$$\mathbf{U}_q \in \mathbf{U}_{bound} \quad (4.4.10)$$

where  $\mathbf{X}_{bound}$  is the set of admissible states that include all the state values within the corresponding upper and lower state bounds of  $\mathbf{x}_u$  and  $\mathbf{x}_l$ , and  $\mathbf{U}_{bound}$  the set of admissible inputs that include all the input values within the corresponding upper and lower input bounds of  $\mathbf{u}_u$  and  $\mathbf{u}_l$ .

### Discrete-Time Dynamic Model

The continuous-time differential equations of the system dynamics 4.3.20 are discretised into discrete-time difference equations,

$$\mathbf{x}(k+1) = \mathbf{x}(k) + \mathbf{f}(\mathbf{x}(k), \mathbf{u}(k))\Delta t \quad (4.4.11)$$

where  $\Delta t$  is the chosen, fixed sampling period of the discrete time step.

### Functional Equation

The continuous-time cost function is discretised and becomes what is known as the functional equation of dynamic programming and represents dynamic programming's application of the principle of optimality mathematically [30]. We can express the total path cost  $J_{ij}$  from the current state  $\mathbf{x}_i(k)$  via the next state  $\mathbf{x}_j(k+1)$ , as the sum of the incremental cost  $\Delta J_{ij}$  of transitioning from the current state to the next state when applying an input  $\mathbf{u}_{ij}(k)$ , and the total optimal path cost of the next state  $J_j^*$  resulting from moving from that next state to a terminal state,

$$J_{ij}(\mathbf{x}_i(k), \mathbf{u}_{ij}(k)) = \Delta J_{ij}(\mathbf{x}_i(k), \mathbf{u}_{ij}(k)) + J_j^*(\mathbf{x}_j(k+1)) \quad (4.4.12)$$

where the incremental cost function  $\Delta J_{ij}$  is obtained by discretising the state transition cost function  $g(\mathbf{x}(t), \mathbf{u}(t), t)$  as follows,

$$\Delta J_{ij}(\mathbf{x}_i(k), \mathbf{u}_{ij}(k+1)) \approx g(\mathbf{x}_i(k), \mathbf{u}_{ij}(k), k)\Delta t \quad (4.4.13)$$

The sought after optimal control input  $\mathbf{u}_{ij}^*(k)$ , when applied at state  $\mathbf{x}_i(k)$ , will lead to the minimum cost  $J_i^*$  using the functional equation,

$$J_i^*(\mathbf{x}_i(k), \mathbf{u}_{ij}^*(k)) = \min_{\mathbf{u}_{ij}(k), j=1\dots n} \left\{ \Delta J_{ij}(\mathbf{x}_i(k), \mathbf{u}_{ij}(k)) + J_j^*(\mathbf{x}_j(k+1)) \right\}, \mathbf{u}_{ij}(k) \in \mathbf{U}_q \quad (4.4.14)$$

where  $\mathbf{u}_{ij}(k)$  is an admissible control input that transitions the system from state  $\mathbf{x}_i(k)$  to some admissible next state  $\mathbf{x}_j(k+1)$ .

### Algorithm Execution

The dynamic programming algorithm needs a table to store the optimal costs and optimal inputs for each state at each time step. We create two data tables to store these solutions, one containing the optimal path costs  $\mathbf{J}^*$  and one to store the optimal inputs  $\mathbf{U}^*$  as follows,

$$\mathbf{J}_{n \times N}^* = \left\{ J_{i,k}^* \right\} \quad (4.4.15)$$

$$\mathbf{U}_{n \times N}^* = \left\{ \mathbf{u}_{i,k}^* \right\} \quad (4.4.16)$$

with  $i \in [1, n]$  and  $k \in [1, N]$  where  $n$  is the number of quantised states and  $N$  is the number of discrete time steps. The tables are initialised with infinite control and infinite cost and then all terminal states' cost is set to the terminal cost defined by the problem, so that the algorithm only considers paths leading to the terminal states,

$$J_{i,N}^* = h(\mathbf{x}_i), \mathbf{x}_i \in [\boldsymbol{\eta}_{fl}, \boldsymbol{\eta}_{fu}] = \mathbf{X}_{final} \quad (4.4.17)$$

The algorithm then starts by iterating backwards in time from the time step  $k = N - 1$  to the beginning time step 1. The algorithm evaluates all the possible state transitions from the current time step to the next time step, trying all the admissible control input values in  $\mathbf{U}_q$  and using the functional equation, before stepping back one more time step. This is done for all state values in  $\mathbf{X}_q$  and for each time step. Equation 4.4.11 is used to calculate the next candidate state transition for each input,

$$\mathbf{x}_j(k + 1) = \mathbf{x}_i(k) + \mathbf{f}(\mathbf{x}_i(k), \mathbf{u}_{ij}(k))\Delta t \quad (4.4.18)$$

The algorithm calculates the total cost of a new path candidate that leads to a terminal state via the next state by summing the incremental transition cost to move from the current state to the next state and the stored optimal cost to move from the next state to the terminal state,

$$J_{i,k}(\mathbf{x}_i(k), \mathbf{u}_i(k)) = \Delta J_{ij}(\mathbf{x}_i(k), \mathbf{u}_{ij}(k)) + J_{j,k+1}^*(\mathbf{x}_j(k + 1)) \quad (4.4.19)$$

With each transition the algorithm checks if the transition yields a new minimum cost and stores the optimal control input with the optimal for the current state if it does,

$$J_{i,k}^* \leftarrow J_{i,k} \quad (4.4.20)$$

$$\mathbf{u}_{i,k}^* \leftarrow \mathbf{u}_{ij} \quad (4.4.21)$$

If the calculated transition state value of  $\mathbf{x}_j(k + 1)$  does not fall on a quantised value in the vector  $\mathbf{X}_q$  then the stored optimal cost value of the next state must be interpolated. Figure 4.4 illustrates an iteration step at the state value  $\mathbf{x}_1(N - 2)$  where all the control inputs are tested and one of the inputs leads to a next state that has a value between  $\mathbf{x}_1$  and  $\mathbf{x}_2$ .

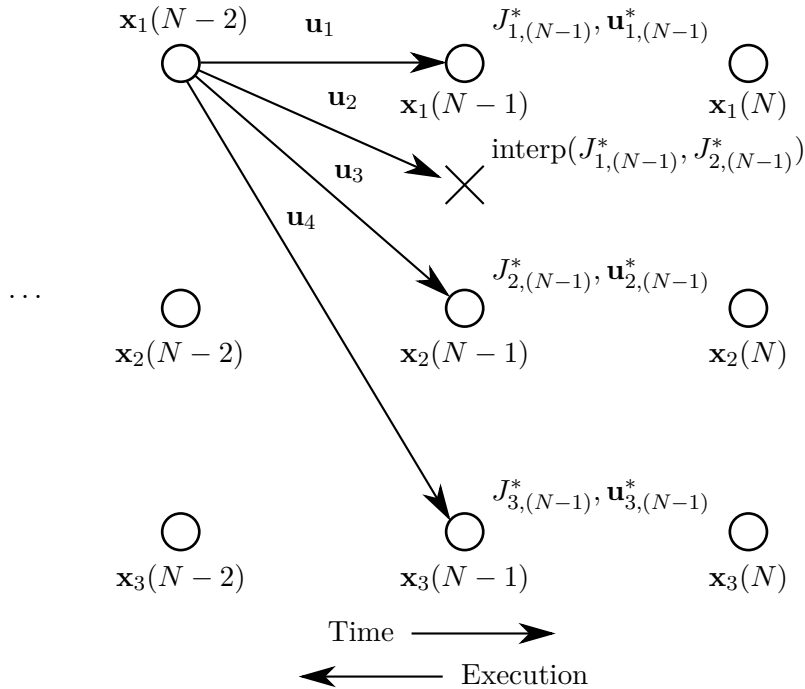


Figure 4.4: Dynamic programming path cost interpolation

A linear interpolation scheme can then be used to calculate the optimal cost from the stored costs  $J_{1,(N-1)}^*$  and  $J_{2,(N-1)}^*$ . When retrieving the optimal inputs from the structure  $\mathbf{U}^*$ , the stored control inputs will also have to be interpolated when using an input that leads to a non-quantised state value. In the case of Figure 4.4, if  $\mathbf{u}_2$  was the optimal input at  $\mathbf{x}_1(N-2)$ , then the next input would need to be interpolated from the values of  $\mathbf{u}_{1,(N-1)}^*$  and  $\mathbf{u}_{2,(N-1)}^*$ .

For higher order systems with multiple control and state dimensions, the method of dynamic programming runs into a severe drawback. Bellman calls this problem “the curse of dimensionality”. The number of calculations required to find the optimal control sequence grows as the state and control variable quantisation becomes finer and the system order increases. This growth becomes difficult to deal with for higher order systems. This problem arises because digital computers only have a finite amount of fast memory space, i.e. on-chip cache available to store the quantised state and control variables being used for the calculations [30].

#### 4.4.4 Dynamic Programming Implimentation

This section describes the specific application of the dynamic programming method to the problem of attitude and flight vector recovery as originally proposed by Engelbrecht [1].

##### Design Decisions

The following key design decisions were made by Engelbrecht [1]:

For the dynamic programming implementation, the thrust dynamics and fast rotational dynamics (such as the angle of attack  $\alpha$  and roll rate  $P_W$ ) of the problem are omitted to keep the problem tractable for dynamic programming and avoid “the curse of dimensionality”. The thrust dynamics are omitted with the rationale that due to the low bandwidth of the engines, the thrust will have little effect on the translational dynamics over the relatively short duration of the recovery.

Furthermore the dynamics are discretised with a sampling period of  $\Delta t = 1$  second. The reasoning is that the translational dynamics are an order of magnitude slower than that of the inner-loop dynamics of the controllers and thus we are assuming time scale separation. The inner-loop controllers have a time constants in the order of 0.3 seconds, which is sufficiently smaller than the chosen sampling time of 1 second. Thus when assuming time scale separation, we assume that the inputs can change instantaneously from the perspective of the translational dynamics.

##### Discrete-Time Dynamic Model

The point mass translational dynamics of Equations 4.3.1 to 4.3.3 is discretised into discrete-time difference equations using Equation 4.4.11,

$$\bar{V}(k+1) = \bar{V}(k) + \frac{1}{m} \left[ T(k) - \frac{1}{2} \rho \bar{V}(k)^2 SC_D(\alpha(k)) - mg \sin(\gamma(k)) \right] \Delta t \quad (4.4.22)$$

$$\gamma(k+1) = \gamma(k) + \frac{1}{\bar{V}(k)m} \left[ \frac{1}{2} \rho \bar{V}(k)^2 SC_L(\alpha(k)) \cos(\Phi(k)) - mg \cos(\gamma(k)) \right] \Delta t \quad (4.4.23)$$

$$\Phi(k+1) = \Phi(k) + P_W(k) \Delta t \quad (4.4.24)$$

If the thrust state is included in the dynamics the following discrete-time difference equation is used which is derived from the zero-order hold z-transform for a first-order system,

$$T(k+1) = e^{-\Delta t/\tau}T(k) + (1 - e^{-\Delta t/\tau})T_c(k) \quad (4.4.25)$$

### Incremental State Transition Cost Function

The cost functions of Equations 4.4.4 to 4.4.6 are discretised and used in the incremental state transition cost function of the dynamic programming as follows,

$$\Delta J^h(\mathbf{x}_i(k), \mathbf{x}_j(k+1)) = \max(-\bar{V}_i(k) \sin(\gamma_i(k)), 0)\Delta t \quad (4.4.26)$$

$$J^{\bar{V}}(\mathbf{x}_i(k), \mathbf{x}_j(k+1)) = \max(\bar{V}_i(k), J^{\bar{V}*}(\mathbf{x}_j(k+1))) \quad (4.4.27)$$

$$\Delta J^{\Phi}(\mathbf{x}_i(k), \mathbf{x}_j(k+1)) = \|\Phi_{W_i}(k)\|\Delta t \quad (4.4.28)$$

where the airspeed transition cost function Equation 4.4.27 is unique in that it is the total cost of transitioning from the current state to the next state instead of the incremental cost due to the maximum function inherently updating the total cost. Note that the transitional costs are all only functions of the current state  $\mathbf{x}_i(k)$  and the next state  $\mathbf{x}_j(k+1)$ . This has important implications to algorithm optimisation as will be explained in the next section regarding control input calculations.

### Control Inputs

We do not have to quantise the control inputs for this specific problem, which is a significant advantage in terms of algorithm optimisation. Since the dynamic system has three states and three inputs, we can simultaneously solve the set of difference equations to calculate the exact input needed to transition the system from the current state  $\mathbf{x}_i(k)$  at time instant  $k$  to the next state  $\mathbf{x}_j$  at time instant  $k+1$ . We then check if the calculated control input falls within the admissible control bounds  $\mathbf{U}_{bound}$  for it to be a valid transition.

The key realisation with this method is that we are not forced to loop through all quantised control inputs in a vector  $\mathbf{U}_q$  and to calculate what the next state will be for each input. This, coupled with the incremental cost functions only being a function of the states, means that as we loop through all current states and next states, we can check if the transition yields a new minimum cost *before* calculating the control input needed for the state transition. In other words, we loop through all current states  $\mathbf{x}_i(k)$  and all next states  $\mathbf{x}_j(k+1)$  and skip all input calculations for transitions that do not yield a new minimum cost, instead of looping through all current states  $\mathbf{x}_i(k)$  and all inputs  $\mathbf{u}_i(k)$  while being forced to calculate all state transition results  $\mathbf{x}_j(k+1)$  before we are able to calculate the incremental transition cost.

The control input needed to transition from a current state  $\mathbf{x}_i(k)$  to a candidate next state  $\mathbf{x}_j(k+1)$  in one time step  $\Delta t$  is calculated as follows,

$$\alpha_i(k) = C_L^{-1} \left( \frac{m\bar{V}_i(k) \frac{\gamma_j(k+1) - \gamma_i(k)}{\Delta t} + mg \cos \gamma_i(k)}{\frac{1}{2}\rho\bar{V}_i^2(k)S \cos \Phi_{W_i}(k)} \right) \quad (4.4.29)$$

$$T_i(k) = \frac{m [\bar{V}_j(k+1) - \bar{V}_i(k)]}{\Delta t} + \frac{1}{2}\rho\bar{V}_i^2(k)SC_D(\alpha_i(k)) + mg \sin \gamma_i(k) \quad (4.4.30)$$

$$P_{W_i}(k) = \frac{\Phi_{W_j}(k+1) - \Phi_{W_i}(k)}{\Delta t} \quad (4.4.31)$$

where it can be noted that the first equation contains a singularity at an airspeed of zero or a bank angle of 90 degrees. The airspeed singularity exists because the aircraft cannot produce lift without any airspeed. The bank angle singularity exists due to the aircraft not being able to produce any lift in the vertical plane with a lift vector at  $\pm 90$  degrees and thus cannot affect the flight path angle. The zero airspeed singularity is avoided by excluding it from the admissible state range. The 90 degree bank angle state however must be accommodated as we would like to consider this state as well as those greater than 90 degrees for inverted upset recovery conditions. We therefore use an alternative set of equations to determine a valid state transition from states at, and near  $\pm 90$  degrees. Because the flight path angle is essentially uncontrolled near this state, we first calculate the flight path angle transition due to gravity alone using,

$$\gamma_j(k+1) = \gamma_i(k) + \frac{-mg \cos(\gamma(k))}{\bar{V}(k)m} \Delta t \quad (4.4.32)$$

and check if this calculated value is close enough to the flight path angle state to which we are transitioning. If the transition is valid, we then choose a trim thrust value and trim angle of attack input to propagate the airspeed transition,

$$T_i(k) = T_{trim} \quad (4.4.33)$$

$$\alpha_i(k) = \alpha_{trim} \quad (4.4.34)$$

$$\bar{V}_j(k+1) = \bar{V}_i(k) + \frac{1}{m} \left[ T_i(k) - \frac{1}{2}\rho\bar{V}_i(k)^2 SC_D(\alpha_i(k)) - mg \sin(\gamma_i(k)) \right] \Delta t \quad (4.4.35)$$

and check if this calculated airspeed value is close enough to the airspeed state to which we are transitioning. If the transition is valid, we then finally calculate the needed roll rate input needed for the transition and check if it is an admissible input in the same manner as the non 90 degree case,

$$P_{W_i}(k) = \frac{\Phi_{W_j}(k+1) - \Phi_{W_i}(k)}{\Delta t} \quad (4.4.36)$$

The thrust response of the GTM aircraft is much slower than the angle of attack and roll rate responses, and we therefore cannot necessarily assume time scale separation between the thrust input response and the translational dynamics using a sampling time of  $\Delta t = 1$  second. Instead of calculating the thrust input command, the thrust can be kept constant during the recovery. If constant thrust is used in the problem, then the transition inputs are calculated using an alternative set of equations. In this case, Equations 4.4.29 to 4.4.31 are instead replaced by

the following alternative equations,

$$T_i(k) = T_{const} \quad (4.4.37)$$

$$\alpha_i(k) = C_L^{-1} \left( \frac{m\bar{V}_i(k) \frac{\gamma_j(k+1) - \gamma_i(k)}{\Delta t} + mg \cos \gamma_i(k)}{\frac{1}{2}\rho\bar{V}_i^2(k)S \cos \Phi_{W_i}(k)} \right) \quad (4.4.38)$$

$$\bar{V}_j(k+1) = \bar{V}_i(k) + \frac{1}{m} \left[ T_i(k) - \frac{1}{2}\rho\bar{V}_i(k)^2 S C_D(\alpha_i(k)) - mg \sin(\gamma_i(k)) \right] \Delta t \quad (4.4.39)$$

$$P_{W_i}(k) = \frac{\Phi_{W_j}(k+1) - \Phi_{W_i}(k)}{\Delta t} \quad (4.4.40)$$

where  $T_{const}$  is the chosen constant thrust value of the problem. For the transition to be valid, we check if the calculated airspeed value in the above equations is close enough to the airspeed state to which we are transitioning. The 90 degree bank angle case in the constant thrust problem is accommodated using the same equations as discussed previously (Equations 4.4.32 to 4.4.36).

### Algorithm Execution

We now give the execution process of the dynamic programming routine. We first quantise our states by creating quantised state vectors  $\bar{\mathbf{V}}_q$  for the airspeed,  $\mathbf{\Gamma}_q$  for the flight path angle and  $\mathbf{\Phi}_{W_q}$  for the bank angle, with the chosen quantisation intervals  $\Delta\bar{V}$ ,  $\Delta\gamma$  and  $\Delta\phi_W$  for the airspeed, flight path angle and bank angle respectively. We then construct the data tables for the optimal cost  $\mathbf{J}_{n \times N}^*$  and optimal control inputs  $\mathbf{U}_{n \times N}^*$ . An additional state transition table  $\mathbf{j}^*$  is used in this implementation. With this approach, we know exactly which next state to transition to, given the current state, while moving forward in time. We can directly retrieve the state transition and optimal inputs for each state without using any of the system's dynamic equations [30], using the state's index value  $j_{i,k}$  in the data tables,

$$\mathbf{j}_{n \times N}^* = \{j_{i,k}^*\} \quad (4.4.41)$$

Thus the state transition table stores the index pointing to the optimal next state to transition to in the quantised state vector  $\mathbf{X}_q$ . Before the main execution loop, the algorithm is initialised as follows,

---

#### Algorithm 1 Dynamic Programming Initialisation

---

- 1:  $\triangleright$  Store Cartesian product of all admissible states, which gives a vector of all state combinations
  - 2:  $\mathbf{X}_q \leftarrow \bar{\mathbf{V}}_q \times \mathbf{\Gamma}_q \times \mathbf{\Phi}_{W_q}$
  - 3:  $\mathbf{J}^* \leftarrow \infty$   $\triangleright$  Initialise entire optimal cost table with infinite values
  - 4: **for**  $i \leftarrow 1, \text{LENGTH}(\mathbf{X}_q)$  **do**
  - 5:   **if**  $\mathbf{x}_i \in \mathbf{X}_{final}$  **then**
  - 6:      $J_i^* \leftarrow 0$   $\triangleright$  Assign terminal cost of 0 to terminal states for all time
  - 7:      $j_i^* \leftarrow i$   $\triangleright$  Optimal next state index of all terminal states is state itself
  - 8:      $\mathbf{u}_i^* \leftarrow \mathbf{u}_{trim}$   $\triangleright$  Optimal input of terminal states is trim input
  - 9:   **end if**
  - 10: **end for**
-



The length of the state combination vector  $\mathbf{X}_q$  is the product of the lengths of the three quantised state vectors  $\bar{\mathbf{V}}_q$ ,  $\mathbf{\Gamma}_q$  and  $\Phi_{\mathbf{W}_q}$ . The optimal inputs for terminal states is calculated to be the input to trim the aircraft at that terminal state. The main algorithm loop then executes as follows,

---

**Algorithm 2** Dynamic Programming Execution
 

---

```

1:  $k \leftarrow N$  ▷ Start at final time instant
2: while  $k > 0$  do
3:    $k \leftarrow k - 1$  ▷ Step back one time instant
4:   for  $i \leftarrow 1, \text{LENGTH}(\mathbf{X}_q)$  do ▷ Iterate through all current states  $\mathbf{x}_i(k)$ 
5:     for  $j \leftarrow 1, \text{LENGTH}(\mathbf{X}_q)$  do ▷ Iterate through all next possible states  $\mathbf{x}_j(k+1)$ 
6:       ▷ calculate hierarchical costs to move from  $\mathbf{x}_i(k)$  to final state through  $\mathbf{x}_j(k+1)$ 
7:        $J_i^h \leftarrow \Delta J_{ij}^h + J_{j,k}^{h*}$ 
8:        $J_i^{\bar{V}} \leftarrow J_i^{\bar{V}}(J_{j,k}^{\bar{V}*})$ 
9:        $J_i^{\Phi} \leftarrow \Delta J_{ij}^{\Phi} + J_{j,k}^{\Phi*}$ 
10:      ▷ Does transition give new minimum cost? (Minimise hierarchical cost function)
11:      if  $(J_i^h < J_i^{h*})$  or  $(J_i^h = J_i^{h*}$  and  $J_i^{\bar{V}} < J_i^{\bar{V}*}) \dots$ 
12:      or  $(J_i^h = J_i^{h*}$  and  $J_i^{\bar{V}} = J_i^{\bar{V}*}$  and  $J_i^{\Phi} < J_i^{\Phi*})$  then
13:        ▷ Calculate input needed to transition from state  $\mathbf{x}_i(k)$  to  $\mathbf{x}_j(k+1)$  using
14:        Equations 4.4.29 to 4.4.36
15:         $\mathbf{u}_{ij} \leftarrow \text{CALCINPUT}(\mathbf{x}_i, \mathbf{x}_j)$ 
16:        ▷ is the calculated input an admissible input?
17:        if  $\mathbf{u}_{ij} \in \mathbf{U}_{\text{bound}}$  and  $n_L(k) \in [n_{L_{\text{max}}}, n_{L_{\text{min}}}]$  then
18:          ▷ store new optimal optimal path costs
19:           $J_{i,1 \rightarrow k}^{h*} \leftarrow J_i^h$ 
20:           $J_{i,1 \rightarrow k}^{\bar{V}*} \leftarrow J_i^{\bar{V}}$ 
21:           $J_{i,1 \rightarrow k}^{\Phi*} \leftarrow J_i^{\Phi}$ 
22:           $j_{i,1 \rightarrow k}^* \leftarrow j$  ▷ store new optimal state index to transition to
23:           $\mathbf{u}_{i,1 \rightarrow k}^* \leftarrow \mathbf{u}_{ij}$  ▷ store new optimal transition input
24:        end if
25:      end if
26:    end for
27:  end for
28: end while

```

---

### Search Optimisations

Similar optimisations were made to the dynamic programming algorithm as proposed by Engelbrecht [1].

1. A check was added so that the algorithm only considers transitions to states with finite cost. If the state has a non finite cost associated with it, it means that that state has no solution to a terminal state and it would not make sense to waste computational time calculating a transition to this next state.
2. When the optimal cost, index and input is stored, it is stored for the first time step all the way up to the current time step  $1 \rightarrow k$ . So when the algorithm steps back one time step it does not calculate suboptimal transitions it already calculated in the previous iteration step.

3. A check was added so that the algorithm only considers next states which optimal costs has been updated in the previous iteration step. If the optimal cost of the next state is the same as it was in a previous time step then the algorithm does not waste computation calculating a transition to this next state again, as it will yield the same result as in a previous time step. This optimisation works along with the previous listed optimisation. If this check is added without the previous listed optimisation the algorithm would never consider an optimal transition that stayed the same from a previous iteration step.
4. The bank angle is only quantised from zero to positive 180 degrees, which is half of the full bank angle state space. This can be done because of the symmetry in the bank angle recovery. A negative bank angle produces the same dynamics as its positive bank angle counterpart.

### Lookup Table Navigation

The dynamic programming algorithm produces lookup tables that store the optimal state trajectories and the optimal input sequences from all initial admissible states. Given an initial state  $\mathbf{x}_i(1)$  that has a finite cost associated with it, i.e. can be recovered from, the optimal state trajectories  $\mathbf{x}^*(k)$  and optimal input sequence  $\mathbf{u}^*(k)$  can be retrieved from the data tables  $\mathbf{j}^*$  and  $\mathbf{U}^*$ . The optimal state path is obtained by using the state indexes  $j^*$  from the state index table.

---

#### Algorithm 3 Dynamic Programming Solution Retrieval

---

```

1:  $k \leftarrow 1$ 
2:  $i \leftarrow i_0$  ▷ Start at index of initial state  $\mathbf{x}_i(1)$ 
3: while  $k \leq N$  do
4:    $\mathbf{x}^*(k) \leftarrow \mathbf{x}_i$  ▷ Value from  $\mathbf{X}_q$ 
5:    $\mathbf{u}^*(k) \leftarrow \mathbf{u}_{i,k}^*$  ▷ Value from  $\mathbf{U}^*$ 
6:    $i \leftarrow j_{i,k}^*$ 
7:    $k \leftarrow k + 1$ 
8: end while

```

---

For states that do not fall on quantised states, either nearest neighbour interpolation can be used for the initial state, or multilinear interpolation can be used in a similar manner as discussed in §4.4.3.

### 4.4.5 Dynamic Programming Results

This section provides some illustrative and exhaustive results of the numerical optimal trajectory solutions from the implemented dynamic programming algorithm. The illustrative results consist of a few example trajectory solutions, while the exhaustive results consist of a comprehensive set of all the trajectory solutions produced by the dynamic programming algorithm. Aspects of the results are discussed and a short summary then is given.

#### Problem Setup

The dynamic programming algorithm's problem variables were set to the values detailed in Table 4.1. The maximum  $\bar{V}$  is arbitrarily chosen to be above the normal trim speed but below the structural integrity envelope. The minimum  $\bar{V}$  is chosen to include the stall region of the

aircraft. The final time instant  $\bar{V}$  bounds are chosen to be within safe cruising speed limits where no extreme inputs are needed to trim the aircraft for these speeds.

The maximum angle of attack is chosen with the motivation that the inner loop angle of attack controller cannot follow angle of attack commands above 21 degrees. The limit on roll rate is artificially chosen such that passenger aircraft are not capable of excessive roll rates due to safety factors. In addition, an artificial roll rate envelope is added to the problem to make the constraints more interesting from [1], illustrated in Figure 4.5.

Note that the bank angle is limited to positive angles, while the roll rate is limited to negative rates. This is because the bank angle solution can be mirrored for negative bank angles and it reduces the search space. Another reason for these limits is to reduce numerical 'oscillation' around  $0^\circ$  as there should be no incentive to bank away from level flight.

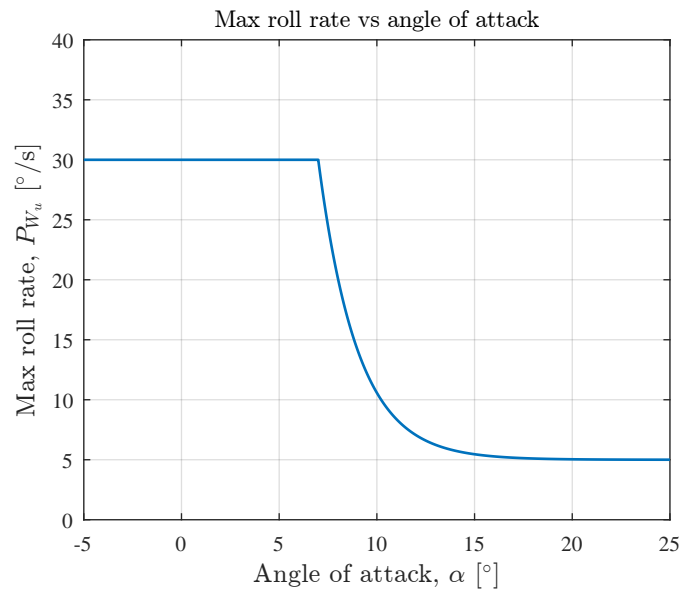


Figure 4.5: Max allowed roll rate as a function of angle of attack

Table 4.1: Problem parameter values

Parameter	Symbol	Value	Units
<i>Physical constants</i>			
Mass	$m$	23.59	kg
Gravitational constant	$g$	9.81	m/s <sup>2</sup>
Wing surface area	$S$	0.548	m <sup>2</sup>
Dynamic pressure	$\rho$	1.225	kg/m <sup>3</sup>
Aerodynamic lift Coefficient	$C_L$	Figure 4.3	-
Aerodynamic drag Coefficient	$C_D$	Figure 4.3	-
<i>State bounds</i>			
Velocity	$\bar{V}_u, \bar{V}_l$	140, 20	kn
Final velocity	$\bar{V}_{fu}, \bar{V}_{fl}$	120, 75	kn
Flight path angle	$\gamma_u, \gamma_l$	30, -80	degrees
Bank angle	$\Phi_u, \Phi_l$	180, 0	degrees
Final flight path angle	$\gamma_f$	0	degrees
Final bank angle	$\Phi_f$	0	degrees
<i>Input bounds</i>			
Angle of attack	$\alpha_u, \alpha_l$	21, 0	degrees
Roll rate	$P_{W_u}, P_{W_l}$	0, -30	degrees/second
Thrust	$T_u, T_l$	136.25, 0	Newton
Load factor	$n_{L_{\max}}, n_{L_{\min}}$	2.5, -1	g
Roll envelope	$P_{W_u}(\alpha)$	Figure 4.5	degrees/second
<i>Time</i>			
Number of time steps	$N$	14	-
Number of grid points	$N + 1$	15	-
Sample time	$\Delta t$	1	seconds
<i>DP quantisation resolution</i>			
Velocity	$\Delta \bar{V}_q$	5	kn
Flight path angle	$\Delta \gamma_q$	5	degrees
Bank angle	$\Delta \Phi_q$	7.5	degrees

### Illustrative Trajectory Results

This section discusses three recovery trajectory solutions for different upset conditions that were chosen to illustrate the results obtained from the dynamic programming algorithm. The optimal state trajectories and optimal input sequences are shown, along with the time history of the change in altitude during the recovery. Figure 4.6 shows the three recovery trajectories for three initial upset conditions. The thrust is kept constant during the recoveries and only changes to a trim thrust input once the aircraft is recovered, i.e. reached a terminal state. This is motivated by the fact that the GTM's thrust input response is much slower than the angle of attack and roll rate inputs, as discussed previously in the implementation of the dynamic programming algorithm in section §4.4.4. Interestingly, the recovery trajectory solutions all make intuitive sense from the perspective of what trained pilot would do in the upset conditions with reference to the pilot training aid [9].

**Case 1 - Overspeed** - Case 1 is an overspeed upset condition, starting at level flight  $\gamma_0 = 0^\circ$ ,  $\Phi_{W_0} = 0^\circ$  and at an airspeed exceeding the allowed safe final airspeed  $\bar{V}_0 = 140$  kn.

For the overspeed condition the algorithm gives a positive angle of attack command to raise the flight path angle above zero degrees to 'bleed off' airspeed using gravity. When the airspeed has been recovered to within the safe limit, a lower angle of attack command is given at four seconds to return the flight path angle back to level flight.

**Case 2 - Inverted bank angle** - Case 2 is an inverted bank angle upset where the aircraft is almost entirely upside down initially  $\Phi_{W_0} = 160^\circ$ , starting at a level flight path angle  $\gamma_0 = 0^\circ$  and an airspeed of  $\bar{V}_0 = 70$  kn.

For the inverted bank angle upset, the algorithm uses a large initial roll rate to recover the bank angle and bring the aircraft to a more 'upright' attitude before recovering the loss in flight path angle. The flight path angle 'dips' during the bank angle recovery, because while the aircraft is inverted the lift vector is pointed downwards and cannot help the flight path angle recover. When the bank angle is lower than 90 degrees the algorithm 'pulls up' to recover the flight path angle, using the maximum angle of attack allowed by the roll rate envelope and load factor constraint, since the lift vector is now pointed upwards.

**Case 3 - Flight path angle with bank angle upset** - Case 3 is a flight path angle upset where the initial flight path angle is at a severe negative angle  $\gamma_0 = -50^\circ$ . In addition, the aircraft is initially banked at a high bank angle below 90 degrees  $\Phi_{W_0} = 50^\circ$  at a low airspeed  $\bar{V}_0 = 40$  kn.

In the case of the flight path angle upset, the bank angle is recovered within two seconds. At two seconds the algorithm gives the maximum angle of attack command possible to recover the flight path angle as quickly as possible, without exceeding the load factor constraint. It is likely that a high angle of attack is not given initially, because the aircraft cannot give a high angle of attack if it wants to recover the bank angle as quickly as possible due the roll rate envelope constraint. The aircraft simultaneously also recovers from its underspeed condition during the manoeuvre.

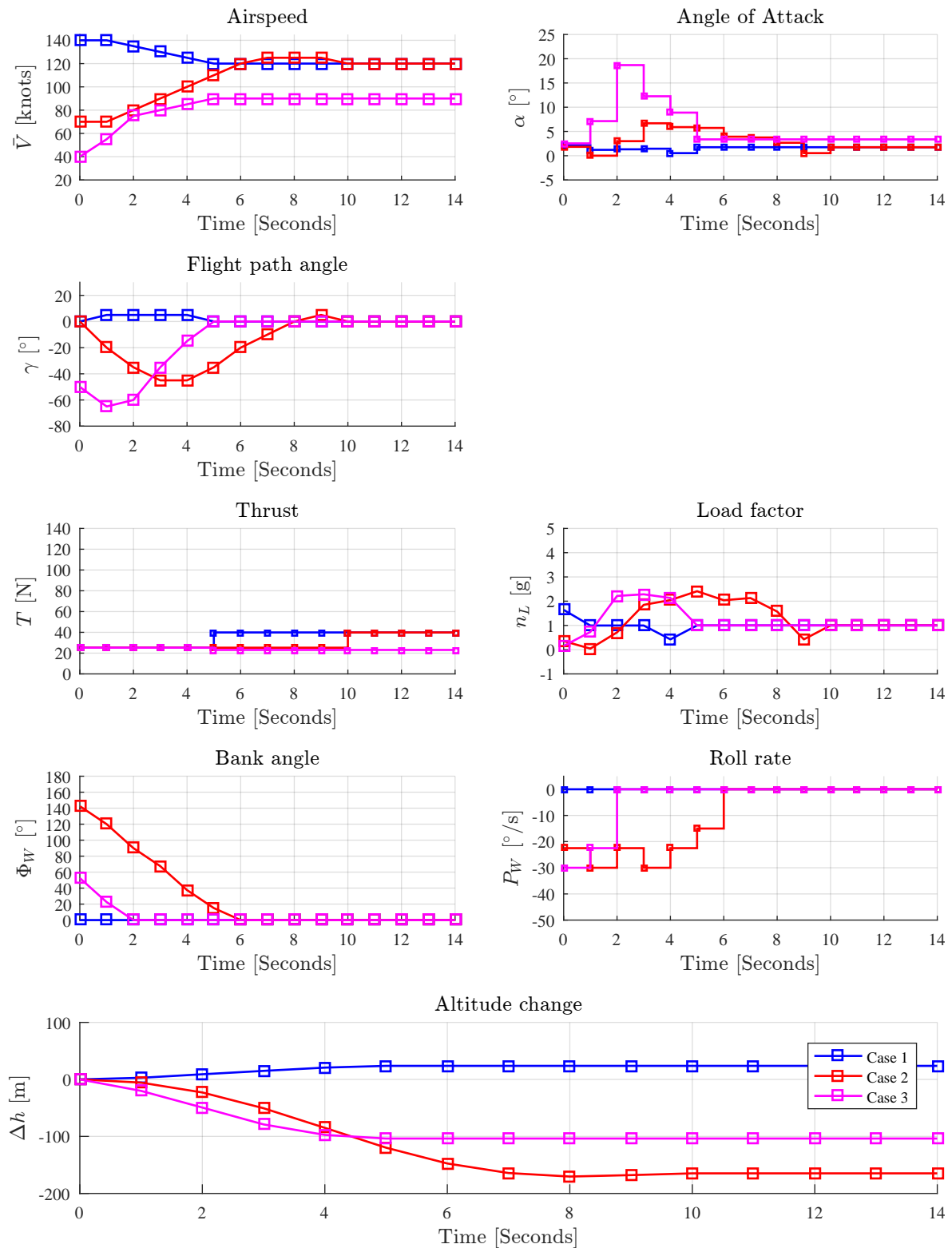


Figure 4.6: Illustrative dynamic programming recovery trajectories

### Exhaustive Trajectory Results

The dynamic programming solution allows us to retrieve the optimal recovery trajectories from *all* recoverable initial states. The comprehensive set of optimal state trajectories and control inputs for all recoverable initial states in the state grid can be retrieved. Figure 4.7 shows a plot of all the state recovery trajectories and Figure 4.8 shows a plot of all the optimal input sequences, including the load factor plot for all the trajectories. For all these recovery trajectories the thrust input is kept constant until the terminal states are reached before switching to a trim thrust command. It can be seen that all of the trajectories remain within the load factor constraints and thus remains in the structural integrity envelope. Figure 4.9 shows the altitude trajectories from all recoverable initial states.

Some recovery trajectories recover within one or two seconds, while others take up to the full 15 second time window to recover optimally. Most trajectories lose between 0 and 300 meters of altitude, while some do not lose any altitude at all and actually gain altitude. These are recoveries where the aircraft starts at a level flight path angle and recovers from overspeed and therefore raises the flight path angle to reduce speed. Thus with a positive flight path angle, altitude is gained. Most altitude trajectories show an initial ‘dip’ and then rises again. This is because the aircraft loses altitude while recovering its flight path angle and bank angle and then picks up excessive speed, which it then loses by using a positive flight path angle, regaining altitude. When the aircraft has finished the recovery, the altitude trajectory flattens to a constant, because the aircraft has returned to level flight and there is no more change in altitude.

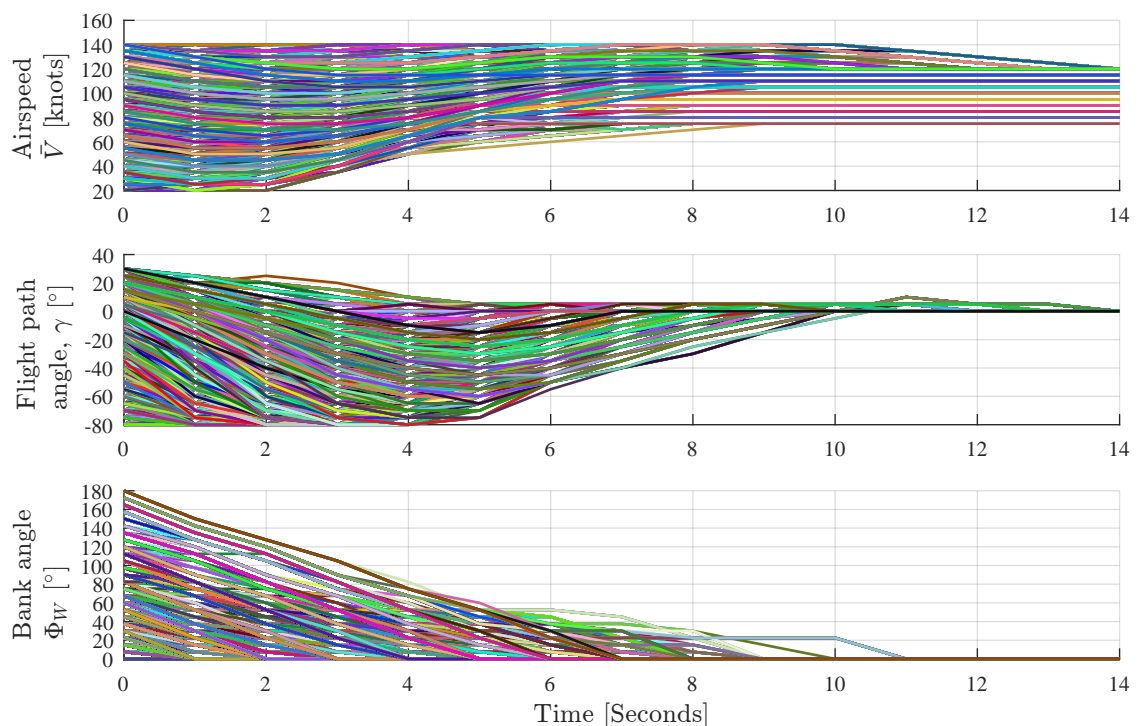


Figure 4.7: Plot of all state recovery trajectories

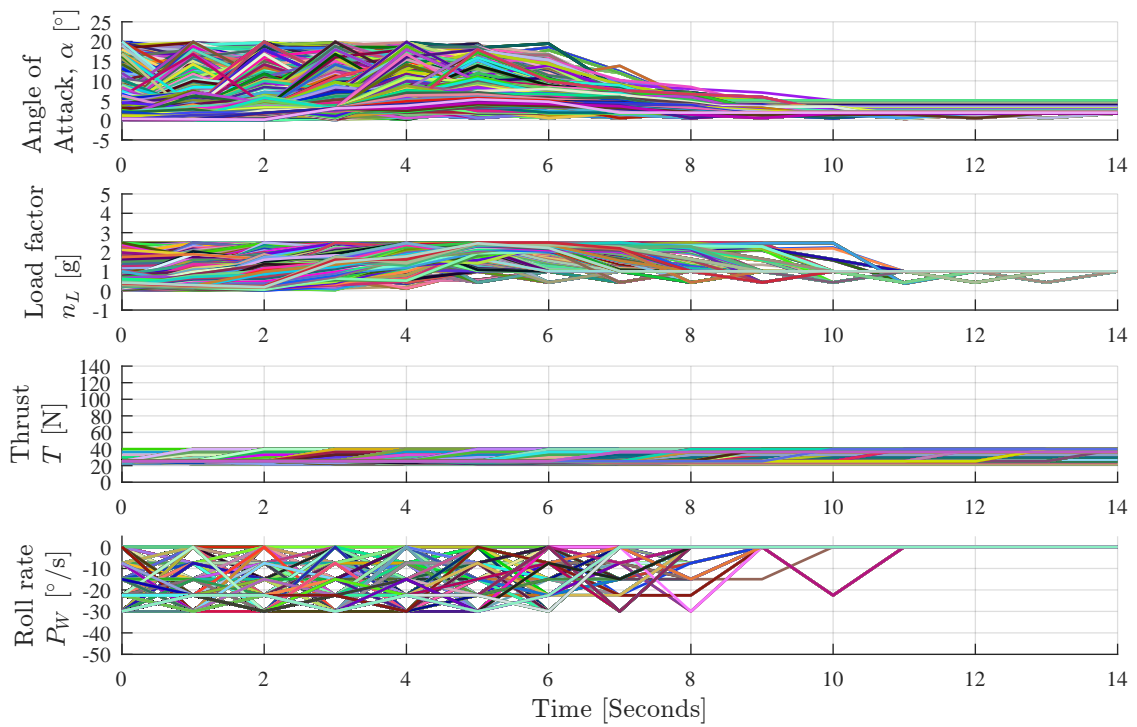


Figure 4.8: Plot of all control input recovery sequences

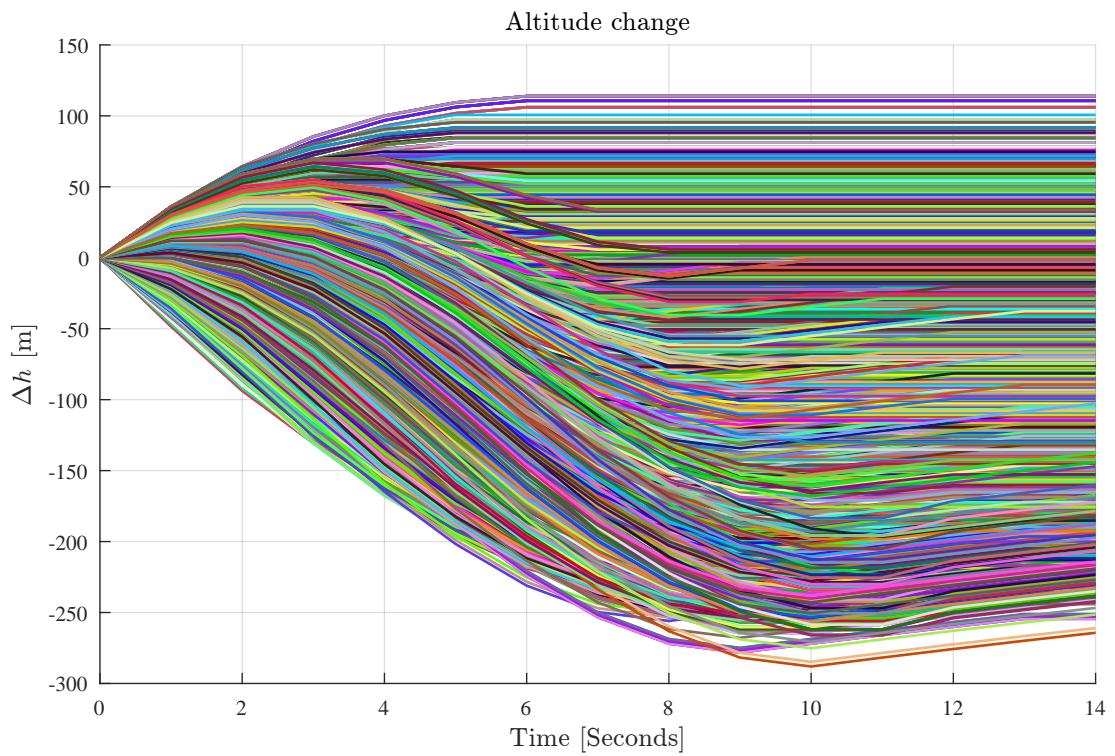


Figure 4.9: Plot of all altitude change recovery trajectories



Unfortunately not all initial states can be recovered from due to the constraints imposed on the system, such as maximum allowed load factor or maximum airspeed. Initial states with infinite cost associated with them are states for which the dynamic programming algorithm could not find an admissible solution. Conversely, initial states with finite cost associated with them indicate that the dynamic programming could find a valid optimal solution from these initial states to a terminal state, and these states are thus considered recoverable states. To get an idea of the distribution of recoverable versus non-recoverable states, a visual map can be made using the cost table values.

A visual grid representing each discrete flight path angle and airspeed state as a block at a bank angle of zero degrees was constructed, illustrated in Figure 4.10, to represent all strictly longitudinal upsets. The states with a finite cost associated with them, i.e. recoverable states are coloured green. The states with an infinite cost associated with them, i.e. unrecoverable states are coloured red. [30].

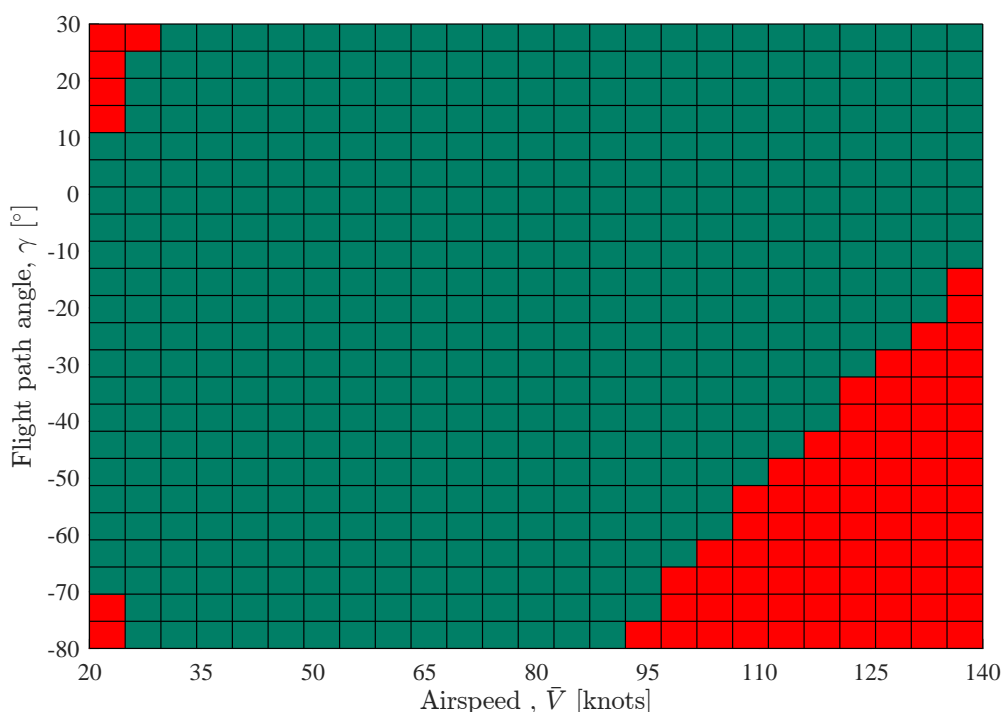


Figure 4.10: Colour map of recoverable and unrecoverable states. Airspeed vs. flight path angle, at a bank angle of 0 degrees. (green = recoverable, red = unrecoverable)

Most initial longitudinal states are recoverable, except for states with a steeply descending flight path angle at high initial airspeed. This is likely due to the algorithm not being able to find a recovery solution without exceeding the maximum airspeed constraint. A few unrecoverable states lie at severe underspeed conditions with high flight path angles or very low flight path angles.

To see how initial bank angle affects recovery, visual grids representing each flight path angle and bank angle initial state as a block at a certain initial speed was constructed, coloured according to recovery state. Several visual recovery grids are shown in Figure 4.11. Each sub-figure represents a different initial speed.

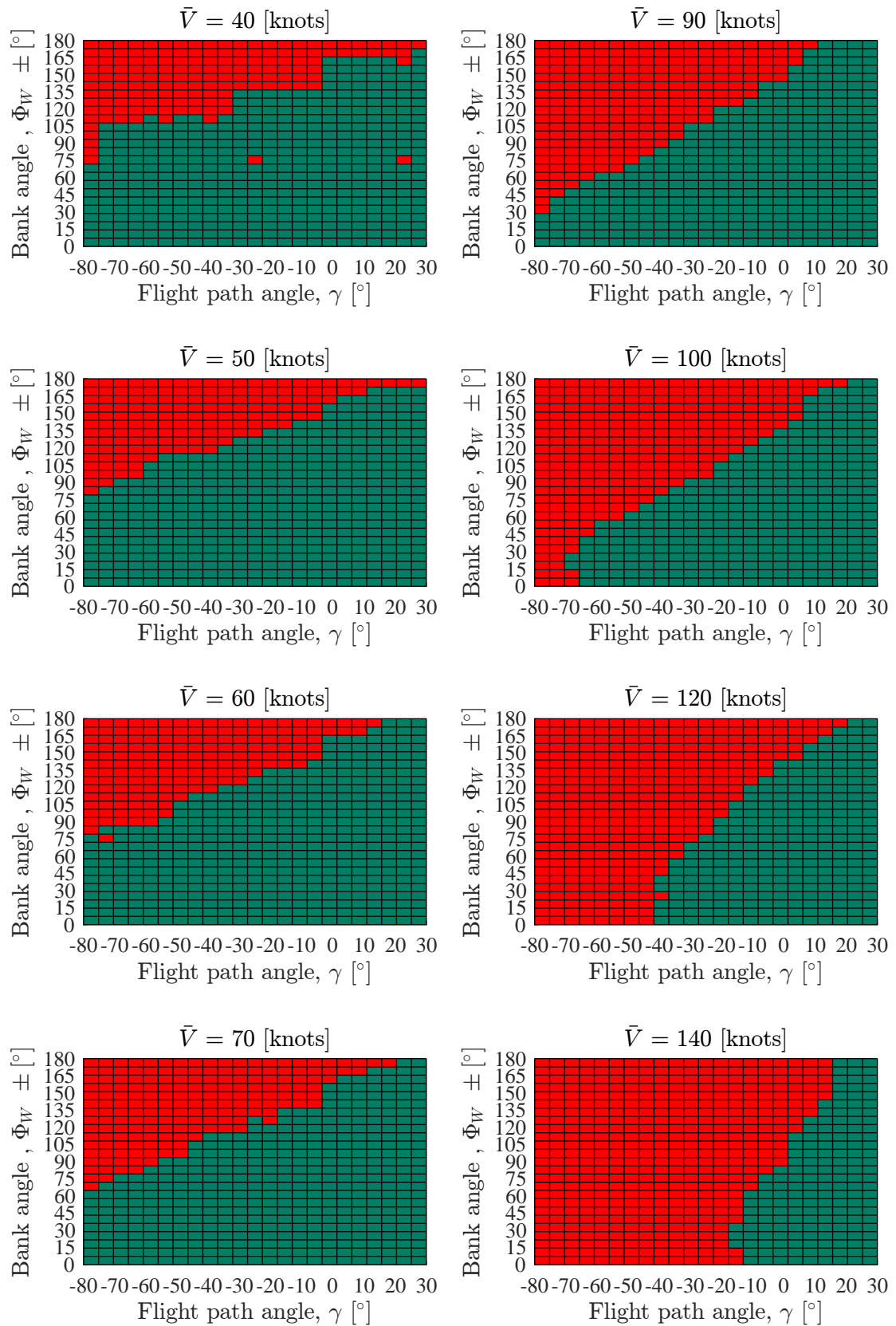


Figure 4.11: Colour map of recoverable and unrecoverable states. Flight path angle vs. bank angle, at various airspeeds. (green = recoverable, red = unrecoverable)

States are generally unrecoverable at high bank angles and steeply descending flight path angles, where the dynamic programming algorithm could not recover without either exceeding the minimum flight path angle or maximum airspeed constraint. More states at lower bank angles and steep descending flight path angles become unrecoverable at higher airspeeds (above 90 knots). At lower airspeeds (below 50 knots) the amount of unrecoverable states increases towards states with high bank angle and level to climbing flight path angles, as the aircraft cannot avoid stalling to below the allowed minimum airspeed.

The unrecoverable states close to some 90 degree bank angle states at an airspeed of 40 knots are due to the chosen quantisation resolution for the flight path angle state. These states are just outside the tolerance condition that decides whether to treat the state as if no lift can be produced and use alternative equations for the state transition. For these states no angle of attack could be found that transitioned the states to another quantised flight path angle state within an admissible lift value and within the allowed load factor constraint. Although the dynamic programming could not find a valid solution for these states, they are expected to be recoverable states, as they are well within the recoverable state region. Increasing the flight path angle quantisation resolution should solve this issue. However, this issue highlights an inherent limitation of the dynamic programming method that must quantise the states.

#### 4.4.6 Results Summary

Numerical results from the dynamic programming algorithm showed that it could optimally recover from various combinations of bank angle, flight path angle and airspeed upsets, which includes severe upsets such as inverted bank angles. All recoverable trajectories recovered the aircraft with minimum altitude loss prioritised above all other secondary costs, and remained within the structural integrity envelope due to adhering to the load factor and airspeeds constraints.

The dynamic programming method is limited by the dimensionality of the problem, i.e. the allowed number of state dynamic equations and the quantisation resolution of the state space, the latter of which can cause the algorithm to fail to find recovery solutions for some states. However, the dynamic programming gives a closed-loop solution and can find the global minimum solution for *all* recoverable initial discrete states.

### 4.5 Trajectory Optimisation using SQP

The dynamic programming trajectory (DP) planner presented in the previous section uses a reduced-order model of the aircraft dynamics that only includes the point mass translational dynamics and omits the fast rotational dynamics and the engine lag dynamics. This simplification of the aircraft dynamics was necessary to make the optimal control problem tractable to be solved with dynamic programming. (Dynamic programming suffers from the “curse of dimensionality” which limits its application to relatively low-dimensional dynamic models.)

We now propose a new approach to trajectory planning that uses sequential quadratic programming to solve the optimal upset recovery problem, as an alternative to the dynamic programming approach. The sequential quadratic programming approach uses direct transcription to transcribe the optimal control problem into a non-linear programming problem (NLP), which is then solved using a constrained optimisation algorithm called sequential quadratic programming. The SQP algorithm does not suffer from the “curse of dimensionality” and therefore allows higher-dimensional models to be used for the trajectory planning. The SQP planner therefore uses a higher-dimensional model of the aircraft dynamics that includes the

fast rotational dynamics (as represented by the inner-loop angle of attack controller and roll rate controller) as well as the engine lag dynamics.

In this section the optimal control problem of flight trajectory optimisation is transcribed into a NLP via a direct transcription method, in order to be solved by Matlab's SQP routine. The general formulation of a NLP is given, followed by a brief look at the SQP's general algorithm. A detailed discussion of the direct transcription method is then given, which involves the implementation of either Euler collocation or Hermite-Simpson collocation, followed by the detailed application of direct transcription to the flight trajectory optimisation problem. The section also discusses how gradient information required by the SQP solver is obtained with the use of an Automatic Differentiation (AD) tool called ADiGator. Finally, the section presents and discusses result cases that compare the recovery trajectory solutions of the SQP method with the solutions of the dynamic programming method. A comparison between SQP method solutions that uses a constant thrust model and SQP method solutions that uses a varying thrust model is also given.

#### 4.5.1 Non-linear Programming Problem

Most non-linear programming (NLP) methods used for solving open-loop trajectory optimisation problems incorporate some type of Newton method to solve for a finite set of unknowns [24]. A NLP is a decisional problem concerning a scalar objective function and a vector of constraints. As opposed to the optimal control problem, no dynamics are involved in a NLP. A NLP takes the following general mathematical form:

Determine the  $n_z$ -vector of decision variables  $\mathbf{z} \in \mathbb{R}^{n_z}$  that minimises,

$$F(\mathbf{z}) \tag{4.5.1}$$

subject to the algebraic constraints,

$$\mathbf{c}_L \leq \mathbf{c}(\mathbf{z}) \leq \mathbf{c}_U \tag{4.5.2}$$

and bounds,

$$\mathbf{z}_L \leq \mathbf{z} \leq \mathbf{z}_U \tag{4.5.3}$$

where  $\mathbf{c}(\mathbf{z}) \in \mathbb{R}^m$ . In this formulation equality constraints can be imposed by setting  $\mathbf{c}_L = \mathbf{c}_U$ .

#### 4.5.2 General SQP algorithm

The SQP method can be viewed as a natural extension of Newton and quasi-Newton methods to the constrained optimisation setting. The SQP method tries to mimic the Newton method for unconstrained optimisation by constructing and solving a subproblem known as a quadratic programming subproblem. Newton's method can be thought of as an optimisation method that continuously minimises a quadratic approximation of the objective function. The idea is to put the cost function and the constraint equations into a single objective function and try and solve it as a single minimisation problem [30].

A brief explanation of the general SQP algorithm is presented here with reference to a numerical survey by Rao [25].

In a gradient-based method such as the SQP, an initial guess is made for the unknown decision vector  $\mathbf{z}$ . At the  $k^{\text{th}}$  iteration the decision vector  $\mathbf{z}^{(k)}$  is updated to a new best approximation  $\mathbf{z}^{(k+1)}$  for the minimum location of the objective function with,

$$\mathbf{z}^{(k+1)} = \mathbf{z}^{(k)} + \tilde{\alpha}^{(k)} \mathbf{d}^{(k)} \quad (4.5.4)$$

where  $\mathbf{d}^{(k)}$  is the search direction in which to change the decision vector  $\mathbf{z}^{(k)}$  and  $\tilde{\alpha}^{(k)}$  is the magnitude of the step size that should be taken. The step size is determined by means of a line search algorithm and is chosen to sufficiently decrease the objective function. The search direction  $\mathbf{d}^{(k)}$  is determined by solving a subproblem known as the quadratic program problem (QP),

$$\min_{\mathbf{d}} \quad \frac{1}{2} \mathbf{d}^T \mathbf{H}^{(k)} \mathbf{d} + \nabla F^T(\mathbf{z}^{(k)}) \mathbf{d} \quad (4.5.5)$$

$$\text{s.t.} \quad \nabla c_i^T(\mathbf{z}^{(k)}) \mathbf{d} - c_i(\mathbf{z}^{(k)}) = \mathbf{0}, \quad (i \in \mathcal{A}) \quad (4.5.6)$$

$$\nabla c_i^T(\mathbf{z}^{(k)}) \mathbf{d} - c_i(\mathbf{z}^{(k)}) < \mathbf{0}, \quad (i \in \mathcal{I}) \quad (4.5.7)$$

where  $\mathcal{A}$  and  $\mathcal{I}$  are the active and inactive constraint sets respectively and  $\mathbf{H}^{(k)}$  is a positive definite approximation of the Hessian of the problem's Lagrangian  $\mathcal{L}$ , where  $\mathcal{L}$  is defined as,

$$\mathcal{L}(\mathbf{z}, \boldsymbol{\lambda}) = F(\mathbf{z}) - \boldsymbol{\lambda}^T \mathbf{c}(\mathbf{z}) \quad (4.5.8)$$

where the elements in  $\boldsymbol{\lambda}$  are known as the Lagrange multipliers. The Lagrange function combines the objective function and constraints using these multipliers. The SQP creates the quadratic programming subproblem by locally approximating the Lagrangian as a second order function (by creating a local quadratic approximation of the Lagrangian), and by approximating the constraints as first order functions (by locally linearising the constraints) [30].

In the case of the SQP, the Hessian  $\mathbf{H}^{(k)}$  is obtained using a quasi-Newton approximation which uses a well-known update method called the Broyden-Fletcher-Goldfarb-Shanno (BFGS) update. Usually  $\mathbf{H}^{(k)}$  is initialised as the identity matrix and each subsequent iteration it is updated with,

$$\mathbf{H}^{(k+1)} = \mathbf{H}^{(k)} + \frac{\mathbf{v}^{(k)} \mathbf{v}^{(k)T}}{\mathbf{v}^{(k)T} \mathbf{s}^{(k)}} - \frac{\mathbf{H}^{(k)} \mathbf{s}^{(k)} \mathbf{s}^{(k)T} \mathbf{H}^{(k)}}{\mathbf{s}^{(k)T} \mathbf{H}^{(k)} \mathbf{s}^{(k)}} \quad (4.5.9)$$

where  $\mathbf{v}^{(k)} = \nabla \mathcal{L}(\mathbf{z}^{(k+1)}, \boldsymbol{\lambda}^{(k)}) - \nabla \mathcal{L}(\mathbf{z}^{(k)}, \boldsymbol{\lambda}^{(k)})$  and  $\mathbf{s}^{(k)} = \tilde{\alpha}^{(k)} \mathbf{d}^{(k)}$ . Note that slight modifications are sometimes made to the BFGS update to ensure  $\mathbf{H}^{(k)}$  stays positive definite. See [31] for in-depth information on the SQP algorithm. The SQP algorithm steps are summarized here.

1. Set  $k = 0$  and estimate initial decision vector  $\mathbf{z}^{(0)}$  with some guess. Set  $\mathbf{H}^{(0)} = \mathbf{I}$ .
2. Evaluate objective function and constraint functions, and their gradients. Update the Hessian using BFGS (4.5.9).
3. Set up and solve the QP subproblem (4.5.5 - 4.5.7) to determine search direction  $\mathbf{d}^{(k)}$ .
4. Evaluate stopping criterion to decide if solution has converged using a tolerance check  $\|\mathbf{d}^{(k)}\| < \epsilon$  and that constraint violation is within tolerance.
5. Perform line search to determine step size  $\tilde{\alpha}^{(k)}$  which sufficiently decreases cost function.
6. Update decision vector with (4.5.4). Set  $k = k + 1$  and go to step 2.

### 4.5.3 Direct Transcription

This thesis focuses on using direct transcription using collocation methods and employing a non-linear programming routine to solve the trajectory optimisation problem. The process of transcribing an optimal control problem essentially converts it into a NLP by means of discrete function approximation. A certain order of discrete integral approximation is chosen for the integral term in Equation 4.3.19 and the dynamics in the form of differential equations are converted into a set of algebraic constraints with the same order of discrete approximation. The material on direct transcription and collocation methods presented here is mainly sourced from numerical method surveys by Betts, Rao and Topputo et al. [24; 25; 32].

For the direct formulation we break up the time window into  $K$  discrete smaller equally spaced intervals,

$$t_0 < t_1 < \dots < t_K = t_f \quad (4.5.10)$$

where the points are referred to as a node, mesh, grid or knot points. We use the notation  $\mathbf{x}_k \equiv \mathbf{x}(t_k)$  and  $\mathbf{u}_k \equiv \mathbf{u}(t_k)$ , ( $k = 0, 1, \dots, K$ ) to indicate the value of the state and control variables at a knot point.

#### Collocation

Collocation is used to transcribe differential dynamic constraints into a set of algebraic constraints. The idea is to choose a  $M^{\text{th}}$ -degree piecewise polynomial over the interval  $[t_k, t_{k+1}]$  to estimate the state,

$$\mathbf{X}(t) \approx \sum_{j=0}^M \mathbf{a}_k (t - t_k)^j, \quad t \in [t_k, t_{k+1}] \quad (4.5.11)$$

The coefficients ( $a_0, \dots, a_M$ ) of the piecewise polynomial are chosen so that the function value at the beginning of the interval matches the state value at that time,

$$\mathbf{X}(t_k) = \mathbf{x}(t_k) \quad (4.5.12)$$

The derivative of the piecewise polynomial is forced to match that of the state at certain points in the interval, called collocation points. The interval  $[t_k, t_{k+1}]$  is divided into  $L$  subintervals  $[\tau_i, \tau_{i+1}]$  where

$$\tau_i = t_k + h_k \alpha_i, \quad h_k = t_{k+1} - t_k, \quad 0 \leq \alpha_i \leq 1, \quad (i = 1, \dots, L) \quad (4.5.13)$$

At these collocation points we require that the approximated state derivative  $\dot{\mathbf{X}}(t)$  must match the right-hand side of the state dynamics equation. This requirement is then known as the *collocation condition* and is expressed as,

$$\dot{\mathbf{X}}(\tau_i) = \mathbf{f}(\mathbf{x}(\tau_i), \mathbf{u}(\tau_i), \tau_i) \quad (4.5.14)$$

The beginning and/or end points of the interval  $[t_k, t_{k+1}]$  can also be included to be collocation points, i.e. knot points can also be collocation points depending on the collocation scheme used. Figure 4.12 shows a graphical representation of a third-order collocation method known as the Hermite-Simpson method. The blue circles are the knot points and the red diamonds are collocation points. Within each interval it has one collocation point in the middle of the

interval at  $\tau_2$ , indicated by the red diamond, with  $\alpha_2 = 0.5$ . The beginning and end knot points of the interval  $t_k$  and  $t_{k+1}$ , indicated by blue dots, are also collocation points. By using a third-order Hermite polynomial and enforcing the collocation condition at the middle point (red diamond), the collocation condition is enforced for each of the three collocation points.

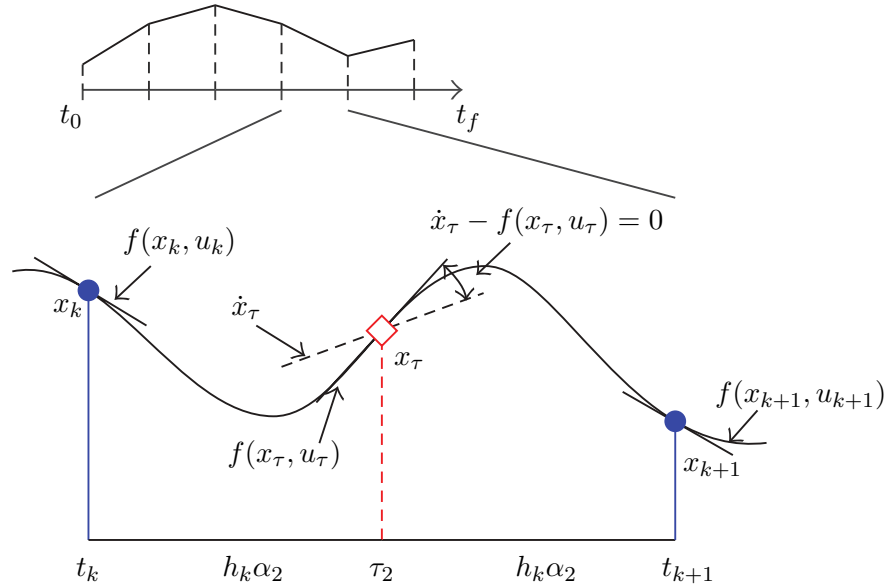


Figure 4.12: Collocation conditions for third-order method (Adapted from [32])

Direct transcription's solution becomes the true solution as the interval time, or sampling time strives to zero,

$$h_k \rightarrow 0 \quad (4.5.15)$$

Increasing the number of intervals should improve the solution accuracy, but this increases the problem dimensions leading to more computational time. Another way of improving the solution accuracy is to use a higher order collocation method, with more collocation points within each interval. This usually leads to the solution being approximated by a higher order piecewise polynomial.

### First-order Runge-Kutta (Euler) Collocation

For the direct method formulation we first have to obtain the discretised form of the optimal control problem equations. We assume a uniform time grid, i.e. our time steps are constant for each interval  $h_k = t_s$  ( $k = 0, 1, \dots, (K - 1)$ ). For Euler collocation this is done in the same manner as was done for the dynamic programming solution. The states are numerically integrated using forward Euler integration which approximates the state derivatives as,

$$\dot{\mathbf{x}}(t) \approx \frac{\mathbf{x}(k+1) - \mathbf{x}(k)}{t_s} \quad (k = 0, 1, \dots, (K - 1)) \quad (4.5.16)$$

where  $t_s$  is the time sample period defined as,

$$t_s = \frac{t_K}{K} \quad (4.5.17)$$

The dynamic state equations of Equation 4.3.20 are then numerically integrated as,

$$\mathbf{x}(k+1) = \mathbf{x}(k) + t_s \mathbf{f}(\mathbf{x}(k), \mathbf{u}(k)) \quad (4.5.18)$$

Now it is important to observe that the Euler method, and by extension Runge-Kutta methods, are also collocation methods. The Euler method is a first-order Runge-Kutta method and similarly a first-order collocation method, with  $M = 1$  in Equation 4.5.11. The only collocation point in the Euler collocation method is the start of each interval,  $t_k$ , i.e. the only collocation points are the knot points themselves.

When a Runge-Kutta method is employed in the form of collocation, the differential equation is said to be solved simultaneously, because all of the unknown parameters are determined at the same time. Collocation methods are said to simulate the dynamics implicitly, because the values of the state at each collocation point are obtained at the same time, as opposed to solving the state sequentially as in a time marching (shooting) method [25].

In order to implement simultaneous simulation, the discretised dynamics are written as defect constraints of the form,

$$\zeta_i = \dot{\mathbf{X}}(\tau_i) - \mathbf{f}(\mathbf{x}(\tau_i), \mathbf{u}(\tau_i), \tau_i) \quad (4.5.19)$$

The differential equations are then replaced by a finite set of defect constraints derived by the numerical integration scheme, in this case we are using Euler integration with  $i = 1$  and  $\alpha_i = 0$  in Equation 4.5.13, and so Equation 4.5.19 becomes:

$$\zeta_k = \dot{\mathbf{X}}_k - \mathbf{f}_k \quad (4.5.20)$$

which is written in the *derivative* form and where  $\mathbf{f}_k = \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k, t_k)$  and,

$$\dot{\mathbf{X}}_k = \frac{\mathbf{x}_{k+1} - \mathbf{x}_k}{t_s} \quad (4.5.21)$$

with the start of every interval being the only collocation point i.e,  $\tau_1 = t_k$ . The defect constraints can then be written in *integral* form as,

$$\zeta_k = \mathbf{x}_{k+1} - \mathbf{x}_k - t_s \mathbf{f}_k \quad (4.5.22)$$

where

$$t_s \mathbf{f}_k \approx \int_{t_k}^{t_{k+1}} \mathbf{f}(\mathbf{x}(\tau), \mathbf{u}(\tau), \tau) d\tau \quad (4.5.23)$$

In the case of Euler collocation the control is assumed to be piecewise constant and the dynamics are also assumed to be piecewise constant. The state is then approximated as a piecewise linear function. A graphical representation of Euler collocation can be seen in Figure 4.13. The goal is to find a solution where all the defects are equal to zero. For a more detailed discussion on direct transcription and collocation applied to optimal control problems see [32].



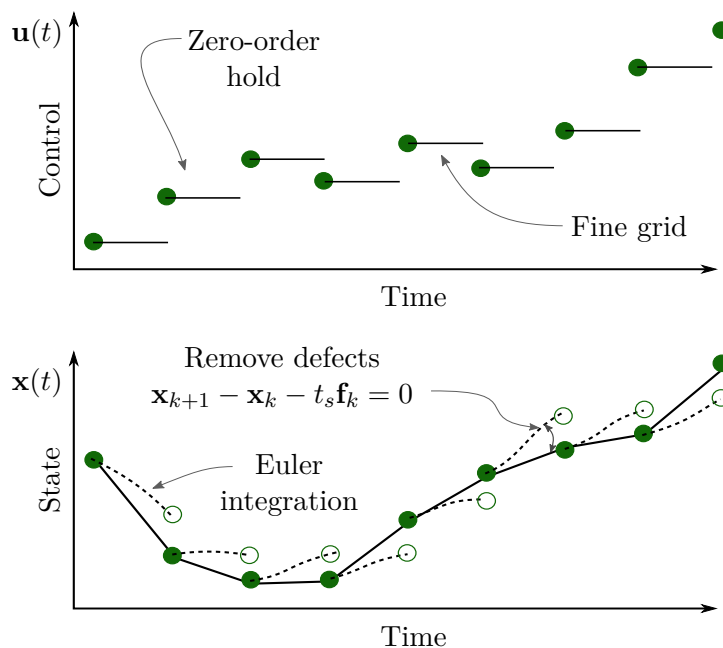


Figure 4.13: Euler Collocation (Adapted from [26])

### Third-order Hermite-Simpson Collocation

A better collocation method than Euler collocation is Hermite-Simpson collocation, a third-order collocation method which is recommended in literature. This method approximates the control as a piecewise linear function and the dynamics as a piecewise quadratic function. The state is then a piecewise cubic Hermite polynomial with a collocation point in the middle of each interval. The control may also be approximated as a zero-order hold or quadratic function. The time domain is divided into  $K$  intervals where the start and end of the interval  $[t_k, t_{k+1}]$  are collocation points with an additional collocation point in the middle of each interval  $t_{k+\frac{1}{2}}$ . This leads to  $N = 2K + 1$  collocation points. Figure 4.14 illustrates a graphical representation of the Hermite-Simpson method.

The objective function integral is approximated with a third-order Simpson quadrature method,

$$\int_0^{t_f} g(\mathbf{x}, \mathbf{u}, t) dt \approx \sum_{k=0}^{K-1} \frac{t_s}{6} (g_k + 4g_{k+\frac{1}{2}} + g_{k+1}) \quad (4.5.24)$$

The dynamics are transcribed into the Hermite interpolant condition and Simpson defect constraints which then make up the collocation constraints for each interval. The Hermite interpolant calculates the value of the state at the collocation point  $t_{k+\frac{1}{2}}$ ,

$$\mathbf{x}_{k+\frac{1}{2}} = \frac{1}{2} (\mathbf{x}_k + \mathbf{x}_{k+1}) + \frac{t_s}{8} (\mathbf{f}_k - \mathbf{f}_{k+1}) \quad (4.5.25)$$

The value of  $\mathbf{x}_{k+\frac{1}{2}}$  can be explicitly calculated from the start and end point variable values, or the collocation point  $\mathbf{x}_{k+\frac{1}{2}}$  can be made a decision variable and Equation 4.5.25 becomes an implicit collocation defect constraint,

$$\zeta_k^h = \frac{1}{2} (\mathbf{x}_k + \mathbf{x}_{k+1}) + \frac{t_s}{8} (\mathbf{f}_k - \mathbf{f}_{k+1}) - \mathbf{x}_{k+\frac{1}{2}} \quad (4.5.26)$$

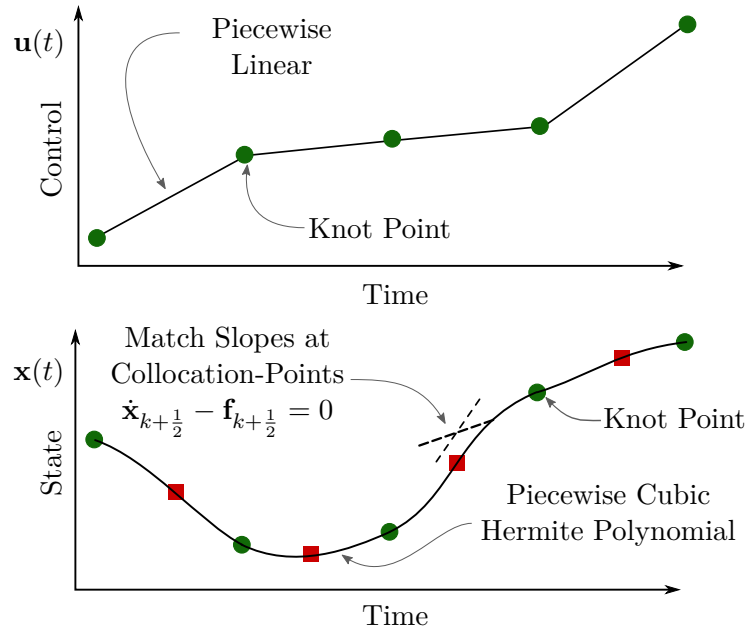


Figure 4.14: Hermite-Simpson Collocation (Adapted from [26])

The difference between these two approaches is discussed in Betts [24] and for this implementation we are using the more common latter case, using the implicit constraint  $\zeta_k^h$ . The value of the control at the middle collocation point can be calculated using simple linear interpolation and is not a decision variable in the method,

$$\mathbf{u}_{k+\frac{1}{2}} = \frac{\mathbf{u}_k + \mathbf{u}_{k+1}}{2} \quad (4.5.27)$$

The dynamics at the middle collocation point is enforced using the Simpson defect constraint,

$$\zeta_k^p = \mathbf{x}_k - \mathbf{x}_{k+1} + \frac{t_s}{6} (\mathbf{f}_k + 4\mathbf{f}_{k+\frac{1}{2}} + \mathbf{f}_{k+1}) \quad (4.5.28)$$

which is in the form of Equation 4.5.14. The full collocation constraint for each segment of the method is then,

$$\zeta_k = \{\zeta_k^h, \zeta_k^p\}^T \quad (4.5.29)$$

which consists of three collocation points per interval/segment. Enforcing these two defect equations will satisfy the dynamics, i.e. the collocation condition, at each of the three collocations points per interval. See [32] for the derivation of these collocation conditions. The Hermite-Simpson method possesses better numerical properties than the Euler method as it implicitly integrates the dynamics when used in this form.

#### 4.5.4 Transcribing the Flight Trajectory Optimisation Problem into an NLP

We are now ready to transcribe our original problem of aircraft trajectory optimisation, with the system described by Equations 4.3.9 to 4.3.12, into a NLP. The Euler collocation transcription implementation is explained for clarity as there is only a slight difference in how the defect constraints are written between it and the Hermite-Simpson collocation method.

### Transcribed Dynamics

The continuous differential equations of Equations 4.3.9 to 4.3.14 governing the aircraft dynamics must be transcribed into a set of discrete algebraic equations. In the case of Euler collocation, the system dynamics must first be written in the discrete form of Equation 4.5.18. The transcription method is presented using the first-order angle of attack model. The direct transcription method's structure makes it easy to extend it to higher dimensions. The directly transcribed aircraft system now becomes a set of algebraic equations that are the discrete system dynamics described by,

$$\bar{V}(k+1) = \bar{V}(k) + \frac{t_s}{m} \left[ T(k) - \frac{1}{2} \rho \bar{V}(k)^2 SC_D(\alpha(k)) - mg \sin(\gamma(k)) \right] \quad (4.5.30)$$

$$\gamma(k+1) = \gamma(k) + \frac{t_s}{\bar{V}(k)m} \left[ \frac{1}{2} \rho \bar{V}(k)^2 SC_L(\alpha(k)) \cos(\Phi(k)) - mg \cos(\gamma(k)) \right] \quad (4.5.31)$$

$$\Phi(k+1) = \Phi(k) + t_s P_W(k) \quad (4.5.32)$$

$$\alpha(k+1) = \alpha(k) + t_s \left[ -\frac{1}{\tau_\alpha} \alpha(k) + \frac{1}{\tau_\alpha} \alpha_c(k) \right] \quad (4.5.33)$$

$$\delta_T(k+1) = \delta_T(k) + t_s \left[ -\frac{1}{\tau} \delta_T(k) + \frac{1}{\tau} \delta_{Tc}(k) \right] \quad (4.5.34)$$

$$P_W(k+1) = P_W(k) + t_s \left[ -\frac{1}{\tau_P} P_W(k) + \frac{1}{\tau_P} P_{Wc}(k) \right] \quad (4.5.35)$$

Where  $\delta_T$  and  $\delta_{Tc}$  are the throttle state and throttle input command of the aircraft respectively. The throttle state is mapped to a specific thrust output via the GTM's non-linear thrust function,

$$T = f^T(\delta_T) \quad (4.5.36)$$

The system state vector for the discrete dynamics is then,

$$\mathbf{x}_k = \left[ \bar{V}_k \quad \gamma_k \quad \Phi_k \quad \alpha_k \quad \delta_{T_k} \quad P_{W_k} \right]^\top \quad (k = 0, 1, \dots, K) \quad (4.5.37)$$

and the input vector is,

$$\mathbf{u}_k = \left[ \alpha_{ck} \quad \delta_{Tck} \quad P_{Wck} \right]^\top \quad (k = 0, 1, \dots, K) \quad (4.5.38)$$

If the thrust is chosen to be kept constant in the recovery problem, then the throttle state and input variables  $\delta_T$  and  $\delta_{Tc}$  are omitted from the problem. The dynamic equation of Equation 4.5.34 is then also omitted from the transcribed dynamics, and the variable  $T(k)$  in Equation 4.5.30 is replaced by the chosen constant thrust value  $T(k) = T_{const}$ .

The system state vector for the discrete dynamics using a constant thrust value is then,

$$\mathbf{x}_k = \left[ \bar{V}_k \quad \gamma_k \quad \Phi_k \quad \alpha_k \quad P_{W_k} \right]^\top \quad (k = 0, 1, \dots, K) \quad (4.5.39)$$

and the input vector for the problem using a constant thrust value is,

$$\mathbf{u}_k = [\alpha_{ck} \quad P_{Wck}]^T \quad (k = 0, 1, \dots, K) \quad (4.5.40)$$

The discretised system equations are written in the form of Equation 4.5.22 to become the defect constraints of the NLP problem. If the Hermite-Simpson collocation method is used, the aircraft dynamics are transcribed into defect constraints using Equations 4.5.26 and 4.5.28 instead.

### Design Variables

The NLP design variables become the values of the states and controls at the knot points of Equation 4.5.10. The length of the time window is also made a variable by adding  $t_K$  to the design variable vector. A fixed grid size of  $K$  segments, resulting in  $N = K + 1$  knot points, is then chosen and the sample time  $\Delta t = t_s$  of Equation 4.5.17 is allowed to vary with  $t_K$ . Alternatively  $t_K$  can be omitted from the design vector and made a fixed constant. The whole design variable vector of the transcribed problem is,

$$\mathbf{z} = \{\mathbf{x}_0, \mathbf{u}_0, \mathbf{x}_1, \mathbf{u}_1, \dots, \mathbf{x}_K, \mathbf{u}_K, t_K\}^T \quad (4.5.41)$$

with  $N$  state decision variables and  $N$  control decision variables. In the case of Hermite-Simpson collocation, there would be  $N = 2K + 1$  state decision variables and  $K + 1$  control decision variables, as the state values at the midpoint collocation points are included as design variables.

### Cost Function

The cost function of Equation 4.3.25 is discretised and is now viewed as the non-linear function  $F(\mathbf{z})$  to be minimised,

$$F(\mathbf{z}) = \sum_{k=t_0}^K \max(-\bar{V}_k \sin(\gamma_k), 0) t_s \quad (4.5.42)$$

subject to defect constraints and the state and control bounds that will be transcribed into NLP constraints and decision variable bounds. In the case of Hermite-Simpson collocation we set  $F(\mathbf{z})$  equal to Equation 4.5.24 instead. We do not use a multi-objective cost function as with the dynamic programming hierarchical cost function of Equations 4.4.4 to 4.4.6, because we would have to implement it as a weighted multi-objective cost function for the NLP and we do not wish to compromise the altitude loss objective function at all. (This is a disadvantage of the sequential quadratic programming approach.)

### Design Variable Bounds

The optimal control problem's state and control bounds of Equations 4.3.26 to 4.3.34 become a discrete set of bounds by applying a specific upper and lower bound on each design variable that are packed into the design variable bound vectors  $\mathbf{z}_L$  and  $\mathbf{z}_U$ ,

$$\mathbf{z}_U = \{\mathbf{x}_{0U}, \mathbf{u}_{0U}, \mathbf{x}_{1U}, \mathbf{u}_{1U}, \dots, \mathbf{x}_{KU}, \mathbf{u}_{KU}, t_{KU}\}^T \quad (4.5.43)$$

with the state and control upper bound vectors as,

$$\mathbf{x}_{kU} = \left[ \bar{V}_{kU} \quad \gamma_{kU} \quad \alpha_{kU} \quad \Phi_{kU} \quad \delta_{TkU} \quad P_{WkU} \right]^\top \quad (k = 0, 1, \dots, (K-1)) \quad (4.5.44)$$

$$\mathbf{u}_{kU} = \left[ \alpha_{ckU} \quad \delta_{TckU} \quad P_{WckU} \right]^\top \quad (k = 0, 1, \dots, K) \quad (4.5.45)$$

with  $\bar{V}_{kU} = \bar{V}_u$ ,  $\gamma_{kU} = \gamma_u$ ,  $\alpha_{kU} = \alpha_u$ ,  $\Phi_{kU} = \Phi_u$ ,  $\alpha_{ckU} = \alpha_{cu}$ ,  $\delta_{TckU} = \delta_{Tcu}$  and  $P_{WkU} = P_{Wu}$ . The final time instant upper bound state vector is then set to any set of unique terminal state bounds. In this case the airspeed has a unique terminal bound. The terminal condition bounds of Equation 4.3.37  $\bar{V}_{fl}$  and  $\bar{V}_{fu}$  are enforced by simply replacing the bounds of Equation 4.3.26  $\bar{V}_l$  and  $\bar{V}_u$  in the final time variables  $\mathbf{x}_{KL}$  and  $\mathbf{x}_{KU}$  in the decision variable bound vectors  $\mathbf{z}_L$  and  $\mathbf{z}_U$  with the values of the terminal condition bounds,

$$\mathbf{x}_{KU} = \left[ \bar{V}_{KU} \quad \gamma_{KU} \quad \alpha_{KU} \quad \Phi_{KU} \quad \delta_{TKU} \quad P_{WKU} \right]^\top \quad (4.5.46)$$

with  $\bar{V}_{KU} = \bar{V}_{fu}$ ,  $\gamma_{KU} = \gamma_u$ ,  $\alpha_{KU} = \alpha_u$ ,  $\Phi_{KU} = \Phi_u$  and  $\delta_{TKU} = \delta_{Tu}$ . The bounds setup is exactly the same for the lower bound decision variable vector  $\mathbf{z}_L$  just with the corresponding lower bound variables set as above. Note that it is not recommended to enforce equality constraints by setting variable bounds  $\mathbf{z}_U = \mathbf{z}_L$  as this introduces a pair of ill-defined extra slack variables inside Matlab's SQP routine. The equality constraint vector  $\mathbf{c}(\mathbf{z})$  is intended for this purpose instead.

### Equality Constraint Vector

The remaining initial, final and defect constraints are all handled as NLP equality constraints in this problem. Thus the  $m$  NLP constraints become,

$$\mathbf{c}(\mathbf{z}) = \begin{bmatrix} \zeta_0 \\ \zeta_1 \\ \vdots \\ \zeta_{K-1} \\ \boldsymbol{\eta}_0 \\ \boldsymbol{\eta}_K \end{bmatrix} = \mathbf{c}_L = \mathbf{c}_U = \mathbf{0} \quad (4.5.47)$$

where  $\boldsymbol{\eta}_K = \boldsymbol{\eta}_f$ . The initial time constraint equation for this case would be,

$$\boldsymbol{\eta}_0 = [-\mathbf{x}_0 - \mathbf{x}_{ini}] \quad (4.5.48)$$

with,

$$\mathbf{x}_0 = \left[ \bar{V}_0 \quad \gamma_0 \quad \alpha_0 \quad \Phi_0 \quad \delta_{T0} \quad P_{W0} \right]^\top \quad (4.5.49)$$

$$\mathbf{x}_{ini} = \left[ \bar{V}_{ini} \quad \gamma_{ini} \quad \alpha_{ini} \quad \Phi_{ini} \quad \delta_{Tini} \quad P_{Wini} \right]^\top \quad (4.5.50)$$

The initial state of the system is enforced by setting the variables with subscript *ini* to the desired initial state value for the numerical simulation. The vector  $\boldsymbol{\eta}_K$  is made up of the terminal state constraint equations and the terminal state trim conditions,

$$\boldsymbol{\eta}_K = \left[ \boldsymbol{\zeta}_T^\top, \quad -\gamma_K - \gamma_f, \quad -\Phi_K - \Phi_f \right]^\top \quad (4.5.51)$$

where the trim state constraint vector  $\boldsymbol{\zeta}_T$  forces the algorithm to calculate the trim inputs at the terminal state of the trajectory and consists of the following set of equations,

$$\boldsymbol{\zeta}_T = \begin{bmatrix} T_K - \frac{1}{2}\rho S \bar{V}_K^2 C_D(\alpha_K) \\ \frac{1}{2}\rho S \bar{V}_K^2 C_L(\alpha_K) - mg \\ P_{WK} \\ -\alpha_K + \alpha_{cK} \\ -\delta_{TK} + \delta_{TcK} \end{bmatrix} \quad (4.5.52)$$

The trim state constraint vector assumes that we are trimming for straight and level flight. The NLP will choose the design variables, i.e. the state and control values at each grid point, so that the defects of Equation 4.5.22 are driven to zero. The collocation function will approximate the true dynamics of the system within the accuracy of the numerical integration scheme chosen. For more information on numerical methods applied to optimal control problems see surveys [24] and [25].

### Adding Non-linear Inequality Constraints

The flight trajectory optimisation problem can also have non-linear inequality constraints or path constraints. The first is in the form of a normal load factor constraint,

$$n_{L_{\min}} \leq \frac{\frac{1}{2}\rho \bar{V}^2 S C_L(\alpha)}{mg} \leq n_{L_{\max}} \quad (4.5.53)$$

The SQP routine used in Matlab's `trim` function does not support non-linear inequality constraints at a higher function level. However, one way NLP routines handle inequality constraints is by converting them into equality constraints using slack variables. The load factor constraint can be included by writing it as two sets of equality constraints applied at every grid point along the trajectory,

$$n_{L_{\min}} - n_{L_k} + s_{k_{L_{\min}}}^2 = 0 \quad (k = 0, 1, \dots, K) \quad (4.5.54)$$

$$-n_{L_{\max}} + n_{L_k} + s_{k_{L_{\max}}}^2 = 0 \quad (k = 0, 1, \dots, K) \quad (4.5.55)$$

where

$$n_{L_k} = \frac{\frac{1}{2}\rho \bar{V}_k^2 S C_L(\alpha_k)}{mg} \quad (k = 0, 1, \dots, K) \quad (4.5.56)$$

The slack variables  $s_k$  are added to the decision variable vector  $\mathbf{z}$  for the problem. The slack variables are unbounded, meaning their upper and lower bounds are set as  $z_U = \infty$  and  $z_L = -\infty$  respectively. The quadratic term in Equations 4.5.54 and 4.5.55 ensure that the slack terms stay positive so that the inequality holds as defined.

The second non-linear inequality constraint is in the form of the roll rate envelope of Figure 4.5 and can be added in a similar manner as above, using its own set of slack variables. Only one set of extra slack variables are used for the roll rate envelope constraint unlike the two sets needed for the load factor constraint, because the roll envelope only bounds the maximum roll rate.

### 4.5.5 Scaling

Scaling is very important when dealing with a NLP. Poor scaling can lead to very poor convergence rates or even divergence. Scaling can also affect the termination tests and numerical conditioning of a NLP [33]. The idea behind scaling is that all the variables should have the same effect on the objective function, meaning their gradients must be of similar scales. When the relative size of variables or constraints in a problem are significantly different, it is recommended to introduce scaling factors to transform them to be of similar scale. For instance, when dealing with trajectory optimisation, it commonly occurs that some state variables may range from  $\pi$  to  $-\pi$  (as in the case of angles), where others may range from 0 to 1000 meters (as in the case of altitude variables). The special structure of the optimal control problem can be exploited to improve the scaling of the underlying NLP subproblem.

Betts [33] suggests making the ranges of the state and control variables more uniform using,

$$\begin{bmatrix} \tilde{\mathbf{x}} \\ \tilde{\mathbf{u}} \end{bmatrix} = \begin{bmatrix} \mathbf{V}_x & \mathbf{0} \\ \mathbf{0} & \mathbf{V}_u \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ \mathbf{u} \end{bmatrix} + \begin{bmatrix} \mathbf{r}_x \\ \mathbf{r}_u \end{bmatrix} \quad (4.5.57)$$

which relates the scaled state  $\tilde{\mathbf{x}}$  and control  $\tilde{\mathbf{u}}$  to the unscaled quantities  $\mathbf{x}$  and  $\mathbf{u}$ . The  $n_x \times n_x$  diagonal matrix  $\mathbf{V}_x$  contains the state variable scale weights and the  $n_u \times n_u$  diagonal matrix  $\mathbf{V}_u$  contains the control scale weights. The corresponding variable shifts are defined by the vectors  $\mathbf{r}_x$  and  $\mathbf{r}_u$ .

The problem constraints, which consists of the defect constraints  $\boldsymbol{\zeta}$  and path constraints  $\mathbf{g}$  are normalised using,

$$\tilde{\mathbf{c}} = \begin{bmatrix} \mathbf{W}_f & \mathbf{0} \\ \mathbf{0} & \mathbf{W}_g \end{bmatrix} \begin{bmatrix} \boldsymbol{\zeta} \\ \mathbf{g} \end{bmatrix} \quad (4.5.58)$$

which relates the scaled constraints  $\tilde{\mathbf{c}}$  to the unscaled constraints  $\boldsymbol{\zeta}$  and  $\mathbf{g}$ . Here  $\mathbf{W}_f$  is an  $n_x \times n_x$  diagonal matrix of differential equation scale weights and  $\mathbf{W}_g$  is an  $n_g \times n_g$  diagonal matrix of path-constraint equation scale weights.

For a well conditioned constraint Jacobian it is suggested by Betts to set,

$$\mathbf{W}_f = \mathbf{V}_x \quad (4.5.59)$$

The path constraint weights  $\mathbf{W}_g$  should be chosen such that the derivatives of the path constraints are of norm one. The variables scales are chosen such that the scaled quantities  $\tilde{x} \sim \mathcal{O}(1)$  and lie in the interval  $[-0.5, +0.5]$ . We use the problem's state and control bounds of Equations 4.3.26 to 4.3.34 by setting,

$$v_k = \frac{1}{x_{kU} - x_{kL}} \quad (4.5.60)$$

$$r_k = \frac{1}{2} - \frac{x_{kU}}{x_{kU} - x_{kL}} \quad (4.5.61)$$

It is also recommended that the objective function should be normalised such that,

$$\tilde{F} = w_F F \approx 1 \quad (4.5.62)$$

which relates the scaled objective function  $\tilde{F}$  to the unscaled objective function  $F$  using a scalar weight  $w_F$ .

### 4.5.6 Obtaining the Objective Gradient and Constraint Jacobian using Automatic Differentiation

The SQP routine being employed is a gradient based method, and thus requires first-order and second-order gradient information in the form of the objective function's gradient, the constraint vector's Jacobian matrix and Lagrangian's Hessian matrix. The objective function's gradient and the constraint vector's Jacobian is either supplied by the user or the routine employs a finite difference approximation, much like 4.5.16. Recall that the form of the Jacobian  $\mathbf{Jf}(\mathbf{x})$  for a vector function  $\mathbf{f}(\mathbf{x})$  is,

$$\mathbf{Jf}(\mathbf{x}) = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \frac{\partial f_2}{\partial x_1} & \cdots & \frac{\partial f_2}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \cdots & \frac{\partial f_m}{\partial x_n} \end{bmatrix} \in \mathbb{R}^{mn} \quad (4.5.63)$$

The Hessian is typically obtained via a quasi-Newton method like the BFGS approximation mentioned in section 4.5.1. Calculating the gradient information for large complex problems a-priori is often not a trivial matter and one has to settle for the routine's inbuilt finite difference method, which can be slow and inaccurate. Calculating the gradients by hand can be a time consuming and error prone process. If one wishes to increase the dimensionality of a problem, the gradients must be recalculated and can become an intractable process. The implementation here uses a method called automatic differentiation (AD) to obtain the problem's gradient information, which automatically constructs the gradient and Jacobian of the problem to pass to the SQP routine and allows them to easily be reconstructed for additional problem dimensions.

As we increase the dimensionality for the problem of optimal attitude and flight vector recovery using SQP to build up a more representative model, we run into some of the previously mentioned issues. For low dimensional cases such as the two-dimensional case, the objective function's gradient and constraint vector's Jacobian was initially calculated by hand, but for the larger dimensional cases that include state lag dynamics and input state dependent constraints, more effective and reliable methods for calculating the gradients were investigated.

The initial choice was to use Matlab's symbolic toolbox to automatically generate the Jacobian, but this process runs into a fundamental flaw due to the nature of the problem. The GTM model uses lookup tables to calculate its non-linear aerodynamic effects, thus no closed form equations existed for the lift and drag functions in the problem definition. The Matlab symbolic toolbox version (2008) that was available alone cannot differentiate such discontinuous piecewise functions. An alternative method was found using a method called automatic differentiation (AD) to obtain the objective function gradient and constraint vector's Jacobian. A relatively new AD software package (as of 2016) licensed under the GNU General Public License called ADiGator [34] was used, that can handle piecewise functions. Automatic differentiation is a numerical differentiation method, but unlike finite difference approximations it does not suffer from numerical inaccuracies and instabilities, and is accurate to machine precision.

### The Concept of Automatic Differentiation

The concept of AD, also known as algorithmic differentiation, started with the advent of object orientated programming (OOP), but has not seen much attention until recent years after its power was 'rediscovered'. Many AD software implementations exist today. The two main implementations of this method are what is known as operator overloading (OO), which uses



OOP, and source code transformation (ST). Recently these two methods have merged, taking the advantages of both. AD can be considered the third alternative in numerical differentiation. It uses exact formulas along with floating-point values, instead of expression strings as in symbolic differentiation, and it involves no approximation error as in numerical differentiation using difference quotients [35]. There are two distinct modes of AD, namely forward mode and reverse mode. Forward mode is easier to implement and therefore easier to understand, but uses more memory than reverse mode. Forward mode is implemented using OO, whereas reverse mode can be implemented with a combination of OO and ST. For a comprehensive introduction to automatic differentiation using OO and OOP see [35].

The key idea of AD is that it uses basic derivative rules from calculus, such as the chain rule, and implement them numerically, and exploits the fact that a general function  $\mathbf{y} = \mathbf{f}(\mathbf{x})$  can be decomposed into a sequence of elementary function operations. In either forward or reverse mode, each link in the calculus chain rule is implemented until the derivative of the input with respect to itself is encountered. For the simple composition  $y = g(h(x)) = g(w)$ , the chain rule states,  $\frac{\partial y}{\partial x} = \frac{\partial y}{\partial w} \frac{\partial w}{\partial x}$ . In forward mode, the operations are performed from the variable of differentiation to the final derivative of the function, i.e.  $\frac{\partial w}{\partial x}$  is computed first and then  $\frac{\partial y}{\partial w}$ . Reverse mode performs the operations from the function back to the variable of differentiation. Consider the evaluation of the function  $f(x_1, x_2) = x_1x_2 + \sin(x_1)$ . The evaluation can be represented by the computational graph in Figure 4.15.

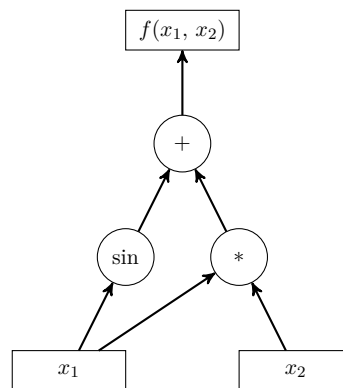


Figure 4.15: Computational graph representing  $f(x_1, x_2) = x_1x_2 + \sin(x_1)$

Suppose that  $x_1 = 2$  and  $x_2 = 3$ . The function will then be evaluated by computing the sequence of values on the right of Table 4.2. The idea of AD is to perform the derivative calculations automatically along with the function evaluation. One way to achieve this is by parsing the function expression  $f(x_1, x_2) = x_1x_2 + \sin(x_1)$  and producing the sequence of expressions in both columns of Table 4.2. This approach is known as the source transformation method and can be quite involved. Alternatively OOP can be used along with OO to define an object that stores its value and its derivative, using the chain rule when basic operators are invoked on the object to also calculate the derivative. The forward and reverse mode methods make use of the latter implementation. Forward mode advances from bottom to top in the computational graph in Figure 4.15. The reverse mode instead uses an initial forward pass to calculate only the values of what is known as the adjoints, and then transverse the graph from top to bottom while accumulating these adjoints using the chain rule. The disadvantage of forward mode becomes apparent when one thinks of the data structure produced as a binary tree which has a value and a derivative branch at each node. It requires  $2^n$  locations for computing  $n + 1$  distinct values. For functions with a large number of independent variables the forward mode runs into memory issues.

Table 4.2: Sequence of value and derivative calculations performed in AD example

$x_1 = 2$	$x'_1 = 1$
$x_2 = 3$	$x'_2 = 1$
$w_1 = \sin(x_1) = 0.909$	$w'_1 = \cos(x_1)x'_1 = -0.416$
$w_2 = x_1 * x_2 = 6$	$w'_2 = x'_1x_2 + x_2x'_1 = 5$
$f = w_1 + w_2 = 6.909$	$f' = w'_1 + w'_2 = 4.584$

The software package used, known as ADiGator, is a Matlab implementation and uses a value/-derivative object in forward mode. The object also then produces a file using ST, that contains the statements that computes the derivative of the original function with respect to the input variable. This file can then be called like a normal function file that has an execution time in the same order as one normal function evaluation, and does not suffer from the memory issues when using forward mode alone.

### Matrix Sparsity in Optimisation Problems

ADiGator exploits sparsity in the Jacobian of functions by only storing non-zero elements of the Jacobian and uses less memory by doing so. A matrix is sparse if most of its elements are zero and conversely a matrix is dense if most of its elements are non-zero. It is well-known that directly transcribed problems result in a sparse Jacobian in most cases [32], as will be shown in this section. Gradient methods such as the SQP also takes advantage of sparsity in the Jacobian. This avoids out-of-memory issues when scaling a direct method to more variables or refining a problem's mesh, because each time grid point for each variable corresponds to a column in the Jacobian matrix. Figure 4.16 shows the Jacobian sparsity pattern of the constraint vector of Equation 4.5.47 of a third order Hermite-Simpson method, and incidently the pattern for the Euler collocation as well.

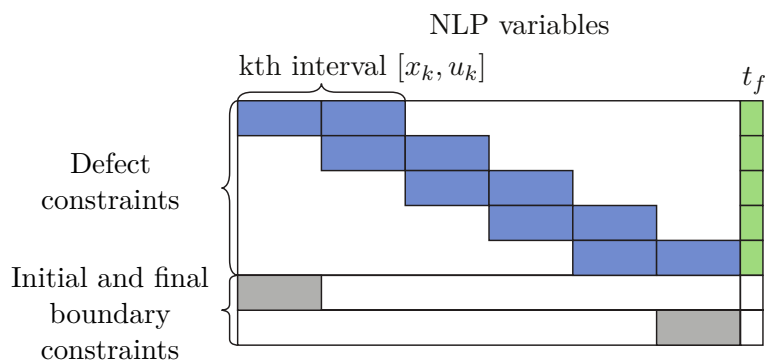


Figure 4.16: Jacobian sparsity pattern of constraints (Adapted from [32])

The shaded blocks represent non-zero elements in the matrix with all other elements being zero. Each shaded blue square corresponds to the variable pair  $[x_k, u_k]$  at time instant  $t_k$ . There are two elements along the diagonal of the matrix corresponding to the defect constraint equations, because each constraint equation is dependant on variables at a time instant  $t_k$  and  $t_{k+1}$ . If the final time is also a variable, then each constraint equation is also dependant on the variable  $t_f$  in the form of Equation 4.5.17, which is represented by the green shaded squares. If the Jacobian for a directly transcribed problem was dense, it would mean that each time

interval's dynamic constraint equation is dependent on variables at every other time instant in the whole time window.

### Using ADiGator for Trajectory Optimisation Problem

The ADiGator software package for Matlab was used to construct the objective function's gradient and constraint vector's Jacobian functions of the directly transcribed problem, which are then passed the SQP routine. The SQP routine expects the objective function and constraint function together in one function which is passed. Likewise, the routine also expects the gradient of the objective function and constraint Jacobian in one function. The GTM model lookup tables are interpolated using a first order linear interpolation within the Simulink model. Using a linear piecewise function for the SQP method is not ideal, due to the discontinuities in the function's gradient. Spline interpolation was used instead to interpolate the static lift and drag lookup table values. Spline interpolation uses piecewise polynomials to represent the problem's non-linear functions to obtain smooth function gradients. ADiGator has support for Matlab's inbuilt piecewise polynomial interpolation objects and as a result can differentiate the spline functions.

ADiGator was set up with the appropriate variables of differentiation and then the objective/constraint function was passed to it. ADiGator then generates a new function file which can be called with the same arguments as the objective/constraint function, but the generated file instead returns the gradient and Jacobian values. Whenever the mesh resolution for the transcription changes, the derivative function file must be regenerated. ADiGator can also be used in vectorised mode which relies on the sparsity pattern of the Jacobian to generate the matrix for any sized mesh. Thus in this mode the function file does not have to be regenerated for different meshes. ADiGator's vectorised mode is more suited to algorithms that implement adaptive meshes, where the optimisation is rerun multiple times with finer meshes at key sections along a solution in order to minimise the maximum numerical error of the solution. The Hessian of the problem's Lagrangian can also potentially be generated by simply passing ADiGator the first-order generated derivative function file, and it will then generate the second-order derivative function file, which would be the Hessian in this case. Having second-order information available means one does not have to rely on the quasi-Newton BFGS methods, and could result in a significant performance gain. Unfortunately, the SQP routine used in the available Matlab version does not support user supplied Hessian information. For more insight on how ADiGator works, see the user guide [36] for details.

#### 4.5.7 Calling the SQP Solver

With the problem transcribed and scaled, the formatted information can be passed to the SQP function that solves the NLP. The function call to the SQP would typically consist of the following output and input parameters,

$$[\tilde{\mathbf{z}}_{soln}] = \text{SQP}(\tilde{\mathbf{z}}_{guess}, \tilde{\mathbf{F}}(\tilde{\mathbf{z}}), \tilde{\mathbf{c}}(\tilde{\mathbf{z}}), \nabla\tilde{\mathbf{F}}(\tilde{\mathbf{z}}), \mathbf{J}\tilde{\mathbf{c}}(\tilde{\mathbf{z}}), \tilde{\mathbf{z}}_U, \tilde{\mathbf{z}}_L) \quad (4.5.64)$$

The SQP function is supplied with a guess vector which consists of the scaled initial values of the decision variables chosen by the user  $\tilde{\mathbf{z}}_{guess}$ , the scaled objective function  $\tilde{\mathbf{F}}(\tilde{\mathbf{z}})$ , the scaled constraint vector  $\tilde{\mathbf{c}}(\tilde{\mathbf{z}})$ , the scaled function gradient  $\nabla\tilde{\mathbf{F}}(\tilde{\mathbf{z}})$ , the scaled constraint function Jacobian  $\mathbf{J}\tilde{\mathbf{c}}(\tilde{\mathbf{z}})$ , the scaled upper bound decision variable vector  $\tilde{\mathbf{z}}_U$  and the scaled lower bound decision variable vector  $\tilde{\mathbf{z}}_L$ . The SQP function then outputs the scaled decision variable vector solution  $\tilde{\mathbf{z}}_{soln}$  if the solver converged. This output should then be unscaled and interpolated to obtain the state and input trajectory solutions to the original problem.

The objective function and constraint vector functions are passed to the ADiGator tool that then generates the objective function gradient and constraint Jacobian functions.

#### 4.5.8 Representing the Solution

The NLP produces a solution represented by the decision variables which are a discrete set of points. The solution to the original continuous time trajectory optimisation problem however, is within a function space. Interpolation should then be used to convert the NLP solution back into a continuous function space solution. Each transcription method has a corresponding interpolation method [27]. A transcription process takes a continuous problem to a discrete problem, while interpolation takes a discrete solution (such as our set of decision variables) and represents it as a continuous solution by taking a discrete set of points back to continuous functions. Direct transcription uses what is known as spline interpolation. A spline function is a sequence of polynomial segments which are all of a given order, i.e. a quadratic spline consists of a sequence of second-order polynomial functions, with one polynomial between each knot point.

#### Euler Collocation

Recall that the definition for Euler collocation is that the control and dynamics are approximated by a piecewise constant function between  $N = K + 1$  knot points. Thus the control solution and dynamics should then be represented using a piecewise constant, i.e. zero-order hold interpolation function,

$$\mathbf{u}(t) = \begin{cases} \mathbf{u}_k, & t \in [t_k, t_{k+1}] \\ 0, & \text{otherwise} \end{cases} \quad (k = 0, 1, \dots, (N - 1)) \quad (4.5.65)$$

$$\mathbf{f}(\mathbf{x}(t), \mathbf{u}(t), t) = \begin{cases} \mathbf{f}_k, & t \in [t_k, t_{k+1}] \\ 0, & \text{otherwise} \end{cases} \quad (k = 0, 1, \dots, (N - 1)) \quad (4.5.66)$$

The state is then the integral of the dynamics, meaning the state solution is represented by an interpolating function which has an order one degree higher than the dynamics. This is true in the case of most direct collocation methods. Thus in the case of Euler collocation, the state solution is represented by a piecewise linear interpolation function and is obtained by integrating the dynamics of Equation 4.5.66,

$$\mathbf{x}(t) = \begin{cases} \mathbf{x}_k + \delta_k \mathbf{f}_k, & t \in [t_k, t_{k+1}] \\ 0, & \text{otherwise} \end{cases} \quad (k = 0, 1, \dots, (N - 1)) \quad (4.5.67)$$

where  $\delta_k = t - t_k$  and the value at the start of each segment  $\mathbf{x}_k$  acts as the constant of integration.

#### Hermite-Simpson Collocation

The definition of the Hermite-Simpson collocation method that was used, approximates the control as a piecewise linear first-order function and the dynamics as a quadratic second-order function on  $K$  segments, with one collocation point between each segment. This equates to  $N = 2K + 1$  collocation points. The control solution is then represented as a piecewise linear spline interpolation function,

$$\mathbf{u}(t) = \begin{cases} \mathbf{u}_k + \delta_k \mathbf{c}, & t \in [t_k, t_{k+1}] \\ 0, & \text{otherwise} \end{cases} \quad (k = 0, 1, \dots, K) \quad (4.5.68)$$

where,

$$\begin{aligned} \delta_k &= t - t_k \\ \mathbf{c} &= \frac{-1}{t_s} (\mathbf{u}_k - \mathbf{u}_{k+1}) \end{aligned} \quad (4.5.69)$$

The dynamics are represented using a piecewise quadratic spline interpolation function derived using the Lagrange interpolation [27] method,

$$\mathbf{f}(t) = \begin{cases} \mathbf{f}_k + \delta_k \mathbf{c}_1 + \delta_k^2 \mathbf{c}_2, & t \in [t_k, t_{k+1}] \\ 0, & \text{otherwise} \end{cases} \quad (k = 0, 1, \dots, K) \quad (4.5.70)$$

where,

$$\begin{aligned} \mathbf{c}_1 &= \frac{-1}{t_s} (3\mathbf{f}_k - 4\mathbf{f}_{k+\frac{1}{2}} + \mathbf{f}_{k+1}) \\ \mathbf{c}_2 &= \frac{2}{t_s^2} (\mathbf{f}_k - 2\mathbf{f}_{k+\frac{1}{2}} + \mathbf{f}_{k+1}) \end{aligned} \quad (4.5.71)$$

The state solution is then represented by a cubic Hermite spline interpolate between each of the  $N$  collocation points,

$$\mathbf{x}(t) = \begin{cases} b_{00}(\delta_n) \mathbf{x}_n + b_{10}(\delta_n) \frac{t_s}{2} \dot{\mathbf{x}}_n + b_{01}(\delta_n) \frac{t_s}{2} \mathbf{x}_{n+1} + b_{11}(\delta_n) \frac{t_s}{2} \dot{\mathbf{x}}_{n+1}, & t \in [t_n, t_{n+1}] \\ 0, & \text{otherwise} \end{cases} \quad (4.5.72)$$

for  $(n = 0, 1, \dots, (N - 1))$  where,

$$\begin{aligned} \delta_n &= \frac{2(t - t_n)}{t_s} \\ b_{00} &= 2\delta_n^3 - 3\delta_n^2 + 1 \\ b_{10} &= \delta_n^3 - 2\delta_n^2 + \delta_n \\ b_{01} &= -2\delta_n^3 + 3\delta_n^2 \\ b_{11} &= \delta_n^3 - \delta_n^2 \end{aligned} \quad (4.5.73)$$

### 4.5.9 Estimating the Error in the Solution

It is useful to determine the accuracy of a given solution, as the choice of the transcription method and discretisation (i.e. the chosen grid size) limits the accuracy of the solution. For most numerical methods a way of quantifying the amount of accuracy lost by using a certain discretisation is needed. Several methods exist to approximate the discretisation error in the solution. A relatively simple method is used here as presented in Betts [33]. We want to know the error between the true solution and our interpolated solution. Unfortunately the true solution is unknown, so the assumption is made that the interpolated control solution that was found, is the true solution for the control. We also know that the true solution would satisfy the collocation constraint between all the grid points,

$$\dot{\mathbf{x}}^*(t) = \mathbf{f}(\mathbf{x}^*(t), \mathbf{u}^*(t)) \quad (4.5.74)$$

However, since we made some discretisation, there is some error between the state derivative and system dynamics between grid points and can be represented as,

$$\boldsymbol{\epsilon}(t) = \dot{\tilde{\mathbf{x}}}(t) - \tilde{\mathbf{f}}(\tilde{\mathbf{x}}(t), \tilde{\mathbf{u}}(t)) \quad (4.5.75)$$

where  $\boldsymbol{\epsilon}(t)$  is the discretisation error in the dynamics,  $\tilde{\mathbf{x}}(t)$  is the interpolated state solution and  $\tilde{\mathbf{u}}(t)$  is the interpolated control solution. The interpolated state derivative  $\dot{\tilde{\mathbf{x}}}(t)$  function is calculated using the interpolation methods such as those of Equations 4.5.66 and 4.5.72, depending on the collocation method used. Since it is assumed that the interpolated control is the true solution, we can use it in the system dynamic equations and assume by extension that  $\tilde{\mathbf{f}}$  represents the true dynamics between knot points,

$$\tilde{\mathbf{u}}(t) \approx \mathbf{u}^*(t) \quad (4.5.76)$$

$$\tilde{\mathbf{f}}(\tilde{\mathbf{x}}(t), \tilde{\mathbf{u}}(t)) \approx \mathbf{f}(\mathbf{x}^*(t), \mathbf{u}^*(t)) \quad (4.5.77)$$

We can now compute an estimate of the total error in the state trajectory for each segment, by integrating the absolute value of the error term  $\boldsymbol{\epsilon}(t)$  for each segment in the grid,

$$\boldsymbol{\xi}_k = \int_{t_k}^{t_{k+1}} |\boldsymbol{\epsilon}(t)| \quad (k = 0, 1, \dots, N-1) \quad (4.5.78)$$

where  $\boldsymbol{\xi}_k$  is the total state discretisation error for each segment. The integral in this case is usually calculated using some form of adaptive quadrature. In this case Romberg quadrature is used as suggested in Betts [33]. In the case of Euler collocation the interpolated state solution  $\tilde{\mathbf{x}}(t)$  is piecewise linear, thus the interpolated state derivative  $\dot{\tilde{\mathbf{x}}}(t)$  is a piecewise constant (zero-order hold) function.

### 4.5.10 Using Z-Transform Difference Equations in Direct Transcription Method

When the Euler collocation method was used, the Euler difference equations Equations 4.5.33 to 4.5.35 were replaced by the following z-transform difference equations, as they gave better

numerical stability with the same amount of mesh points,

$$\alpha(k+1) = e^{-t_s/\tau_\alpha} \alpha(k) + (1 - e^{-t_s/\tau_\alpha}) \alpha_c(k) \quad (4.5.79)$$

$$T(k+1) = e^{-t_s/\tau} T(k) + (1 - e^{-t_s/\tau}) T_c(k) \quad (4.5.80)$$

$$P_W(k+1) = e^{-t_s/\tau_P} P_W(k) + (1 - e^{-t_s/\tau_P}) P_{W_c}(k) \quad (4.5.81)$$

Using the above difference equation in the Euler collocation method is not strictly consistent with the method. However, both the Euler difference equations and z-transform difference equation are explicit methods, i.e. uses a previous value of the state to calculate the next value. The z-transform difference equation gives better numerical solutions when the time constant becomes small as in the case of the angle of attack dynamics, meaning less mesh points is required to get an accurate result.

An important distinction between the two discretisation equations is that the z-transform equation is strictly a discrete solution. It only gives the value of the next state at the knot points and no meaningful interpolation can be used between the points. The error estimation technique discussed in the previous section does not apply to the states using the z-transform difference equations. The correct definition of the Euler collocation method is that the state is assumed to be linear between knot points. Direct transcription's solution is a continuous function approximation of the dynamics. Using the z-transform difference equation is more suited to the dynamic programming method due to the explicit nature of the equations.

When an implicit collocation method is used, such as in the case of the Hermite-Simpson method, the z-transform difference equations cannot be used. It was decided that to increase numerical accuracy, a higher-order implicit method should be used, rather than mixing difference methods, to stay consistent with the direct transcription method's definition. Thus for the final implementation of the SQP trajectory optimisation algorithm, the Hermite-Simpson collocation method is used.

The initial numerical solutions obtained using the z-transform variants of the input dynamics used the solution of the dynamic programming method as the initial guess for the SQP routine. The trajectory results mainly matched the dynamic programming results, but this is expected as it was initialised with the solution as the initial guess. Appendix B section §B.2 also shows the numerical results for various dimensional cases of the problem using the Euler method with z-transformed dynamics.

#### 4.5.11 SQP Trajectory Results

Illustrative recovery trajectory results for the SQP algorithm are presented and discussed in this subsection. Several recovery trajectory cases from different upsets are shown ranging from overspeed recovery to simultaneous flight path angle, bank angle and airspeed recovery cases. In each case the SQP solution is compared to the dynamic programming solution, followed by a comparison of recoveries using constant thrust versus varying thrust (with the thrust delay model). Finally a summary and short conclusion on the results are given.

##### Problem Setup

The SQP algorithm's problem constants and variables were setup up as listed in Table 4.3, with the problem bounds similar to those of the dynamic programming setup for consistency.

Instead of a first-order model representing the angle of attack input delay, a second-order model was used, derived from the inner-loop controller's closed-loop short period response. The rationale became apparent after the first trajectory control scheme iteration in §5.4.6. This also highlights the strength of the SQP method, as the model dimensions could easily be expanded as needed. For all the result cases the Hermite-Simpson collocation method is used and thus the trajectory results are continuous in the form of interpolated piecewise third-order Hermite spline state trajectories and interpolated piecewise linear input trajectories. The final time was also included as a design variable, thus the algorithm could in effect vary the time window and the overall sample time between knot points  $t_s$  which remains even for the whole trajectory.

### Constant Thrust SQP vs Dynamic Programming

This subsection provides recovery trajectory results from a range of upset scenarios where the trajectory results of the SQP using constant thrust are compared to the dynamic programming's constant thrust trajectory results.

The thrust response of the GTM aircraft is much slower than the angle of attack and roll rate responses. Thus for the dynamic programming method, we cannot necessarily assume time scale separation between the thrust input response and the translational dynamics using a sampling time of  $\Delta t = 1$  second. For this reason, the thrust is kept constant for the recovery in the implementation of the dynamic programming method. Although the SQP method can model the thrust input delay and use a varying thrust input, it is decided to use a constant thrust value in the SQP method in order to fairly compare the SQP recovery trajectory results with those of the dynamic programming method.

The upset scenarios presented include flight path angle, bank angle, overspeed and underspeed upsets and a combination of these upsets. Recovery trajectories from the following upset scenarios are given:

- Flight path angle, steep decent upset
- Flight path angle, steep climb upset
- Bank angle upset
- Underspeed upset
- Overspeed upset
- Flight path angle and bank angle, steep decent upset
- Flight path angle, bank angle, steep decent and overspeed upset
- Flight path angle, bank angle, steep climb and underspeed upset

For each of these upset scenarios, the recovery strategy of the trajectory results is discussed. The initial angle of attack for all the result cases was set to a trim value of  $\alpha = 3^\circ$  and the initial roll rate is set to zero, with the reasoning that an aerodynamic envelope recovery system is already in place that would have recovered the angle of attack back to a normal value inside the aerodynamic envelope and would have recovered the angular rates to near zero. The thrust for all cases was kept constant at a trim thrust value. Overall the SQP algorithm follows almost exactly the same recovery strategies as the dynamic programming algorithm, but in most cases loses slightly less altitude. This is mainly due to the SQP's ability to use continuous state and



input values and vary the fixed sample time between intervals, allowing it to recover the states more quickly.

Table 4.3: Problem parameter values

Parameter	Symbol	Value	Units
<i>Physical constants</i>			
Mass	$m$	23.59	kg
Gravitational constant	$g$	9.81	m/s <sup>2</sup>
Wing surface area	$S$	0.548	m <sup>2</sup>
Dynamic pressure	$\rho$	1.225	kg/m <sup>3</sup>
Aerodynamic lift Coefficient	$C_L$	Figure 4.3	-
Aerodynamic drag Coefficient	$C_D$	Figure 4.3	-
Angle of attack 2nd order model	$\mathbf{A}_\alpha, \mathbf{B}_\alpha$	§5.4.6	-
Engine lag time constant	$\tau$	2.5	seconds
Roll rate lag time constant	$\tau_P$	0.3	seconds
<i>State bounds</i>			
Velocity	$\bar{V}_u, \bar{V}_l$	145, 20	kn
Final velocity	$\bar{V}_{fu}, \bar{V}_{fl}$	120, 75	kn
Flight path angle	$\gamma_u, \gamma_l$	30, -80	degrees
Bank angle	$\Phi_u, \Phi_l$	180, 0	degrees
Angle of attack	$\alpha_u, \alpha_l$	21, 0	degrees
Roll rate	$P_{W_u}, P_{W_l}$	0, -30	degrees/second
Throttle state	$\delta_{Tu}, \delta_{Tl}$	100, 0	%
Load factor	$n_{L_{\max}}, n_{L_{\min}}$	2.5, -1	g
Final flight path angle	$\gamma_f$	0	degrees
Final bank angle	$\Phi_f$	0	degrees
<i>Input bounds</i>			
Angle of attack command	$\alpha_{c_u}, \alpha_{c_l}$	$\infty, -\infty$	degrees
Roll rate command	$P_{W_{c_u}}, P_{W_{c_l}}$	0, -30	degrees/second
Throttle command	$\delta_{T_{c_u}}, \delta_{T_{c_l}}$	100, 0	%
Roll envelope	$P_{W_u}(\alpha)$	Figure 4.5	degrees/second
<i>Time</i>			
Final time bounds	$t_{KU}, t_{KL}$	30, 1	seconds
Number of time steps (intervals)	$K$	14	-
Number of mesh/grid points	$N = 2K + 1$	29	-

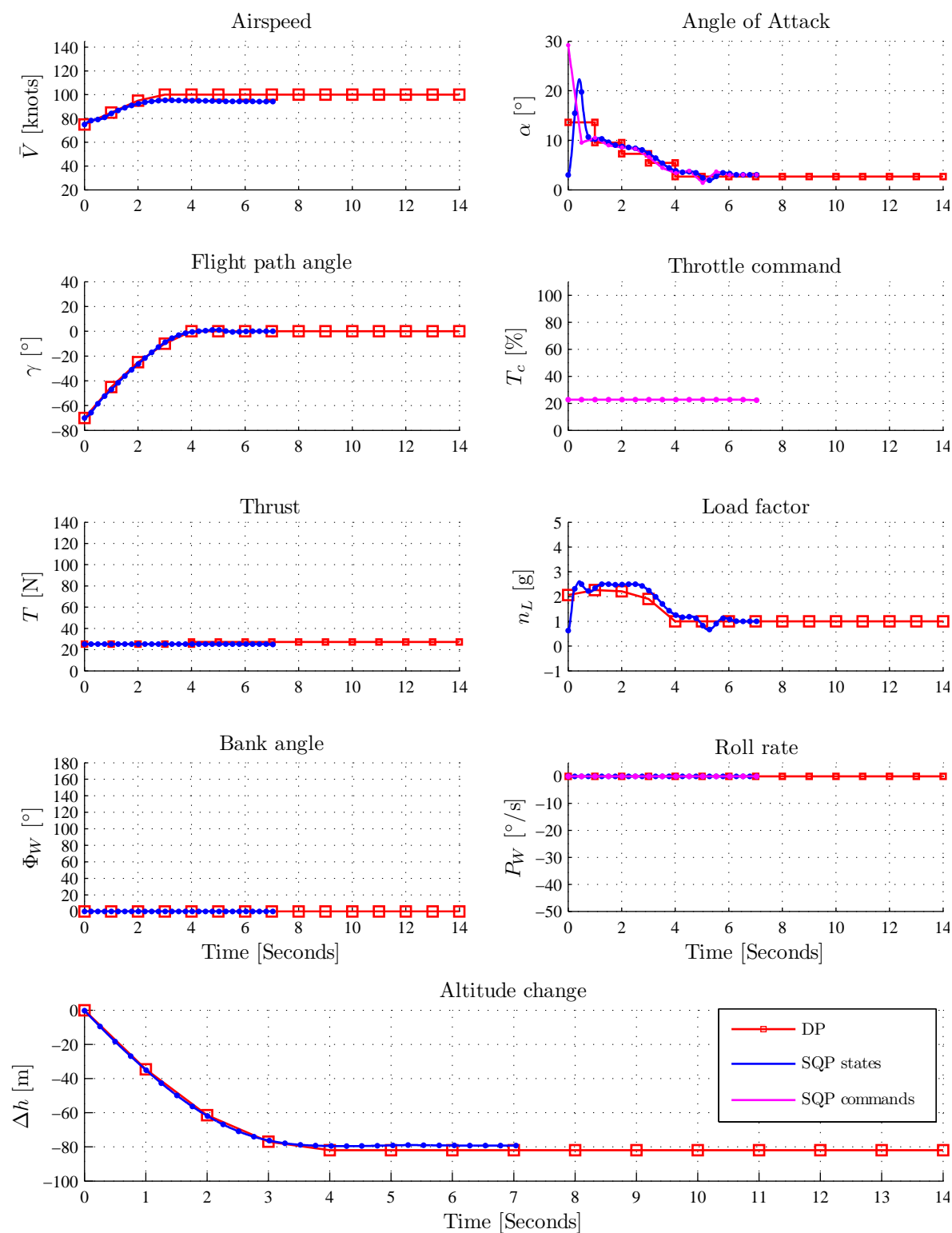


Figure 4.17: SQP vs DP for step flight path angle descent upset

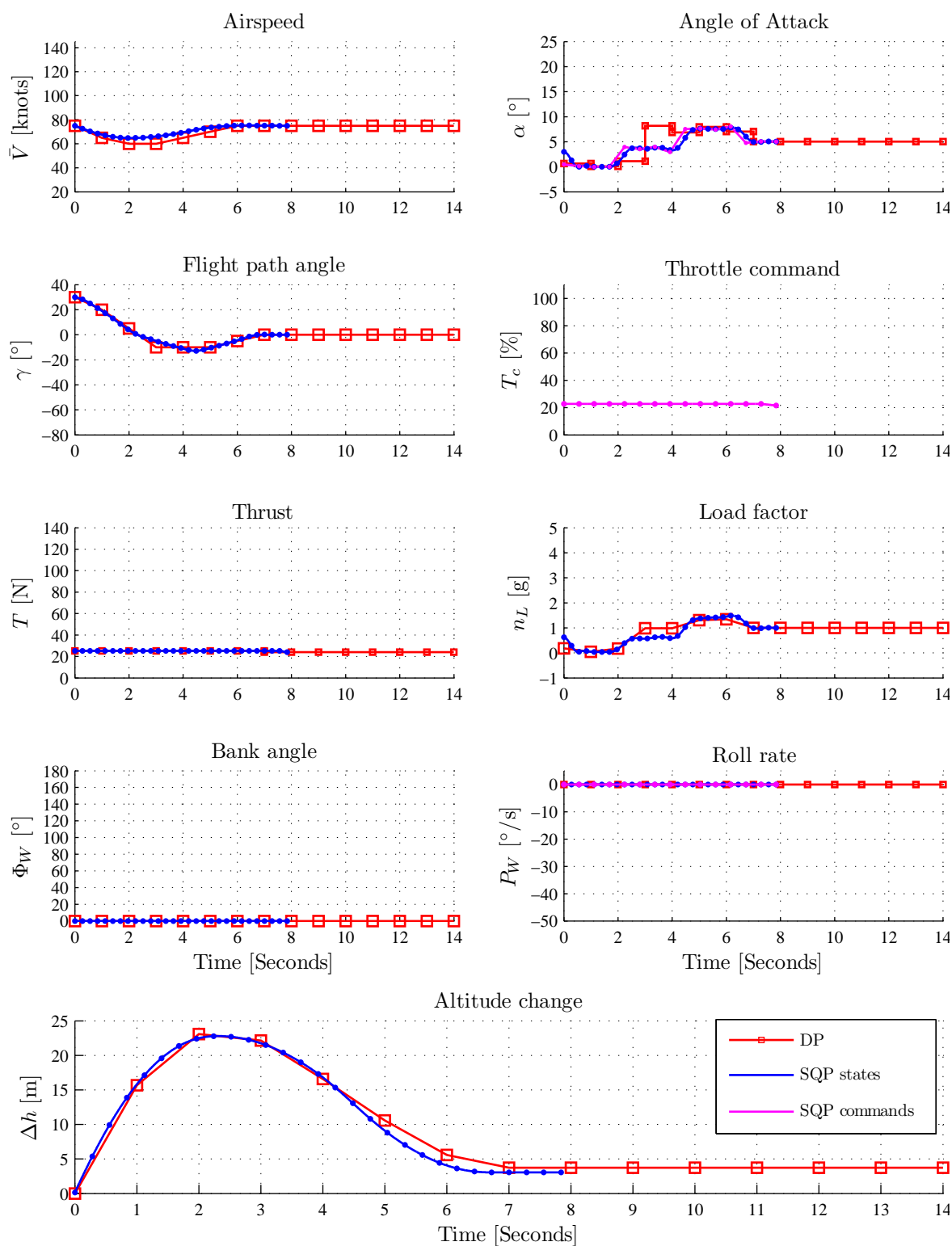


Figure 4.18: SQP vs DP for step flight path angle climb upset

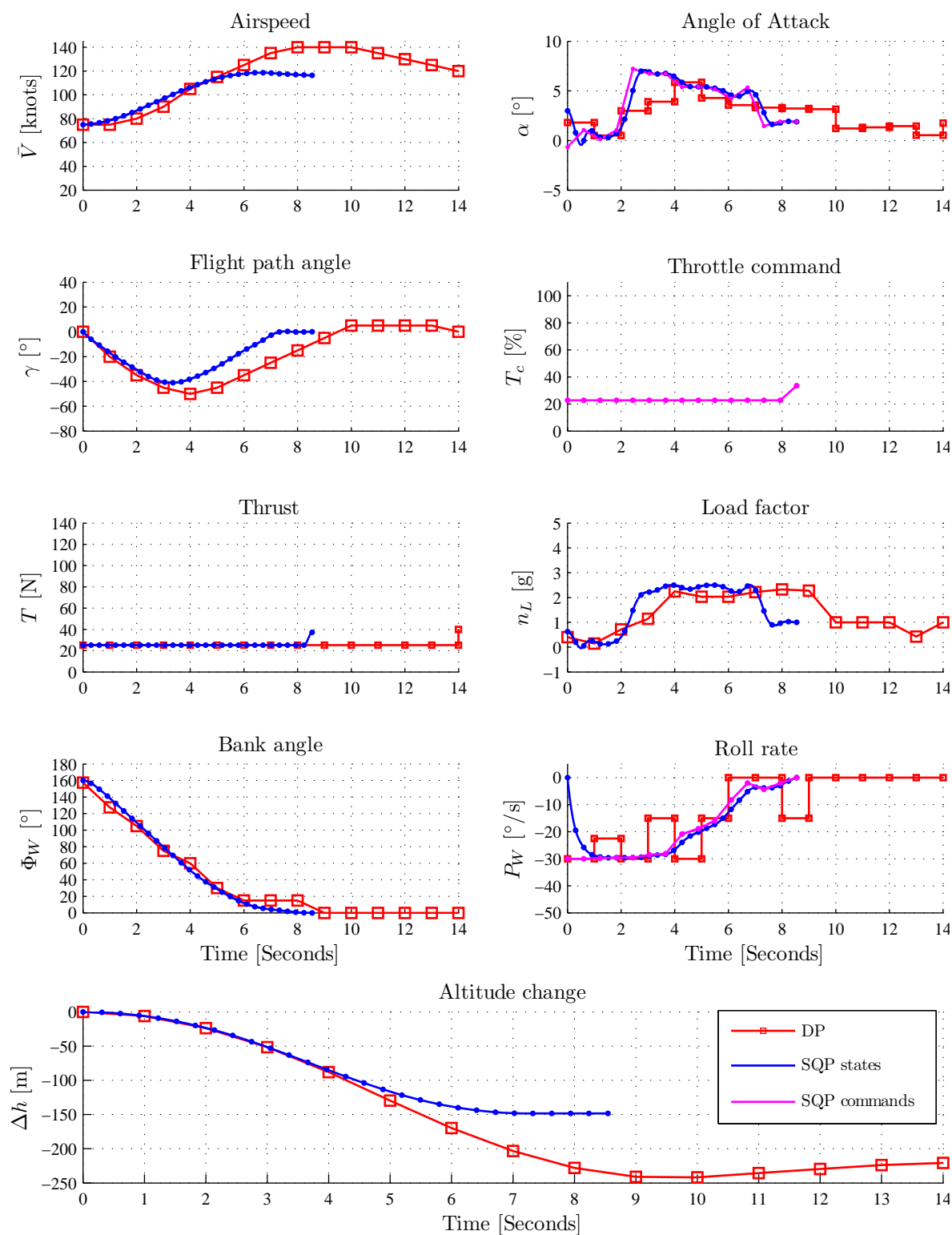


Figure 4.19: SQP vs DP for bank angle upset

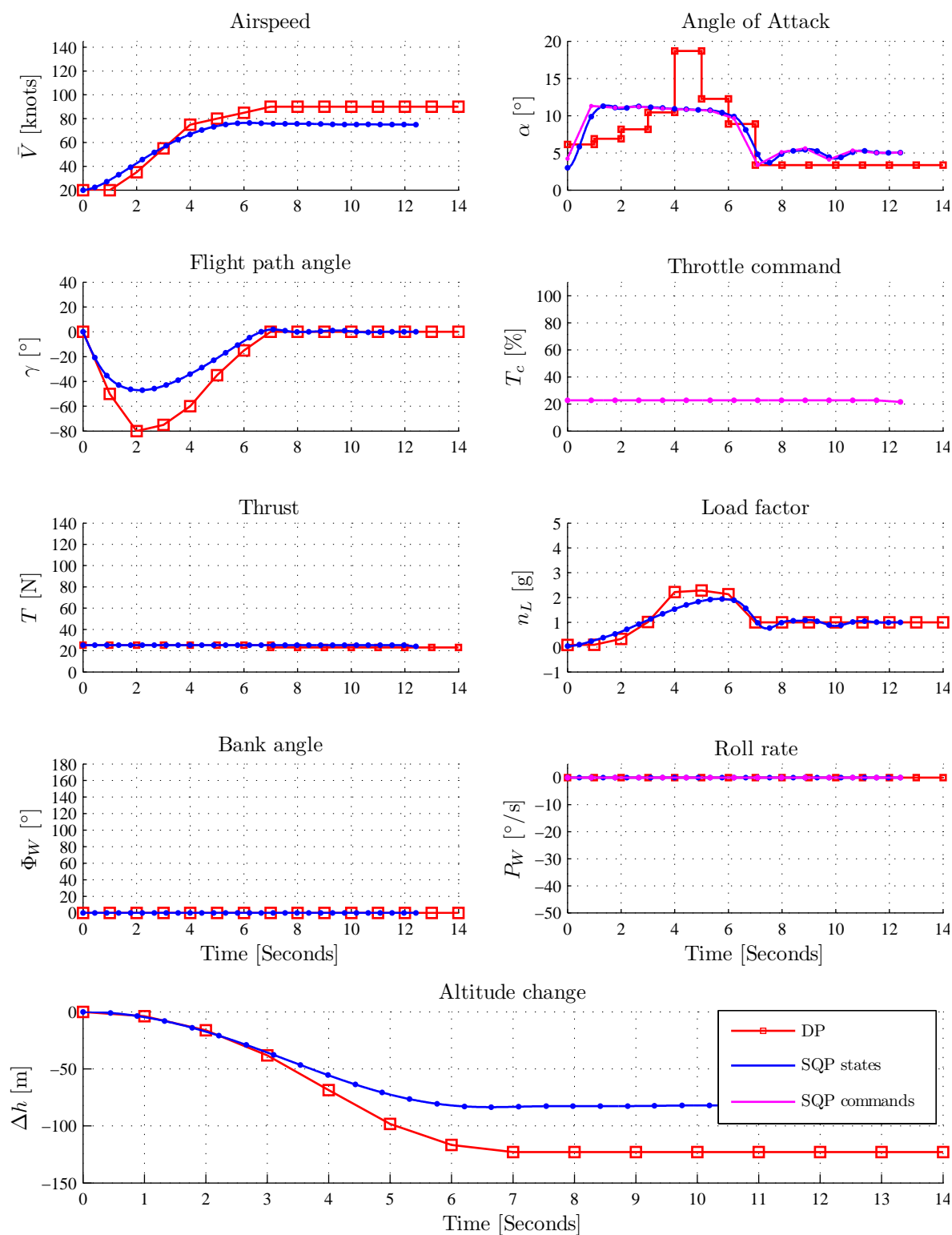


Figure 4.20: SQP vs DP for underspeed upset

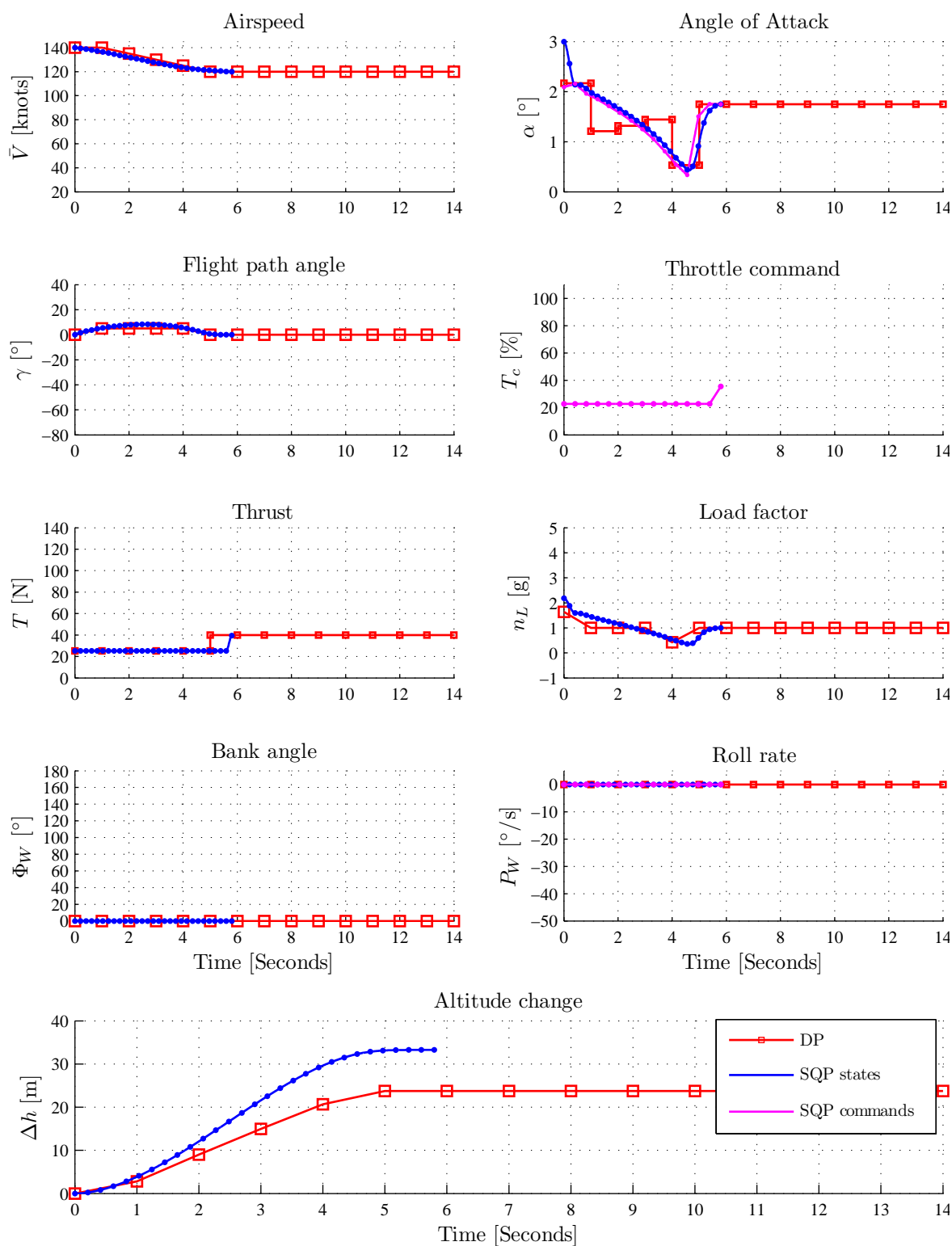


Figure 4.21: SQP vs DP for overspeed upset

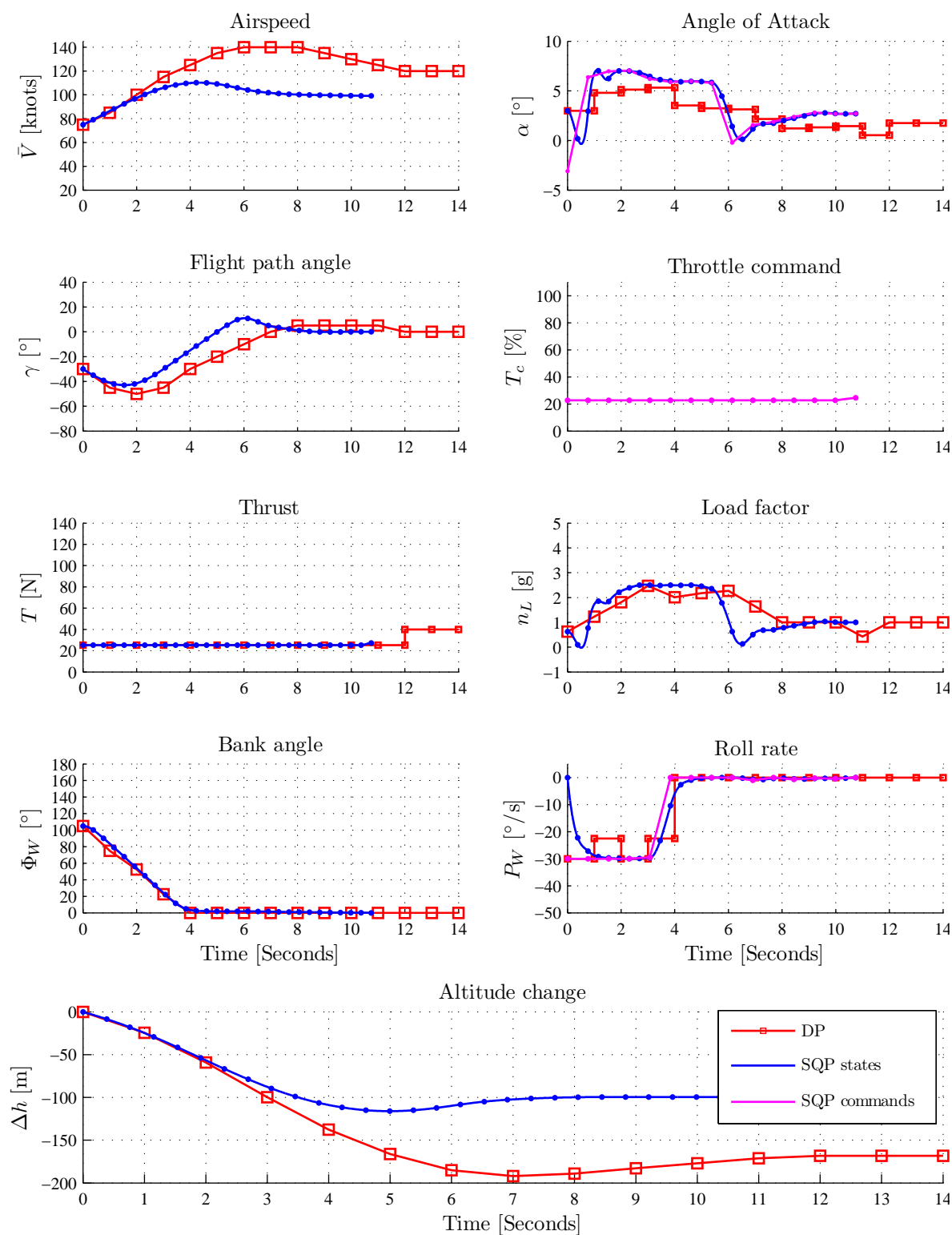


Figure 4.22: SQP vs DP for flight path angle step descent and bank angle upset

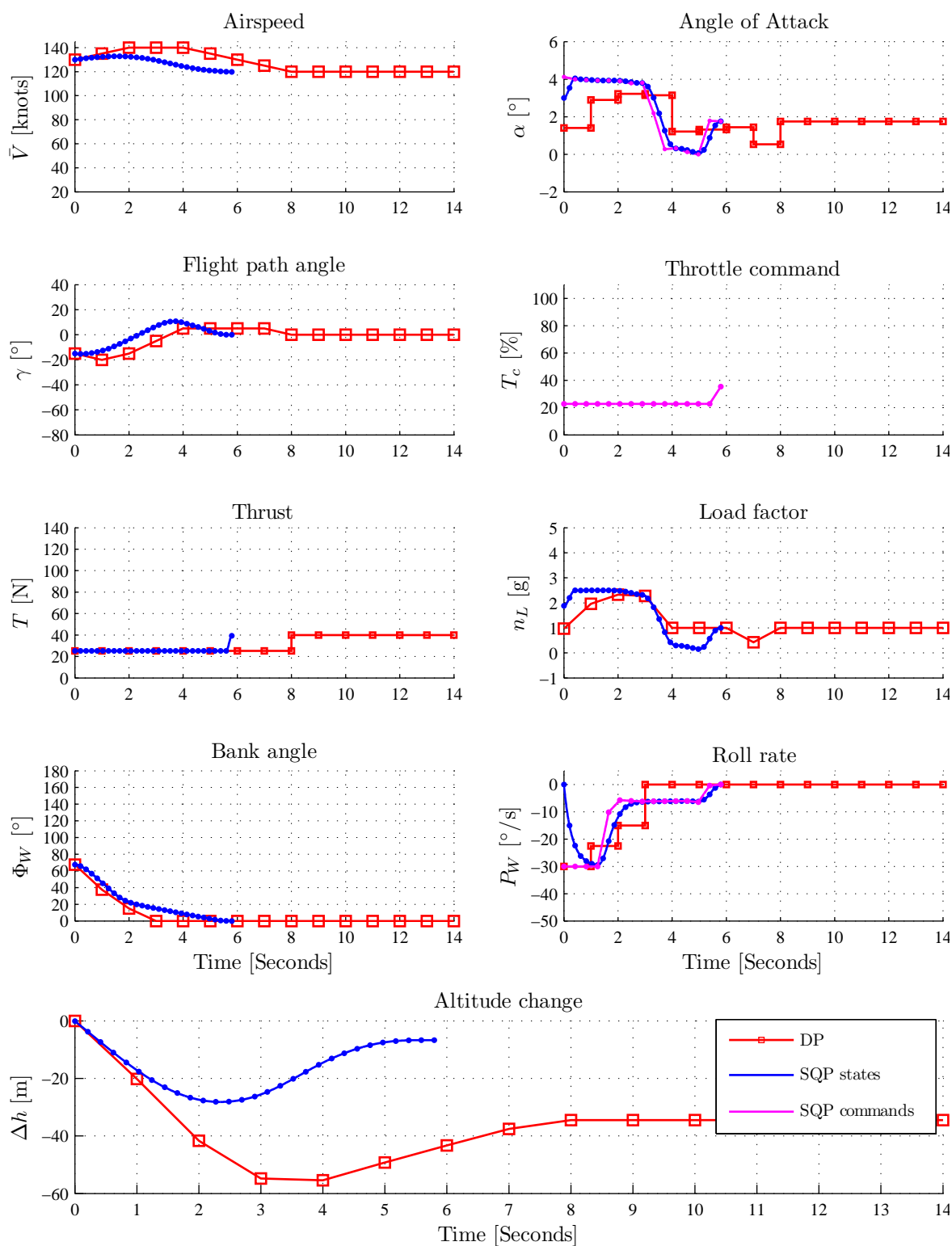


Figure 4.23: SQP vs DP for flight path angle step descent with bank angle in overspeed upset



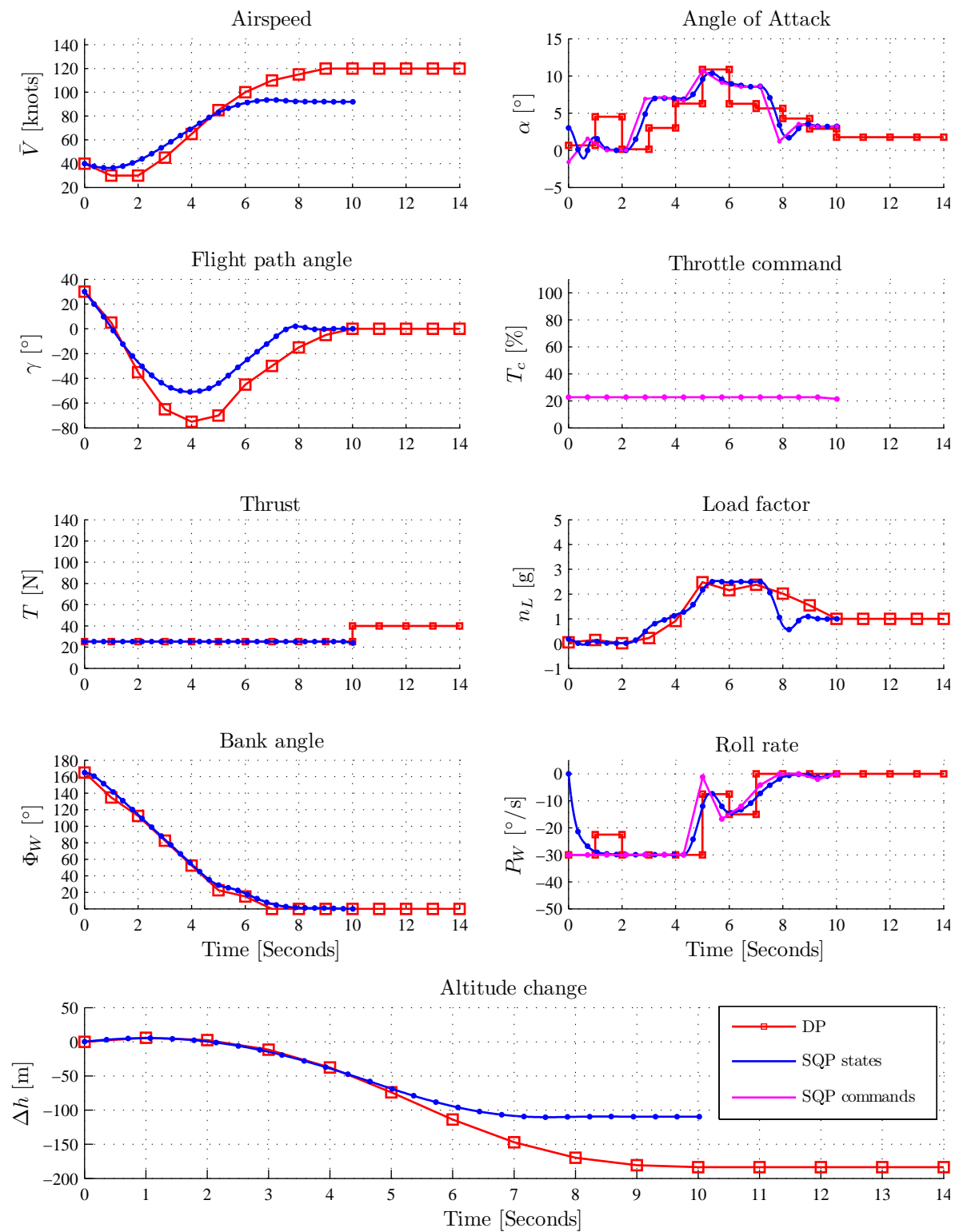


Figure 4.24: SQP vs DP for flight path angle step climb with bank angle in underspeed upset

**Flight path angle, steep decent upset** - For this upset scenario the aircraft is initialised with a steep descending flight path angle of  $\gamma = -70^\circ$  at a nominal trim airspeed at a zero bank angle attitude. The SQP recovery algorithm immediately issues a hard ‘pull-up’ command, i.e. a large angle of attack command, as large as the normal load factor limit allows for, to recover the flight path angle as quickly as possible to level flight  $\gamma = 0^\circ$ . The airspeed increases during the recovery due to the negative flight path angle, as a large component of gravity is accelerating the velocity vector of the aircraft. The aircraft recovers at about 4 seconds with a maximum altitude loss of 80 meters. The SQP algorithm reaches a trim state at roughly 7 seconds.

**Flight path angle, steep climb upset** - For this upset scenario the aircraft is in a steep climb with a flight path angle of  $\gamma = 30^\circ$  starting at a nominal trim airspeed and a zero bank angle attitude. The SQP algorithm immediately gives the lowest angle of attack command allowed to try and decrease the flight path angle. The airspeed decreases initially due to the component of gravity accelerating the aircraft in the opposite direction of the velocity vector, i.e. downwards. The airspeed decreases until the flight path angle reaches zero at 2 seconds before starting to increase again as the flight path angle goes negative. The flight path angle dips below zero degrees during the recovery to help recover the lost airspeed back to a safe minimum allowed final value of 75 knots. The airspeed and flight path angle are recovered simultaneously to their final values at level flight. The recovery does not lose any altitude but instead gains an amount of 4 meters of altitude due to the positive flight path angle.

**Bank angle upset** - This upset scenario starts with a level flight path angle of  $\gamma = 0^\circ$  at a nominal trim airspeed with a near fully inverted bank angle attitude of  $\Phi_W = 160^\circ$ . The SQP algorithm immediately gives the maximum allowed negative roll rate command in order to recover the bank angle below 90 degrees as quickly as possible, because while the aircraft is inverted the lift vector cannot be used to positively affect the flight path angle and counter gravity. Due to this, the flight path angle dips to negative values as the bank angle is recovered. The angle of attack is kept low while the aircraft is above 90 degrees bank angle, as giving a positive angle of attack value while the aircraft is ‘upside down’ will only increase the flight path angle’s descent rate. At around 2 seconds, the SQP pre-emptively gives a hard ‘pull-up’ command, i.e. the maximum angle of attack command allowed by the load factor limit and roll rate envelope, so that the maximum angle of attack state is reached just as the aircraft transitions to below 90 degree bank angle at roughly 3 seconds. As soon as the aircraft transitions below 90 degrees bank angle, the lift vector can once again positively affect the flight path angle. SQP keeps the angle of attack as high as possible to recover the flight path angle after 3.5 seconds, while still recovering the rest of the bank angle to wings level as quickly as possible. There is a roll rate envelope constraint active that constraints the maximum allowed roll rate as a function of angle of attack, so the SQP balances the angle of attack and roll rate to recover the flight path angle and bank angle as optimally as possible. The airspeed increases during the recovery due to the negative flight path angle, but is kept within safe limits during recovery.

The dynamic programming algorithm does not give minimum angle of attack commands while the aircraft is upside down, most likely because it must try to reach an admissible airspeed quantised value using a certain amount of drag. The same reasoning can be applied to the variations in the roll rate commands compared to the SQP algorithm. With no rate limits on the roll rate, the dynamic programming simply has to ensure the integral of the bank angle decreases as quickly as possible using an arbitrary sequence of roll rate commands. Dynamic programming reaches its first bank angle value below 90 degrees at about the same time as the SQP. The SQP loses significantly less altitude than the dynamic programming in this case, losing 150 meters versus dynamic programming’s almost 250 meters and the SQP recovers more quickly at roughly 9 seconds versus dynamic programming’s 14 seconds. This is most

likely due to dynamic programming being limited by its state quantisation. All 3 major states play a role in altitude loss in this recovery (airspeed, flight path angle and bank angle), and by varying the states continuously, the SQP algorithm can achieve a more optimal solution to this upset case.

**Underspeed upset** - This upset scenario starts at an airspeed below the stall speed at 20 knots with a level flight path angle and zero bank angle attitude. At this low airspeed the aircraft cannot produce enough lift to keep the flight path angle level and it starts to dip immediately. The SQP simultaneously recovers the flight path angle and airspeed, using negative flight path angle and gravity to recover the airspeed. SQP also does not use maximum angle of attack commands to recover the flight path angle as it does not want to hinder the airspeed recovery with excessive amounts of drag. As the airspeed recovers, more lift can be produced and at around 2 seconds, with the airspeed transitioning past 45 knots, the flight path angle begins to increase again. Although both the SQP and dynamic programming recovers within 7 seconds, the SQP recovers with less altitude loss than the dynamic programming algorithm due to the same reasoning in the bank angle upset case, namely due to the state and time quantisation resolution limitations of dynamic programming. The SQP loses roughly 75 meters of altitude, while the dynamic programming loses roughly 125 meters.

**Overspeed upset** - This scenario starts at the maximum admissible unsafe airspeed value of 140 knots with a level flight path angle and a bank angle attitude of zero degrees. As the algorithm recovers the airspeed back to the maximum safe final airspeed of 120 knots, the flight path angle climbs slightly due to the lift generated by the excess airspeed. The angle of attack is slowly dropped to counter the rise in flight path angle, while still producing enough drag to slow the aircraft down. The dynamic programming recovers in 5 seconds with the SQP recovering one second later at 6 seconds due to the modelled delay in the angle of attack input. This manoeuvre does not lose any altitude with the SQP gaining slightly more altitude than the dynamic programming at roughly 35 meters gained versus dynamic programming's 24 meters.

**Flight path angle and bank angle, steep decent upset** - This upset scenario illustrates a simultaneous flight path angle and bank angle recovery, with an initial descending flight path angle of  $\gamma = -30^\circ$ , an inverted bank angle attitude of  $\Phi_W = 105^\circ$  and a starting nominal airspeed value. The SQP prioritises bank angle recovery by immediately giving and maintaining the maximum allowed roll rate command until the bank angle is recovered. The airspeed increases during the recovery due to the negative flight path angle. The negative flight path angle grows initially as the lift vector cannot affect it positively at bank angles higher than 90 degrees. As with the bank angle recovery case, the SQP gives the highest 'pull-up' command in angle of attack allowed by either the load factor limit or roll rate envelope as the aircraft transitions below 90 degrees bank angle to start recovering the flight path angle.

The SQP uses a higher angle of attack than the dynamic programming between 1 and 6 seconds, possibly because it can use continuous flight path angle states to transition to, while dynamic programming is forced to use angles of attack that will transition it to the next most ideal, but reachable quantised flight path angle value. Due to the higher angle of attack trajectory in the SQP recovery, more drag is produced, the flight path angle is transitioned to a less negative value and less airspeed is gained as a result, preventing the aircraft from going into an overspeed condition (airspeed above 120 knots) as in the dynamic programming's case. Both the SQP and dynamic programming algorithms prioritise bank angle recovery, but the dynamic programming does not maintain maximum roll rate between 0 and 3 seconds. This is because the dynamic programming has to use the roll rate to transition between discrete bank angle states and has no roll rate delay. It must discretely integrate the bank angle, and the 'notch' in dynamic programming's roll rate command is a product of this. It can be seen that both the SQP and dynamic programming end up at the same bank angle state at 2 seconds. The SQP

loses roughly 110 meters of altitude while the dynamic programming loses roughly 190 meters. The SQP loses less altitude than the dynamic programming, because of its lower airspeed trajectory and faster flight path angle recovery during the recovery, meaning the altitude rate is less negative.

**Flight path angle, bank angle, steep decent and overspeed upset** - This scenario puts the aircraft in an initial condition with a descending flight path angle of  $\gamma = -15^\circ$ , a bank angle attitude close to  $\Phi_W = 70^\circ$  and in overspeed with an airspeed of 130 knots. The SQP recovers the airspeed, flight path angle and bank angle simultaneously, and uses a higher angle of attack than the dynamic programming between 0 and 3 seconds, because it is not limited to discrete flight path angle values as discussed in the previous upset case. Maximum allowed angle of attack is immediately given unlike in the previous case, because we are already below 90 degrees bank angle, so the flight path angle can immediately be positively affected by the lift vector.

The SQP does not prioritise recovering the bank angle to exactly zero degrees unlike the dynamic programming, because there is no tertiary cost in the SQP algorithm that minimises the bank angle integral. This difference plays a role in this case, because at around 2 seconds, the SQP has already lost maximum altitude and has recovered the flight path angle to above zero degrees, while the bank angle is still non-zero. The SQP now only has to recover the airspeed, and has no incentive to immediately suppress the bank angle to zero if enough lift is being generated while at positive flight path angles and recovering airspeed. The SQP loses less altitude than the dynamic programming at 28 meters lost versus dynamic programming's 55 meters lost.

**Flight path angle, bank angle, steep climb and underspeed upset** - This scenario starts with a positive climbing flight path angle of  $\gamma = 30^\circ$  with an inverted bank angle attitude of  $\Phi_W = 165^\circ$  while in underspeed with an airspeed of 40 knots. The SQP once again follows a similar strategy to the dynamic programming, where the bank angle recovery is prioritised. While the aircraft is upside down, the lift vector cannot counter gravity and the flight path angle immediately starts to decrease. Between 0 and 2 seconds the SQP keeps the angle of attack low, because the lift vector is pointed downwards. Just as the aircraft transitions to a bank angle below 90 degrees at 3 seconds a 'pull-up' command is given using the maximum allowed angle of attack by the roll rate envelope to recover the flight path angle. The roll rate is reduced after 4 seconds and a higher angle of attack can be given due to the roll rate envelope. The airspeed is recovered simultaneously with the flight path angle.

Both the SQP and dynamic programming recover at the same time at 10 seconds. The larger angle of attack commanded by the dynamic programming at 1 second is most like due to the discrete flight path angle state limitation of the dynamic programming. The SQP also recovers with a lower maximum final airspeed than the dynamic programming. The SQP loses less altitude than the dynamic programming at 100 meters lost versus the dynamic programming's 184 meters. This is likely due to the SQP's advantage of using continuous input and state values.

### SQP Varying Thrust vs Constant Thrust

Using the same upset scenarios as in the previous subsection, we compare the SQP's recovery trajectories using a constant thrust value and varying thrust values, to investigate the potential advantage of using the thrust input while modelling the thrust delay in the recovery. The varying thrust implementation models the thrust input as a first-order delay. Figures 4.25 to 4.32 show the SQP recovery trajectories comparing the constant thrust and varying thrust cases for each of the upset scenarios of the previous subsection.

There is very little difference between the recovery trajectories produced by the constant thrust approach and the recovery trajectories produced by the variable thrust approach. Overall, no real advantage is gained from using the varying thrust input in terms of total altitude lost in the cases presented. The acceleration produced by the thrust is too low to significantly change the airspeed over the relatively short duration of the recovery manoeuvre. The thrust actuator is therefore not able to make an effective contribution to the airspeed recovery. The only significant advantage gained from using the varying thrust was seen in the ‘underspeed only’ case, which makes intuitive sense as airspeed must be recovered as soon as possible, and with extra axial acceleration from increased thrust, the recovery process is sped up. However, the throttle inputs tend to aggressively transition between minimum and maximum values during recovery and in practice this behaviour is undesirable, as it puts unnecessary strain on the jet engines.

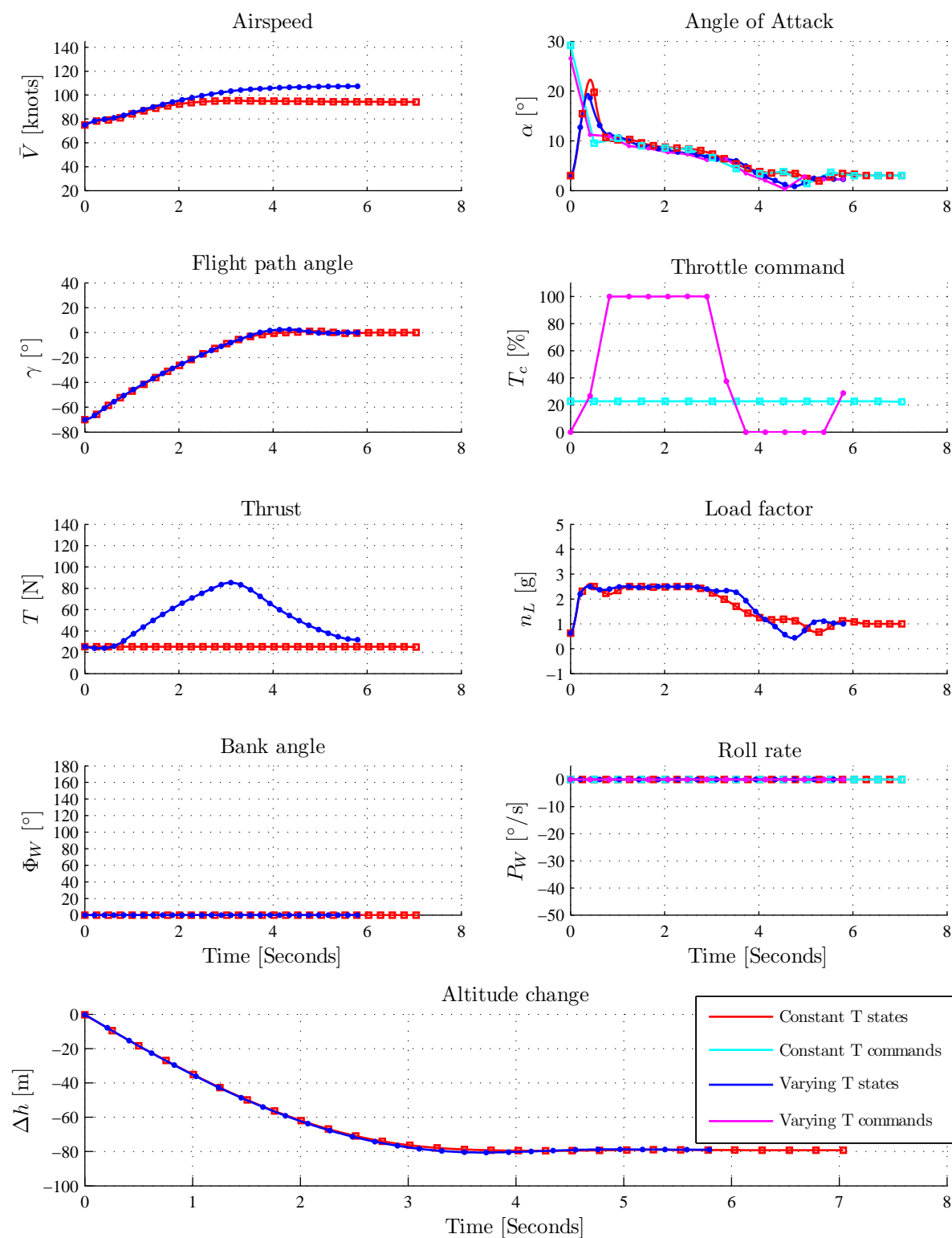


Figure 4.25: SQP vs DP for step flight path angle descent upset

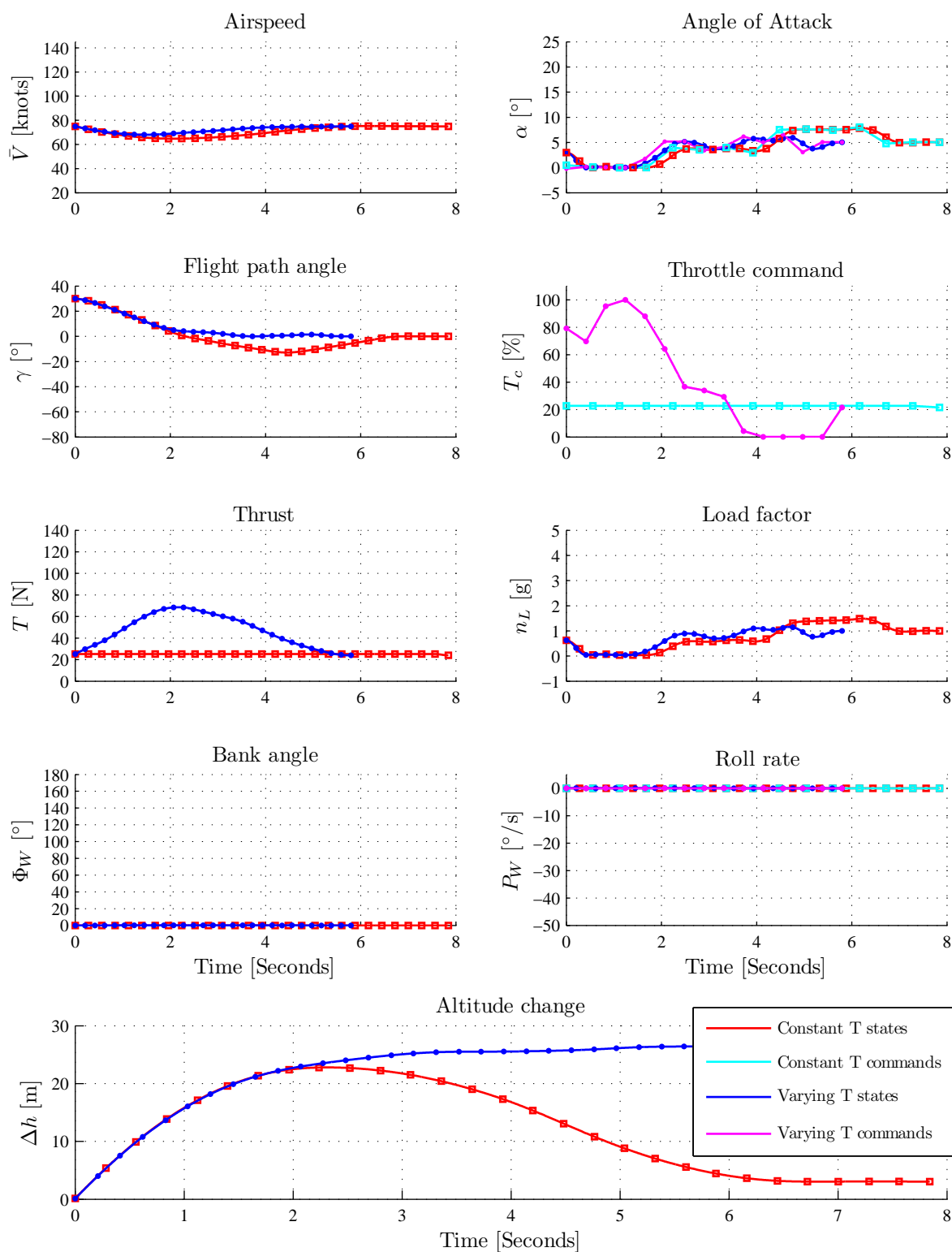


Figure 4.26: SQP vs DP for step flight path angle climb upset

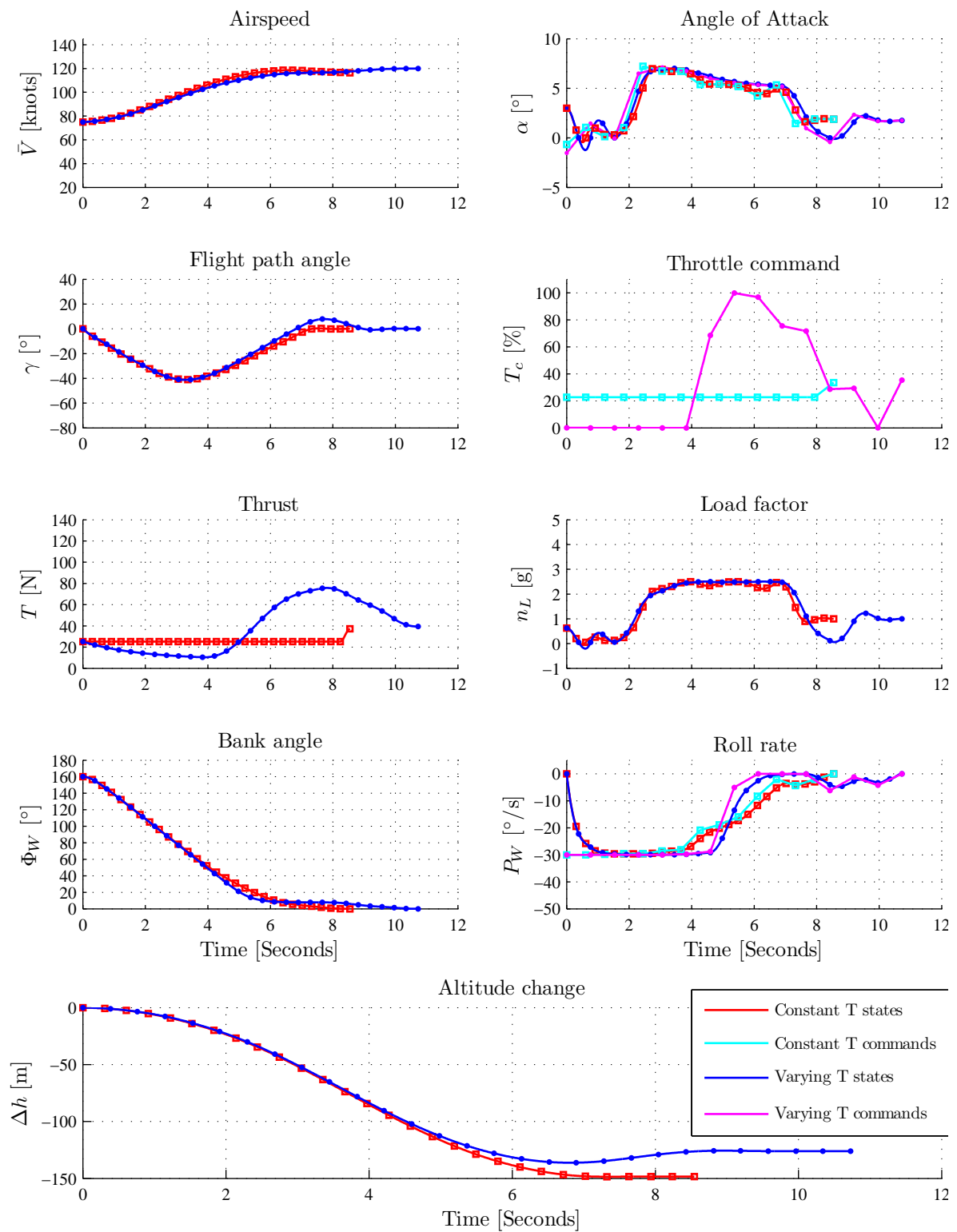


Figure 4.27: SQP vs DP for bank angle upset



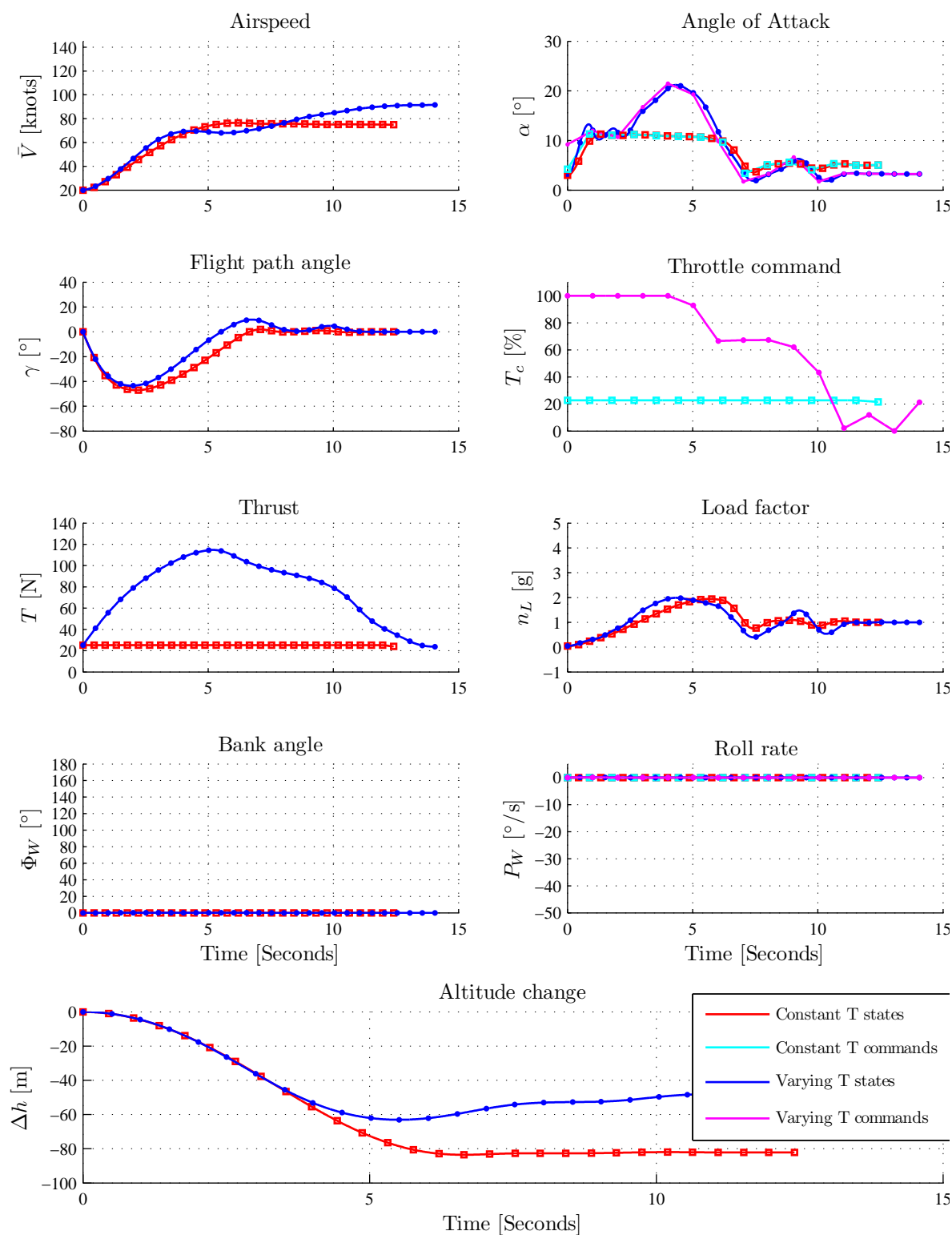


Figure 4.28: SQP vs DP for underspeed upset

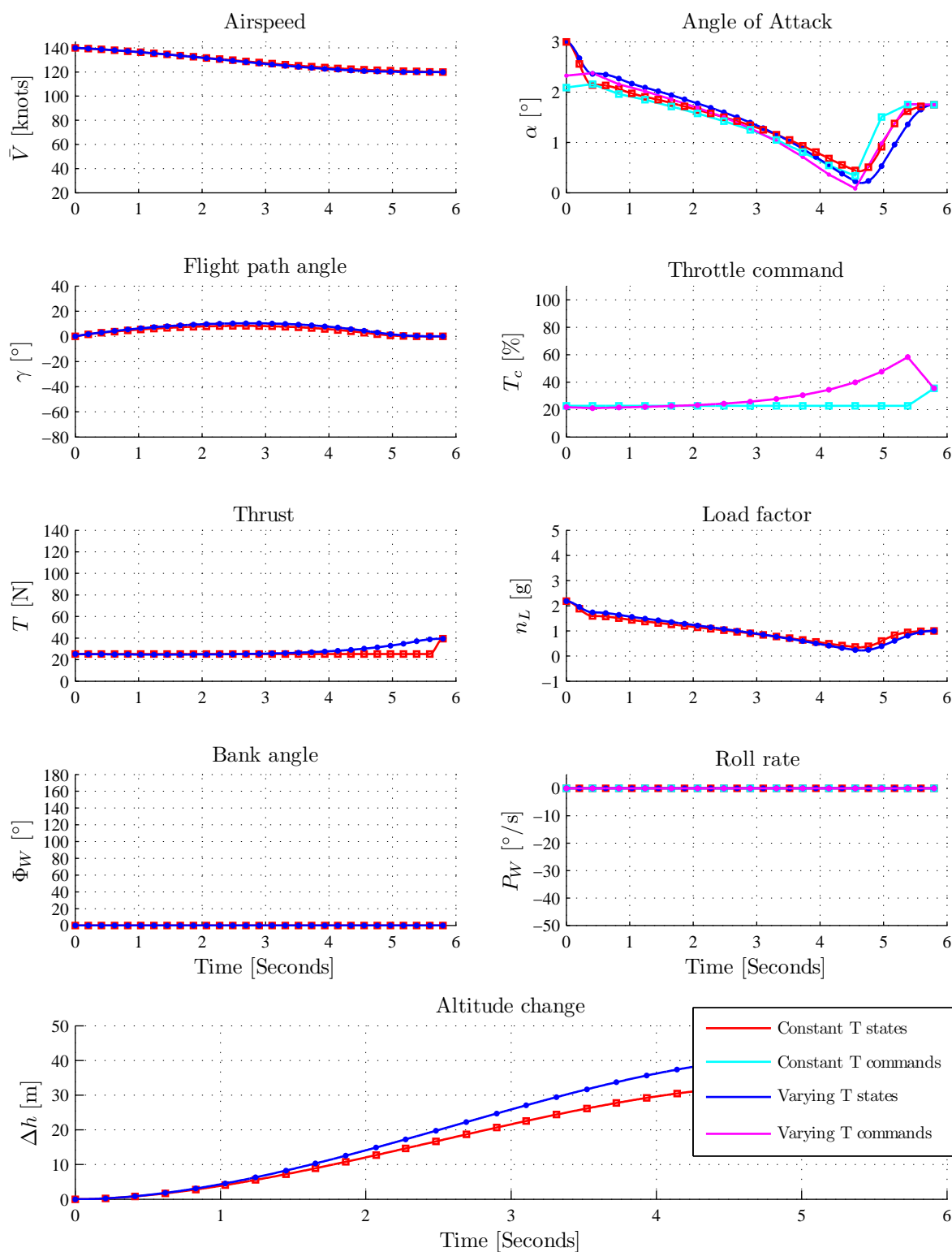


Figure 4.29: SQP vs DP for overspeed upset

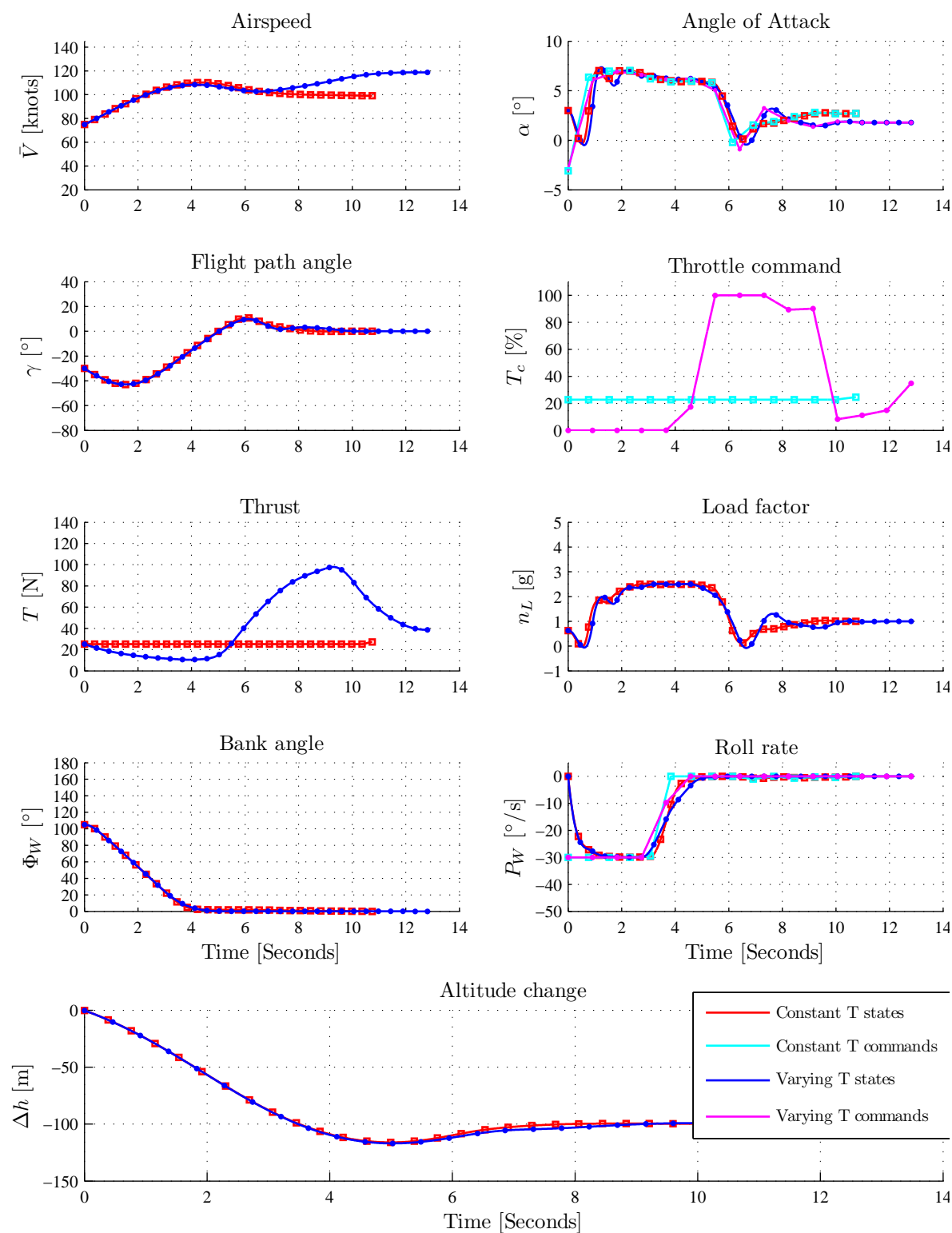


Figure 4.30: SQP vs DP for flight path angle step descent and bank angle upset

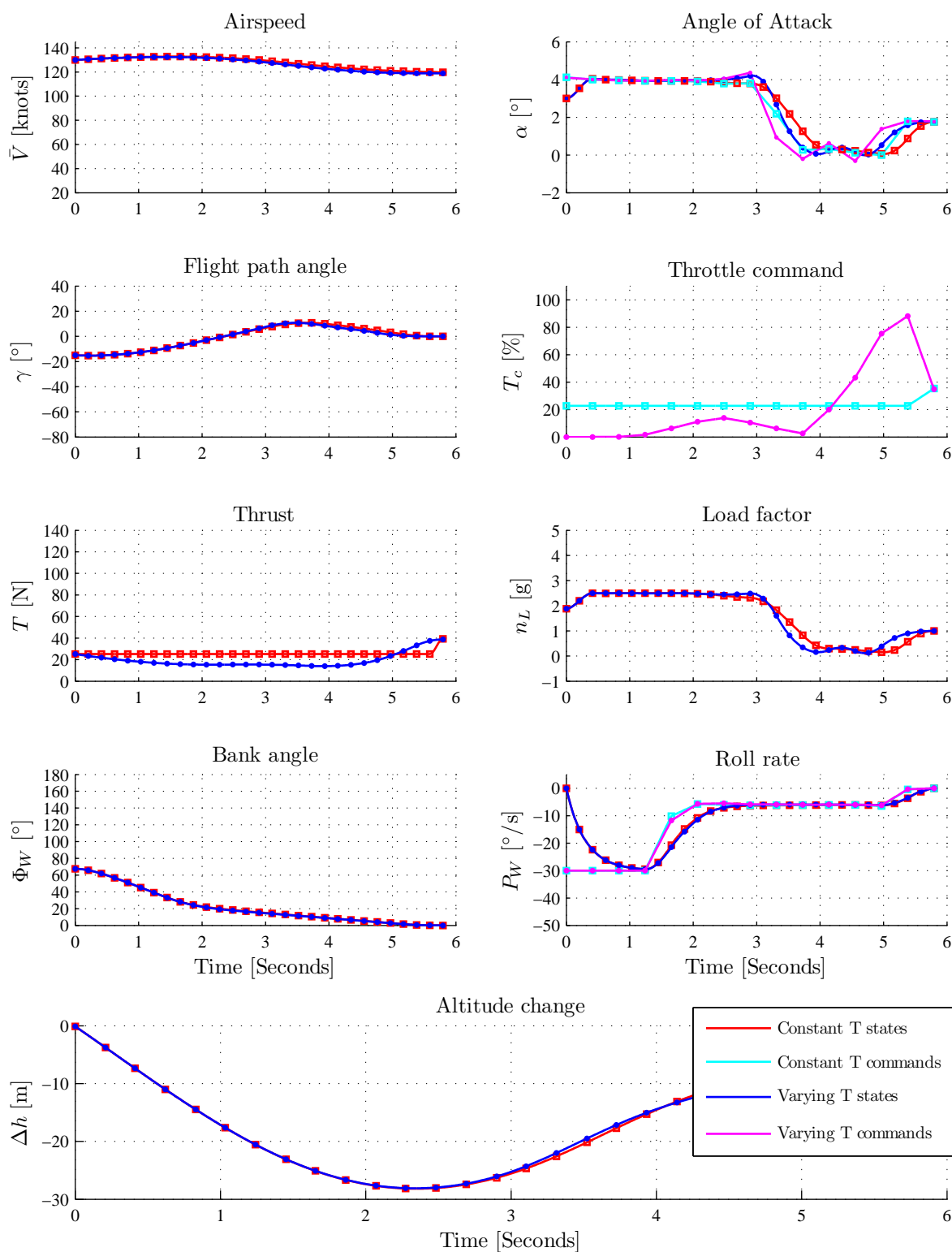


Figure 4.31: SQP vs DP for flight path angle step descent with bank angle in overspeed upset

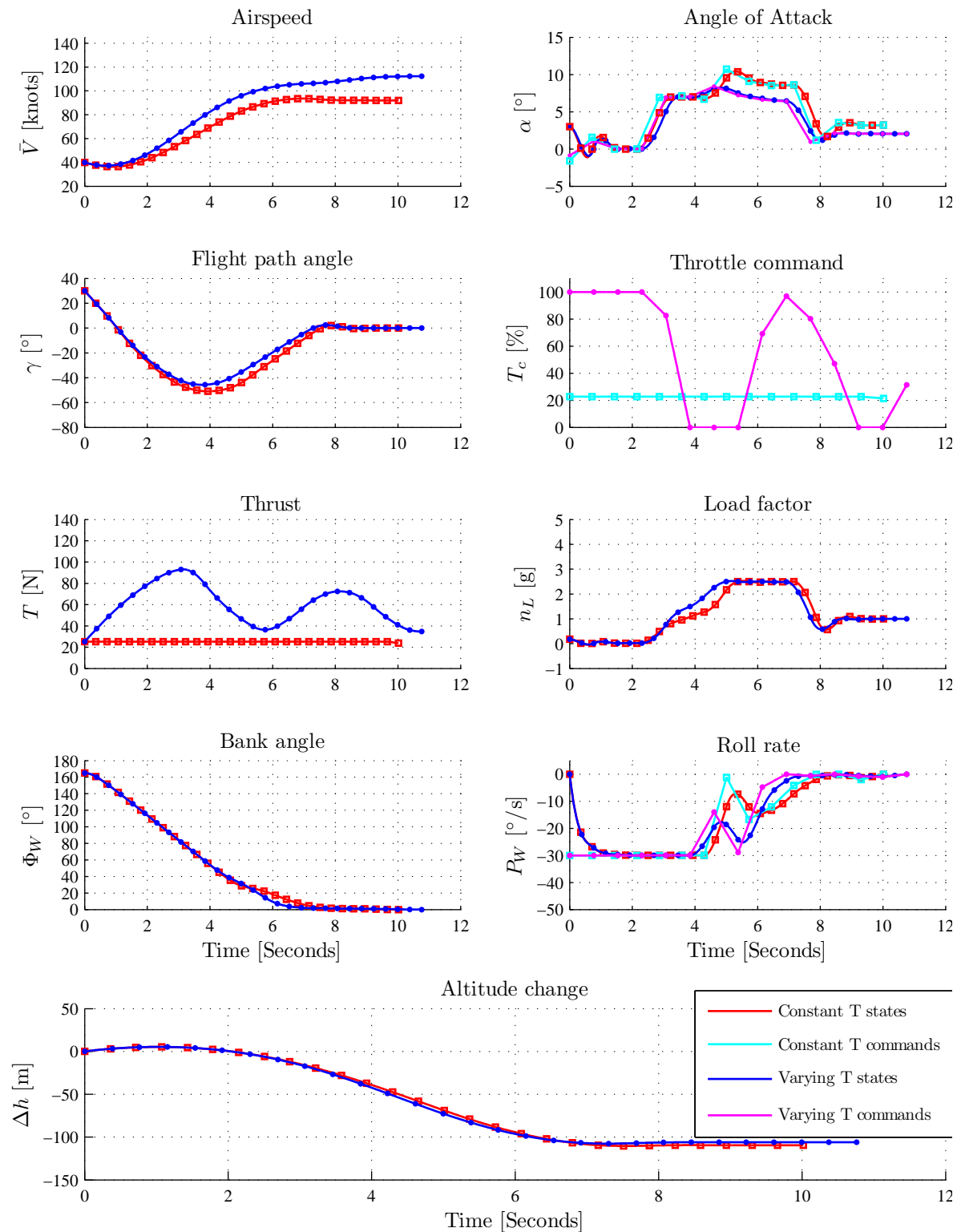


Figure 4.32: SQP vs DP for flight path angle step climb with bank angle in underspeed upset

The estimated numerical errors were calculated for the eight SQP upset recovery scenarios for the thrust varying cases illustrated here (using the equations discussed in §4.5.9). The calculated error estimates for the recovery trajectories are generally sufficiently small, with larger errors present at knot points where the dynamics are changing quickly. However this is to

be expected as numerical integration errors are magnified with large changes in the dynamics over an interval. Appendix B section §B.1 gives the numerical error estimate plots showing the calculated numerical estimates. These error estimates are mainly rough estimates, as they assume the NLP has found a good solution and use the reduced-order dynamics in the calculations. The trajectory execution simulations presented in Chapter 5 are a much more accurate validation method for the numerical solutions than the error estimate calculations. This is due to the fact that the executed trajectories are simulated using the full non-linear aircraft model, and these executed trajectory results are then compared to the planned trajectories.

### General Observations

It was found that the SQP algorithm was very sensitive to the initial final-time guess that is passed to the SQP function, i.e. the value of the  $t_f$  variable guess initially passed before execution. In most cases, if the final-time guess was lower than the minimum time the aircraft could be expected to recover from (without violating the dynamic constraints and structural integrity of the aircraft), the algorithm would struggle to converge, or not converge at all. Setting the final-time guess to a higher and more lenient value, resulted in the algorithm converging much more quickly and with significantly more success. The algorithm then adjusts the final-time variable  $t_f$  during execution to a lower and more optimal value. However, setting the final-time guess to a value that is too lenient (high), could cause the algorithm to converge to a suboptimal local minima. For example, we can use the final time of dynamic programming's solutions as a guideline: For the same initial upset, if the recovery time of a dynamic programming solution is 6 seconds and the SQP final-time guess is set to 15 seconds, the SQP algorithm might converge to a recovery trajectory solution that completes in 11 seconds. Setting the SQP final-time guess to 7 seconds instead, causes the SQP algorithm to converge to a more optimal recovery trajectory with a final time that is closer to the final time of the dynamic programming trajectory solution of 6 seconds.

It was noted that the SQP could find solutions for some initial upset states that the dynamic programming algorithm could not. These were states that were generally inside and close to the border of the unrecoverable 'envelope' of the state grid map of Figure 4.11. However, as expected, the SQP could also not find solutions for the tested initial state cases that are 'deeper' inside the unrecoverable region, and this resulted in the SQP failing to converge for these initial states.

A limitation of the SQP algorithm is that it must recompute the trajectory solution for each different initial upset condition (since it is a gradient based method), while the dynamic programming method can compute all the trajectories from all of the recoverable states offline and use a fast online search to simply retrieve the stored solutions. However, the online computational burden of the SQP is still much less than dynamic programming's offline computational burden, and the SQP has the potential of being implemented as a real-time algorithm.

#### 4.5.12 Results Summary

Illustrative results were presented that showed the optimal state and input recovery trajectories generated by both the SQP optimal control algorithm and the dynamic programming algorithm from various types of initial upset conditions. The SQP's trajectory results are in the form of continuous piecewise third-order spline interpolated state trajectories and continuous piecewise linear interpolated input trajectories. The SQP's trajectory results were compared to those of dynamic programming's results using constant thrust input for each of the upset conditions.

The SQP followed similar recovery strategies than dynamic programming and validates the dynamic programming's trajectory solutions. In most result cases presented, the SQP solution

trajectories resulted in less peak altitude from being lost compared to the peak altitude loss of the dynamic programming solution trajectories. This result can be attributed to an advantage of the SQP method, namely the ability to use continuous state values, and the fact that the dynamic programming method is limited by only using discrete quantised values for the states.

Results were also presented that compared SQP recovery trajectories using constant throttle inputs versus recovery trajectories using varying throttle inputs (that modelled the thrust delay). From the comparisons it was clear that using the thrust as a varying input gave no significant advantage in terms of peak altitude loss except for severe underspeed upsets.

## 4.6 Conclusions

This chapter presented the design and implementation of two optimal control algorithms for attitude and flight vector recovery for large transport aircraft, namely the dynamic programming (DP) method and the sequential quadratic programming (SQP) method. The attitude and flight vector recovery problem was formulated as an optimal control problem that used a reduced-order model of the aircraft dynamics, that was then solved using dynamic programming and sequential quadratic programming.

The first algorithm, called dynamic programming, gives the recovery solution from *all* recoverable initial states in the form of a large state transition lookup table and gives a global minimum solution in a closed-loop form solution. However, for the aircraft recovery problem dynamic programming is forced to use a reduced-order model of only the point mass translational dynamics, where the fast rotational dynamics and thrust dynamics are omitted, because of a major limitation known as “the curse of dimensionality”.

The second method, namely the sequential quadratic programming method does not suffer from dynamic programming’s limitations and can include the fast rotational dynamics and thrust dynamics. The SQP can also use continuous-valued states and inputs, unlike dynamic programming, which is limited to quantised state values. This enables the SQP method to use a more representative model of the aircraft dynamics in the recovery trajectories. In most cases, these advantages enabled the SQP to produce recovery trajectories from upset conditions while losing less peak altitude compared to the dynamic programming method, while staying within the constraints imposed on the system. The SQP method produced recovery trajectories that used similar recovery strategies than the dynamic programming method, and validates the dynamic programming method’s assumption that a reduced-order model of the aircraft’s translational dynamics can be used to produce near-optimal recovery trajectories. Finally, it could be seen that modelling the thrust as an input with a delay did not provide any significant advantages to the optimal recovery trajectory solutions above using a constant thrust input.

The SQP routine can be used as a trajectory planner that would generate optimal recovery solutions that would then be executed by control schemes such as those presented in the next chapter.

## Chapter 5

# Trajectory Execution

This chapter details methods to execute a planned optimal recovery trajectory with a large transport aircraft, using the conventional aircraft flight controllers that were designed in Chapter 3. These methods would be used in an integrated attitude and flight vector recovery system: The system would detect the upset and if it is identified as an unusual attitude upset, the optimal guidance law would generate optimal recovery trajectories from the initial upset state, that would then be executed by commanding the middle-loop and/or inner-loop flight controllers. When the aircraft has successfully recovered, the fly-by wire Normal control laws would reactivate for nominal flight.

The solution generated by the optimal guidance law (in this case the SQP routine) uses an approximated reduced-order model and is also expected to have numerical integration errors. These errors can be present even when using a very high-order integration scheme, and despite even being relatively small, they might cause the system to deviate from the optimal trajectory. If disturbances are present, the deviation could be worse or even cause the system to become unstable. The goal is therefore to regulate the full non-linear system to follow the planned optimal recovery trajectories. The numerical trajectory solutions generated by the SQP routine can also be verified on the full non-linear aircraft simulation model with these control schemes.

The material in this chapter is presented as follows: First some background is given on the trajectory execution problem and motivation is given for using conventional aircraft controllers. Four different control schemes are proposed to control the aircraft to follow the planned optimal recovery trajectories generated by the SQP algorithm, namely input commands only, state commands only, combined input commands and state commands, and state commands only using compensated references. All four control schemes are designed, implemented, and verified in simulation on the full non-linear NASA Generic Transport Model (GTM). The next section then discusses and compares all the results of the four control schemes implemented on the GTM, and also discusses design decisions made as a result of observations during implementation. A section then provides a summary of the results following the detailed results and finally, the chapter gives a conclusion on the performance of the implemented control schemes.

### 5.1 Background and Motivation

The investigation of different trajectory execution schemes to control the aircraft to execute the planned optimal recovery trajectories was motivated by recommendations made by Engelbrecht in his PhD thesis [1]. Engelbrecht proposed two trajectory execution flight control architectures for attitude and flight vector recovery: one that supplies only the optimal input sequences to the inner-loop controllers, and another that supplies the optimal input sequences as references



to the inner-loop controllers and the optimal state trajectories as references to the middle-loop controllers. He performed preliminary simulations using the full NASA GTM model to illustrate the ability of both architectures to execute the optimal recovery trajectories provided by a dynamic programming guidance law. However, he recommended that the integration of the optimal trajectory planning guidance law and the trajectory execution control laws be investigated in more depth. Engelbrecht also recommended that the use of an angle of attack controller versus a normal load factor controller as the inner-loop controller for flight path angle recovery should be further investigated. The development of the four different trajectory execution flight control architectures (control schemes) presented in this chapter is therefore a more thorough investigation and an elaboration of the ideas proposed by Engelbrecht in his PhD thesis.

The first architecture proposed by Engelbrecht is shown in Figure 5.1. A dynamic programming guidance law generates the optimal input sequences (angle of attack, roll rate, and thrust) and the optimal state trajectories (airspeed, flight path angle, and bank angle) to perform the recovery. The optimal inputs are supplied as references to inner-loop controllers that control the angle of attack, roll rate, and thrust. The optimal state trajectories are not used by this control scheme. Also note that Engelbrecht used a Lyapunov-based inner-loop controller to control the angle of attack and roll rate, and not a conventional angle of attack controller (or load factor controller), nor a conventional roll rate controller.

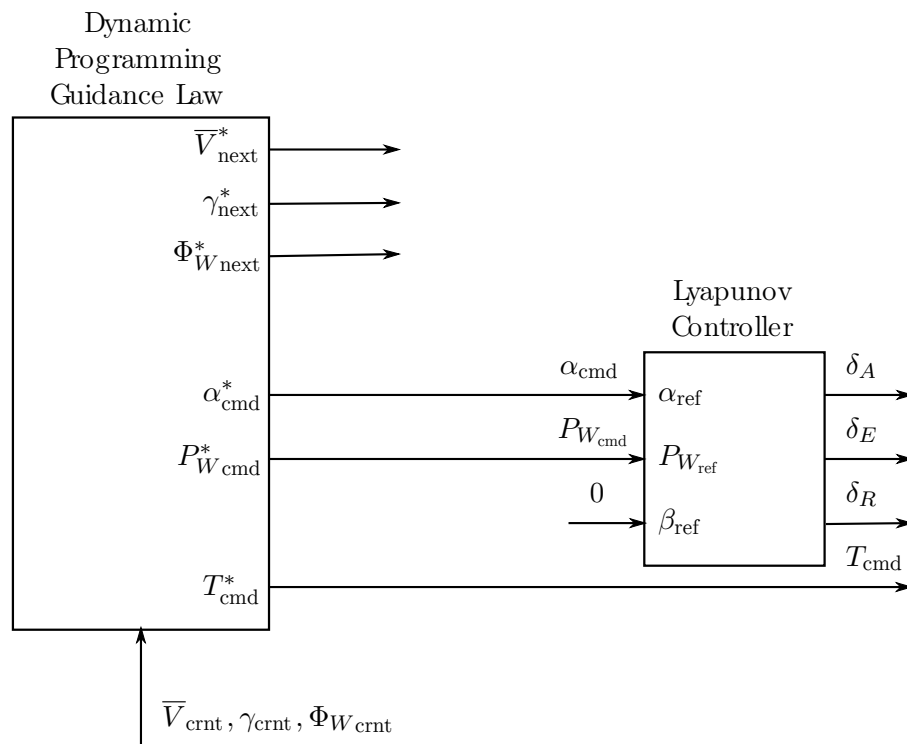


Figure 5.1: Integrated flight envelope recovery with dynamic programming outer-loop and Lyapunov controller inner-loop (Taken from [1])

The second architecture proposed by Engelbrecht is shown in Figure 5.2. A dynamic programming guidance law generates the optimal input sequences (angle of attack, roll rate, and thrust) and the optimal state trajectories (airspeed, flight path angle, and bank angle) to perform the recovery. The optimal inputs are supplied as references to the inner-loop controllers (thrust, load factor control, and roll rate control), while the optimal state trajectories are supplied as references to the middle-loop controllers (airspeed control, flight path angle control, and bank angle control). The optimal inputs are superimposed as feed-forward references onto the ‘feedback’ references supplied by the middle-loop controllers to the inner-loop controllers. The optimal angle of attack command is translated to a load factor command to be compatible with the load factor controller.

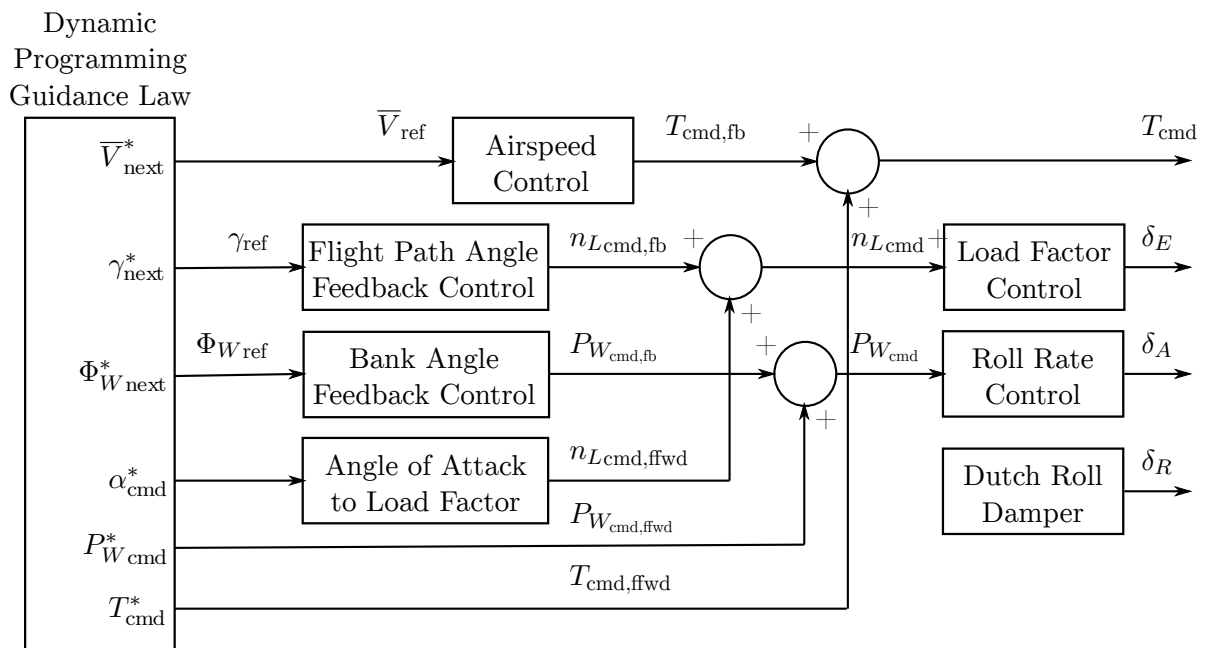


Figure 5.2: Integrated flight envelope recovery architecture with dynamic programming outer-loop and conventional flight control middle and inner-loops (Taken from [1])

A research decision was made to only employ conventional inner-loop and middle-loop control laws similar to those used in fly-by-wire flight control systems used on modern commercial passenger aircraft, rather than ‘exotic’ control laws such as Engelbrecht’s Lyapunov-based inner-loop controller. This decision was motivated by the fact that the research project was performed in collaboration with Airbus, and that their experts were interested in the feasibility of applying the optimal attitude and flight vector recovery to a flight control architecture similar to those used on their aircraft.

The implementation of the control schemes that perform trajectory execution in this chapter also serves to validate the reduced-order dynamic models used by the dynamic programming and SQP algorithms and validate that it is possible for a full non-linear aircraft model to follow these optimal trajectories using representative fly-by-wire controllers of commercial aircraft.

## 5.2 Overview of Control Schemes

In this chapter different control schemes are evaluated and compared. This section provides an overview of the four different control schemes that were proposed for trajectory execution, namely:

- **Input Commands Only ('Input only')**: The optimal inputs are supplied as references to the inner-loop controllers. The optimal state trajectories are not supplied to the flight control system. The aircraft is therefore controlled to execute the optimal state trajectories in an open-loop fashion.
- **State Trajectories Only ('State only')**: The optimal state trajectories are supplied as references to the middle-loop controllers. The optimal inputs are not supplied to inner-loop controllers. The aircraft is therefore controlled to execute the optimal state trajectories in a closed-loop fashion.
- **Combined State Trajectories and Optimal Inputs ('Combined state and input')**: The optimal inputs are provided as references to the inner-loop controllers and the optimal state trajectories are provided as references to the middle-loop controllers. The aircraft is therefore controlled to execute the optimal state trajectories in a closed-loop fashion, but with feed-forward references to the inner-loop controllers that anticipate the commands required to execute the trajectory.
- **Compensated State Trajectories ('Compensated state')**: The optimal state trajectories are supplied as references to the middle-loop controllers, but the optimal inputs are not supplied as references to the inner-loop controllers. Instead, the state references to the middle-loop controllers are modified to account for the 'lag' introduced by the middle-loop controller dynamics.

The first control scheme only uses the optimal input commands, which are directly passed to the inner-loop controllers as references, and can be seen as open-loop control of the aircraft state trajectory. This scheme is similar to the first scheme used by Engelbrecht (shown in Figure 5.1), but instead uses conventional flight controllers as the inner-loop controllers. Figure 5.3 shows the block diagram representation of the 'input only' control scheme. Here we use the star notation  $\mathbf{x}^*(t) \equiv \mathbf{x}^*$  and  $\mathbf{u}^*(t) \equiv \mathbf{u}^*$  to denote the time-varying optimal trajectory and optimal input solutions generated by the SQP. The middle-loop flight controllers were not used in this control scheme. This scheme is expected to execute the planned state trajectory with no lag, but does not provide disturbance rejection or robustness to model uncertainty, and the executed state trajectory will gradually deviate from the optimal trajectory over time, since no state feedback is used.

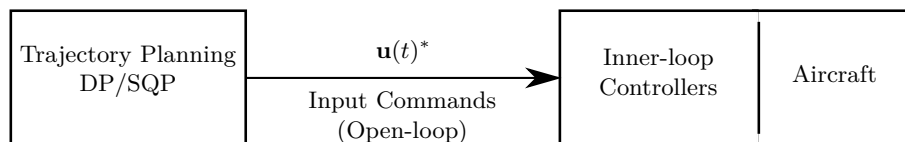


Figure 5.3: Architecture of 'input only' control scheme

The second control scheme only uses the optimal state trajectories, which are passed to the middle-loop controllers as time-varying reference signals. Figure 5.4 shows the block diagram

representation of the ‘state only’ control scheme. This scheme provides robustness and disturbance rejection due to the state feedback used by the middle-loop flight controllers and does not modify the flight controllers in any significant way. This scheme is expected to exhibit an executed state trajectory that lags behind the optimal reference trajectory, but with no steady-state error.

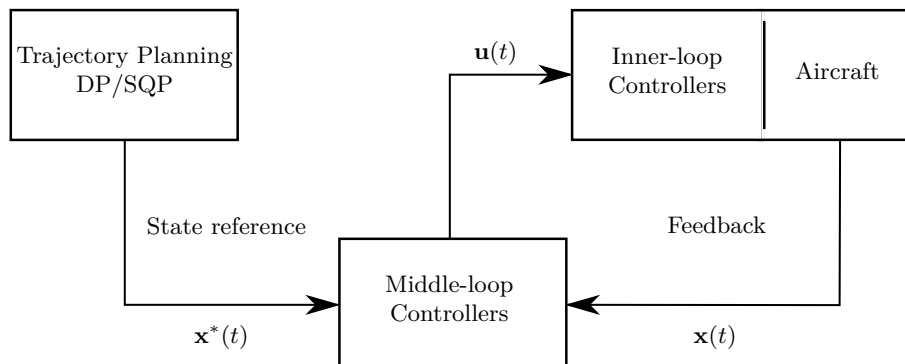


Figure 5.4: Architecture of ‘state only’ trajectory control scheme

The third control scheme uses both the optimal state and optimal input command trajectories in a state feedback with feed-forward input mixing configuration to try and gain the advantages of both of the previous two schemes. Figure 5.5 shows the block diagram representation of the ‘combined state and input’ control scheme. The middle-loop controllers use state feedback to provide corrective input actions, that are added to the optimal feed-forward input commands from the SQP. This scheme is expected to give better performance than the previous two schemes, with no lag in the executed state trajectory, with disturbance rejection and robustness to model uncertainty provided by the feedback control, and with minimal steady-state error.

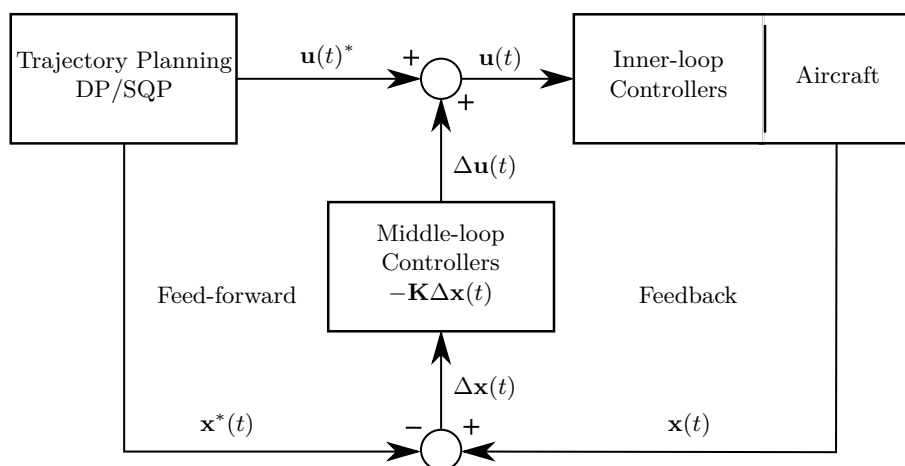


Figure 5.5: Architecture of ‘combined state and input’ control scheme

The fourth scheme, illustrated in Figure 5.6, is an extension of the second, ‘state only’, scheme, but compensates the state trajectory references that are supplied to the middle-loop controllers. The compensation results in reference commands  $\mathbf{x}_c^*(t)$  that take the dynamics of the middle-loop flight controllers into account. This scheme is expected to eliminate the trajectory lag exhibited by the ‘state only’ scheme, while still providing the advantages of feedback control,

namely disturbance rejection, robustness to model uncertainty, and steady-state command tracking.

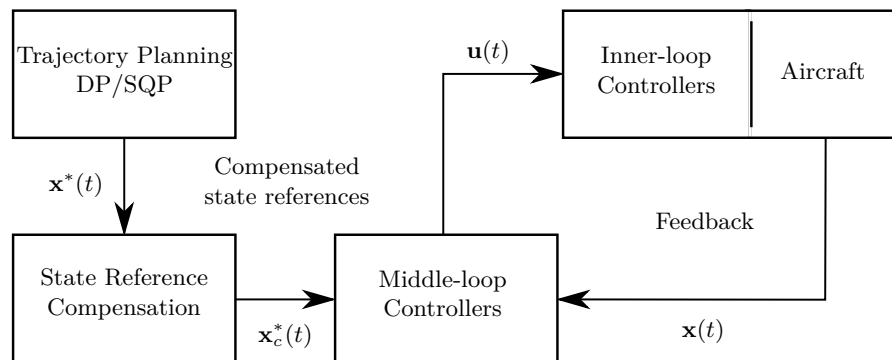


Figure 5.6: Architecture of ‘compensated state’ trajectory control scheme

Note that the optimal trajectory plan from the SQP is formulated in terms of wind-axis variables, such as wind-axis bank angle and wind-axis roll rate, while the conventional flight controllers and sensors operate in terms of body-axis variables, such as bank angle and body-axis roll rate. Thus as part of the control scheme design, the wind-axis state references and wind-axis input commands of the optimal trajectories are converted into body-axis commands for the conventional flight controllers. Furthermore, the body-axis sensor measurements and state estimates are converted back to the wind-axis in order to compare the trajectories that were actually executed with the optimal trajectories that were planned.

The subsequent sections address the design of the four schemes in more detail and provide simulation results of recovery trajectories tested on the full non-linear simulation model of the NASA GTM for each of the schemes.

### 5.3 Design of Control Schemes

This section details the design of each of the four proposed trajectory control schemes and how the conventional flight controllers were used within them. This is then followed by subsections detailing the sensor and command conversions from wind-axis to body-axis and visa versa used in the control schemes.

#### 5.3.1 Input Commands Only Control Scheme

Figure 5.7 shows the architecture block diagram representing the ‘input only’ control scheme that uses the conventional fly-by-wire controllers that were designed in Chapter 3. For this control scheme only the optimal input commands  $\mathbf{u}^*$  from the SQP solution are used to command the inner-loop controllers.

The optimal input commands planned by the DP/SQP algorithm, namely the optimal thrust command  $\delta_{Tc}^*$ , the optimal angle of attack command  $\alpha_c^*$  (or the optimal normal acceleration command  $a_{z_c}^*$ ), and the optimal wind-axis roll rate command  $P_{Wc}^*$ , are provided as references to the aircraft’s thrust input, the angle of attack (or normal acceleration) controller, and the roll rate and sideslip angle controller, respectively. The optimal wind-axis roll rate command  $P_{Wc}^*$  is converted to an equivalent body-axis bank angle rate command  $\dot{\Phi}_c$  for the roll rate and sideslip angle controller (The details of the conversion from wind-axis roll rate to body-axis

bank angle rate will be presented in section §5.3.6). The sideslip angle reference for the roll rate and sideslip angle controller is set to a constant zero degrees.

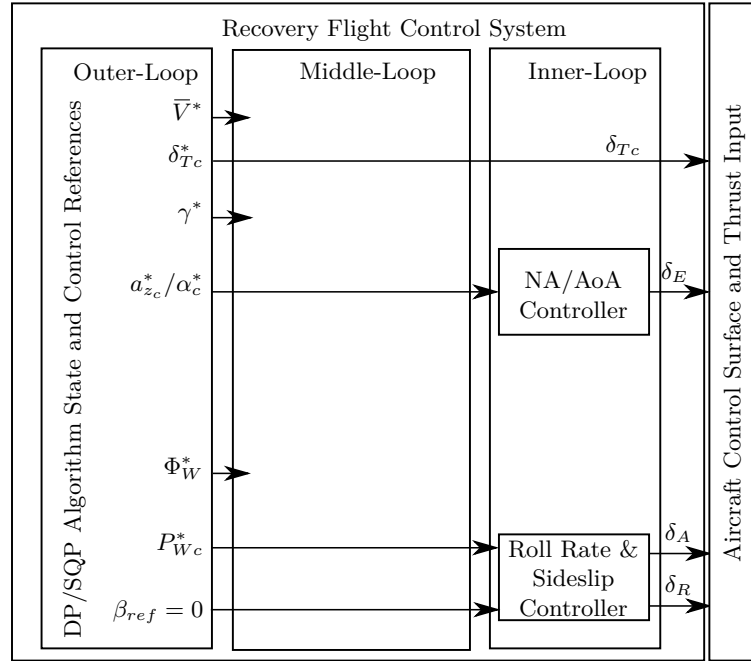


Figure 5.7: Flight controller architecture for ‘input only’ control scheme

Engelbrecht [1] originally formulated the optimal control problem to use angle of attack as input, and suggests if conventional flight controllers are used, that the angle of attack input be converted to a normal load factor command (which is essentially a normal acceleration command). From this suggestion, we investigated the use of both an angle of attack and normal acceleration inner-loop controller and compare their performance using the ‘input only’ control scheme.

The SQP’s optimal angle of attack command  $\alpha_c^*$  solution must be translated into a normal acceleration command before it can be used as reference to the DQ controller. The optimal normal acceleration input trajectory is calculated from the optimal state and input trajectories with a wind-axis to body-axis transformation using the optimal angle of attack angle command and sideslip angle assumed zero,

$$a_{z_c}^* = \sin \alpha_c^* a_{W_x}^* + \cos \alpha_c^* a_{W_z}^* \quad (5.3.1)$$

where  $a_{W_x}^*$  is the axial acceleration in the wind-axis and  $a_{W_z}^*$  is the normal acceleration in the wind-axis, i.e. the lift-axis acceleration,

$$a_{W_x}^* = \frac{1}{m} \left[ T^* - \frac{1}{2} \rho \bar{V}^{*2} S C_D(\alpha_c^*) - mg \sin \gamma^* \right] \quad (5.3.2)$$

$$a_{W_z}^* = \frac{1}{m} \left[ \frac{1}{2} \rho \bar{V}^{*2} S C_L(\alpha_c^*) + mg \cos \gamma^* \cos \Phi^* \right] \quad (5.3.3)$$

Note that the translation is performed offline (using the planned airspeed and flight path angle trajectories) to produce a pre-generated optimal normal acceleration input *before* the trajectory

is executed. The translation from angle of attack to normal acceleration is not performed online (using instantaneous measurements of airspeed or flight path angle) during the execution of the trajectory.

The equivalent body-axis bank angle rate command  $\dot{\Phi}_c$  has been converted from the optimal wind-axis roll rate command  $P_{Wc}^*$ , which is interpreted as a wind-axis bank angle rate command  $\dot{\Phi}_{Wc}^* = P_{Wc}^*$ , because of the SQP's assumption that  $\dot{\Phi}_W^* = P_W^*$ . The conversion is necessary to take into account the effects of the wind-axis pitch and yaw rates on wind-axis bank angle rate, and make the command compatible with the roll rate and sideslip controller (This process uses the conversion equations discussed in section §5.3.6). Note that the DPDR controller is not strictly an inner-loop controller as it controls the bank angle state and only implicitly controls the roll rate, but from the perspective of the trajectory planning routine, it controls an input variable. In the 'input only' control scheme, the DPDR controller is therefore used as an inner-loop bank angle rate controller, and the bank angle rate command  $\dot{\Phi}_c$  is supplied to the bank angle rate input (integrator) of the DPDR controller.

Initially, the baseline angle of attack controller that is supplied with the NASA GTM simulation model was used as the angle of attack controller. However, it was found that this baseline controller delivered poor angle of attack tracking at low airspeeds (Some example simulation results using the baseline angle of attack controller, showing the poor angle of attack tracking, are included in Appendix D). A new angle of attack controller was therefore designed using a Linear Quadratic Integral (LQI) architecture with gain scheduling for different dynamic pressures, i.e. airspeeds (The design of the new gain-scheduled angle of attack controller was already presented in Chapter 3, section §3.4).

### 5.3.2 State Trajectories Only Control Scheme

Figure 5.8 shows the architecture block diagram representing the 'state only' control scheme that uses the designed conventional fly-by-wire controllers. For this control scheme only the optimal state reference solutions  $\mathbf{x}^*$  from the SQP solution are used to command the middle-loop controllers.

The optimal state trajectories planned by the DP/SQP algorithm, namely the optimal flight path angle  $\gamma^*$  and the optimal wind-axis bank angle  $\Phi_W^*$  are provided as references to the flight path angle controller and the bank angle and sideslip angle controller, respectively. The middle-loop flight path angle controller in turn supplies normal acceleration commands to the inner-loop normal acceleration controller. The optimal wind-axis bank angle  $\Phi_W^*$  is converted to an equivalent body-axis bank angle reference  $\Phi^*$  for the bank angle and sideslip angle controller (The details of the conversion from wind-axis bank angle to body-axis bank angle will be presented in section §5.3.6). The optimal thrust command  $\delta_{Tc}^*$  is directly provided as reference to the aircraft's thrust input. The sideslip angle reference for the roll rate and sideslip angle controller is set to a constant zero degrees. In the 'state only' control scheme, the DPDR controller is used as a middle-loop bank angle controller, and the bank angle reference therefore bypasses the bank angle rate input (integrator) of the DPDR controller.

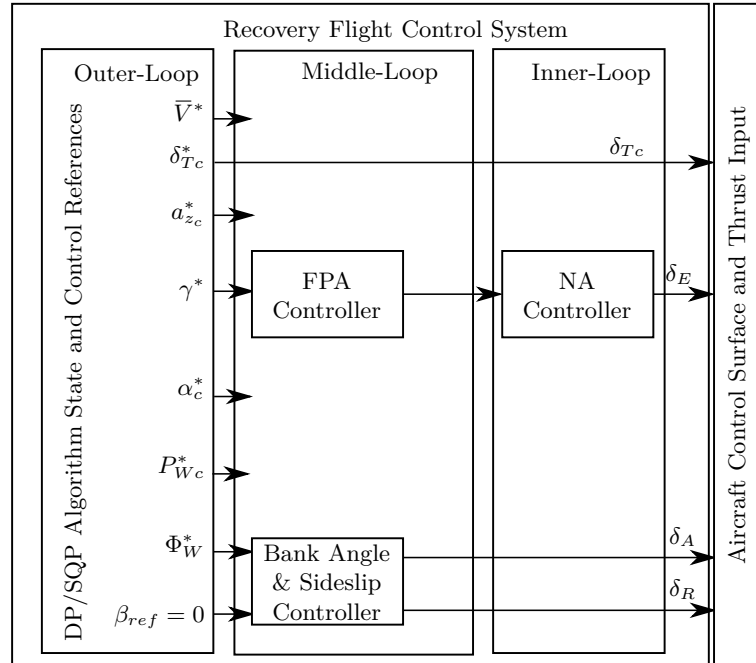


Figure 5.8: Flight controller architecture for ‘state only’ control scheme

In theory, the optimal airspeed trajectory  $\bar{V}^*$  could also be supplied as a reference to the airspeed controller. However, the timescale over which the airspeed controller is able to change the airspeed using thrust is much longer than the duration of a typical recovery. The airspeed can be changed much more rapidly by changing the flight path angle and using gravity to increase or decrease the airspeed. The trajectory planning algorithm already exploits this fact, and uses the flight path angle more than the thrust to perform both underspeed and overspeed recovery. It was also found that the airspeed controller’s thrust commands were too aggressive when the airspeed is changing quickly, due to the state feedback gain on axial acceleration. For these reasons, it was decided not to use the airspeed controller for the recovery, and rather to supply the optimal thrust command directly to the aircraft’s thrust input. The airspeed controller is only activated after recovery is completed and trims the aircraft to the optimal terminal airspeed to achieve steady state-tracking of the optimal reference.

### 5.3.3 Combined State and Input Control Scheme

The architecture of the ‘combined state and input’ scheme is shown in Figure 5.9. For this control scheme we try to gain both the immediate response of the ‘input only’ scheme and the disturbance rejection, robustness to model uncertainty, and good steady-state tracking of the ‘state only’ scheme. The optimal input commands are supplied as references to the inner-loop controllers and the optimal state trajectories are supplied as references to the middle-loop controllers. The outputs of the middle-loop controllers are then superimposed on the optimal input commands that are supplied directly to the inner-loop controllers. The total references supplied to the inner-loop controllers therefore contain a feed-forward component to eliminate trajectory lag, and a feedback component to correct deviations from the optimal state trajectories.

The optimal flight path angle  $\gamma^*$  is provided as a reference to the middle-loop flight path angle controller, while the optimal normal acceleration input command  $a_{zc}^*$  is fed forward to the inner-loop normal acceleration controller. The total normal acceleration reference supplied



to the inner-loop normal acceleration controller is therefore the sum of a feed-forward term provided by the trajectory planner and a correction term provided by the middle-loop flight path angle controller. Note that there is also a correction term provided by an angle of attack controller, which shall be discussed later. The flight path angle controller is switched back to its original nominal trim mode by disabling the optimal feed-forward term after the aircraft has successfully recovered in order to trim the aircraft to wings level flight.

The optimal wind-axis bank angle  $\Phi_W^*$  is provided as a reference to the middle-loop bank angle controller, while the optimal wind-axis roll rate input command  $P_{Wc}^*$  is fed forward along with the commands from the bank angle controller to form a total wind-axis roll rate command  $P_{Wc}$ . The total wind-axis roll rate reference supplied to the inner-loop roll rate and sideslip angle controller is therefore the sum of a feed-forward term provided by the trajectory planner and a correction term provided by the middle-loop bank angle controller. However, note that the total wind-axis roll rate command  $P_{Wc}$  is first converted to an equivalent body-axis bank angle rate command  $\dot{\Phi}_c$  before it is given as the reference to the roll rate and sideslip angle controller (The details of the conversion from wind-axis bank angle to body-axis bank angle, and from wind-axis bank angle rate to body-axis bank angle rate, will be presented in section 5.3.6). The sideslip angle reference for the roll rate and sideslip angle controller is set to a constant zero degrees.

The ‘combined state and input’ control scheme also does not use the airspeed controller (for the same reasons as the ‘state only’ control scheme) and instead the optimal thrust command is supplied directly to the aircraft’s thrust input. However, the airspeed controller is activated after the recovery has been completed, to trim the airspeed in level flight.

Finally, the presence of the middle-loop angle of attack controller must be explained. During simulation testing it was found that the executed airspeed trajectory deviated significantly from the planned airspeed trajectory. The main reason for this was that the drag acting on the aircraft differed significantly between the planned and executed trajectories. Since the airspeed is controlled in an open-loop fashion, and not corrected by a middle-loop airspeed controller, the airspeed trajectory is especially susceptible to model uncertainty. The angle of attack controller was therefore added to correct deviations between the planned angle of attack and the executed angle of attack. The angle of attack controller keeps the executed angle of attack closer to the planned angle of attack, which keeps the actual drag experienced by the aircraft closer to the planned drag, and ultimately keeps the executed airspeed trajectory closer to the planned airspeed trajectory.

The middle-loop angle of attack controller controls the angle of attack using normal acceleration commands to the inner-loop normal acceleration controller. The optimal angle of attack state  $\alpha^*$  from the trajectory planning algorithm (DP/SQP) is supplied as a reference to the angle of attack controller. The normal acceleration command provided by the angle of attack controller to correct the angle of attack deviation, is then added to the feed-forward normal acceleration term provided by the trajectory planner, and the feedback normal acceleration term provided by the flight path angle controller to correct the flight path angle deviation. The angle of attack regulator is also turned off when the flight path angle controller is switched back to its nominal trim mode after the aircraft has successfully recovered.

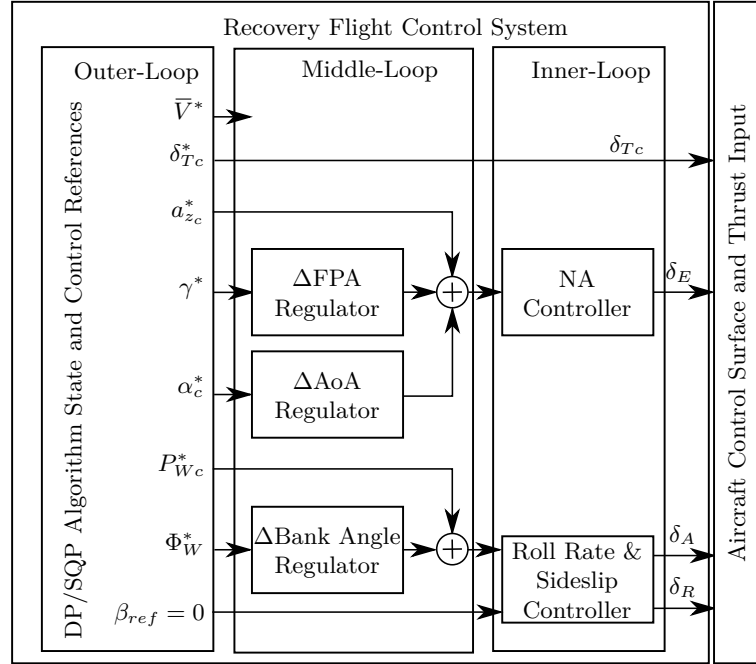


Figure 5.9: Modified flight controller architecture for ‘combined state and input’ control scheme

A relatively straight-forward method that is used in literature, uses a Linear Quadratic Regulator (LQR) to obtain a trajectory stabilisation feedback controller [37]. The conventional flight controllers are in essence linear controllers and are designed to work around a nominal trim state and not a varying state trajectory. The original flight controllers are used to work with the input commands generated by the SQP routine by using input mixing, while still using the middle-loop controllers to give steady-state tracking and robustness to disturbances to follow the reference state trajectories.

The architecture of the ‘combined state and input’ scheme can be derived from the architecture of a designed Time Varying Linear Quadratic Regulator (TVLQR) in [37]. Figure 5.5 shows the control scheme that uses the feed-forward control inputs  $\mathbf{u}^*$  and state references  $\mathbf{x}^*$  provided by the SQP routine together with the feedback stabilisation control  $\Delta \mathbf{u}(t)$  provided by the state regulator controller gains. Following this general architecture, the middle-loop flight controllers become the state regulator component in the ‘combined state and input’ scheme.

The next subsections details how the original middle-loop control laws were used to correct the deviation of the executed state trajectory from the optimal planned state trajectory.

### Middle-loop Control Laws

The command laws of the conventional middle-loop flight path angle controller and middle-loop bank angle controller (DPDR controller) that were designed, are recalled here. The flight path angle control law commands the normal acceleration command for the DQ controller,

$$a_{z_c} = \frac{-\bar{V} [K_\gamma (\gamma_c - \gamma)] - a_{\gamma_{gravity}}}{\cos \Phi} + a_{z_{gravity}} \quad (5.3.4)$$

with,

$$a_{z_{gravity}} = g \cos \Theta \cos \Phi, \quad a_{\gamma_{gravity}} = g \cos \Theta \quad (5.3.5)$$

This normal acceleration command can be rewritten in an alternate form to highlight the state regulation term,

$$\begin{aligned}
 a_{z_c} &= \frac{-\bar{V} [K_\gamma (\gamma_c - \gamma)] - g \cos \Theta}{\cos \Phi} + g \cos \Theta \cos \Phi \\
 &= \left[ \frac{-g \cos \Theta}{\cos \Phi} + g \cos \Theta \cos \Phi \right] + \left[ \frac{-\bar{V} K_\gamma}{\cos \Phi} (\gamma_c - \gamma) \right] \\
 &= a_{z_{nom}} + \Delta a_{z_\gamma}
 \end{aligned} \tag{5.3.6}$$

From Equation 5.3.6 we can see that there is a nominal normal acceleration input  $a_{z_{nom}}$  in the command law that effectively trims the aircraft to a certain flight path angle rate. The flight path angle controller only calculates correction command  $\Delta a_{z_\gamma}$ , which is the normal acceleration control effort needed to regulate the flight path angle state  $\gamma$  to the state reference  $\gamma_c$ . The nominal normal acceleration command is added to give a non-zero control effort when the flight path angle error is zero  $\gamma = \gamma_c$ , because it accounts for gravity to maintain a flight path angle rate of zero at trim.

The bank angle controller is technically a middle-loop controller that directly controls the aircraft control surfaces to control the bank angle state. However, it can implicitly control the roll rate and be seen as an inner-loop controller by using an integrator to integrate a bank angle rate reference into a bank angle reference,

$$\Phi_{ref} = \int_0^\infty \dot{\Phi}_{ref} dt \tag{5.3.7}$$

Here the bank angle rate term  $\dot{\Phi}_{ref}$  represents the roll rate input of the DPDR controller (if it is used as a roll rate controller) and the integral represents the integrator component in the DPDR controller that converts a roll rate input into a bank angle reference inside the controller.

The autothrust airspeed controller was not used for speed regulation due to its low bandwidth. Either a constant thrust command was given, or the open-loop throttle commands from the SQP. The autothrust controller was used simply to trim the aircraft once recovered.

These flight control laws along with the inner-loop controllers were used to track a time-varying reference state trajectory  $\mathbf{x}^*$  while using a feed-forward input term  $\mathbf{u}^*$ . The general case of the state regulation formulation deals with non-linear systems, and the error dynamics that govern the deviation from the reference state trajectories are re-linearised at different points along the time-varying state reference trajectory. This results in a time varying control law that uses time-varying gains to ensure robustness.

In our design, we use the conventional flight controllers that work from a single nominal trim point, since the aircraft is assumed to be within its aerodynamic envelope due to the formulation of the upset recovery problem. The flight controllers are fixed-gain controllers and are already designed to be sufficiently robust within the aerodynamic envelope. Thus the designed conventional flight controllers are used as state trajectory regulator controllers and the trajectory regulation formulation reduces to a time-invariant gain solution.

### Middle-loop Control Laws used to Correct Trajectory Deviations

For the ‘combined state and input’ scheme, the final resulting command laws for the middle-loop controllers are all in the state regulator form of Figure 5.5,

$$\begin{aligned}\mathbf{u} &= \mathbf{u}^* + \Delta\mathbf{u} \\ \Delta\mathbf{u} &= -\mathbf{K}(\mathbf{x} - \mathbf{x}^*)\end{aligned}\tag{5.3.8}$$

For the flight path angle controller the command law becomes,

$$\begin{aligned}a_{z_c} &= a_{z_c}^* + \Delta a_{z_\gamma} + \Delta a_{z_\alpha} \\ &= a_{z_c}^* + \frac{-\bar{V}K_\gamma(\gamma^* - \gamma)}{\cos\Phi} + \frac{K_\alpha(\alpha^* - \alpha)\bar{q}SC_{Z_\alpha}}{m}\end{aligned}\tag{5.3.9}$$

where  $a_{z_c}^*$  is the optimal open-loop feed-forward normal acceleration command expected to produce the optimal flight path angle trajectory (which already accounts for gravity),  $\Delta a_{z_\gamma}$  is the normal acceleration command needed to minimise the error in the flight path angle state from the optimal flight path angle state, and  $\Delta a_{z_\alpha}$  is the normal acceleration command needed to minimise the error in the angle of attack state from the optimal angle of attack state. The derivation and details of this flight path angle command law is discussed later in this section.

The DPDR controller is used as an inner-loop roll rate controller in the ‘combined state and input’ scheme. Thus a middle-loop bank angle command law is created instead that commands the DPDR controller. The new middle-loop bank angle command law uses the state regulator form of Figure 5.5 to regulate the wind-axis bank angle to track the planned wind-axis bank angle. The wind-axis bank angle controller issues a corrective wind-axis roll rate command which is proportional to the error between the wind-axis bank angle and the commanded (or planned) wind-axis bank angle. This corrective command is summed with a feed-forward wind-axis roll rate term from the trajectory planner to form the total wind-axis roll rate command,

$$\begin{aligned}P_{Wc} &= P_{Wc}^* + \Delta P_W \\ &= P_{Wc}^* + K_{\Phi_W}(\Phi_W^* - \Phi_W)\end{aligned}\tag{5.3.10}$$

where  $P_{Wc}^*$  is the open-loop optimal wind-axis roll rate feed-forward command which is expected to produce the optimal wind-axis bank angle trajectory, and  $\Delta P_W$  is the wind-axis roll rate command to minimise the error in the wind-axis bank angle state from the optimal wind-axis bank angle state. This total wind-axis roll rate command is then converted into an equivalent bank angle rate command  $\dot{\Phi}_c$  that is issued as a reference to the roll rate input of the DPDR controller,

$$\dot{\Phi}_{ref} = \dot{\Phi}_c, \quad \dot{\Phi}_c = f^{W \rightarrow B}(P_{Wc})\tag{5.3.11}$$

where the total wind-axis roll rate command  $P_{Wc}$  is converted into the body-axis bank angle rate command  $\dot{\Phi}_c$  with the function  $f^{W \rightarrow B}$  representing the conversion process of section §5.3.6, to account for the SQP’s assumption that  $\dot{\Phi}_W^* = P_{Wc}^*$ .

The state regulator of Figure 5.5 can be used as theoretical basis to derive the final flight path angle command law. We can write the original flight path angle middle-loop control law of

Equation 5.3.4 in the state regulator form of Equation 5.3.8,

$$\begin{aligned}
 u &= u^* + \Delta u \\
 &= u^* - K(x - x^*) \\
 a_{z_c} &= \frac{-\bar{V} [K_\gamma (\gamma_c - \gamma)] - a_{\gamma_{gravity}}}{\cos \Phi} + a_{z_{gravity}} \\
 &= a_{z_{nom}} + \Delta a_{z_\gamma} \\
 &= a_{z_{nom}} - \frac{-\bar{V} K_\gamma}{\cos \Phi} (\gamma - \gamma_c)
 \end{aligned} \tag{5.3.12}$$

with,

$$\begin{aligned}
 u^* &= a_{z_{nom}} = \frac{-g \cos \Theta}{\cos \Phi} + g \cos \Theta \cos \Phi \\
 x^* &= \gamma_c
 \end{aligned} \tag{5.3.13}$$

$$\Delta u = \Delta a_{z_\gamma} = -K (\gamma - \gamma_c), \quad K = \frac{-\bar{V} K_\gamma}{\cos \Phi}$$

The initial mixing scheme incorrectly superimposed two nominal inputs  $a_{z_{nom}}$ , namely the original command law nominal input, and the optimal feed-forward command  $a_z^*$ , by summing the feed-forward term with the total normal acceleration command  $a_{z_c}$ . Intuitively, we want to use the time-varying optimal feed-forward command instead of the original command law nominal input, because we want the system to be regulated along the optimal flight path angle rate trajectory  $\dot{\gamma}^*$  and not the nominal flight path angle rate (of zero). Thus the correct way of mixing the optimal feed-forward input with the middle-loop controller command is to *replace* the original nominal input of the controller  $a_{z_{nom}}$  with that of the optimal feed-forward command  $a_z^*$ , giving the flight path angle control law of,

$$a_{z_c} = a_z^* + \Delta a_{z_\gamma} \tag{5.3.14}$$

with,

$$\Delta a_{z_\gamma} = -\frac{-\bar{V} K_\gamma}{\cos \Phi} (\gamma - \gamma^*) \tag{5.3.15}$$

Note that in this control scheme the flight path angle controller gives zero control effort when the flight path angle error is zero  $\gamma = \gamma^*$  and the normal acceleration control command reduces to only that of the optimal feed-forward command.

As discussed previously and in section §5.3.1, using an angle of attack controller instead of a DQ controller caused the executed airspeed trajectory to deviate less from the planned airspeed trajectory. Using the state regulator form of Equation 5.3.8 we can add a normal acceleration control effort component  $\Delta a_{z_\alpha}$  to the flight path angle command law of Equation 5.3.14 to minimise the error in the angle of attack state. Thus we still have the advantage of using an acceleration based inner-loop controller and still be able to regulate the angle of attack state for better airspeed tracking,

$$a_{z_c} = a_z^* + \Delta a_{z_\gamma} + \Delta a_{z_\alpha} \quad (5.3.16)$$

with,

$$\Delta a_{z_\alpha} = -\frac{K_\alpha \bar{q} S C_{Z_\alpha}}{m} (\alpha - \alpha^*), \quad C_{Z_\alpha} = \left. \frac{\partial C_{Z,Static}}{\partial \alpha} \right|_T \quad (5.3.17)$$

The angle of attack error is multiplied by the stability derivative  $C_{Z_\alpha}$  to get the approximate linear change in normal lift force, and thereby change in normal acceleration needed to minimise the error in the angle of attack state.

### 5.3.4 Compensated State Trajectories Control Scheme

The ‘compensated state’ control scheme is similar to the ‘state only’ control scheme, but attempts to eliminate the trajectory ‘lag’ by modifying the state references supplied to the middle-loop controllers in order to compensate for the ‘lag’ introduced by middle-loop controller dynamics. Instead of using feed-forward input commands to the inner-loop controllers, the ‘compensated state’ control scheme essentially replans the state references to take the middle-loop dynamics into account during the trajectory planning phase. This approach could be used for existing fly-by-wire flight control architectures where the references to the inner-loop controllers are not accessible when the middle-loop controllers are active.

The architecture of the ‘compensated state’ scheme is shown in Figure 5.10. Linear models of the middle-loop controller dynamics are used to convert the optimal state trajectories to compensated state references for the middle-loop controllers.

The optimal state trajectories planned by the DP/SQP algorithm, namely the optimal flight path angle  $\gamma^*$  and the optimal wind-axis bank angle  $\Phi_W^*$ , are passed through compensation blocks that convert them to the modified state references for the flight path angle controller and the bank angle and sideslip angle controller, respectively. The middle-loop flight path angle controller in turn supplies normal acceleration commands to the inner-loop normal acceleration controller. The compensated optimal wind-axis bank angle reference  $\Phi_{W_c}^*$  is converted to an equivalent body-axis bank angle reference  $\Phi_{ref}$  for the bank angle and sideslip angle controller. (The details of the conversion from wind-axis bank angle to body-axis bank angle will be presented in section §5.3.6) The sideslip angle reference for the roll rate and sideslip angle controller is set to a constant zero degrees. As with the ‘state only’ scheme, the ‘compensated state’ control scheme uses the DPDR controller as a middle-loop bank angle controller, and the bank angle reference therefore bypasses the bank angle rate input (integrator) of the DPDR controller.

The ‘compensated state’ scheme also does not use the airspeed controller (for the same reasons as the ‘state only’ scheme) and instead the optimal thrust command is supplied directly to the aircraft’s thrust input. (The thrust lag has already been accounted for in the original trajectory planning phase, and does not have to be compensated again.)

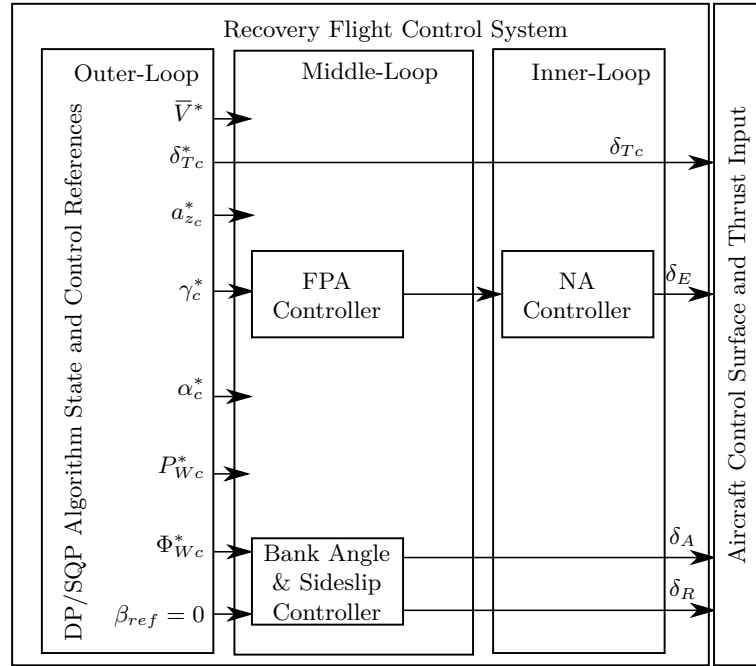


Figure 5.10: Flight controller architecture for ‘compensated state’ scheme

A post-optimisation calculation is done using the linear model dynamic equations of the middle-loop controllers with the optimal state trajectory solutions to calculate the new optimal state reference commands to give to the middle-loop controllers. The SQP routine solution supplies us with the interpolated optimal state trajectories  $\mathbf{x}^*(t)$  and the interpolated optimal state dynamics  $\mathbf{f}^*(t)$ . These variables are then used in a first-order dynamics equation of the middle-loop controllers to calculate the compensated state reference commands. The flight path angle and bank angle controller are approximated by a first-order response model with their respective time constants,

$$\dot{\gamma}^* = -\frac{1}{\tau_\gamma}\gamma^* + \frac{1}{\tau_\gamma}\gamma_c^* \quad (5.3.18)$$

$$\dot{\Phi}_W^* = -\frac{1}{\tau_P}\Phi_W^* + \frac{1}{\tau_P}\Phi_{Wc}^* \quad (5.3.19)$$

which can be rearranged to calculate the compensated state reference terms,

$$\gamma_c^* = \frac{1}{\tau_\gamma}\dot{\gamma}^* + \gamma^* \quad (5.3.20)$$

$$\Phi_{Wc}^* = \frac{1}{\tau_P}\dot{\Phi}_W^* + \Phi_W^* \quad (5.3.21)$$

where  $\tau_\gamma$  is the first-order time constant of the flight path angle controller model calculated as  $\tau_\gamma = 1.15$  seconds and  $\tau_P$  is the first-order time constant of the bank angle controller model calculated as  $\tau_P = 0.3$  seconds. The variables  $\gamma_c^*$  and  $\Phi_{Wc}^*$  are the desired optimal middle-loop compensated references for the flight path angle and bank angle controllers respectively that are calculated. The optimal flight path angle dynamics  $\dot{\gamma}^*$  and optimal bank angle dynamics  $\dot{\Phi}_W^*$

are second-order interpolated polynomial functions supplied by the SQP, that were calculated using Equation 4.5.70 in Chapter 4.

There is no angle of attack regulation component in this scheme as the default flight path angle controller does not include such an element. As with all the previous schemes the airspeed controller is not used due to the GTM's very slow thrust to airspeed response. The airspeed for the recovery trajectories is mainly influenced by the drag forces and the designed airspeed controller cannot regulate the airspeed by just using the thrust input.

### 5.3.5 Sensing Wind-Axis Bank Angle and Wind-Axis Roll Rate

The SQP trajectory planner is formulated in terms of the wind-axis bank angle  $\Phi_W$  and wind-axis roll rate  $P_W$ , while the conventional fly-by-wire controllers expect a body-axis bank angle reference  $\Phi_{ref}$  and a body-axis bank angle rate reference  $\dot{\Phi}_{ref}$ . The onboard state estimator also reports the aircraft state in terms of body-axis variables and not in terms of wind-axis variables. A wind-axis to body-axis conversion is therefore required to translate the planned optimal state trajectories and command inputs to references for the fly-by-wire controllers, and a body-axis to wind-axis conversion is required to compare the executed state trajectory with the planned state trajectory. The wind-axis bank angle  $\Phi_W$  can be calculated from the angle of attack  $\alpha$  and the sideslip angle  $\beta$  measured by the anemometric sensors, and the body-axis roll angle  $\Phi$ , pitch angle  $\Theta$ , and yaw angle  $\Psi$  provided by the onboard state estimator. The calculation is performed as follows:

The attitude of the wind-axis system relative to the inertial-axis system  $\mathbf{DCM}_{I \rightarrow W}$  is the product of the attitude of the body-axis system relative to the inertial-axis system  $\mathbf{DCM}_{I \rightarrow B}$  and the attitude of the wind-axis system relative to the body-axis system  $\mathbf{DCM}_{B \rightarrow W}$  [1]. This relationship can be expressed mathematically using the applicable direction cosine matrix (DCM) matrices,

$$\mathbf{DCM}_{I \rightarrow W} = \mathbf{DCM}_{B \rightarrow W} \mathbf{DCM}_{I \rightarrow B} \quad (5.3.22)$$

where the inertial- to body-axis transformation matrix  $\mathbf{DCM}_{I \rightarrow B}$  is defined as,

$$\mathbf{DCM}_{I \rightarrow B} = \begin{bmatrix} C_\Psi C_\Theta & S_\Psi C_\Theta & -S_\Theta \\ C_\Psi S_\Theta S_\Phi - S_\Psi C_\Phi & S_\Psi S_\Theta S_\Phi + C_\Psi C_\Phi & C_\Theta S_\Phi \\ C_\Psi S_\Theta C_\Phi + S_\Psi S_\Phi & S_\Psi S_\Theta C_\Phi - C_\Psi S_\Phi & C_\Theta C_\Phi \end{bmatrix}, S_{(\cdot)} = \sin(\cdot), C_{(\cdot)} = \cos(\cdot) \quad (5.3.23)$$

and the body to wind-axis transformation  $\mathbf{DCM}_{B \rightarrow W}$  is calculated using,

$$\mathbf{DCM}_{B \rightarrow W} = \begin{bmatrix} \cos \alpha \cos \beta & \sin \beta & \sin \alpha \cos \beta \\ -\cos \alpha \sin \beta & \cos \beta & -\sin \alpha \sin \beta \\ -\sin \alpha & 0 & \cos \alpha \end{bmatrix} \quad (5.3.24)$$

We then recognise that the inertial to wind-axis transformation matrix  $\mathbf{DCM}_{I \rightarrow W}$  is also an Euler 3-2-1 sequence of rotations, but using the flight path heading  $\Psi_W$  for the z-axis rotation, the flight path angle  $\gamma$  for the y-axis rotation, and the wind-axis bank angle  $\Phi_W$  for the x-axis rotation. The direction cosine matrix  $\mathbf{DCM}_{I \rightarrow W}$  for the inertial- to wind-axis transformation, expressed in terms of the angles  $\Psi_W$ ,  $\gamma$ ,  $\Phi_W$ , is therefore,



$$\mathbf{DCM}_{I \rightarrow W} = \begin{bmatrix} C_{\Psi_W} C_{\gamma} & S_{\Psi_W} C_{\gamma} & -S_{\gamma} \\ C_{\Psi_W} S_{\gamma} S_{\Phi_W} - S_{\Psi_W} C_{\Phi_W} & S_{\Psi_W} S_{\gamma} S_{\Phi_W} + C_{\Psi_W} C_{\Phi_W} & C_{\gamma} S_{\Phi_W} \\ C_{\Psi_W} S_{\gamma} C_{\Phi_W} + S_{\Psi_W} S_{\Phi_W} & S_{\Psi_W} S_{\gamma} C_{\Phi_W} - C_{\Psi_W} S_{\Phi_W} & C_{\gamma} C_{\Phi_W} \end{bmatrix}, \quad (5.3.25)$$

$S_{(\cdot)} = \sin(\cdot), C_{(\cdot)} = \cos(\cdot)$

We can therefore calculate the wind-axis bank angle  $\Phi_W$  from the elements of the inertial- to wind-axis transformation matrix  $\mathbf{DCM}_{I \rightarrow W}$  as follows:

$$\Phi_W = \arctan_2 \left( \frac{[\mathbf{DCM}_{I \rightarrow W}]_{23}}{[\mathbf{DCM}_{I \rightarrow W}]_{33}} \right) \quad (5.3.26)$$

The wind-axis bank angle rate  $\dot{\Phi}_W$  can be calculated using the angle of attack  $\alpha$  and the sideslip angle  $\beta$  measured by the anemometric sensors, the body-axis angular rates  $P$ ,  $Q$ , and  $R$  measured by the gyroscopes, and the body-axis total accelerations  $a_{B_x}$ ,  $a_{B_y}$  and  $a_{B_z}$  obtained from the gravity-compensated accelerometer sensor measurements, and the groundspeed  $\bar{V}$  supplied by the inertial navigation system. The calculation is performed as follows:

The wind-axis roll rate  $P_W$  can be calculated using the rigid body-rotational dynamics of the aircraft derived in the work of Peddle [17],

$$\begin{bmatrix} \dot{\alpha} \\ \dot{\beta} \\ P_W \end{bmatrix} = \begin{bmatrix} -\cos \alpha \tan \beta & 1 & -\sin \alpha \tan \beta \\ \sin \alpha & 0 & -\cos \alpha \\ \cos \alpha \sec \beta & 0 & \sin \alpha \sec \beta \end{bmatrix} \begin{bmatrix} P \\ Q \\ R \end{bmatrix} + \frac{1}{\bar{V}} \begin{bmatrix} \sec \beta & 0 \\ 0 & 1 \\ -\tan \beta & 0 \end{bmatrix} \begin{bmatrix} a_{W_z} \\ a_{W_y} \end{bmatrix} \quad (5.3.27)$$

where  $a_{W_z}$  and  $a_{W_y}$  are the wind-axis normal and lateral accelerations respectively. The body-axis rates  $P$ ,  $Q$  and  $R$  are sensed by the onboard gyroscopes and the wind-axis normal and lateral accelerations are calculated by transforming the body-axis total acceleration vector  $\mathbf{a}_B$  into a wind-axis total acceleration vector  $\mathbf{a}_W$  using the body-axis to wind-axis transformation matrix  $\mathbf{DCM}_{B \rightarrow W}$ ,

$$\mathbf{a}_W = \mathbf{DCM}_{B \rightarrow W} \mathbf{a}_B \quad (5.3.28)$$

The body-axis acceleration vector  $\mathbf{a}_B$  is obtained from the gravity-compensated acceleration sensor measurements. The wind-axis bank angle rate  $\dot{\Phi}_W$  can now be calculated from the wind-axis angular rates  $P_W$ ,  $Q_W$  and  $R_W$ , and the wind-axis bank angle  $\Phi_W$  and flight path angle  $\gamma$  that were previously calculated, using the following equation,

$$\begin{bmatrix} \dot{\Phi}_W \\ \dot{\gamma} \\ \dot{\Psi}_W \end{bmatrix} = \begin{bmatrix} 1 & \sin \Phi_W \tan \gamma & \cos \Phi_W \tan \gamma \\ 0 & \cos \Phi_W & -\sin \Phi_W \\ 0 & \sin \Phi_W \sec \gamma & \cos \Phi_W \sec \gamma \end{bmatrix} \begin{bmatrix} P_W \\ Q_W \\ R_W \end{bmatrix}, \quad |\gamma| \neq \frac{\pi}{2} \quad (5.3.29)$$

Note that the wind-axis roll rate  $P_W$  was calculated using the third row of Equation 5.3.27, and that the wind-axis pitch rate  $Q_W$  and the wind-axis yaw rate  $R_W$  are calculated from the wind-axis normal and lateral accelerations  $a_{W_z}$  and  $a_{W_y}$ , and the groundspeed  $\bar{V}$ , as follows,

$$Q_W = \frac{-a_{W_z}}{\bar{V}} \quad (5.3.30)$$

$$R_W = \frac{a_{W_y}}{\bar{V}} \quad (5.3.31)$$

### 5.3.6 Converting Wind-Axis Reference Angles

The optimal wind-axis bank angle state trajectory  $\Phi_W^*$  and the optimal wind-axis roll rate command input  $P_W^*$  supplied by the trajectory planner must be converted to a body-axis bank angle command  $\Phi_c$  and a body-axis bank angle rate command  $\dot{\Phi}_c$  to be used as references for the DPDR controller. The following sections discuss how to convert the wind-axis roll rate and wind-axis bank angle into their body-axis bank angle rate and body-axis bank angle counterparts.

#### Converting Wind-Axis Bank Angle Rate $\dot{\Phi}_W$ to Body-Axis Bank Angle Rate $\dot{\Phi}$

The optimal wind-axis roll rate command  $P_{Wc}^*$  calculated by the trajectory planner (DP/SQP) actually represents a wind-axis bank angle rate command  $\dot{\Phi}_{Wc}$ , and not a wind-axis roll rate command  $P_{Wc}$ . This is due to the fact that the reduced-order point mass translational dynamics model that is used for the trajectory planning assumes that the wind-axis bank angle rate equals the wind-axis roll rate, i.e. that  $\dot{\Phi}_W = P_W$  (see Equation 4.3.11). Although this is a good approximation, it is not strictly true. The true relationship between wind-axis roll rate and wind-axis bank angle rate is expressed by Equation 5.3.29 in the previous section.

We will therefore consider the optimal wind-axis roll rate command  $P_{Wc}^*$  to be a wind-axis bank angle rate command  $\dot{\Phi}_{Wc}$ , and then convert it to an equivalent body-axis bank angle rate command  $\dot{\Phi}_c$ . The body-axis bank angle rate command  $\dot{\Phi}_c$  is then supplied as a reference to the middle-loop bank angle controller (DPDR controller). The calculation is performed as follows:

We first calculate the wind-axis roll rate command  $P_{Wc}$  using the wind-axis bank angle rate command  $\dot{\Phi}_{Wc}$ , the wind-axis pitch rate  $Q_W$  and the wind-axis yaw rate  $R_W$ , and substituting them into the first row of Equation 5.3.29 and rearranging terms,

$$\dot{\Phi}_{Wc} = P_{Wc}^* \quad (5.3.32)$$

$$P_{Wc} = \begin{bmatrix} 1 & -\sin \Phi_W \tan \gamma & -\cos \Phi_W \tan \gamma \end{bmatrix} \begin{bmatrix} \dot{\Phi}_{Wc} \\ Q_W \\ R_W \end{bmatrix}, \quad |\gamma| \neq \frac{\pi}{2} \quad (5.3.33)$$

with,

$$Q_W = \frac{-a_{Wz}}{\bar{V}} \quad (5.3.34)$$

$$R_W = \frac{a_{Wy}}{\bar{V}} \quad (5.3.35)$$

The wind-axis angular rates  $P_{Wc}$ ,  $Q_W$  and  $R_W$ , are then converted into the equivalent body-axis angular rates  $P$ ,  $Q$  and  $R$ , using an inverse form of Equation 5.3.27 derived in [17] as follows,

$$\begin{bmatrix} P \\ Q \\ R \end{bmatrix} = \begin{bmatrix} 0 & \sin \alpha \\ 1 & 0 \\ 0 & -\cos \alpha \end{bmatrix} \begin{bmatrix} \dot{\alpha} \\ \dot{\beta} \end{bmatrix} + \mathbf{DCM}_{W \rightarrow B} \begin{bmatrix} P_{Wc} \\ Q_W \\ R_W \end{bmatrix} \quad (5.3.36)$$

where the wind-axis to body-axis  $\mathbf{DCM}_{W \rightarrow B}$  is the inverse of the body-axis to wind-axis  $\mathbf{DCM}_{B \rightarrow W}$  shown in the previous section,

$$\mathbf{DCM}_{W \rightarrow B} = (\mathbf{DCM}_{B \rightarrow W})^{-1} = (\mathbf{DCM}_{B \rightarrow W})^T \quad (5.3.37)$$

The body-axis angular rates  $P$ ,  $Q$  and  $R$  are then converted into the body-axis bank angle rate command  $\dot{\Phi}_c$  using Equation 2.3.5 as follows,

$$\dot{\Phi}_c = \begin{bmatrix} 1 & \sin \Phi \tan \Theta & \cos \Phi \tan \Theta \end{bmatrix} \begin{bmatrix} P \\ Q \\ R \end{bmatrix}, \quad |\Theta| \neq \frac{\pi}{2} \quad (5.3.38)$$

The body-axis bank angle rate command  $\dot{\Phi}_c$  is then supplied as the bank angle rate reference  $\dot{\Phi}_{ref}$  for the DPDR controller (acting as inner-loop roll rate controller).

### Converting Wind-Axis Bank Angle $\Phi_W$ to Body-Axis Bank Angle $\Phi$

To translate a wind-axis bank angle reference  $\Phi_W$  into a body-axis bank angle  $\Phi$  we simply use the reverse of the method used to sense the wind-axis bank angle, discussed in §5.3.5. The body-axis bank angle command  $\Phi_c$  can be calculated from the angle of attack  $\alpha$  and the sideslip angle  $\beta$  measured by the anemometric sensors, the optimal wind-axis bank angle  $\Phi_W^*$  supplied by the trajectory planner (DP/SQP), and the estimated flight path angle  $\gamma$ , and the estimated flight path heading  $\Psi_W$  supplied by the aircraft's onboard state estimator. The calculation is performed as follows:

We first calculate the inertial- to body-axis transformation matrix  $\mathbf{DCM}_{I \rightarrow B}$  using,

$$\mathbf{DCM}_{I \rightarrow B} = \mathbf{DCM}_{W \rightarrow B} \mathbf{DCM}_{I \rightarrow W} \quad (5.3.39)$$

where the wind-axis to body-axis transformation matrix  $\mathbf{DCM}_{W \rightarrow B}$  is calculated by taking the inverse of the body-axis to wind-axis transformation matrix  $\mathbf{DCM}_{B \rightarrow W}$  (calculated using Equation 5.3.24), and the inertial-axis to wind-axis transformation matrix  $\mathbf{DCM}_{I \rightarrow W}$  is calculated by substituting the optimal wind-axis bank angle  $\Phi_W^*$ , the estimated flight path angle  $\gamma$  and estimated flight path heading  $\Psi_W$  into Equation 5.3.25. The body-axis bank angle command  $\Phi_c$  is then calculated using the elements of the inertial- to body-axis transformation matrix  $\mathbf{DCM}_{I \rightarrow B}$ , as follows,

$$\Phi_c = \arctan_2 \left( \frac{[\mathbf{DCM}_{I \rightarrow B}]_{23}}{[\mathbf{DCM}_{I \rightarrow B}]_{33}} \right) \quad (5.3.40)$$

The body-axis bank angle command  $\Phi_c$  is then supplied as the bank angle reference  $\Phi_{ref}$  for the middle-loop bank angle controller (DPDR controller).

Note that the estimated flight path angle  $\gamma$  and estimated flight path heading  $\Psi_W$  can be calculated using the estimated inertial-axis velocity co-ordinates of the aircraft, namely the north velocity  $V_N$ , east velocity  $V_E$ , and down velocity  $V_D$ ,

$$\gamma = \arctan_2 \left( \frac{-V_D}{\sqrt{V_N^2 + V_E^2}} \right) \quad (5.3.41)$$

$$\Psi_W = \arctan_2 \left( \frac{V_E}{V_N} \right) \quad (5.3.42)$$

where the velocity co-ordinates can be expressed by the simple differential equations relating the aircraft position to its velocity,

$$\begin{bmatrix} \dot{N} \\ \dot{E} \\ \dot{D} \end{bmatrix} = \begin{bmatrix} V_N \\ V_E \\ V_D \end{bmatrix} \quad (5.3.43)$$

## 5.4 Trajectory Execution Results

All four control schemes presented in the previous section were implemented and verified using the full non-linear NASA GTM simulation model. The outer-loop trajectory planner generated the optimal recovery trajectory references using the SQP algorithm for a specified initial upset condition and fed the commands to the trajectory execution controllers on the GTM model which then executed the trajectories. Note that the SQP trajectory planner uses a second-order model of the angle of attack dynamics for all the final result cases, with the reasoning discussed in section §5.4.6.

First a comparative result case is given where the executed trajectory of all four control schemes is compared. A more detailed discussion of different result cases is then given for each of the four control schemes. A subsection then also discusses the noteworthy issues encountered during the implementation of the control schemes and their resolution. This is then followed by a summary of all the control scheme results. Finally, a conclusion is given on the performance of the four control schemes.

### 5.4.1 Comparative Control Scheme Results

Figure 5.11 shows an illustrative comparison of the state trajectory execution results of all four the control schemes presented in this chapter. The plots show the optimal state trajectory ( $\gamma^*$ ,  $\bar{V}^*$ ,  $\Phi_W^*$ ) planned by the SQP trajectory planner, and the actual state trajectories ( $\gamma$ ,  $\bar{V}$ ,  $\Phi_W$ ) executed by each of the four different control schemes. The optimal altitude loss trajectory  $\Delta h^*$  and the actual altitude loss trajectories  $\Delta h$  are also shown.

All recovery trajectories recover from the same initial upset condition of Table 5.1, where the aircraft is at an inverted bank angle with a descending flight path angle. The initial airspeed is within the normal range, with the angle of attack set to a normal trim value, as well as the initial engine thrust set to the trim setting. The sideslip angle is initialised to zero.

Table 5.1: Initial upset condition

State	Value	Units
$\bar{V}_{initial}$	110	kn
$\gamma_{initial}$	-15	degrees
$\alpha_{initial}$	3	degrees
$\beta_{initial}$	0	degrees
$\Phi_{W_{initial}}$	130	degrees
$T_{initial}$	25.2	Newton

The simulation results show that the ‘input-only’ control scheme executes the planned state trajectory with no lag, but that the executed state trajectory gradually deviates from the

planned trajectory over time, since no state feedback is used. The peak altitude loss of the executed trajectory agrees well with the planned altitude loss predicted by the SQP algorithm. After the recovery is complete, the aircraft slowly loses altitude, since no state feedback is used. This suggests that the normal flight control system should be re-activated immediately after the recovery has been completed in order to maintain the altitude.

The simulation results show that the ‘state only’ control scheme produces an executed state trajectory that lags significantly behind the planned trajectory, but that the flight path angle and wind-axis bank angle eventually follow their final references with zero steady-state error. The trajectory lag causes the peak altitude loss of the executed trajectory to be significantly more than the planned altitude loss predicted by the SQP algorithm.

The simulation results show that the ‘combined state and input’ control scheme and the ‘compensated state’ control scheme both produce executed state trajectories with minimal lag, and also that their flight path angles and wind-axis bank angles follow their final references with zero steady-state error. The ‘combined state and input’ control scheme gives the best performance overall, while the ‘compensated state’ scheme has slight over and undershoot in the executed flight path angle trajectory due to no explicit angle of attack regulation. However, the ‘compensated state’ control scheme can be used for flight control architectures where the references to the inner-loop controllers are not accessible when the middle-loop controllers are active. Both the ‘combined state and input’ and ‘compensated state’ scheme’s altitude loss trajectory matches closely to the predicted altitude loss trajectory.

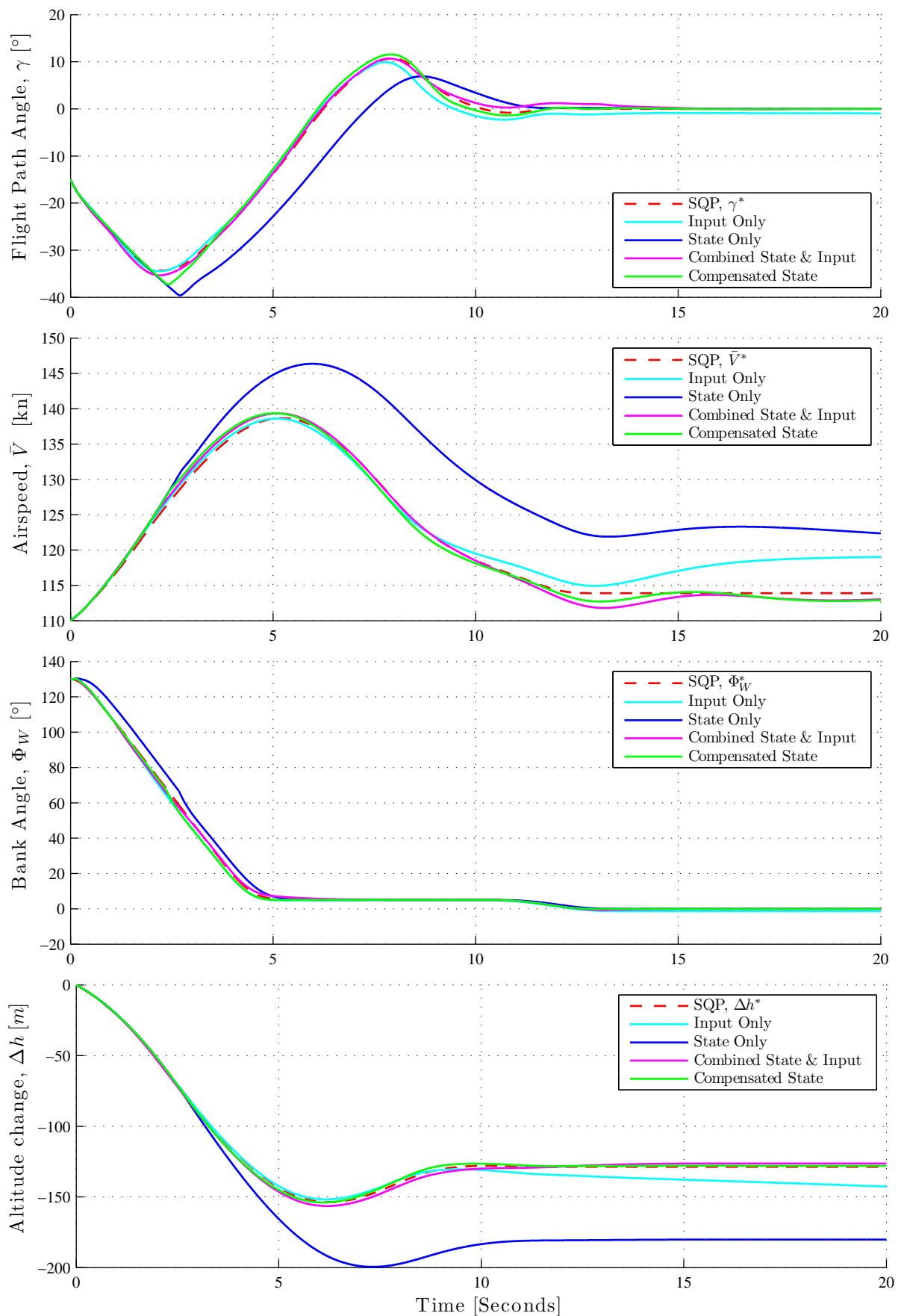


Figure 5.11: Comparison of state trajectories and altitude loss for all four control schemes presented

### 5.4.2 Input Commands Only Control Scheme

Figures 5.12 to 5.14 show simulation results of the GTM system responses for recovering from the same initial upset condition of Table 5.2 using the ‘input only’ scheme. The aircraft is at a high initial bank angle with a steep descending flight path angle. The initial airspeed is set to an underspeed condition, approaching stall speeds, with the angle of attack set to a normal trim value, as well as the initial engine thrust set to the trim setting. The sideslip angle is initialised to zero.

Table 5.2: Initial upset condition

State	Value	Units
$\bar{V}_{initial}$	40	kn
$\gamma_{initial}$	-30	degrees
$\alpha_{initial}$	3	degrees
$\beta_{initial}$	0	degrees
$\Phi_{W_{initial}}$	75	degrees
$T_{initial}$	25.2	Newton

Figure 5.12 shows a comparison of the state trajectories executed by the ‘input only’ control scheme that uses a normal acceleration controller (DQ controller) variant and one that uses a gain-scheduled angle of attack controller variant. Figure 5.13 shows the input signals associated when using the normal acceleration controller variant and Figure 5.14 shows the input signals associated with the gain scheduled angle of attack controller variant.

The SQP trajectory planner’s solution simultaneously recovers the flight path angle, airspeed and bank angle. The flight path angle dips initially, since the lift force vector is insufficient to maintain the flight path angle at high bank angles. As the bank angle is recovered and the lift force vector regains more effect, the SQP gives a ‘pull up’ command (high angle of attack command) to recover the flight path angle. The descending flight path angle causes the airspeed to increase, and the SQP uses gravity with the negative flight path angle to recover the airspeed.

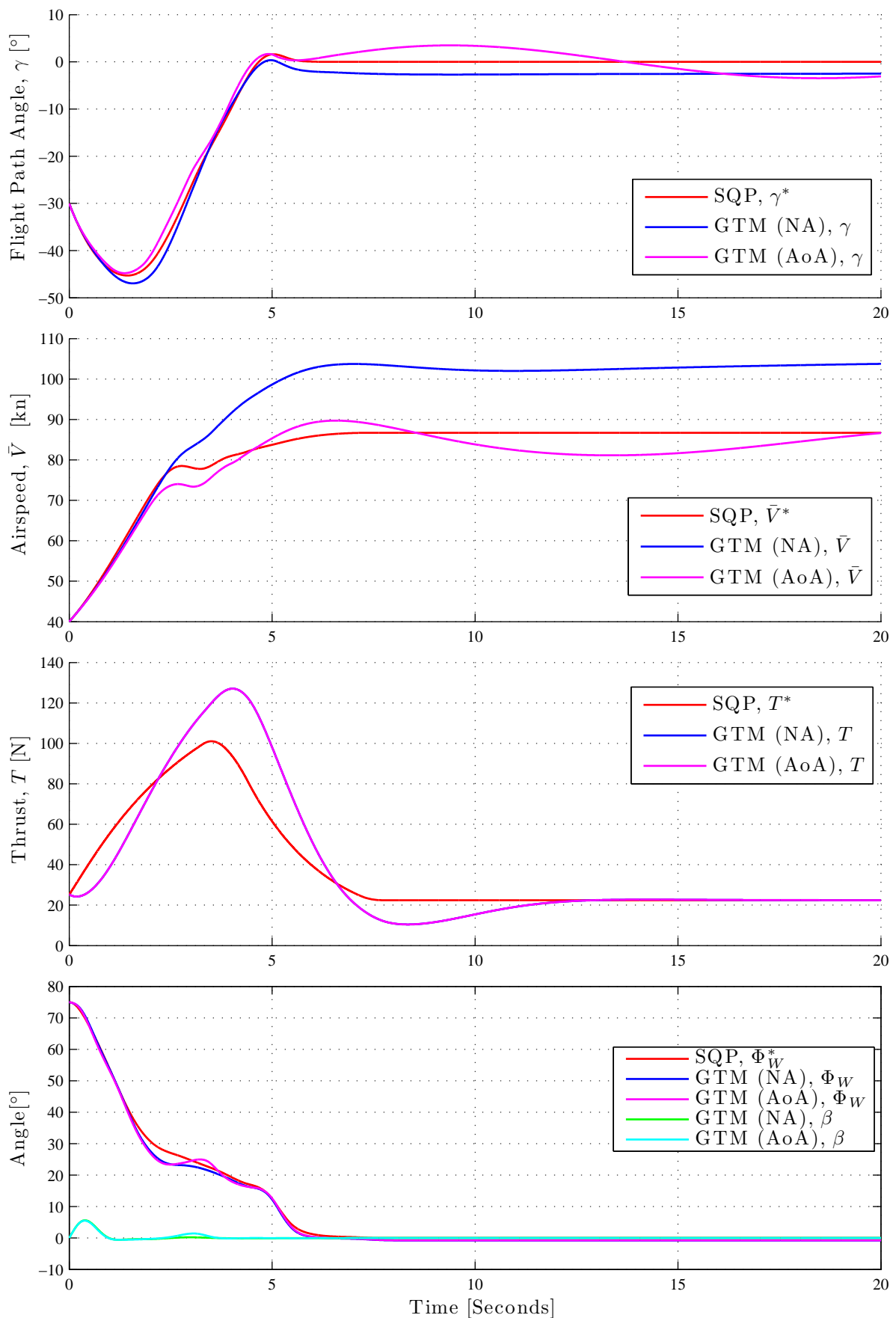


Figure 5.12: State trajectory response using 'input only' control scheme that compares normal acceleration and angle of attack controller variants for a recovery trajectory that uses higher angles of attack



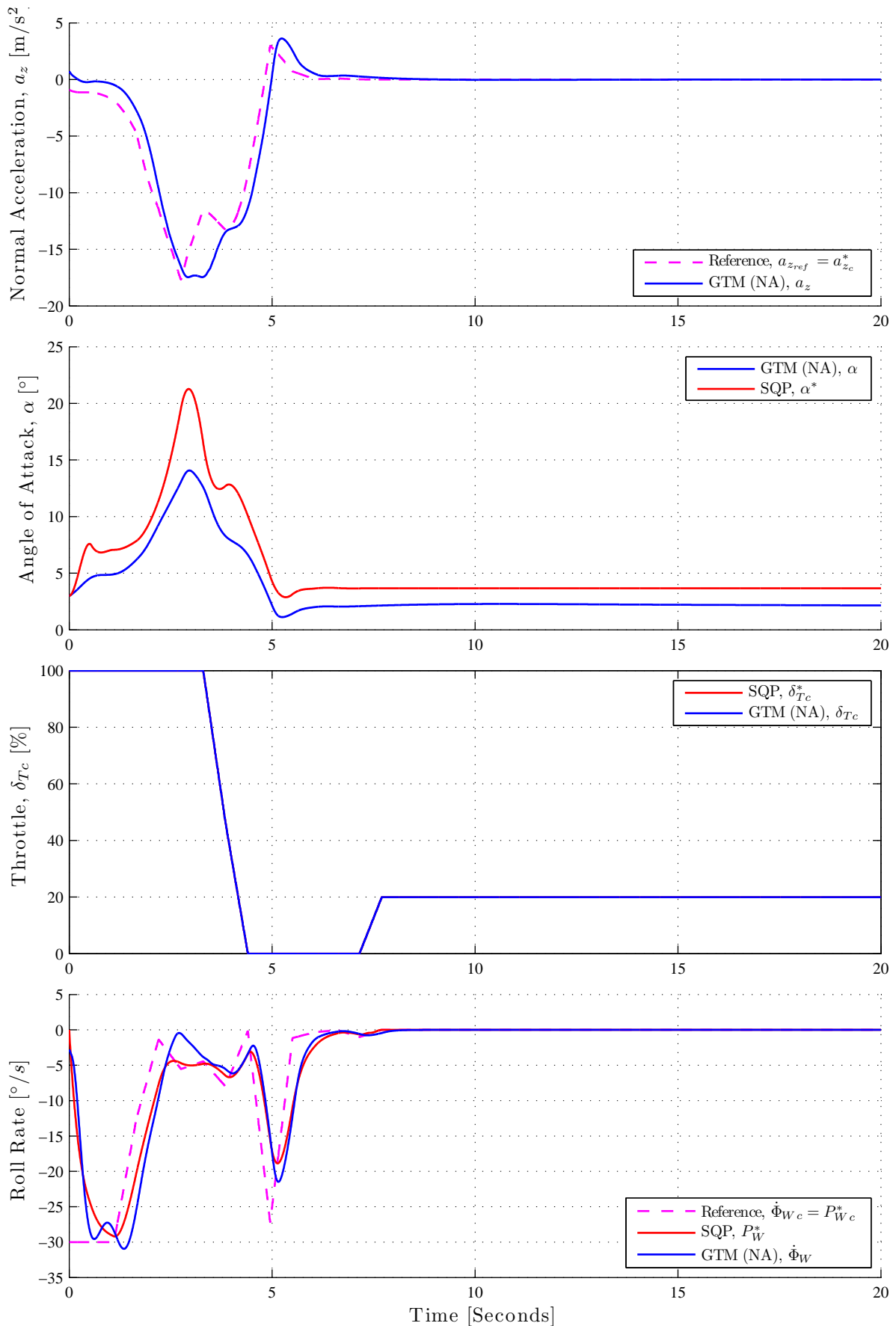


Figure 5.13: Input trajectory response using ‘input only’ control scheme for normal acceleration controller variant for a recovery trajectory that uses higher angles of attack

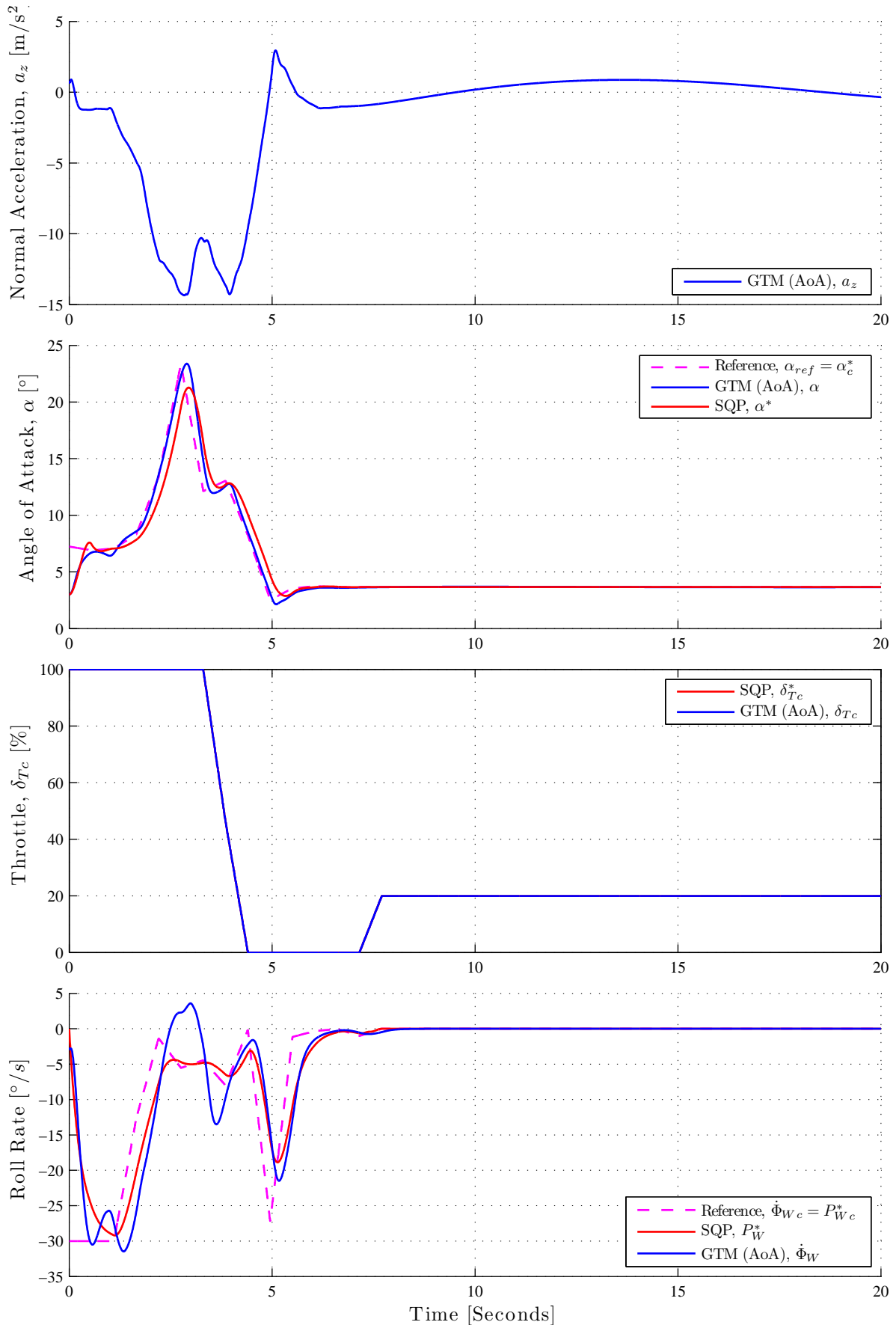


Figure 5.14: Input trajectory response using ‘input only’ control scheme for angle of attack controller variant for a recovery trajectory that uses higher angles of attack

Figure 5.12 shows that the executed flight path angle trajectories of both the normal acceleration variant and the angle of attack variant follow the planned flight path angle trajectory well and remain within the flight path angle limits imposed on the system. The executed flight path angle trajectory of the normal acceleration variant ends with a constant steady-state error, while the executed trajectory of the angle of attack variant exhibits phugoid motions after the flight path angle has been recovered. The executed airspeed trajectory of the angle of attack variant follows the planned airspeed trajectory well, however the executed airspeed trajectory of the normal acceleration variant initially follows the airspeed well, but then deviates significantly once the aircraft starts ‘pulling up’ (using higher angles of attack). The final airspeed for the normal acceleration variant is also much higher than the planned final airspeed. Phugoid motions are also visible in the executed airspeed trajectory of the angle of attack variant. The phugoid motions are primarily a natural result of the exchange between flight path angle and airspeed in a kinematic exchange of potential and kinetic energy. Both the normal acceleration and angle of attack variant’s executed airspeed trajectories remain well within the maximum airspeed limits superscribed by the structural integrity envelope. Both the executed thrust responses of the normal acceleration variant and angle of attack variant are exactly the same, and this is the reason why only the angle of attack variant’s purple plot is visible. The executed thrust response uses more peak thrust than the planned thrust, because GTM’s actual thrust response uses a second-order model, while the SQP modelled the thrust dynamics as a first-order model. This deviation in thrust does not have a significant effect on the executed trajectories, due to the GTM’s very slow thrust to airspeed response. The thrust also stays within the maximum allowed thrust value imposed on the system (maximum value of 136 Newton allowed). The executed wind-axis bank angle trajectories of both the normal acceleration and the angle of attack variants follow the planned wind-axis bank angle trajectory well. Note that the executed wind-axis bank angle trajectories are obtained by converting the measured body-axis bank angle to wind-axis bank angle using the equations in section §5.3.5. There is a slight steady-state error in the executed wind-axis bank angle, due to the trajectories being executed in an open-loop manner. The deviation in the executed bank angle trajectory for both variants at around  $t = 2.5$  seconds is due to a slight deviation in the executed wind-axis bank angle rate response from the planned wind-axis bank angle rate response. The sideslip angle response for both variants are regulated to remain close to zero.

Figure 5.13 shows that the executed normal acceleration follows the planned normal acceleration command well for the normal acceleration controller variant, and with zero steady-state error. The normal acceleration controller performs well and is able to control the normal acceleration output to track the time varying normal acceleration command. The executed normal acceleration response also stays within the load factor limits prescribed by the structural integrity envelope (minimum of -1 g and a maximum of 2.5 g, which is equivalent to a minimum of -9.8 m/s and a maximum of 24.5 m/s respectively). The executed angle of attack signal for the normal acceleration controller variant has a similar shape to the planned angle of attack signal, but with an offset. The executed angle of attack signal is significantly lower than the planned angle of attack during the recovery (planned peak value of 21 degrees vs executed peak value of 14 degrees), which accounts for the large difference between the planned airspeed trajectory and the executed airspeed trajectory. Due to the lower angle of attack, the aircraft will experience significantly less drag in the executed trajectory relative to the planned trajectory. The GTM’s throttle input is exactly the same as the throttle input planned by the SQP, as it uses the SQP’s input as reference. The executed wind-axis bank angle rate follows the planned wind-axis bank angle rate very well and exhibits zero tracking error in steady state. The wind-axis bank angle lags slightly behind the wind-axis bank angle command, but the SQP algorithm planned for this by taking the roll rate delay into account. The sharp ‘dip’ in the wind-axis bank angle rate at  $t = 5$  seconds is due to the trajectory planner’s solution, and is not caused by the trajectory execution. The control scheme simply tries to track the

wind-axis bank angle rate references given to the roll rate and sideslip controller. The executed wind-axis bank angle rate stays inside the bank angle rate limits that were imposed on the system.

Figure 5.14 shows that the normal acceleration does not explicitly follow a normal acceleration reference, but is now an indirect result of the angle of attack control in the case of the angle of attack controller variant. The phugoid motion seen in the airspeed and flight path angle states also manifests in the normal acceleration, mainly due to the variation in the airspeed. (The lift varies because the airspeed varies, while the angle of attack remains constant.) However, the executed load factor signal remain within the load factor limits prescribed by the structural integrity envelope. The executed angle of attack signal follows the planned angle of attack very well and follows the angle of attack reference with zero error in steady state (unlike the angle of attack for the normal acceleration variant, which had an offset). The angle of attack controller controls the angle of attack very well, as is evident from the fact that the angle of attack output follows the time-varying angle of attack reference well. The fact that the executed angle of attack follows the planned angle of attack well, is the reason why the executed airspeed trajectory follows the planned airspeed trajectory well. The executed angle of attack slightly exits the angle of attack limit imposed on the system (peak value of 23 degrees vs 21 degree limit), but still remains far from reaching the stall angle of attack. The executed wind-axis bank angle rate follows the planned wind-axis bank angle rate relatively well and exhibits zero tracking error in steady state. Note that the wind-axis bank angle rate is calculated by using the measured body-axis angular rates, angles and accelerations using the conversion equations from section §5.3.5. The larger deviations between  $t = 2$  and  $t = 4$  seconds is due to larger pitch rates having an effect on the wind-axis bank angle rate, due to the higher executed angles of attack. As with the normal acceleration variant, the wind-axis bank angle lags slightly behind the wind-axis bank angle command, but the SQP algorithm's solution accounted for this. The sharp 'dip' in the wind-axis bank angle rate at  $t = 5$  seconds is due to the same reason as discussed previously in the case of the normal acceleration controller variant, and is part of the trajectory planner's solution. The executed wind-axis bank angle rate also stays inside the bank angle rate limits.

The simulation results show that the 'input only' scheme using the angle of attack controller performs better than the 'input only' scheme using the normal acceleration controller. This is attributed to the fact that the angle of attack controller results in better airspeed trajectory tracking than the normal acceleration controller, especially for recovery trajectories that use high angles of attack. This is due to the airspeed response being primarily influenced by the drag produced over the time scales of these trajectories, and the drag is strongly influenced by the angle of attack. Normal acceleration is insensitive to airspeed, thus by directly commanding angle of attack we have a stronger influence on the airspeed. The executed wind-axis bank angle trajectory shows a steady-state error in both the normal acceleration and angle of attack controller cases. This is due to the fact that the state trajectories are executed in an open-loop manner, and at high angles of attack and deviations from the optimal angle of attack, an error is introduced in the conversion from wind-axis bank angle rate to body-axis bank angle rate, and this error is then integrated to a final non-zero reference offset.

It was found that when the planned optimal angle of attack trajectory only uses low angles of attack (smaller than 10 degrees), the performance of the 'input only' scheme using a normal acceleration controller improves. The translated normal acceleration commands produce the expected angle of attack trajectory and therefore improves the airspeed trajectory tracking. This is illustrated by a state trajectory executed by the 'input only' control scheme example using the normal acceleration controller from the initial upset condition defined in Table 5.3. Figures 5.15 and 5.16 show a simulation result of the GTM's state trajectory and input signal responses respectively for recovering from this particular initial condition, using lower angles

of attack. The initial upset condition is the same inverted bank angle and descending flight path angle upset used in the previous section.

The planned SQP solution prioritises the bank angle recovery, while also recovering flight path angle and airspeed simultaneously. The flight path angle initially dips due to the aircraft being inverted and the lift vector is pointed downwards, and thus cannot oppose gravity and is unable to positively affect the flight path angle. As the aircraft returns to ‘right-side up’, i.e. a bank angle below 90 degrees, the lift vector can once again oppose gravity and the SQP gives a ‘pull up command’ (high angle of attack) to recover the flight path angle. The negative flight path angle causes the aircraft to go into an overspeed condition due to gravity, and the SQP then uses positive flight path angle to recover the airspeed.

Table 5.3: Initial upset condition

State	Value	Units
$\bar{V}_{initial}$	110	kn
$\gamma_{initial}$	-15	degrees
$\alpha_{initial}$	3	degrees
$\beta_{initial}$	0	degrees
$\Phi_{W_{initial}}$	130	degrees
$T_{initial}$	25.2	Newton

Figure 5.15 shows that the executed flight path angle follows the planned flight path angle very well, but with a small steady-state error in the flight path angle after the recovery has been completed. (This is due to the open-loop nature of the ‘input only’ control scheme.) The executed flight path angle also remain within acceptable flight path angle limits. The executed airspeed trajectory follows the planned airspeed trajectory very well during the recovery manoeuvre, and remains within the maximum airspeed limit prescribed by the structural integrity envelope. After the recovery has been completed, the executed airspeed slowly deviates from the planned airspeed, due to the open-loop nature of the ‘input only’ control scheme. The overshoot in the executed thrust signal is due to the same reason as discussed in the previous result case. The thrust still remains within the maximum thrust limit (136 N). The executed wind-axis bank angle trajectory follows the planned trajectory very well, but there is a small steady-state error in the wind-axis bank angle after the recovery has been completed (due to the open-loop nature of the ‘input only’ control scheme). However, the bank angle steady-state error has improved compared to the previous case, as less error is introduced at lower angles of attack in the conversion from wind-axis roll rate to body-axis bank angle rate. The sideslip angle is successfully controlled to remain close to its zero reference point.

Figure 5.16 shows that the executed normal acceleration using the normal acceleration controller variant follows the planned normal acceleration very well, and follows the normal acceleration reference with zero error in steady state. The normal acceleration controller does an excellent job of controlling the normal acceleration output to follow the time-varying normal acceleration reference in this case and the executed normal acceleration still obeys the load factor limits prescribed by the structural integrity envelope. A offset can still be observed between the executed angle of attack signal and the planned angle of attack signal using the normal acceleration controller variant, but the difference is much smaller for this lower angle of attack recovery trajectory. The executed angle of attack signal has a similar shape to the planned angle of attack signal, and for the lower angle of attack trajectory the largest difference is about 1 degree, while for the higher angle of attack trajectory the largest difference was 7

degrees (planned peak value of 21 degrees versus executed peak value of 14 degrees). The fact that the executed angle of attack remains closer to the planned angle of attack is the reason for the much smaller difference between the planned airspeed trajectory and the executed airspeed trajectory. The executed angle of attack also remains well within the specified angle of attack limits. As said previously, the GTM uses the same throttle input as the planned throttle input signal. The executed wind-axis bank angle rate follows the planned wind-axis bank angle rate relatively well, with the wind-axis bank angle lagging slightly behind the wind-axis bank angle command. However, as stated previously, this has been taken into account by the SQP trajectory planner. The ‘bump’ fluctuation in the wind-axis bank angle rate at  $t = 11$  to 13 seconds is due to the same reason as in the previous cases, namely it is a result of the trajectory planning, and is not caused by the trajectory execution. The wind-axis bank angle rate exhibits zero tracking error in steady state and only slightly exits the bank angle rate limits (with a peak value of 33 degrees per second which is 3 degrees per second higher than the 30 degrees per second limit).

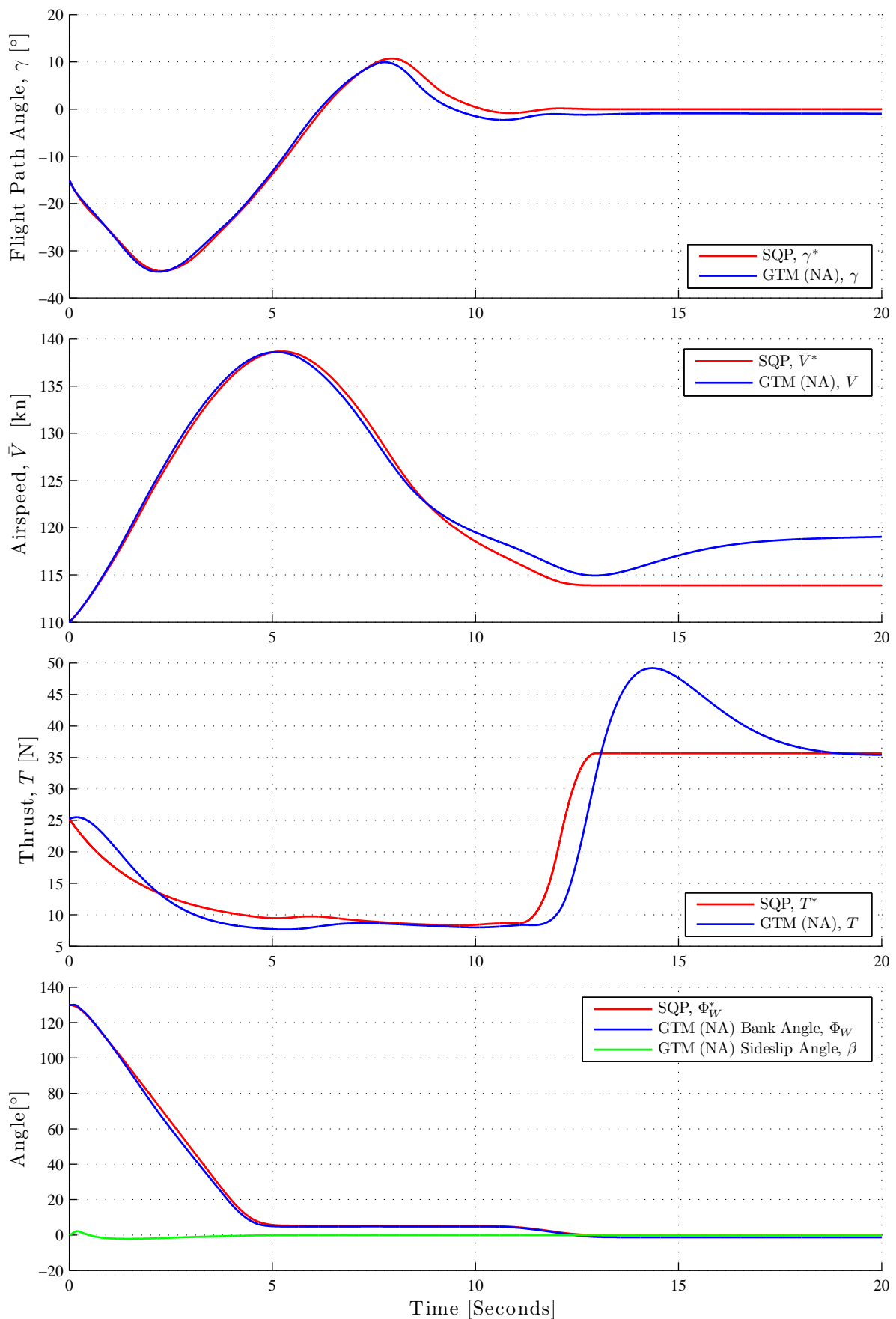


Figure 5.15: State trajectory response using ‘input only’ control scheme and normal acceleration controller variant for a recovery trajectory that uses lower angles of attack

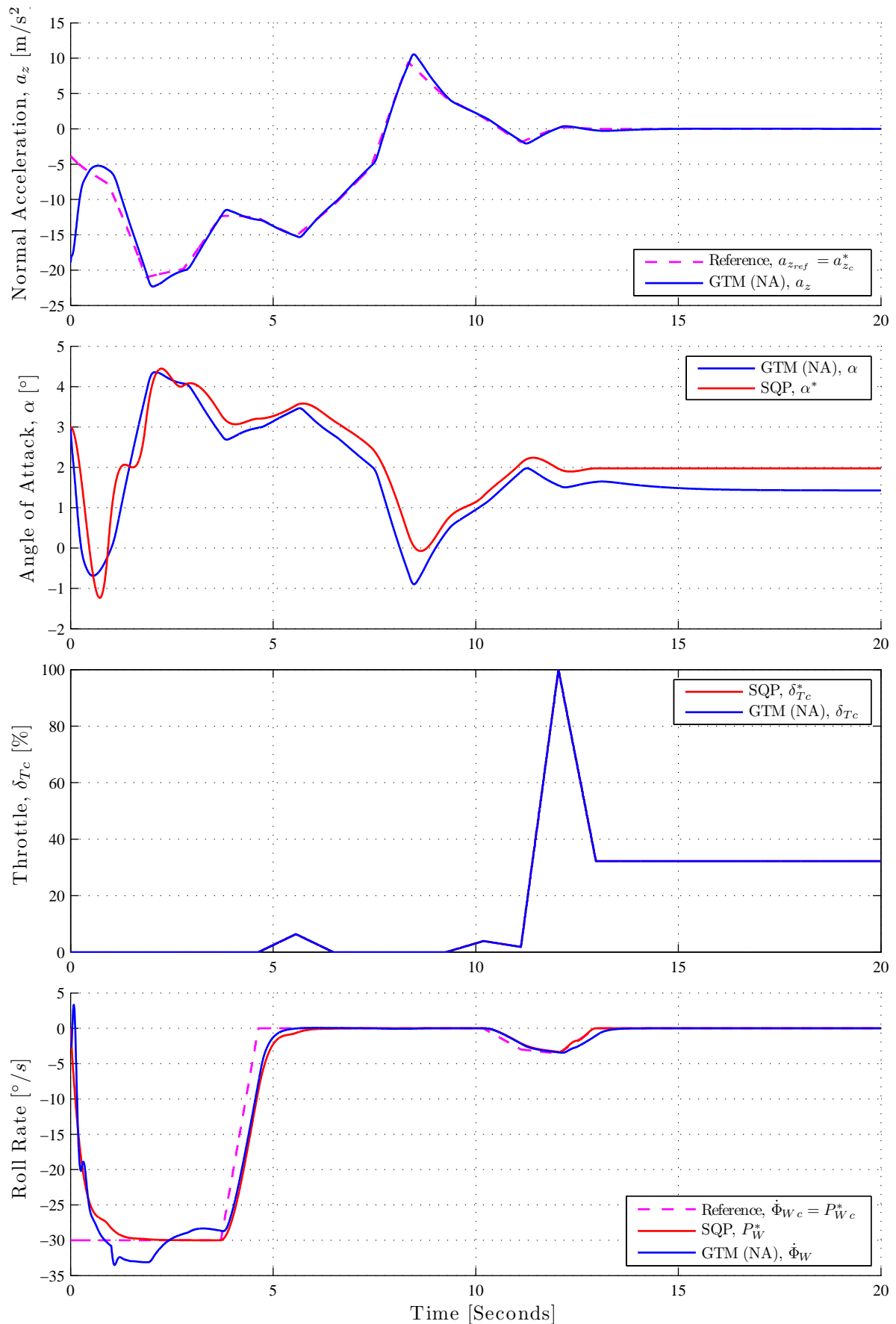


Figure 5.16: Input trajectory response using ‘input only’ scheme and normal acceleration controller variant for a recovery trajectory that uses lower angles of attack



The simulation shows that the normal acceleration controller variant performs better at lower angles of attack, and this is due to the fact that the normal acceleration controller is closer to its single angle of attack linearisation point at lower angles of attack.

The optimal wind-axis roll rate command  $P_{Wc}^*$  can be interpreted as a wind-axis bank angle rate command  $\dot{\Phi}_{Wc}^*$  and is converted into a body-axis bank angle rate command for the roll rate and sideslip controller (DPDR controller). The GTM wind-axis bank angle rate tracks the SQP's expected wind-axis roll rate response fairly well, meaning that the conversion process of section §5.3.6 is able to compensate for the assumption that  $\dot{\Phi}_W = P_W$ . Thus the conversion takes most of the effect of the pitch and yaw rates on bank angle rate into account.

The deviations in the thrust response is due to the fact that the GTM's actual thrust delay model uses a second order response model and we approximated the delay with a first order model. However, because of the time scales of the recoveries and the slow thrust to airspeed response of the GTM, these deviations in the thrust does not have a major effect on the recovery trajectory.

The simulation results show that the 'input only' control scheme produces an executed state trajectory that closely follows the planned state trajectory. This means that the optimal inputs supplied to the inner-loop controllers produce an executed state trajectory that agrees with the planned state trajectory, purely through open-loop control and without using state feedback. This, in turn, indicates that the reduced-order model used for the trajectory planning (by the SQP algorithm) is sufficiently representative of the full aircraft dynamics, and validates the use of the reduced-order model for the trajectory planning.

The 'input only' scheme is not guaranteed to be robust as we are only giving the commands in an open-loop configuration to the inner-loop controllers which cannot correct expected numerical integration errors in the solution or disturbances. The executed trajectory shows a steady-state error in the airspeed and flight path angle states. The normal acceleration controller variant exhibits a constant flight path angle steady-state error whereas the angle of attack controller variant exhibits a steady-state error in the form of undesirable phugoid motions in the airspeed and flight path angle states. This is to be expected as the inner-loop longitudinal controllers only control the short-period motions of the aircraft. This configuration has no disturbance rejection capabilities, so if there is a disturbance force such as wind, the states will deviate more from the expected optimal state trajectories.

### 5.4.3 State Commands Only Control Scheme

Figures 5.17 and 5.18 show a simulation result of the GTM's state trajectory and input signal responses respectively for recovering from the initial upset condition of Table 5.4 using the 'state only' control scheme. The same inverted bank angle with descending flight path angle initial upset condition is used as for the previous section. The purple reference command plots are the references that the individual controllers are given during the recovery. In this case the normal acceleration controller's acceleration command is given only from the flight path angle controller.

Table 5.4: Initial upset condition

State	Value	Units
$\bar{V}_{initial}$	110	kn
$\gamma_{initial}$	-15	degrees
$\alpha_{initial}$	3	degrees
$\beta_{initial}$	0	degrees
$\Phi_{W_{initial}}$	130	degrees
$T_{initial}$	25.2	Newton

Figure 5.17 shows that the executed flight path angle trajectory follows the planned flight path angle, but with a significant lag, but exhibits no steady-state error when the recovery is completed. The reason is that the middle-loop flight path angle controller is a type 1 system, and can only track a varying ramp signal with a constant tracking error. Furthermore, when only feedback control is used, it is expected that there will always be some execution lag, since the feedback controller does not have any foreknowledge of the reference trajectory (that a feed-forward term would provide) and first needs to measure a tracking error in order to produce the correct commands to follow the trajectory. However, the executed flight path angle remains within the flight path angle limits imposed on the system. The sharp negative flight path angle peak at  $t = 2.5$  seconds, is when the flight path angle controller begins to issue non-zero normal specific acceleration commands when the bank angle is below 70 degrees. The flight path angle controller is designed to give a specific acceleration command of zero and only let gravity affect the flight path angle while the bank angle is above 70 degrees, in order to avoid giving equivalent negative load factor commands. Thus when the bank angle transitions below 70 degrees the flight path angle controller is allowed to give non-zero specific acceleration commands to control the flight path angle. The executed airspeed trajectory initially follows the planned trajectory, but then severely deviates and overshoots the planned airspeed trajectory. This is due to the angle of attack deviating significantly from the planned angle of attack, producing less drag than expected. The peak executed airspeed briefly and slightly exits the maximum airspeed limits due to the poor tracking of the planned airspeed, and the airspeed trajectory also shows an error in steady-state. The executed thrust signal shows the exact same response as in the previous section and the overshoot was also discussed in the previous section. The executed wind-axis bank angle trajectory follows the planned wind-axis bank angle trajectory, but with a noticeable lag. This is also due to the fact that the middle-loop bank angle controller does not have foreknowledge of the reference trajectory as discussed previously in the case of the flight path angle controller. However, there is no steady-state error present in the executed wind-axis bank angle trajectory. The slight constant offset from zero in the wind-axis bank angle trajectory between  $t = 5$  to 11 seconds is due to the trajectory planner's solution, and not the trajectory execution. The wind-axis bank angle trajectory solution generated by the SQP exhibits this 10 degree offset, and the middle-loop bank angle controller simply tracks the planned trajectory given to it as reference. The sideslip angle is also successfully controlled to remain close to zero degrees.

Figure 5.18 shows that the normal acceleration controller controls the normal acceleration output very well and tracks the commanded normal acceleration from the flight path angle controller closely with zero steady-state error. However, the normal acceleration does slightly exit the load factor limit at  $t = 2.5$  seconds. This is because the flight path angle controller begins to issue non-zero normal specific acceleration commands when the bank angle transitions below 70 degrees at  $t = 2.5$  seconds, and this transition causes an aggressive normal acceleration command. The executed angle of attack does not follow the planned angle of attack well, with

a large deviation observed while the flight path angle controller only allows gravity to affect the normal acceleration command between  $t = 0$  to 2.5 seconds. The normal acceleration signal from the flight path angle controller does not produce the equivalent planned angle of attack signal. However, the angle of attack does stay well within the angle of attack limits. The throttle signal has the same response as seen in the previous section. The wind-axis bank angle rate tracks the planned wind-axis bank angle rate, but with some lag. This is due to the fact that the executed wind-axis bank angle rate is not controlled directly, because we are using a bank angle and sideslip controller, but the wind-axis bank angle tracks the planned wind-axis bank angle with zero steady-state error. The ‘spike’ deviation in wind-axis bank angle at  $t = 2.5$  seconds is caused by the large pitch rate disturbance which affects the bank angle rate, and the pitch rate is produced by the large normal acceleration response at  $t = 2.5$ . This also causes the wind-axis bank angle rate to exceed the wind-axis bank angle rate limits of 30 degrees per second. The ‘bump’ fluctuation in the wind axis bank angle rate at  $t = 12$  seconds is due to the same reason as discussed in the previous section, and is due to the trajectory planner not the trajectory execution.

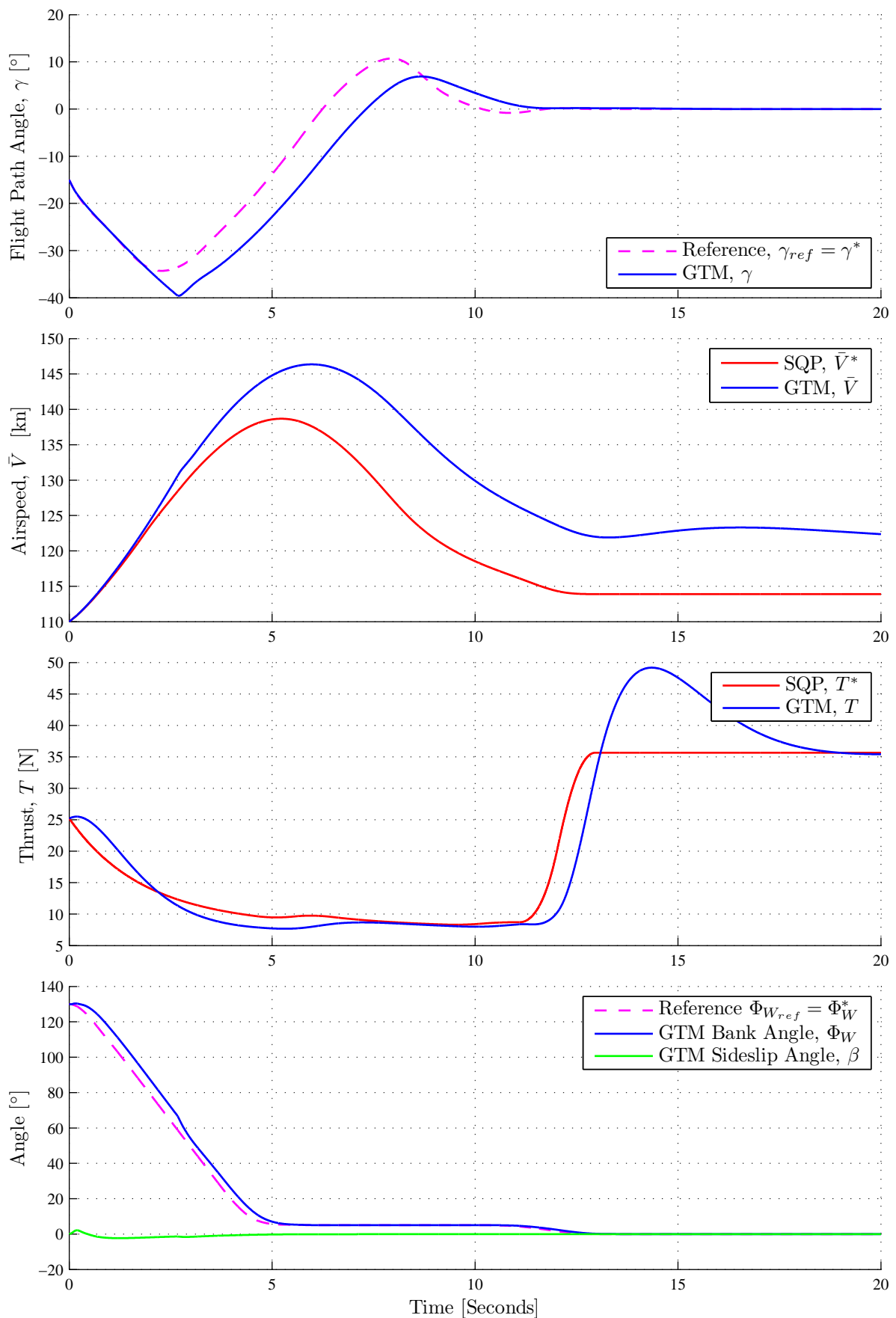


Figure 5.17: State trajectory response using 'state only' control scheme and normal acceleration controller for initial upset condition.

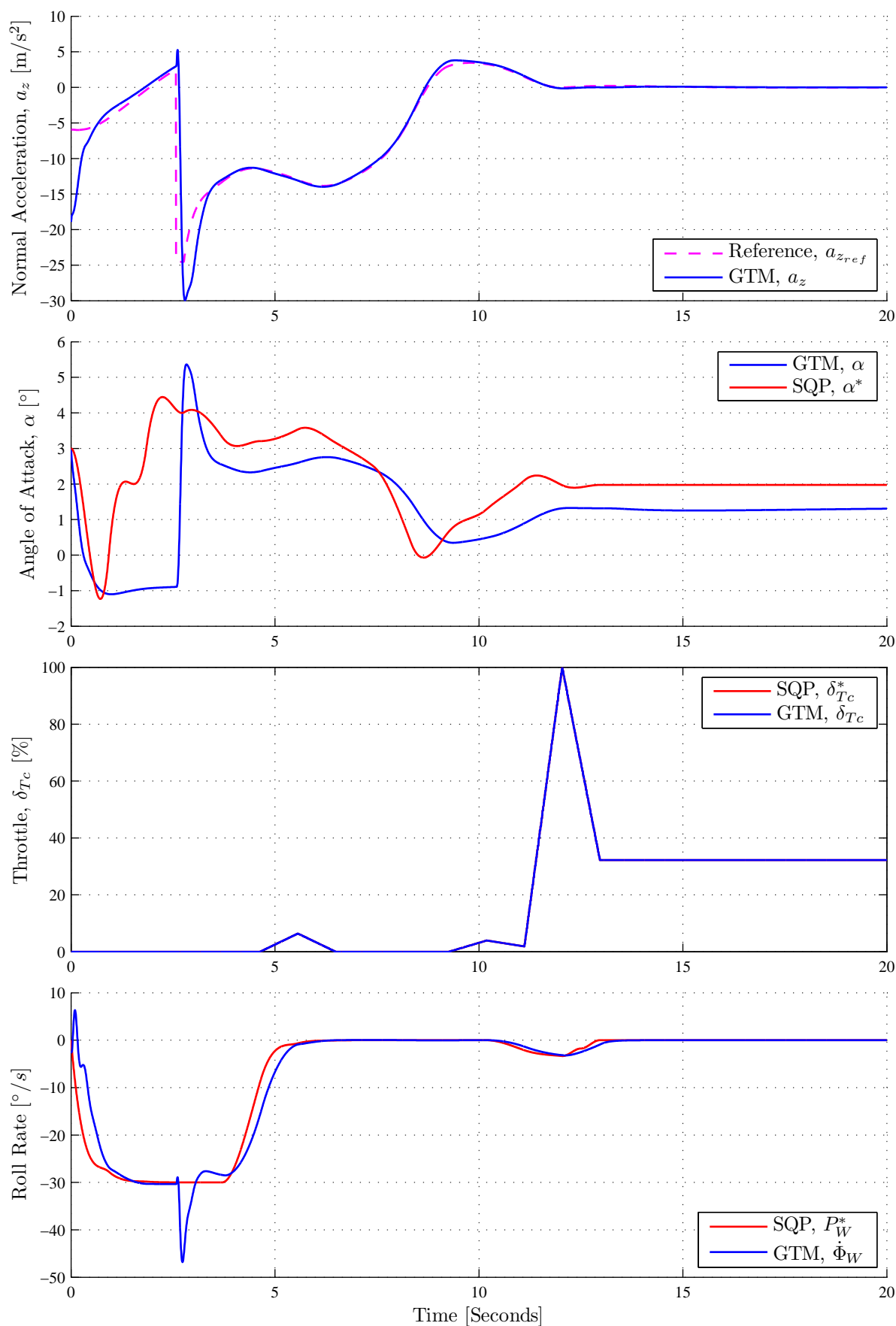


Figure 5.18: Input trajectory response using ‘state only’ control scheme and normal acceleration controller for initial upset condition

The simulation results show that the executed flight path angle and wind-axis bank angle trajectories lag behind the optimal planned state trajectory references from the SQP. However, the ‘state only’ scheme executed trajectory has no flight path angle and wind-axis bank angle steady-state error due to state feedback, unlike the ‘input only’ scheme that had a steady-state error.

The executed airspeed trajectory significantly overshoots the planned airspeed trajectory, and this can be attributed to the same reason for the airspeed overshoot in the case of the ‘input only’ scheme result that uses a normal acceleration controller (for a recovery trajectory using lower angles of attack) in the previous section. The executed angle of attack of the ‘state only’ scheme does not follow the planned angle of attack, resulting in poor airspeed tracking. This is due to the fact that the flight path angle controller’s normal acceleration commands to the normal acceleration controller do not result in the equivalent planned angle of attack commands. The ‘state only’ scheme also controls the airspeed in an open-loop fashion as with the ‘input only scheme’, by only giving the planned input throttle command directly to the GTM’s throttle input. Thus there is no middle-loop controller to minimise the steady-state error in airspeed.

The deviation of the bank angle rate response at around  $t = 2.5$  seconds is due to the fact that the ‘state only’ scheme uses a middle-loop bank angle controller rather than an inner-loop bank angle rate controller, and cannot take the effect of the pitch rate into account. When the flight path angle controller starts issuing non-zero normal specific acceleration commands when the bank angle becomes less than 70 degrees at around  $t = 2.5$  seconds, a big normal acceleration command is given to recover the flight path angle, as seen in the purple normal acceleration reference command. This is what causes the large pitch rates, which in turn also affects the wind-axis bank angle rate significantly at low flight path angles. The conversions from wind-axis bank angle to body-axis bank angle command for the middle-loop bank angle controller cannot take the effect of pitch and yaw rate on bank angle rate into account, and the pitch rate causes a disturbance on the bank angle rate. The ‘input only’ scheme did not suffer from this, since it uses an inner-loop bank angle rate controller, which could take pitch rate into account.

#### 5.4.4 Combined State and Input Control Scheme

Figures 5.19 to 5.21 show a simulation result of the GTM’s state trajectory, input signal and altitude response respectively for recovering from the initial upset condition of Table 5.5 using the ‘combined state and input’ control scheme. The same inverted bank angle with descending flight path angle initial upset condition is used as for the previous sections. The purple normal acceleration reference plot is the total reference signal sent to the normal acceleration controller that includes the superimposed optimal normal acceleration command, the flight path angle correction component and the angle of attack correction component signal.

Table 5.5: Initial upset condition

State	Value	Units
$\bar{V}_{initial}$	110	kn
$\gamma_{initial}$	-15	degrees
$\alpha_{initial}$	3	degrees
$\beta_{initial}$	0	degrees
$\Phi_{W_{initial}}$	130	degrees
$T_{initial}$	25.2	Newton

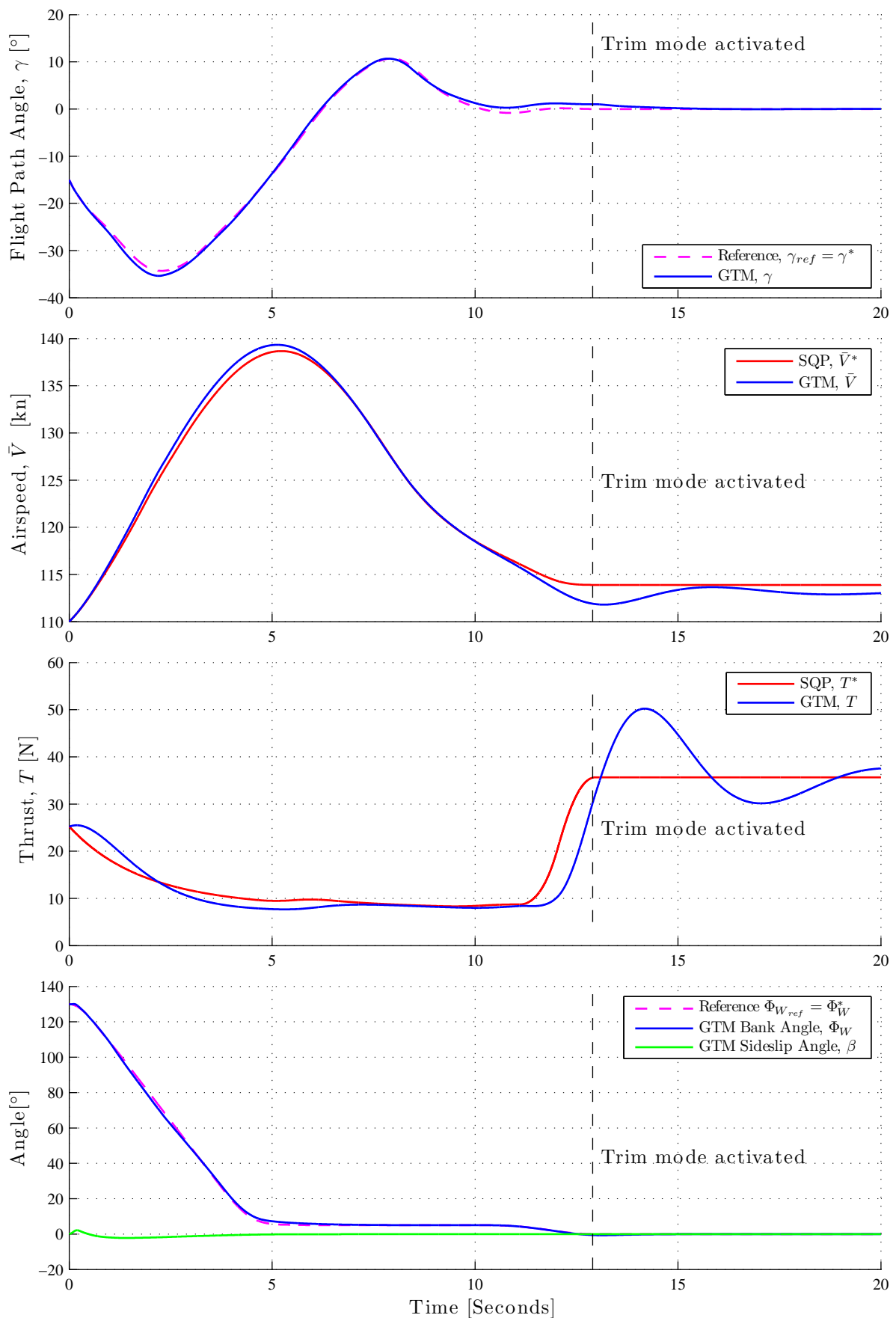


Figure 5.19: State trajectory response using ‘combined state and input’ scheme and normal acceleration controller for a recovery trajectory that uses lower angles of attack

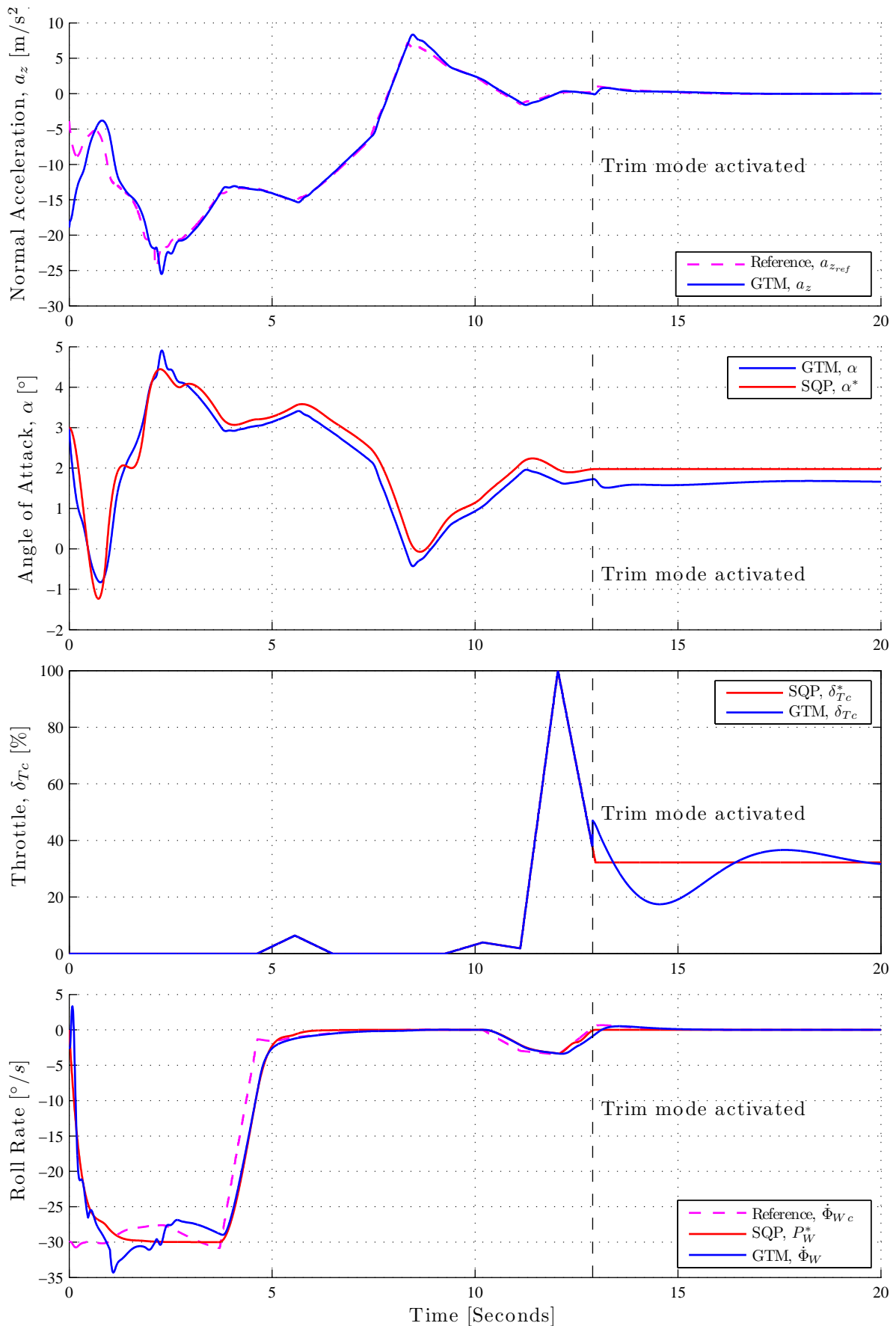


Figure 5.20: Input trajectory response using ‘combined state and input’ scheme and normal acceleration controller for a recovery trajectory that uses lower angles of attack



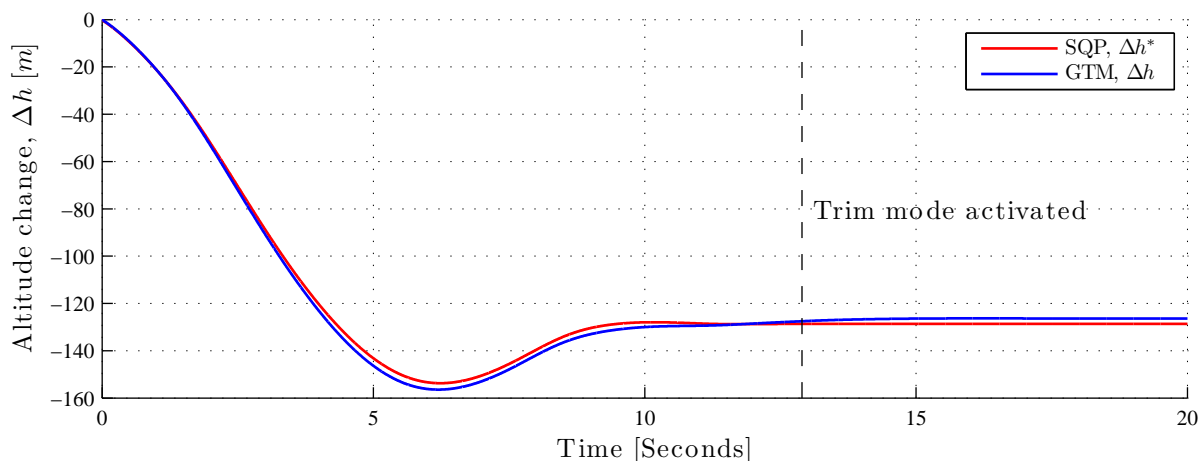


Figure 5.21: Altitude change response using ‘combined state and input’ scheme and normal acceleration controller for a recovery trajectory that uses lower angles of attack

Figure 5.19 shows that the executed flight path angle trajectory tracks the planned flight path angle trajectory very well with minimal lag, and exhibits no steady-state error after recovery is completed (due to the feedback provided by the middle-loop flight path angle controller). The flight path angle controller is switched back to its nominal trim mode after recovery at around  $t = 13$  seconds to trim the aircraft to wings level flight. The executed flight path angle stays within the flight path angle limits imposed on the system. The executed airspeed tracks the planned airspeed quite well, and stays within the maximum airspeed limit prescribed by the structural integrity envelope. The final airspeed is regulated close to the final planned airspeed after the trim airspeed controller is switched on after recovery. The executed thrust signal stays within the maximum thrust limits (of 136 N) and the overshoot in thrust response is due to the same reason as discussed in previous sections. The thrust response does not converge to the final planned thrust value, because the airspeed controller is commanding the thrust to regulate the final airspeed. The wind-axis bank angle tracks the planned wind-axis bank angle very closely and with no steady-state error, due to the middle-loop wind-axis bank angle regulator providing feedback. The slight constant offset from zero in the wind-axis bank angle trajectory between  $t = 5$  to 11 seconds is due to the trajectory planner’s solution, and not the trajectory execution, as discussed in the previous section. The sideslip angle is also regulated to stay close to zero.

Figure 5.20 shows that the normal acceleration controller tracks the normal acceleration command very well and the response exhibits a zero error in steady-state. The normal acceleration controller performs well at controlling the normal acceleration output, and the executed normal acceleration remains within the load factor limits imposed by the structural integrity envelope (minimum of -1 g and a maximum of 2.5 g, which is equivalent to a minimum of -9.8 m/s and a maximum of 24.5 m/s respectively). The middle-loop angle of attack regulator regulates the angle of attack so that the executed angle of attack closely follows the planned angle of attack, with a very slight offset, and this results in good tracking of the planned airspeed trajectory. The angle of attack also stays well within the angle of attack limits. The angle of attack regulator is turned off after recovery at around  $t = 13$  seconds when the Normal flight control laws take back over, and thus there is a steady-state error in the final angle of attack. The throttle signal follows the planned throttle signal exactly until around  $t = 13$  seconds when the airspeed controller is switched back on and commands the throttle after recovery is completed. The executed wind-axis bank angle rate follows the planned wind-axis bank angle rate fairly well, and slightly lags the wind-axis bank angle rate command. However, this is expected,

because the SQP planner takes this into account as discussed in the previous sections. The ‘bump’ fluctuation in the executed wind-axis bank angle rate at around  $t = 13$  seconds is due to the same reason as discussed in previous sections and is due to the trajectory planner and not the trajectory execution. The executed wind-axis bank angle rate exhibits zero steady-state and only slightly exceeds the wind-axis bank angle rate limits (of 30 degrees per second) at  $t = 2$  seconds by 4 degrees per second. This is due to the increased pitch rates at that time in the recovery manoeuvre which is caused by the increased normal acceleration signal.

Figure 5.21 shows that the executed altitude loss trajectory matches the planned altitude loss trajectory of the SQP fairly well, with slightly more peak altitude lost than the planned altitude loss. (The executed trajectory lost around 155 meters altitude versus the planned altitude loss of 150 meters.)

The simulation results show that the executed state trajectories using the ‘combined state and input’ scheme have no lag and follow the planned trajectories well as in the case of the ‘input only’ scheme, but do not deviate as much as the ‘input only’ scheme, and also exhibit no steady-state tracking errors in the executed flight path angle and wind-axis bank angle trajectories.

### Recovery Trajectory that uses Higher Angles of Attack

With the added angle of attack middle-loop feedback controller, the ‘combined state and input’ scheme with the normal acceleration controller now has good angle of attack tracking when higher optimal angle of attack trajectories are used. Recall that the ‘input only’ control scheme variant that uses a normal acceleration controller had poor angle of attack tracking performance for recovery trajectories that use higher angles of attack. Figures 5.22 and 5.24 show a simulation result of the GTM’s state trajectory, input signal and altitude response respectively for recovering from the initial upset condition of Table 5.6 using the ‘combined state and input’ control scheme. To recover from this particular upset condition, the trajectory planner supplies a recovery trajectory that uses higher angles of attack. The same high bank angle with steeply descending flight path angle initial upset condition is used as for the previous sections.

Table 5.6: Initial upset condition

State	Value	Units
$\bar{V}_{initial}$	40	kn
$\gamma_{initial}$	-30	degrees
$\alpha_{initial}$	3	degrees
$\beta_{initial}$	0	degrees
$\Phi_{W_{initial}}$	75	degrees
$T_{initial}$	25.2	Newton

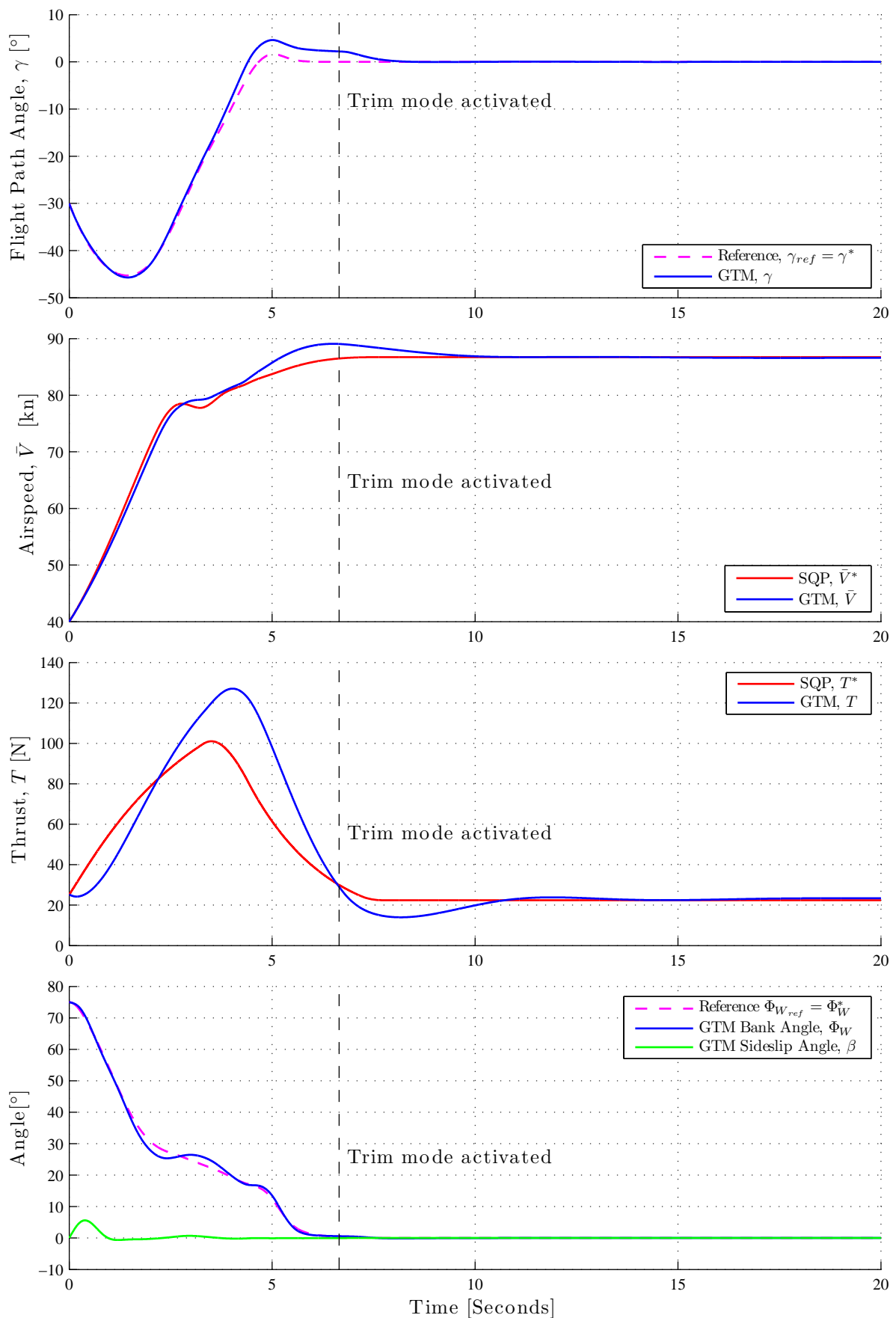


Figure 5.22: State trajectory response using 'combined state and input' scheme and normal acceleration controller for a recovery trajectory that uses higher angles of attack

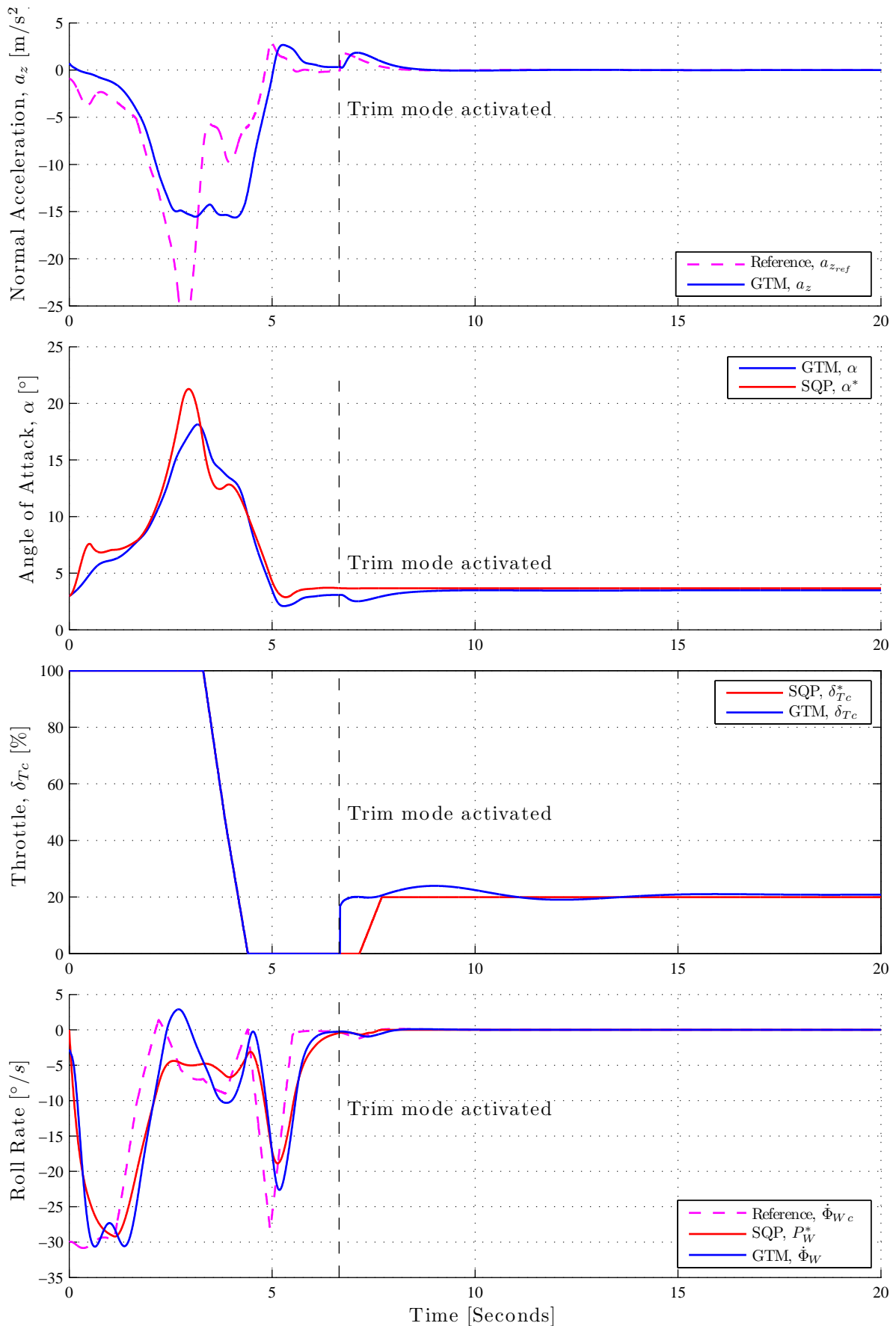


Figure 5.23: Input trajectory response using ‘combined state and input’ scheme and normal acceleration controller for a recovery trajectory that uses higher angles of attack

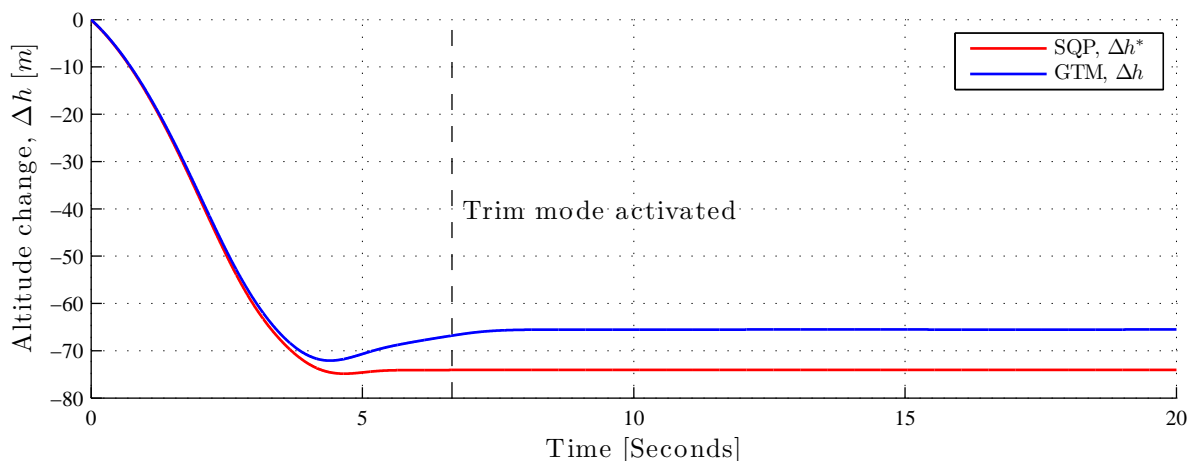


Figure 5.24: Altitude change response using ‘combined state and input’ scheme and normal acceleration controller for a recovery trajectory that uses higher angles of attack

Figure 5.22 shows that the executed flight path angle trajectory follows the planned trajectory quite well and with zero steady-state error (due to the feedback provided by the middle-loop flight path angle controller). There is some slight overshoot in the executed flight path angle at  $t = 5$  seconds due to deviations in the normal acceleration signal from the commanded normal acceleration signal, but the middle-loop flight path angle controller successfully regulates the executed flight path angle trajectory to stay close to the planned trajectory. The flight path angle controller is switched back to its nominal trim mode at around  $t = 7$  seconds after the recovery has been completed in order to trim the aircraft to wings level flight. The executed flight path angle also stays within the flight path angle limits. The airspeed trajectory follows the planned airspeed trajectory closely and settles to a zero error at steady-state, since the airspeed controller is switched on at  $t = 7$  seconds after the recovery has been completed and regulates the airspeed to the final planned airspeed. The airspeed does not exceed the maximum airspeed limit prescribed by the structural integrity envelope. The executed thrust signal overshoots the planned thrust signal, for the same reason as discussed in previous sections, but remains within the maximum thrust limit (136 N). The executed wind-axis bank angle trajectory tracks the planned trajectory fairly well with minimal delay. The slight deviations in the executed wind-axis bank angle trajectory from  $t = 2$  to 4 seconds is due to the deviations in the executed wind-axis bank angle rate from the planned wind-axis bank angle rate. The realised wind-axis bank angle tracks the final planned wind-axis bank angle with zero steady-state error due to the wind-axis bank angle middle-loop regulator. The executed sideslip angle is also regulated to remain close to zero degrees.

Figure 5.23 shows that the executed normal acceleration signal mostly tracks the commanded normal acceleration, but deviates from the command signal between  $t = 2.5$  and 4 seconds. This is due to the fact that the angle of attack is higher, and the normal acceleration controller is thus operating further from its single trim linearisation point at a low angle of attack (of 3 degrees). The normal acceleration signal tracks the final planned normal acceleration with a zero steady-state error and the executed normal acceleration signal stays within the load factor constraints imposed on the system by the structural integrity envelope. The executed angle of attack follows the planned angle of attack signal relatively closely due to the added middle-loop angle of attack regulator that keeps the angle of attack close to the planned angle of attack. The angle of attack regulator is switched off at around  $t = 7$  seconds after the recovery is completed and thus the executed angle of attack exhibits a small steady-state error. The angle of attack also stays within the angle of attack limits imposed on the system. The executed throttle

signal is exactly the same as the planned throttle signal (as discussed in previous sections) and only deviates when the airspeed controller is switched on at around  $t = 7$  seconds that commands the throttle to control the airspeed after recovery is completed. (Remember the airspeed controller is not used during the recovery manoeuvre.) The executed wind-axis bank angle rate follows the planned wind-axis bank angle rate well and with zero steady-state error. The slight deviations in the executed wind-axis bank angle rate from the planned wind-axis bank angle rate at around  $t = 3$  seconds and the sharp ‘dip’ at  $t = 5$  seconds is due to the same reason as discussed in the case of Figures 5.13 and 5.14 in the previous result section §5.4.2 for the ‘input only’ controls scheme. The executed wind-axis bank angle rate slightly lags the command signal, but this is expected as discussed in previous sections and is expected because this lag was taken into account by the SQP trajectory planner. The executed wind-axis bank angle also stays within the wind-axis bank angle rate limits (30 degrees per second) that were imposed on the system.

Figure 5.24 shows that the executed altitude trajectory follows the planned altitude trajectory of the SQP and then deviates to settle with a constant offset. This is due to the executed flight path angle overshoot which results in the GTM gaining more altitude than the planned trajectory. The executed altitude trajectory does not converge to the final planned trajectory, due to the fact that an altitude controller is not used to explicitly control the altitude after the recovery has been completed, and when the flight path angle is recovered to zero, the altitude change is also zero, and remains at a constant value. The peak altitude loss of the executed recovery trajectory is slightly less than the peak altitude loss of the planned trajectory (72 meters altitude lost compared to the planned altitude loss of 75 meters).

The simulation results of the executed trajectories using the ‘combined state and input’ scheme for a recovery trajectory that uses higher angles of attack show that the executed airspeed trajectory now tracks the optimal airspeed trajectory well for this upset case, due to the added angle of attack feedback controller that regulates the angle of attack to follow the planned angle of attack. The angle of attack now tracks the planned angle of attack much closer than in the case of ‘input only’ control scheme (Figure 5.14) for the same recovery trajectory. The slight overshoot in flight path angle causes the realised altitude change trajectory to have slightly less peak altitude loss than the predicted peak altitude loss by the SQP.

#### 5.4.5 Compensated State Trajectories Control Scheme

Figures 5.25 to 5.27 show a simulation result of the GTM’s state trajectory, input signal and altitude responses respectively for recovering from the initial upset condition of Table 5.7 using the ‘compensated state’ control scheme. The same inverted bank angle with descending flight path angle initial upset condition is used as for the previous sections.

Table 5.7: Initial upset condition

State	Value	Units
$\bar{V}_{initial}$	110	kn
$\gamma_{initial}$	-15	degrees
$\alpha_{initial}$	3	degrees
$\beta_{initial}$	0	degrees
$\Phi_{W_{initial}}$	130	degrees
$T_{initial}$	25.2	Newton

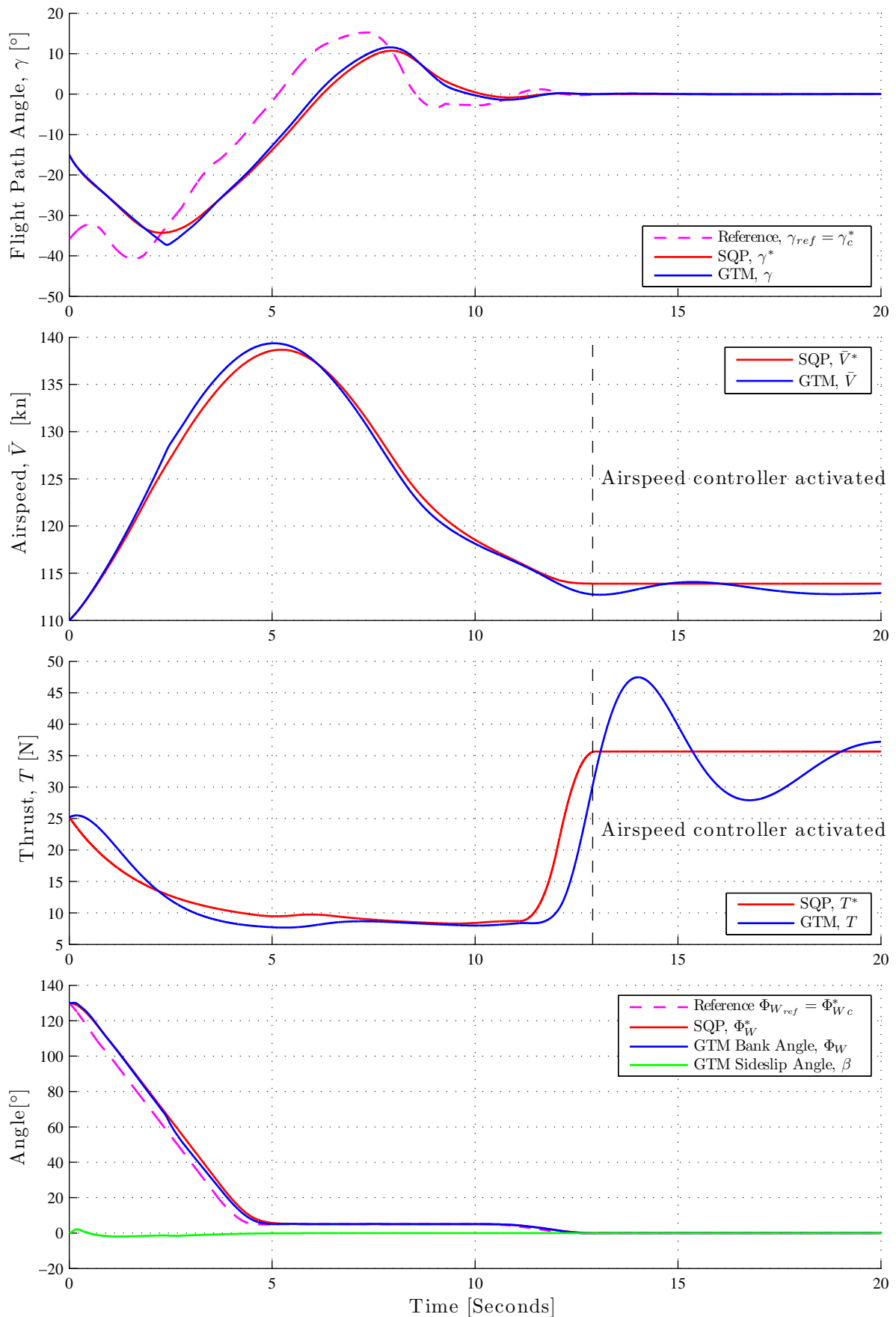


Figure 5.25: State trajectory response using 'compensated state' scheme and normal acceleration controller for a recovery trajectory that uses lower angles of attack

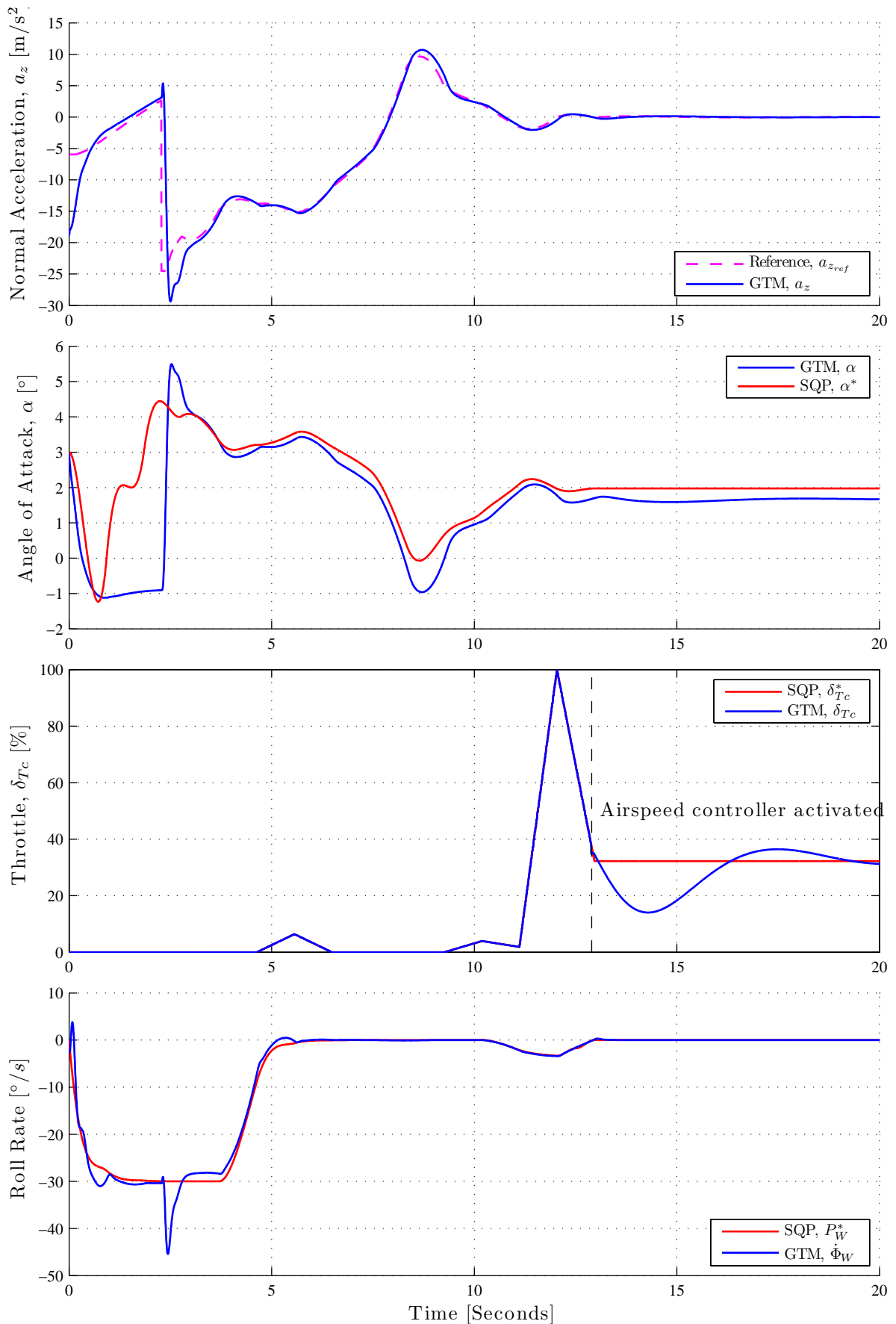


Figure 5.26: Input trajectory response using ‘compensated state’ scheme and normal acceleration controller for a recovery trajectory that uses lower angles of attack



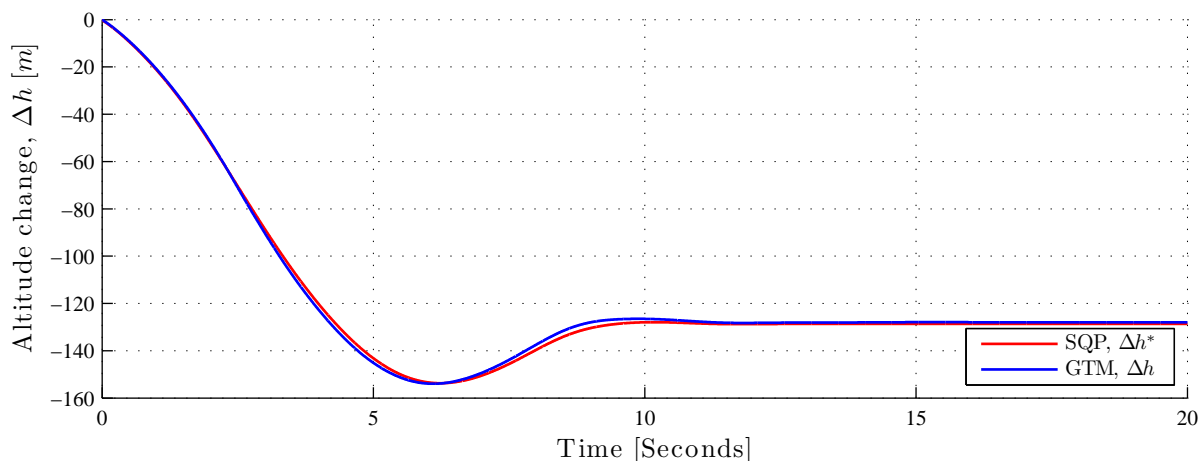


Figure 5.27: Altitude change response using ‘compensated state’ scheme and normal acceleration controller for initial upset condition using low angles of attack

Figure 5.25 shows that the executed flight path angle trajectory follows the planned trajectory very well and exhibits a zero error in steady-state (due to the feedback from the middle-loop flight path angle controller). The executed flight path angle lags the purple flight path angle command, which in turn leads the planned flight path angle trajectory. This is to be expected as we compensated the planned flight path angle command to take the lag of the middle-loop flight path angle controller into account. The slight ‘dip’ in the executed flight path angle at  $t = 2.5$  seconds is because the flight path angle controller only starts to issue non-zero normal specific acceleration commands when the bank angle transitions below 70 degrees, as discussed in previous sections (section §5.4.3). The flight path angle remains within the flight path angle limits. The executed airspeed tracks the planned airspeed relatively well, and the slight offset between the executed and planned airspeed trajectory can be attributed to the executed angle of attack not following the planned angle of attack exactly. The airspeed controller is switched on at around  $t = 13$  seconds to regulate the airspeed close to the planned final airspeed. The airspeed remains within the maximum airspeed limit prescribed by the structural integrity envelope. The overshoot in the thrust response is due to the same reasons as discussed in previous sections. The final thrust does not converge to the final planned thrust value due to the airspeed controller controlling the throttle input to track the final airspeed. The thrust remains within the maximum thrust limits (136 N). The executed wind-axis bank angle tracks the planned wind-axis bank angle well, and with zero steady state error, because we are using a middle-loop bank angle and sideslip controller (that provides feedback control). The executed wind-axis bank angle lags the wind-axis bank angle command, which in turn leads the planned wind-axis bank angle trajectory. This is expected, as we compensated the planned wind-axis bank angle reference to account for the bank angle controller’s dynamics. The slight constant offset from zero in the wind-axis bank angle trajectory between  $t = 5$  to 11 seconds is due to the trajectory planner’s solution, and not the trajectory execution, as discussed in previous sections. The sideslip is also successfully regulated to remain close to zero.

Figure 5.26 shows that the executed normal acceleration signal tracks the normal acceleration command from the flight path angle controller very well, and exhibits zero steady-state error. The sharp ‘dip’ in the command signal at  $t = 2.5$  seconds is when the flight path angle controller begins to issue non-zero normal specific acceleration commands when the bank angle becomes less than 70 degrees, as discussed in previous sections (section §5.4.3). The normal acceleration mostly remains within the load factor limits prescribed by the structural integrity envelope, but slightly exceeds the limit at  $t = 2.5$  seconds due to the aggressive normal acceleration

command by the flight path angle controller. The executed angle of attack initially does not follow the planned angle of attack, but after the flight path angle controller starts issuing non-zero normal specific acceleration commands at  $t = 2.5$  seconds, the resulting angle of attack follows the planned angle of attack with a slight offset. Note that the executed angle of attack is only the result of the middle-loop flight path angle controller commanding the normal acceleration, as there is no middle-loop angle of attack regulator or feed-forward angle of attack command. The angle of attack stays well within the angle of attack limits. The executed throttle signal exactly matches the planned throttle input signal until the airspeed controller is switched on at around  $t = 13$  seconds. The wind-axis bank angle rate mostly follows the planned wind-axis bank angle rate, but note that the executed wind-axis bank angle rate is not controlled directly and is the result of the middle-loop bank angle controller controlling the wind-axis bank angle. The ‘spike’ at  $t = 2.5$  seconds and the ‘bump’ fluctuation at around  $t = 13$  seconds in the executed wind-axis bank angle rate is due to the same reasons as previously discussed in the result case of section §5.4.3 with the ‘state only’ scheme.

Figure 5.27 shows that the executed altitude trajectory matches the planned altitude trajectory very closely, with the same peak altitude loss of around 156 meters.

The executed trajectories show good flight path angle, airspeed and bank angle tracking, but the angle of attack is not regulated. As with the ‘state only’ scheme, the flight path angle controller begins to issue non-zero normal specific acceleration commands when the bank angle becomes less than 70 degrees at around  $t = 2.5$  seconds, resulting in the flight path angle controller giving a large normal acceleration command to recover the flight path angle. This causes the same disturbance in the bank angle rate response, due to the pitch rate as discussed in the case of the ‘state only’ response. The GTM’s altitude change matches the predicted altitude loss by the SQP. There is no guarantee that the airspeed trajectory will track the optimal airspeed, as the angle of attack is unregulated.

For low optimal angle of attack trajectories the first-order flight path angle controller model is a good representation, but at higher optimal angles of attack there were some overshoot in the executed flight path angle trajectory. Figures 5.28 and 5.30 show a simulation result of the GTM’s state trajectory, input signal and altitude response respectively for recovering from the initial upset condition of Table 5.8 using the middle-loop control scheme, that uses a higher angle of attack recovery trajectory. The same high bank angle with steeply descending flight path angle initial upset condition is used as for the previous sections.

Table 5.8: Initial upset condition

State	Value	Units
$\bar{V}_{initial}$	40	kn
$\gamma_{initial}$	-30	degrees
$\alpha_{initial}$	3	degrees
$\beta_{initial}$	0	degrees
$\Phi_{W_{initial}}$	75	degrees
$T_{initial}$	25.2	Newton

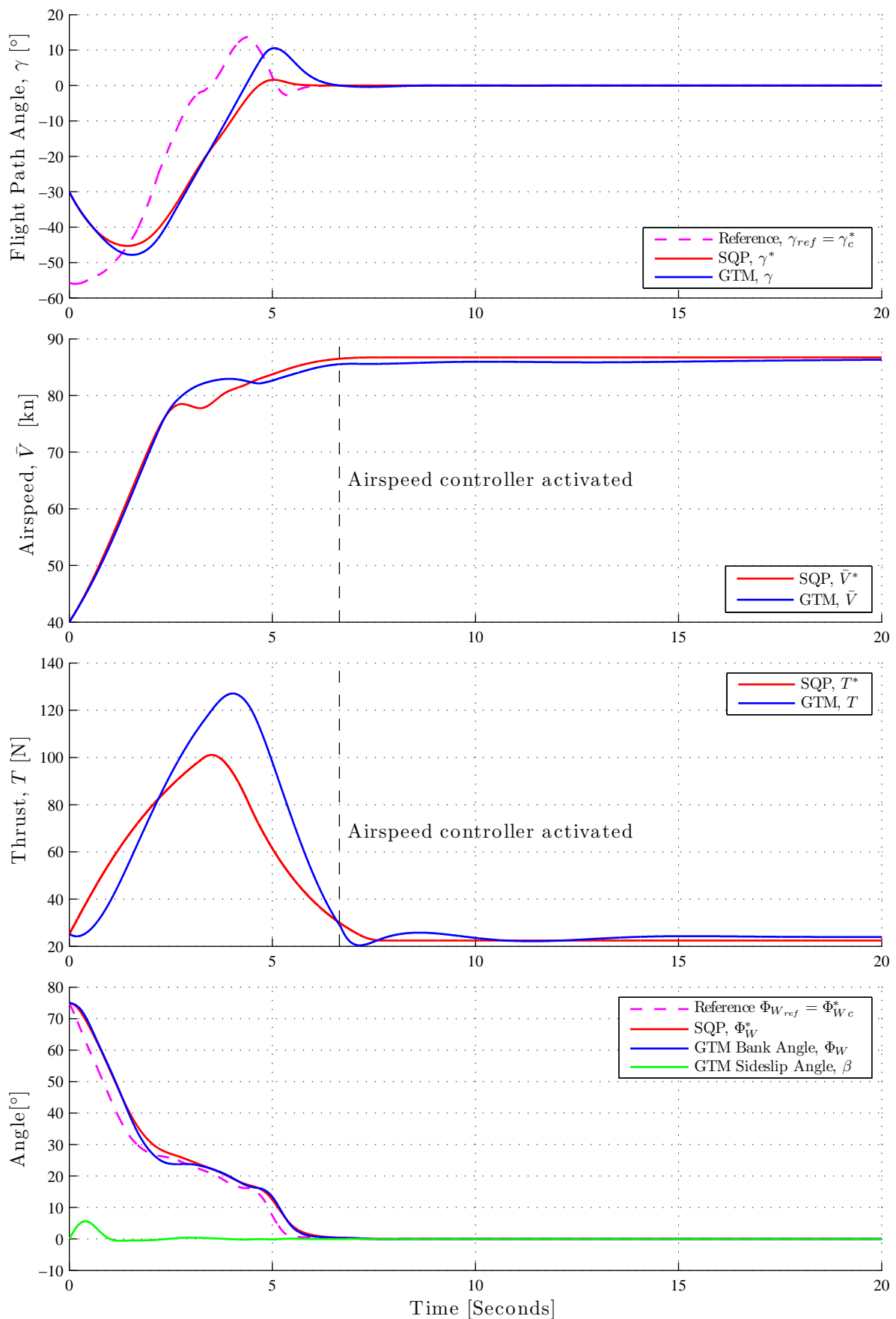


Figure 5.28: State trajectory response using ‘compensated state’ scheme and normal acceleration controller for a recovery trajectory that uses higher angles of attack

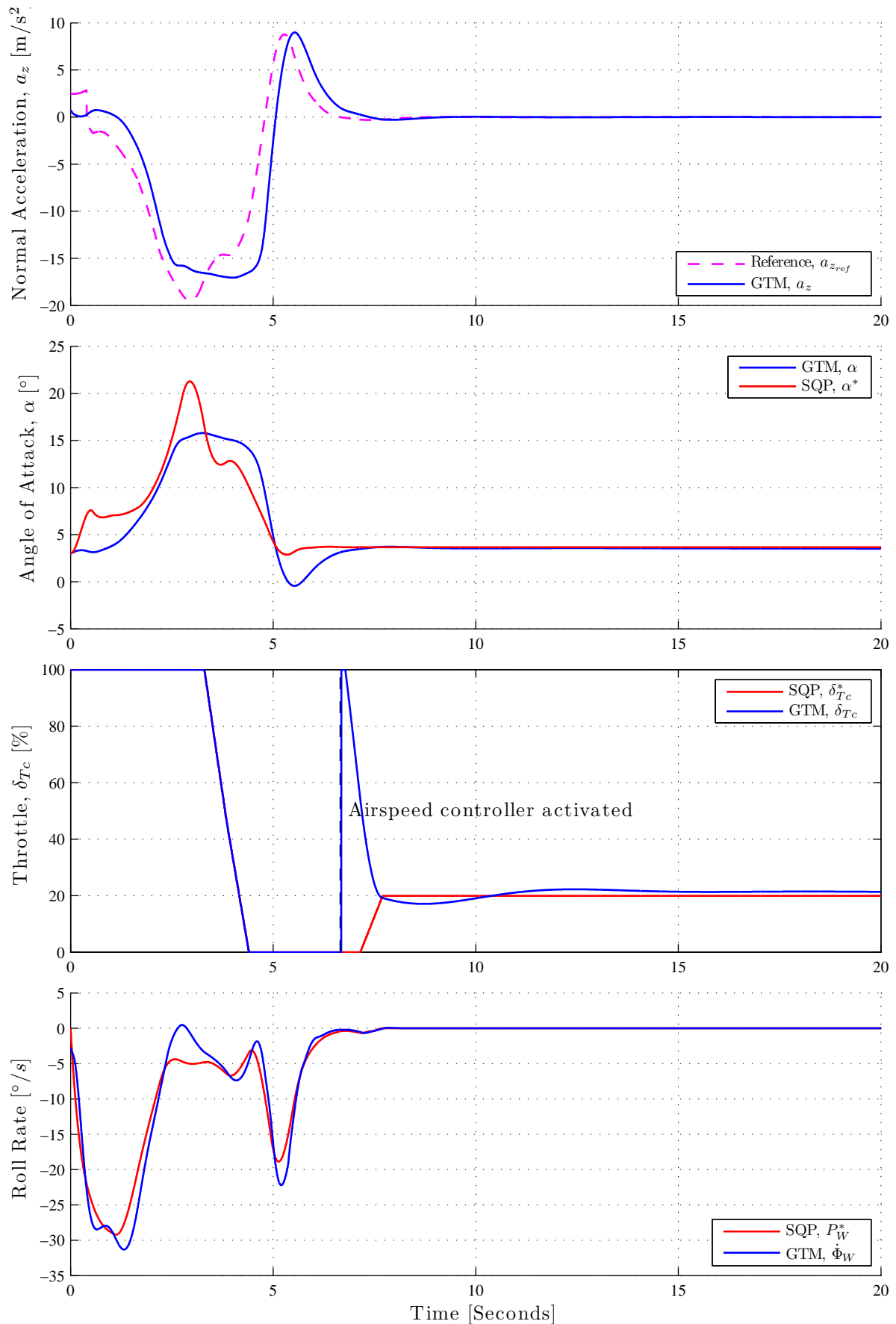


Figure 5.29: Input trajectory response using ‘compensated state’ scheme and normal acceleration controller for a recovery trajectory that uses higher angles of attack

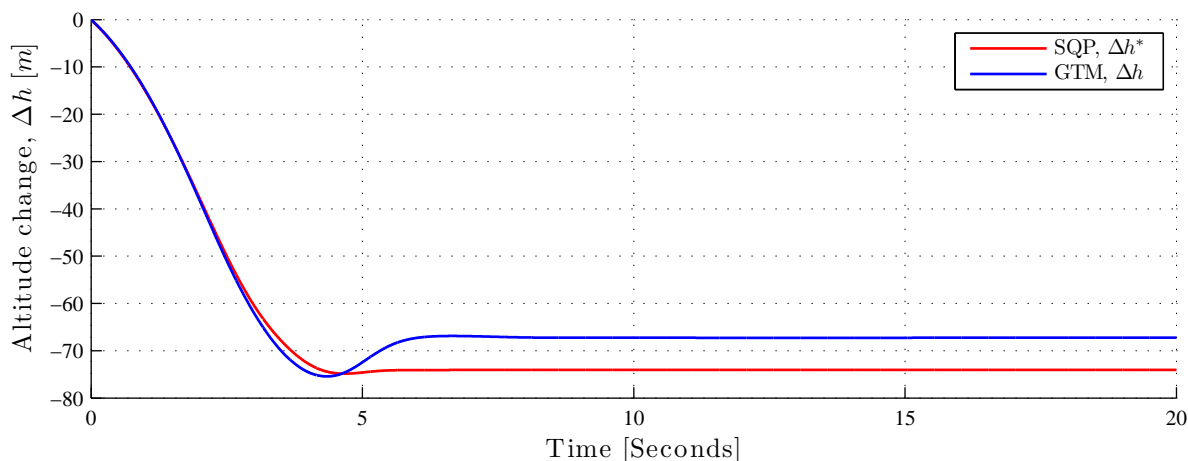


Figure 5.30: Altitude change response using ‘compensated state’ scheme and normal acceleration controller for a recovery trajectory that uses higher angles of attack

Figure 5.28 shows that the executed flight path angle tracks the planned flight path angle, and the executed flight path angle lags the flight path angle command, which in turn leads the planned flight path angle as discussed in the previous simulation result. However the executed trajectory does show slight undershoot at  $t = 1.5$  seconds and noticeable overshoot at  $t = 5$  seconds. This is due to the flight path angle controller model (which was used to calculate the command) being less accurate at higher angles of attack, further from the linearisation point of the controller, which is at a low angle of attack. The executed flight path angle tracks the final planned flight path angle with zero steady state error and the flight path angle remains within the flight path angle limits. The executed airspeed follows the planned airspeed well and is regulated close to the final planned airspeed after the airspeed controller switches on at around  $t = 6.5$  seconds. However, the executed airspeed only tracks the planned airspeed coincidentally, as the angle of attack in turn is not explicitly regulated, and there is a slight overshoot in the executed airspeed trajectory at around  $t = 2.5$  seconds due to the executed angle of attack not following the peak planned angle of attack. The airspeed also remains within the maximum airspeed limits prescribed by the structural integrity envelope. The thrust response overshoot the planned thrust response due to the same reason as discussed in previous sections, and the thrust response remains within the maximum thrust limits (136 N). The executed wind-axis bank angle trajectory follows the planned wind-axis bank angle rate fairly well, with the executed wind-axis bank angle lagging behind the wind-axis bank angle reference, which in turn leads the planned wind-axis bank angle trajectory. This is due to the same reason as discussed in the previous simulation result case. The executed wind-axis bank angle also tracks the final planned wind-axis bank angle with zero steady-state error, due to the feedback from the middle-loop bank angle controller. The ‘kink’ in the wind-axis bank angle at around  $t = 2.5$  seconds is due to the slight deviations in the executed wind-axis bank angle rate from planned wind-axis bank angle rate. The sideslip angle is also successfully regulated to remain close to the setpoint of zero.

Figure 5.29 shows that the executed normal acceleration follows the normal acceleration command from the flight path angle controller relatively well, and exhibits zero steady-state error. The executed normal acceleration also remains within the normal load factors limits imposed by the structural integrity envelope. The executed angle of attack does not follow the planned angle of attack explicitly, as the angle of attack is a result of the controlled normal acceleration only, which in turn is only a result of the middle-loop flight path angle controller commands. There is no middle angle of attack regulation controller to regulate the angle of attack. The

angle of attack remains within the angle of attack limits imposed on the system. Similar to the previous simulation result case, the realised throttle signal is exactly the same as the planned throttle until the airspeed controller switches on at around  $t = 6.5$  seconds. The executed wind-axis bank angle rate follows the planned wind-axis bank angle rate very well and with zero steady-state error. The slight deviation in the executed wind-axis bank angle rate at around  $t = 2.5$  seconds is due to the same reason as discussed in previous sections, and the wind-axis bank angle rate is not explicitly controlled as a middle-loop bank angle controller is used. The pitch rates produced by the normal acceleration trajectory acts as a disturbance on the wind-axis bank angle rate in this case as the middle-loop controller reference does not take the pitch rate into account. The sharp ‘dip’ at  $t = 5$  seconds is due to the same reason as discussed in previous sections. The wind-axis bank angle also rate remains within the wind-axis bank angle constraint imposed on the system.

Figure 5.30 shows that the executed altitude trajectory follows the planned altitude trajectory very well, but then deviates from the planned trajectory at around  $t = 5$  seconds. This is due to the overshoot in the executed flight path angle causing more altitude to be regained. The peak altitude loss of executed altitude trajectory matches that of the peak altitude loss of the planned altitude trajectory, at 75 meters altitude lost.

The simulation results show that at higher angles of attack, the normal acceleration controller is further from its linearisation point, and the flight path angle controller’s first-order model becomes less representative of the actual response, causing the flight path angle overshoot. For this recovery example the GTM’s total altitude loss still matches that of the SQP’s predicted altitude loss, with the GTM regaining more altitude than predicted due to the overshoot in flight path angle.

A possible solution to the overshoot is to design a gain-scheduled normal acceleration controller to give the same response characteristics at higher angles of attack than for low angles of attack so that the first-order model of the flight path angle controller remains representative for a wider range of angles of attack states.

The executed angle of attack trajectories of the ‘compensated state’ scheme does not follow the planned angle of attack trajectories as well as in the case of the ‘combined state and input’ scheme. This is because the ‘compensated state’ scheme does not have a middle-loop angle of attack regulator that explicitly regulates the angle of attack trajectory to track the planned angle of attack trajectory. Thus there is no guarantee that the ‘compensated state’ scheme’s executed airspeed trajectory will follow the planned airspeed as well, because of the unregulated angle of attack trajectory not necessarily producing the expected drag during the recovery.

#### 5.4.6 Observations from Trajectory Execution Using SQP Trajectory Planning

We used the control schemes designed in the previous section with the full non-linear GTM Simulink model to verify our SQP method’s recovery solutions. This section discusses the observations and conclusions that can be drawn from design decisions that were made with the SQP method and the control schemes used.

##### Angle of Attack Inner-loop Controller Model

The initial reduced-order formulation of the dynamics modelled the inner-loop angle of attack controller as a first-order response. Using this model proved to produce poor tracking of the angle of attack state when using the angle of attack feed-forward command from the SQP.

The SQP tries to exploit the first-order behaviour of the model with aggressive angle of attack commands, but because the angle of attack controller's true response shows second-order behaviour, severe overshoot is caused when trying to follow the SQP's feed-forward commands. Due to this behaviour the reduced-order model was updated to model the angle of attack controller as a second-order response. This adds an extra state to the reduced-order model. The model can be represented in state space form,

$$\begin{bmatrix} \dot{\alpha} \\ \ddot{\alpha} \end{bmatrix} = \mathbf{A}_\alpha \begin{bmatrix} \alpha \\ \dot{\alpha} \end{bmatrix} + \mathbf{B}_\alpha \alpha_c \quad (5.4.1)$$

where the state space matrices  $\mathbf{A}_\alpha$  and  $\mathbf{B}_\alpha$  are obtained from the two pole system representing the closed-loop normal acceleration (DQ) controller,

$$p_{dq} = -7.48 \pm 3.75i \quad (5.4.2)$$

This two pole system is chosen to represent both the normal acceleration controller and gain scheduled angle of attack controller, as both controllers were designed to have the same response characteristics (and both controllers control the short-period mode dynamics of the aircraft).

Simply using Euler integration for the second-order linear dynamics when directly transcribing the model proved to produce large numerical errors in the angle of attack trajectory solution when using the same amount of mesh points. It was found that the required amount of mesh points proved too intractable for the Matlab implementation. An alternative method that was explored, was to transform the dynamics of the angle of attack controller into a difference equation using the z-transform, and not the standard Euler method. The direct transcription then uses the z-transformed difference equation for the inner-loop controller dynamics and Euler integration for the rest of the dynamics. This method required less mesh points to give sensible results.

However, even with this alternative method the solutions proved to be too inaccurate due to the small time scale of the angle of attack dynamics. When not assuming time scale separation between the inner-loop dynamics and the transversal states while using Euler integration, the mesh must be significantly finer to ensure accurate solutions. Furthermore, it was realised that using the z-transform with the direct transcription method is mathematically inconsistent. When modelling the inner-loop controllers to this extent with the transversal dynamics, time-scale separation can no longer be assumed. To use less mesh points, but still maintain the needed numerical accuracy, it was instead decided to use a higher order Hermite-Simpson collocation method to solve this problem.

Using the second-order model proved to reduce the overshoot, and the angle of attack trajectory tracking performance in trajectory execution simulations improved considerably. Appendix D shows a comparison between the first-order and second-order angle of attack delay model by illustrating recovery trajectories using each model for the same upset case in Figures D.25 and D.26.

## 5.5 Summary of Results

A summary of the results from the four implemented control schemes investigated in this chapter is given in this section.

The realised trajectories of the 'input only' scheme followed the planned trajectories well with no lag, but did so in an open-loop fashion and showed a noticeable steady-state error. The

scheme does not have any disturbance rejection capabilities and won't be able to deal with model uncertainties. The performance of the 'input only' scheme also indicates that the SQP uses a fairly representative model of the GTM's dynamics and verifies the numerical SQP algorithm results.

The 'state only' scheme produced executed trajectories that significantly lagged behind the predicted state trajectories, but had no steady-state tracking errors. The feedback from the middle-loop controllers does however enable this scheme to potentially compensate for disturbances and model uncertainty.

The 'combined state and input' scheme gave the best performance out of all the schemes by combining the advantages of the 'input only' scheme and the 'state only' scheme. The trajectory lag of the 'state only' scheme is eliminated by using the input commands as feed-forward commands and deviations from the optimal trajectories are corrected by the feedback of the middle-loop regulators. This scheme tracked the planned trajectories well with no lag, and with no steady-state errors, and has potential disturbance rejection capability and is expected to have robustness against model uncertainty.

The final scheme investigated, namely the 'compensated state' scheme, used the unmodified middle-loop controllers in the same manner as the 'state only' control scheme. The scheme compensated the state references by having them lead the original state references using a simplified model of the middle-loop controller dynamics. This scheme's executed trajectories followed the planned trajectory with no lag and had no steady-state error, but relies on a representative first-order model of the middle loop controllers and does not guarantee good airspeed tracking.

Finally it was found that a first-order model for the angle of attack dynamics in the original formulation of the system was inadequate and the SQP trajectory planning algorithm had to be expanded to use a second-order model representing the closed-loop normal acceleration (DQ) and angle of attack controller dynamics.

## 5.6 Conclusions

Four trajectory executed control schemes that aim to execute a planned optimal recovery trajectory were investigated in this chapter, namely an 'input only' scheme, a 'state only' scheme, a 'combined state and input scheme' and finally a 'compensated state' scheme. The 'combined state and input' scheme gave the best performance of all the schemes overall, giving good overall trajectory tracking, and also potentially provides robustness to model uncertainty and disturbance rejection capability. The middle-loop angle of attack regulator component added to the 'combined state and input' control scheme ensured that this scheme could provide better airspeed tracking performance than all the other schemes investigated that used a normal acceleration inner-loop controller (as the angle of attack trajectory significantly affected the airspeed trajectory). The 'combined state and input' scheme would therefore be the recommended regulation architecture to be used for upset recovery. That being said, the 'compensated state' scheme is however a viable alternative control scheme when access to the flight controller architecture is restricted.

The good trajectory tracking results achieved in simulation verify that the proposed trajectory execution control schemes can successfully control the aircraft to follow a planned recovery trajectory, and also validate the use of the simplified reduced-order model for the trajectory planning.



## Chapter 6

# Conclusions and Recommendations

This chapter gives the final conclusions on the implementation of the two major parts of a proposed attitude and flight vector recovery system for large transport aircraft, that consist of an optimal trajectory planning component, and a practical trajectory execution component.

### 6.1 Optimal Trajectory Planning

The first major part of this thesis presented the design and implementation of two optimal control algorithms, namely dynamic programming (DP) and sequential quadratic programming (SQP), for attitude and flight vector recovery. The methods used a reduced-order aircraft model to determine simultaneous bank angle, flight path angle, and airspeed recovery trajectories from an initial flight upset condition.

The problem was formulated as an optimal control problem using a model of the aircraft's point mass translational dynamics, with the fast rotational dynamics approximated as first and second-order responses in angle of attack and roll rate, and the thrust dynamics modelled as a first order response. This optimal control problem was then solved using dynamic programming and sequential quadratic programming.

The dynamic programming method is a policy based algorithm that uses a technique that is known as the principle of optimality to solve the optimal control problem. The algorithm is a multi-stage decision process that uses discretised time, as well as a discretised state space to construct a set of state decisions for each time instant. The algorithm then starts at the final time instant and works backwards in time to determine the optimal path from each intermediate state decision towards the terminal state. The algorithm continues to move backwards in time, one time instant at a time, until all initial states have an optimal path to the terminal state, or the beginning of the time window is reached. Dynamic programming gives the solution from *all* recoverable initial states in the form of a large state transition lookup table and gives a global minimum solution in a closed-loop form. The calculation of the solution table is computationally heavy, but can be done offline, with a light online implementation that simply indexes the solution table. However, for the aircraft recovery problem dynamic programming is forced to use a reduced-order model of only the point mass translational dynamics, where the fast rotational dynamics and thrust dynamics are omitted, in order to keep the problem tractable to be solved. This is due to a major limitation known as “the curse of dimensionality”.

The second alternative method, namely the sequential quadratic programming method, uses a numerical method called direct transcription and collocation to solve the optimal control problem. The direct transcription method transcribes the continuous optimal control problem into a discrete-time problem known as a non-linear programming problem (NLP). The NLP

is then solved using a NLP solver known as sequential quadratic programming (SQP), which is a gradient-based constrained minimisation algorithm. The gradients for this transcribed problem were supplied using an automatic differentiation (AD) software tool called ADiGator. ADiGator generates an executable function from the problem, that returns the objective function's gradient and the constraint vector's Jacobian, which is then passed to the SQP routine. The discrete solution produced by the SQP routine is then interpolated using the appropriate interpolation method to form a continuous trajectory solution. The SQP method provides an optimal trajectory solution in an open-loop form from a single initial upset condition, and the trajectory solution must be recalculated for each initial condition. However, the computational burden of the SQP method's online solution is much less than the dynamic programming method's offline computational burden, and the SQP algorithm has the potential of being implemented in real time. The advantage of the SQP approach, is that it does not suffer from the dynamic programming method's limitations and can include the fast rotational dynamics and thrust dynamics. The SQP approach can also use continuous-valued states and inputs, unlike the dynamic programming method, which is limited to quantised state values. This enables the SQP method to use a more representative model of the aircraft dynamics in the recovery trajectories.

Both the dynamic programming method and the SQP method were able to recover the attitude and flight vector of the aircraft from initial upset conditions while staying within the constraints imposed on the system. The SQP method produced similar recovery strategies as the dynamic programming method, and validates the dynamic programming method's assumption that a reduced-order model of the aircraft's translational dynamics can be used to produce near-optimal recovery trajectories. In most cases, the SQP method was able to produce trajectories that have less peak altitude loss than the trajectories produced by the dynamic programming method, due to the SQP's advantage of using continuous state values.

## 6.2 Trajectory Execution

In the second major part of this thesis, four control schemes were proposed and investigated to execute a planned upset recovery trajectory using conventional fly-by-wire controllers typically found on large commercial transport aircraft. These control schemes would be used in an integrated attitude and flight vector recovery system. In such a system, the conventional inner and/or middle-loop flight controllers would be controlled by a guidance law consisting of a trajectory planning algorithm such as one of the algorithms presented in the first part of this thesis.

The four control schemes for trajectory execution that were investigated are namely, an 'input only' scheme, a 'state only' scheme, a 'combined state and input' scheme and a 'compensated state' scheme. All four schemes were designed, implemented and verified on the full non-linear NASA GTM simulation model. Since the NASA GTM is not supplied with fly-by-wire controllers included, some custom inner-loop and middle-loop controllers were designed specifically for the aircraft and then added to the existing simulation model. The following conventional fly-by-wire controllers were designed and implemented for the NASA GTM: angle of attack controller (inner-loop), normal acceleration controller (inner-loop), airspeed controller (middle-loop), flight path angle controller (middle-loop) and bank angle and sideslip angle controller (middle-loop, but also doubles as an inner-loop roll rate controller).

The 'input only' scheme supplies only the planned inputs generated by the trajectory planner directly to the inner-loop controllers. Thus the trajectories are executed in an open-loop fashion. Two 'input only' variants were investigated, namely a variant that uses a normal acceleration inner-loop controller, and a variant that uses an angle of attack inner-loop controller.

The trajectories executed by the ‘input only’ scheme followed the planned state trajectories with no lag, but exhibited steady-state errors. Due to the absence of a middle-loop state feedback component, this scheme is expected to lack robustness to model uncertainty and would have no disturbance rejection capability. The trajectories executed using the angle of attack controller variant followed the planned airspeed trajectories better than the normal acceleration controller variant, because the angle of attack variant produced better airspeed tracking.

The ‘state only’ control scheme supplies only the planned state trajectories generated by the trajectory planner as references to the conventional middle-loop controllers (with the middle-loop controllers in turn issuing commands to the inner-loop controllers). Thus the trajectories are executed in a closed-loop fashion. The trajectories executed by the ‘state only’ scheme significantly lagged the planned trajectories, but due to the middle-loop controllers providing state feedback, the trajectories exhibited no steady state error. The middle-loop controllers can also provide robustness to model uncertainty and disturbance rejection.

The ‘combined state and input’ scheme combines the advantages of both the ‘input only’ and the ‘state only’ control schemes. The planned input commands are supplied as references to the inner-loop controllers as feed-forward terms and the optimal state trajectories are supplied as references to the middle-loop controllers. The outputs of the middle-loop controllers are then superimposed on the feed-forward input commands, which then become the total reference commands to the inner-loop controllers. The trajectories executed by the ‘combined state and input’ scheme follow the planned trajectories with no lag due to the feed-forward term, and with no steady-state error due to the feedback terms. The middle-loop controllers also provide disturbance rejection and robustness to model uncertainty. An angle of attack middle-loop feedback controller was added that superimposes a normal acceleration correction command onto the feed-forward command as well, in order to regulate the angle of attack to follow the optimal angle of attack signal. The improved angle of attack tracking provided by the feedback controller leads to better tracking of the planned airspeed trajectory, which ultimately results in better tracking of the planned recovery trajectory.

The ‘compensated state’ scheme also uses only the middle-loop controllers (like the ‘state only’ scheme), but modifies the state references supplied to the middle-loop controllers to compensate for the middle-loop controller dynamics. The trajectories executed by the ‘compensated state’ scheme follows the planned trajectory with no lag and with no steady state error. The scheme also still retains the robustness and disturbance rejection provided by the middle-loop controllers, but does not contain an angle of attack feedback controller to provide improved airspeed tracking.

None of the schemes that employ middle-loop controllers use an airspeed controller to regulate the airspeed to follow the planned airspeed state. The planned throttle input is instead supplied directly to the throttle input of the GTM. The airspeed controller’s bandwidth is too slow to affect the trajectories, due to the time scales in which the recovery trajectories are executed. Additionally, it was found that if an airspeed controller is used, then it gives commands that are too aggressive when the airspeed is changing rapidly. Regulating the angle of attack with a feedback control component provides a much more significant improvement in the airspeed tracking of the control schemes. This is due to the angle of attack affecting the aerodynamic drag produced, which in turn directly affects the airspeed of the aircraft.

The ‘combined state and input’ scheme provided the best trajectory tracking of all the control schemes that were investigated, and would be the recommended control scheme (when using conventional flight controllers) to execute upset recovery trajectories that were generated by a recovery trajectory algorithm. The addition of the angle of attack feedback controller in the ‘combined state and input’ scheme ensures much better airspeed tracking performance than all the other schemes that used a normal acceleration inner-loop controller. However, the

‘compensated state’ control scheme provides an adequate alternative if access to inner- and middle-loop controllers are unavailable.

The good tracking performance of the four trajectory execution control schemes validates that the optimal recovery trajectories produced by the SQP trajectory planner are realistic trajectories and verifies that the ‘real aircraft’ (represented by the full non-linear aircraft model) can actually follow the optimal reference trajectories. Furthermore, the good trajectory execution results show that conventional aircraft fly-by-wire controllers can potentially be used as trajectory execution controllers in an attitude and flight vector recovery system.

### 6.3 Future Work

Following on the research presented in this thesis, we briefly discuss some areas that could be explored further in future work:

- The strength of the direct transcription method using sequential quadratic programming can be explored further by increasing the dimensionality of the problem even more. A SQP algorithm that uses the full 6-degrees-of-freedom model of the NASA Generic Transport Model (or other aircraft simulation models) could be investigated.
- The Matlab implementation of the SQP algorithm is not suited for real-time implementation. The real-time execution potential of the algorithm can be investigated by rewriting and optimising the algorithm in a C code implementation. It could be possible with an optimised version to run multiple executions of the SQP method in parallel for the same initial upset condition using different initial guesses and then choose the best result.
- If real-time execution is possible for the SQP algorithm, research could be done on implementing the SQP as an upset recovery guidance algorithm on a real system such as a UAV to validate its simulation results and investigate practical issues such as robustness.

## Appendix A

# Mathematical Aircraft Modelling

This appendix recalls the mathematical notation and modelling of the aircraft. The conventional axis systems and notations are established, the standard six degrees of freedom equations of motion is given, and the forces and moments model of the aircraft are also given. Finally a section presents the mathematical linearisation process of the full aircraft model into the decoupled model using small disturbance theory. The material presented here is adapted from [14], with additional reference material from [1] and [15], and is reproduced for the convenience of the reader.

### A.1 Axis Systems and Notation

Three standard axis systems are commonly used when modelling aircraft dynamics in conventional aerospace applications: the inertial-, body- and wind-axis systems. These axis systems, along with their notation, is discussed in the following sections.

#### A.1.1 Axis Systems

##### Inertial-Axis System

To apply Newton's equations of motion we require an inertial axis system. The standard North-East-Down (NED) axis system, illustrated in Figure A.1, is often used for this as an inertial reference frame. The NED axis system assumes a flat, non-rotating earth. For short flight distances, and for the upset recovery trajectories considered in this work, this adequately approximates an inertial reference frame.

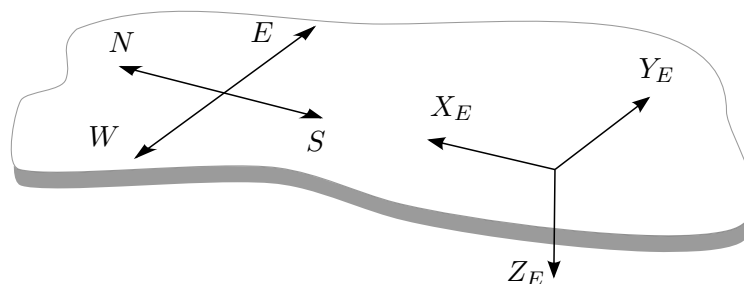


Figure A.1: North-East-Down axis system

The x-axis points in the North direction, the y-axis points in the East direction, and the z-axis points in the Down direction. This forms a complete right-handed, orthogonal axis system which centre is chosen at a convenient place on the earth's surface, such as the centre of a runway for instance.

### Body-Axis System

The body-axis system, illustrated in Figure A.2, is fixed to the aircraft body with its origin coinciding with the aircraft's centre of mass. The x-axis lies in the plain of symmetry, pointing through the nose along the zero lift line of the wing. The y-axis lies perpendicular to the plane of symmetry pointing in the direction of the starboard wing. The z-axis points downwards through the underside of the aircraft and completes the right-handed orthogonal axis system.

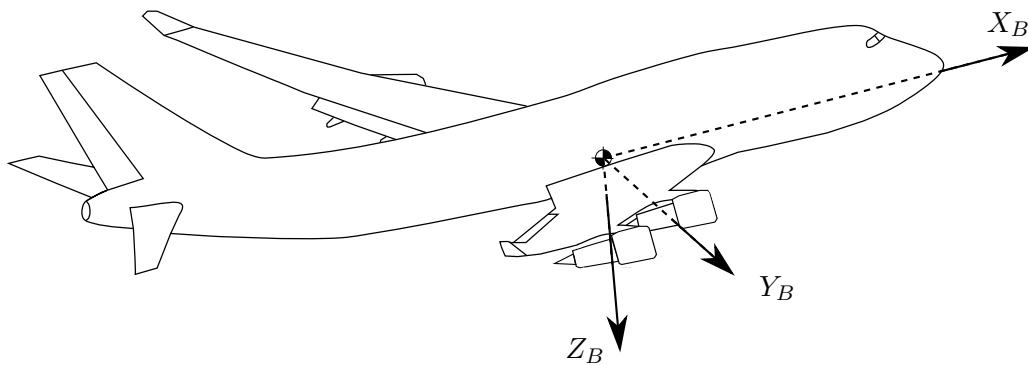


Figure A.2: body-axis system (Adapted from [16])

### Wind-Axis System

The wind-axis, illustrated in Figure A.3, can be seen as a version of the body-axis system where the x-axis  $X_W$  points in the direction of the velocity vector of the aircraft. For nominal flight, the z-axis of the wind-axis still points roughly downwards through the underside of the aircraft and the y-axis of the wind-axis points roughly in the direction of the right-hand (starboard) wing.

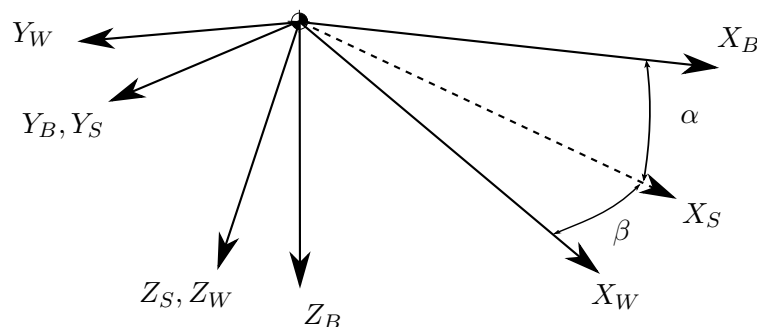


Figure A.3: wind-axis system

The terms  $X_S$ ,  $Y_S$  and  $Z_S$  are the x-axis, y-axis and z-axis of the stability-axis system respectively. The wind-axis is obtained through two rotations and results in another right-handed orthogonal axis system. The first rotation rotates the body-axis about the  $Y_B$ -axis in a negative direction by an angle  $\alpha$ , resulting in the so called stability axis. The stability axis is then rotated about the  $Z_S$  axis in the direction of the velocity vector by an angle  $\beta$  to obtain the wind-axis. The x-axis of the wind-axis points in the direction of the velocity vector, which is also the direction from which the freestream airflow hits the aircraft.

### A.1.2 Aircraft Notation

The following aircraft notation, illustrated in Figure A.4, is used in the body-axis system for the aircraft model

- $X, Y, Z$ : Co-ordinates of the force vector in the body-axis (axial, lateral, and normal force)
- $L, M, N$ : Co-ordinates of the moment vector in the body-axis (roll, pitch, and yaw moment)
- $U, V, W$ : Co-ordinates of the linear velocity vector in the body-axis (axial, lateral, and normal velocity)
- $P, Q, R$ : Co-ordinates of the angular velocity vector in the body-axis (roll, pitch, and yaw rates)
- $\delta_A, \delta_R, \delta_E$ : Aileron, rudder and elevator control surface deflections. A positive deflection is defined by it producing a negative moment.

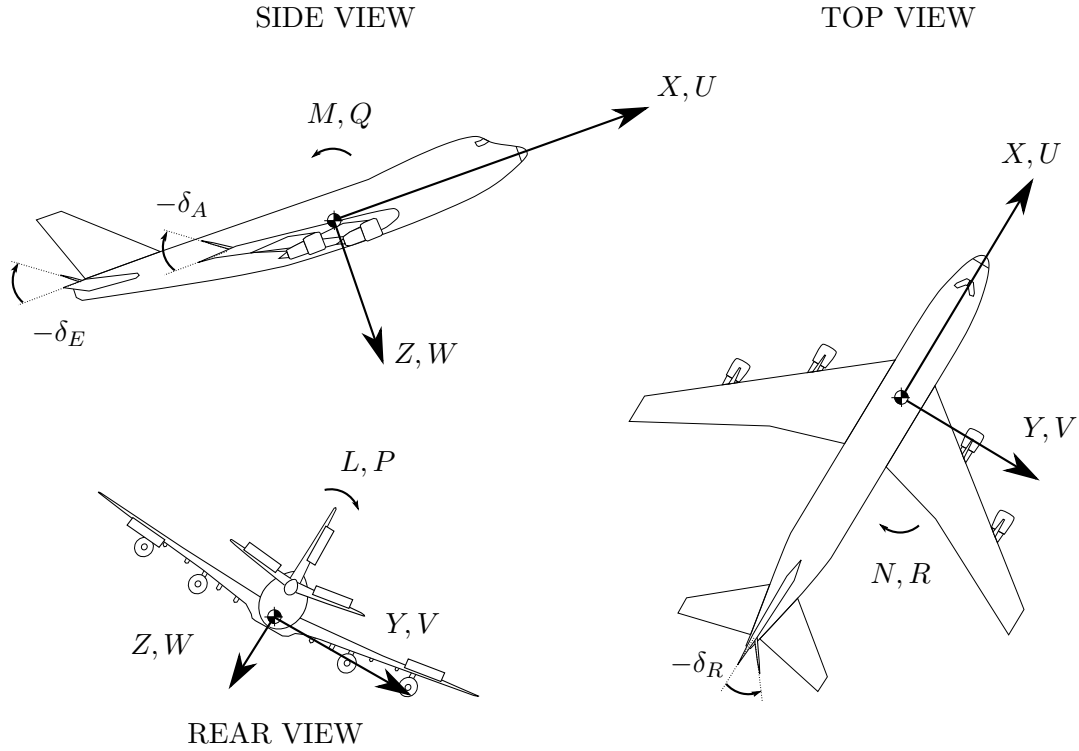


Figure A.4: Standard aircraft notation

It is often useful to express the velocity variables in polar co-ordinates, as a velocity magnitude and two angles. The magnitude of the aircraft's velocity vector, i.e. the airspeed magnitude  $\bar{V}$ , angle of attack  $\alpha$  and sideslip angle  $\beta$  are important aerodynamic variables that affect the aircraft's dynamics. The angle of attack can be seen as the angle between the aircraft's nose and the velocity vector in the longitudinal plane, and the sideslip angle can be seen as the angle between the nose of the aircraft and the velocity vector in the lateral plane. The relationship of these aerodynamic variables to the aircraft's body-axis velocity co-ordinates can be expressed as,

$$\bar{V} = \sqrt{U^2 + V^2 + W^2} \quad (\text{A.1})$$

$$\alpha = \arctan \frac{W}{U} \quad (\text{A.2})$$

$$\beta = \arcsin \frac{V}{\bar{V}} \quad (\text{A.3})$$

The inverse relationship is,

$$U = \bar{V} \cos \alpha \cos \beta \quad (\text{A.4})$$

$$V = \bar{V} \sin \beta \quad (\text{A.5})$$

$$W = \bar{V} \sin \alpha \cos \beta \quad (\text{A.6})$$



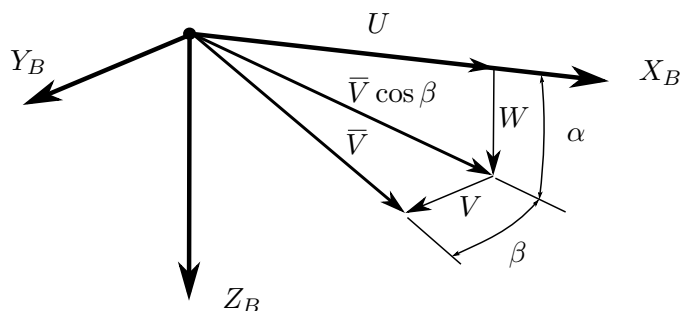


Figure A.5: Polar velocity co-ordinates (Adapted from [32])

## A.2 Six Degrees of Freedom Equations of Motion

We can model the aircraft dynamics quite well using a six-degrees-of-freedom rigid body for control system design purposes. A rigid body implies that the position of all points that are part of the aircraft's body remains constant relative to one another for all time. For larger aircraft, the modes of motion caused by structural flexibility are usually outside the bandwidth of conventional controllers and do not need to be taken into account for this model [14]. This section presents the six degrees of freedom equations of motion model, represented in Figure A.6.

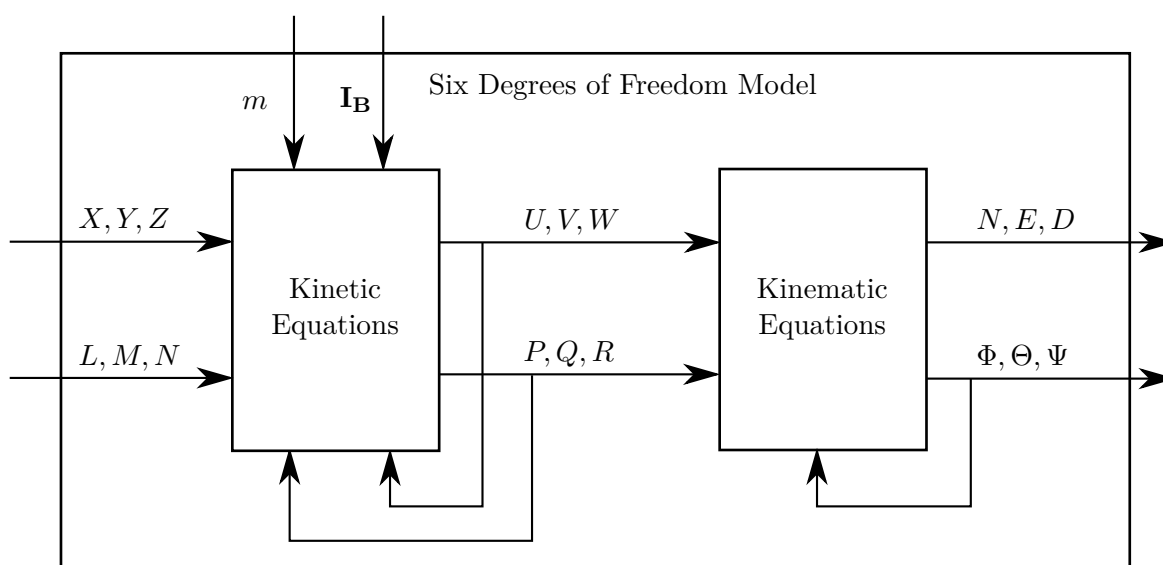


Figure A.6: Block Diagram Overview of 6DOF EOM

### A.2.1 Kinetics

Kinetic equations of motion relate the forces and moments acting on an aircraft to the kinematic state of the aircraft, i.e. its position, velocity and acceleration [14]. The relationship can be modelled using Newton's laws of motion in the body-axis of the aircraft. The resulting kinetic equations of motion as derived in [17] are given as,

$$\mathbf{F}_B = m \left( \frac{d\mathbf{V}_B}{dt} + \boldsymbol{\omega}_B \times \mathbf{V}_B \right) \quad (\text{A.1})$$

$$\mathbf{M}_B = \frac{d}{dt} (\mathbf{I}_B \boldsymbol{\omega}_B) + \boldsymbol{\omega}_B \times \mathbf{I}_B \boldsymbol{\omega}_B \quad (\text{A.2})$$

where  $\mathbf{F}_B$  and  $\mathbf{M}_B$  are the force and moment vectors acting on the aircraft, and  $\mathbf{V}_B$  and  $\boldsymbol{\omega}_B$  are the velocity and angular rate vectors of the aircraft. All these vectors are co-ordinated in the body axis. The term  $m$  is the mass of the aircraft, and  $\mathbf{I}_B$  is the moment of inertia matrix of the aircraft defined as,

$$\mathbf{I}_B = \begin{bmatrix} I_{xx} & I_{xy} & I_{xz} \\ I_{xy} & I_{yy} & I_{yz} \\ I_{xz} & I_{yz} & I_{zz} \end{bmatrix} \quad (\text{A.3})$$

where it is assumed that all the inertia components stay constant for all time. Equations A.1 and A.2 relate the forces and moments acting on the aircraft to the time rate of change of its linear velocity and angular rate. Due to the force, moment, velocity and angular rate vectors being co-ordinated in the body-axis, tangential vector velocity terms arise in the form of the cross product terms due to rotary motion with respect to the inertial frame.

These kinetic vector equations can be displayed in scalar form as a set of six non-linear, coupled differential equations that consists of 3 translational and 3 rotational degrees of freedom equations,

$$X = m (\dot{U} - VR + WQ) \quad (\text{A.4})$$

$$Y = m (\dot{V} - UR + WP) \quad (\text{A.5})$$

$$Z = m (\dot{W} - UQ + VP) \quad (\text{A.6})$$

$$L = \dot{P}I_{xx} + QR(I_{zz} - I_{yy}) \quad (\text{A.7})$$

$$M = \dot{Q}I_{yy} + PR(I_{xx} - I_{zz}) \quad (\text{A.8})$$

$$N = \dot{R}I_{zz} + PQ(I_{yy} - I_{xx}) \quad (\text{A.9})$$

where  $I_{xx}$ ,  $I_{yy}$  and  $I_{zz}$  are the principle moments of inertia about the respective body-axis. The above equations make the following simplifying assumptions,

- The aircraft is symmetrical about its XZ-plane in the body-axis. Thus the cross products of inertia  $I_{xy}$  and  $I_{yz}$  are exactly zero.

- The cross product of inertia  $I_{xz}$  is negligibly small.
- A rigid body with constant mass and moments of inertia.

Given the forces and moments that act on the aircraft body together with its mass and moment of inertia properties, Equations A.4 and A.9 allow the linear and angular velocity to be propagated over time.

## A.2.2 Kinematics

The kinematic equations of motion relate the motion variables (such as linear velocity, angular rate, position and attitude) to each other over time, without reference to the forces and moments [14]. The position and angular position (referred to as attitude) of the aircraft are represented by the following variables,

- $N, E, D$ : Co-ordinates of the position vector in inertial axes (north, east and down position)
- $\Phi, \Theta, \Psi$ : The Euler 3-2-1 attitude parameters of the body-axis system relative to the inertial-axis system (roll, pitch, and yaw angle)

To model external forces on the aircraft body such as gravity, the orientation of the aircraft relative to the inertial-axis must be parameterised.

### Attitude Representation

The Euler angles are typically used to represent the attitude of the body-axis system relative to the inertial-axis system. The Euler angles use three angles, namely the roll, pitch and yaw angles, of which the order is important. The Euler 3-2-1 sequence, illustrated in Figure A.7, is the most commonly used. It starts with the two axis systems aligned and then moves the body-axis system through the following set of ordered rotations,

1. Yaw the body-axis system by rotating it about its z-axis through the yaw angle  $\Psi$
2. Pitch the resulting first intermediate axis system about its y-axis through the pitch angle  $\Theta$
3. Roll the resulting second intermediate axis system about its x-axis through the roll angle  $\Phi$

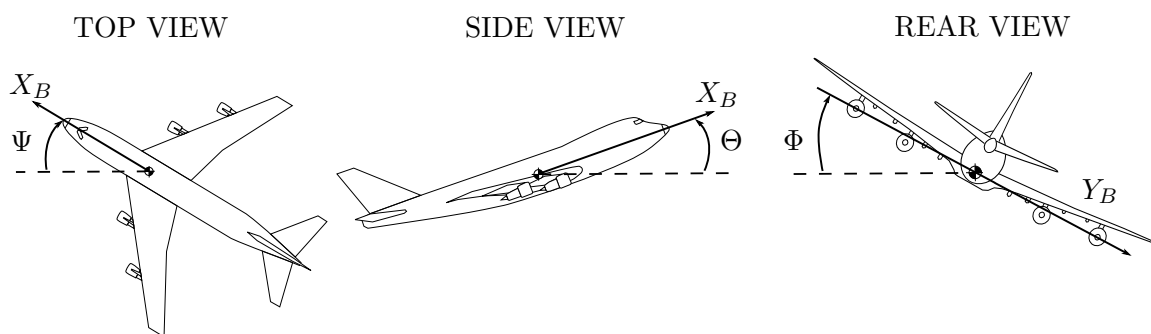


Figure A.7: Sequence of 3-2-1 Euler angles

The co-ordinates of a vector in the inertial-axis system may be transformed to co-ordinates in the body-axis system using the direction cosine matrix, denoted **DCM**, which is a function of the Euler angles as follows,

$$\mathbf{DCM}_{I \rightarrow B} = \begin{bmatrix} C_\Psi C_\Theta & S_\Psi C_\Theta & -S_\Theta \\ C_\Psi S_\Theta S_\Phi - S_\Psi C_\Phi & S_\Psi S_\Theta S_\Phi + C_\Psi C_\Phi & C_\Theta S_\Phi \\ C_\Psi S_\Theta C_\Phi + S_\Psi S_\Phi & S_\Psi S_\Theta C_\Phi - C_\Psi S_\Phi & C_\Theta C_\Phi \end{bmatrix}, S_{(\cdot)} = \sin(\cdot), C_{(\cdot)} = \cos(\cdot) \quad (\text{A.10})$$

Conversely, the co-ordinates of a vector in the body-axis system may be transformed to co-ordinates in the inertial-axis system by using the inverse direction cosine matrix. The DCM is an orthogonal matrix, therefore matrix inverse is simply its transpose,

$$\mathbf{DCM}_{B \rightarrow I} = \mathbf{DCM}_{I \rightarrow B}^{-1} = \mathbf{DCM}_{I \rightarrow B}^T \quad (\text{A.11})$$

### Attitude Dynamics

The attitude dynamics equation describing how the body angular rates  $P$ ,  $Q$ , and  $R$  relate to the time rates of changes of the Euler angles is given by,

$$\begin{bmatrix} \dot{\Phi} \\ \dot{\Theta} \\ \dot{\Psi} \end{bmatrix} = \begin{bmatrix} 1 & \sin \Phi \tan \Theta & \cos \Phi \tan \Theta \\ 0 & \cos \Phi & -\sin \Phi \\ 0 & \sin \Phi \sec \Theta & \cos \Phi \sec \Theta \end{bmatrix} \begin{bmatrix} P \\ Q \\ R \end{bmatrix}, \quad |\Theta| \neq \frac{\pi}{2} \quad (\text{A.12})$$

The above equation describes how roll, pitch and yaw rates in the body-axis relate to time rates of changes of the roll, pitch and yaw Euler angles. Note the singularity that occurs at  $\pm 90^\circ$  pitch angles. At this orientation an ambiguity exists between roll and pitch angles which mathematically gives rise to the singularity. For conventional flight however, the pitch angle is far from  $\pm 90^\circ$  at all times, thus allowing the singularity to be ignored. For the scope of the upsets considered, the pitch attitude is constrained within  $\pm 90^\circ$  to avoid this singularity.

### Position Dynamics

The time rate of change of the aircraft position is related to the aircraft velocity co-ordinated in body axes using the inverse DCM through the following kinematic equation,

$$\begin{bmatrix} \dot{N} \\ \dot{E} \\ \dot{D} \end{bmatrix} = \mathbf{DCM}_{B \rightarrow I} \begin{bmatrix} U \\ V \\ W \end{bmatrix} \quad (\text{A.13})$$

## A.3 Forces and Moments

For the conventional aircraft considered in this thesis, there are three main categories that contribute to the forces and moments that act on the aircraft, namely the aerodynamic, thrust and gravitational forces and moments. The force and moment equations can thus be expressed

as the sum of the components of the three categories,

$$\mathbf{F} = \mathbf{F}^A + \mathbf{F}^T + \mathbf{F}^G \quad (\text{A.1})$$

$$\mathbf{M} = \mathbf{M}^A + \mathbf{M}^T + \mathbf{M}^G \quad (\text{A.2})$$

where the superscripts  $A$ ,  $T$ , and  $G$  denote aerodynamic, thrust, and gravitational components respectively.

### A.3.1 Aerodynamic Model

The aerodynamic forces and moments introduce most of the uncertainty into the aircraft model and are by far the most complex to model [14]. For subsonic flight the aerodynamic forces and moments are proportional to the dynamic pressure experienced by the aircraft, denoted  $\bar{q}$ ,

$$\bar{q} = \frac{1}{2}\rho\bar{V}^2 \quad (\text{A.3})$$

where  $\rho$  is the air density. The aerodynamic force and moment co-ordinates are expanded as follows,

$$X^A = \bar{q}SC_X \quad (\text{A.4})$$

$$Y^A = \bar{q}SC_Y \quad (\text{A.5})$$

$$Z^A = \bar{q}SC_Z \quad (\text{A.6})$$

$$L^A = \bar{q}SbC_l \quad (\text{A.7})$$

$$M^A = \bar{q}S\bar{c}C_m \quad (\text{A.8})$$

$$N^A = \bar{q}SbC_n \quad (\text{A.9})$$

where  $S$  is the wing area,  $b$  is the wingspan,  $\bar{c}$  is the mean aerodynamic chord and  $C(\cdot)$  are the non-dimensional aerodynamic force and moment coefficients. The non-dimensional aerodynamic coefficients capture the aerodynamic properties of the shape of the aircraft, independently from of its size.

### Wide-Envelope Aerodynamic Model

The aerodynamic model of the GTM is represented using non-dimensional aerodynamic force and moment coefficients and span a wide aerodynamic envelope. The coefficients are characterised using a summation of a baseline static term and incremental terms for control surface deflections and dynamic effects from angular rates [13],

$$\begin{aligned} C_i = & C_{i,Static}(\alpha, \beta) + \Delta C_{i,\delta}(\alpha, \beta, \delta_A, \delta_E, \delta_R) \\ & + \Delta C_{i,\hat{q}_{osc}}(\alpha, \hat{q}_{osc}) + \Delta C_{i,\hat{\omega}_{ss}}(\alpha, \beta, \hat{\omega}_{ss}) \end{aligned} \quad (\text{A.10})$$

$$\begin{aligned} C_j = & C_{j,Static}(\alpha, \beta) + \Delta C_{j,\delta}(\alpha, \beta, \delta_A, \delta_E, \delta_R) \\ & + \Delta C_{j,\hat{p}_{osc}}(\alpha, \hat{p}_{osc}) + \Delta C_{j,\hat{r}_{osc}}(\alpha, \hat{r}_{osc}) + \Delta C_{j,\hat{\omega}_{ss}}(\alpha, \beta, \hat{\omega}_{ss}) \end{aligned} \quad (\text{A.11})$$

where  $i = X, Z, m$  and  $j = Y, l, n$ . The baseline static coefficients  $C_{Static}$  are only a function of the angle of attack  $\alpha$  and sideslip angle  $\beta$ . The incremental dynamic coefficients are made up of two types of terms. A rotary balance data term  $\Delta C_{\dot{\omega}_{ss}}$  associated with a steady-state angular rate, and forced oscillation data terms  $\Delta C_{\dot{p}_{osc}}$ ,  $\Delta C_{\dot{q}_{osc}}$  and  $\Delta C_{\dot{r}_{osc}}$  associated with oscillatory angular rates. The incremental control surface coefficients  $\Delta C_{\delta}$  model effects from control surface deflections  $\delta_A$ ,  $\delta_E$ , and  $\delta_R$ .

The aerodynamic model of the GTM is characterised by blending the forced oscillation and rotary balance data together using a method known as the Hybrid Kalviste method. The Hybrid Kalviste method breaks up the total angular rate vector  $\bar{\Omega}$  into a steady-state component  $\omega_{ss}$  along the velocity vector and oscillatory components  $\bar{\Omega}_{osc}$  along the aircraft's body-axes, illustrated in Figure A.8. The Hybrid Kalviste method uses three possible decomposition cases for the calculation of the angular rates, shown in Figure A.9. The decomposition cases depend on where the projection of the angular rate vector onto the xz-plane is relative to the x-axis and z-axis [1]. Table A.1 shows the equations used for converting the body-axis angular rates  $p_b$ ,  $q_b$  and  $r_b$  into the steady-state  $\omega_{ss}$  and oscillatory components  $p_{osc}$ ,  $q_{osc}$  and  $r_{osc}$ .

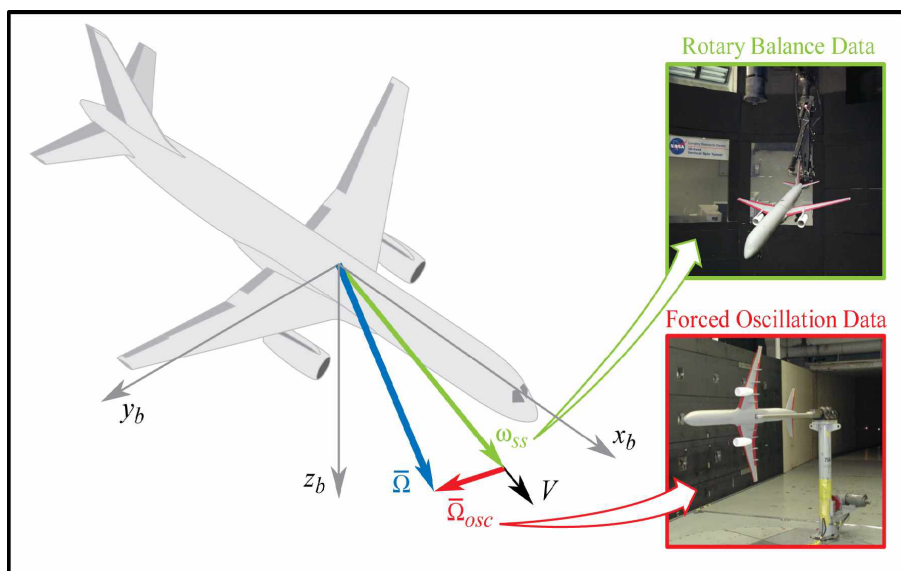


Figure A.8: Decomposition of the total angular rate vector into steady state and oscillatory components [13]

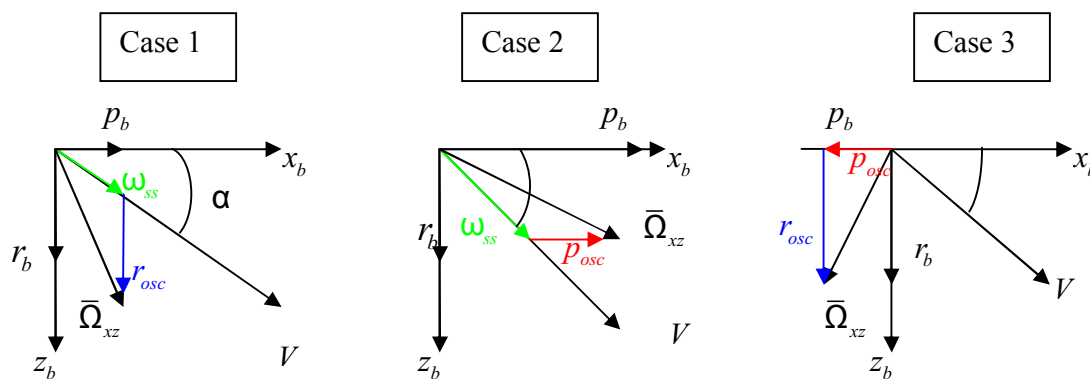


Figure A.9: Three decomposition schemes used with the Kalviste methods [13]

Table A.1: Equations used in the Hybrid Kalviste method [13]

Angular Rate Term	Case 1	Case 2	Case 3
$\omega_{ss} =$	$\frac{p_b}{\cos \alpha \cos \beta}$	$\frac{r_b}{\sin \alpha \cos \beta}$	0
$p_{osc} =$	0	$p_b - \omega_{ss} \cos \alpha \cos \beta$	$p_b$
$q_{osc} =$	$q_b - \omega_{ss} \sin \beta$	$q_b - \omega_{ss} \sin \beta$	$q_b$
$r_{osc} =$	$r_b - \omega_{ss} \sin \alpha \cos \beta$	0	$r_b$

The GTM uses the non-dimensionalised forms of the angular rates in the aerodynamic model. The non-dimensionalised steady-state and oscillatory rates are calculated as follows,

$$\hat{\omega}_{ss} = \frac{\omega_{ss} b}{2\bar{V}} \quad (\text{A.12})$$

$$\hat{p}_{osc} = \frac{p_{osc} b}{2\bar{V}} \quad (\text{A.13})$$

$$\hat{q}_{osc} = \frac{q_{osc} \bar{c}}{2\bar{V}} \quad (\text{A.14})$$

$$\hat{r}_{osc} = \frac{r_{osc} b}{2\bar{V}} \quad (\text{A.15})$$

The aerodynamic coefficient terms in the aerodynamic model of Equations A.10 and A.11 are each structured as an n-dimensional lookup table, where n is equal to the number of independent variables for that term. The aerodynamic coefficient terms are calculated using these tables by linearly interpolating the table breakpoints [13].

### A.3.2 Thrust Model

The thrust forces and moments are produced by the engines of the aircraft. Here we assume a standard engine configuration for large transport aircraft with twin underwing-mounted engines, one mounted below each wing, such as in the case of the NASA GTM model used in this thesis. The thrust forces and moments are functions of the throttle settings of the engines  $\delta_T$ , as well as the air density and the airspeed and can be expressed as,

$$\mathbf{F}^T = \mathbf{f}^T(\delta_T, \rho, \bar{V}) \quad (\text{A.16})$$

$$\mathbf{M}^T = \mathbf{m}^T(\delta_T, \rho, \bar{V}) \quad (\text{A.17})$$

where  $\mathbf{F}^T$  and  $\mathbf{M}^T$  are the thrust force and moment vectors, and  $\mathbf{f}^T$  and  $\mathbf{m}^T$  are general multivariable non-linear functions that are determined by the characteristics of the specific engines used on the aircraft.

The thrust force vector produced by the engines lies primarily along the positive body x-axis. Due to the alignment of the engines relative to the aircraft body, there may be small components in the body y-axis and body z-axis directions. Typical commercial aircraft have their engines pointed slightly upwards and inwards towards the fuselage. If the engines produce equal thrust as in most cases, the y-axis thrust components should cancel out, but not the z-axis components, resulting in a small thrust component expected in the body z-axis.

The thrust moment vector exists due to the engine thrust vector not acting through the aircraft's centre of mass, as well as the gyroscoping torques from the angular momentums of the two engines. In the case of aircraft with underwing-mounted engines, the thrust vector acts through a point below the centre of mass, which produces a dominant nose-up pitching moment proportional to the total engine thrust. The rolling and yawing moments due to the thrust vectors not acting through the centre of mass, tend to oppose each other and cancel out, due to the symmetry of the aircraft and the fact that the left and right engines are normally operated to produce equal thrust. The angular momentums of the engines also tend to oppose each other and cancel out, due to the fact that the engines are designed to rotate in opposite directions and are typically operated at equal engine speeds [1].

### A.3.3 Gravitational Model

In a flat earth NED axis system, the gravitational acceleration vector is adequately modelled as providing a force equivalent to the aircraft's mass in the down direction, that does not vary with latitude and longitude. The corresponding gravitational force co-ordinate vector in inertial axes is thus,

$$\mathbf{F}_I^G = \begin{bmatrix} 0 \\ 0 \\ mg \end{bmatrix} \quad (\text{A.18})$$

where standard gravitational acceleration  $g$  is used. The gravitational forces co-ordinated into the body-axis are functions of the attitude of the body-axis system relative to the inertial-axis system and is obtained using the DCM transformation matrix,

$$\mathbf{F}^G = \text{DCM}_{I \rightarrow B} \mathbf{F}_I^G = \begin{bmatrix} -\sin \Theta \\ \cos \Theta \sin \Phi \\ \cos \Theta \cos \Phi \end{bmatrix} mg \quad (\text{A.19})$$

Finally, because in a uniform gravitational field the centre of gravity coincides with the centre of mass, the gravitational force produces no moment on the aircraft. Thus,

$$\mathbf{M}^G = \mathbf{0} \quad (\text{A.20})$$

## A.4 Linearisation and Linear Analysis

To apply linear systems theory, the non-linear dynamics of the aircraft must be linearised around the trim condition. The mathematical linearisation process presented here is a recall from course notes [14] with reference from [38]. Additionally, a more in-depth linear analysis on the natural modes of the GTM model is given.



### A.4.1 Linearisation Derivation

The dynamics of Equations A.4 to A.9 and Equation A.12 can be written in a more concise non-linear state space form,

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}) \quad (\text{A.1})$$

where the absolute system states and inputs are,

$$\begin{aligned} \mathbf{x} &= [U, V, W, P, Q, R, \Phi, \Theta]^\top \\ \mathbf{u} &= [\delta_A, \delta_E, \delta_R, T]^\top \end{aligned} \quad (\text{A.2})$$

and  $\mathbf{f}$  is the vector function representing the respective dynamic equations. The dynamics governing the states  $\Psi$ ,  $N$ ,  $E$  and  $D$  are not linearisation variables, because they do not re-couple back into the above dynamics and thus do not constitute towards the fundamental aircraft dynamics [14]. Using small disturbance theory, each state and control can be written as the sum of a trim value and a perturbation about trim,

$$\mathbf{x} = \mathbf{x}_T + \Delta\mathbf{x} \quad (\text{A.3})$$

$$\mathbf{u} = \mathbf{u}_T + \Delta\mathbf{u} \quad (\text{A.4})$$

where the perturbation states and controls are,

$$\begin{aligned} \mathbf{x} &= [u, v, w, p, q, r, \phi, \theta]^\top \\ \mathbf{u} &= [\delta_a, \delta_e, \delta_r, \Delta T]^\top \end{aligned} \quad (\text{A.5})$$

Expanding Equation A.1 in a Taylor series about the trim condition yields,

$$\dot{\mathbf{x}}_T + \Delta\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}_T + \Delta\mathbf{x}, \mathbf{u}_T + \Delta\mathbf{u}) = \mathbf{f}(\mathbf{x}_T, \mathbf{u}_T) + \left. \frac{\partial \mathbf{f}}{\partial \mathbf{x}} \right|_T \Delta\mathbf{x} + \left. \frac{\partial \mathbf{f}}{\partial \mathbf{u}} \right|_T \Delta\mathbf{u} + \mathcal{O} \quad (\text{A.6})$$

At trim the time rate of change of the states are zero,

$$\dot{\mathbf{x}}_T = \mathbf{f}(\mathbf{x}_T, \mathbf{u}_T) = 0 \quad (\text{A.7})$$

and assuming that the perturbations from the trim states are small, the higher order terms in the above equation can be ignored and the dynamics approximated by the linearised sensitivities about trim,

$$\Delta\dot{\mathbf{x}} \approx \mathbf{A}_T \Delta\mathbf{x} + \mathbf{B}_T \Delta\mathbf{u}, \quad \mathbf{A}_T = \left. \frac{\partial \mathbf{f}}{\partial \mathbf{x}} \right|_T, \quad \mathbf{B}_T = \left. \frac{\partial \mathbf{f}}{\partial \mathbf{u}} \right|_T \quad (\text{A.8})$$

It is common practice to simplify the partial derivative state space matrices  $\mathbf{A}_T$ , and  $\mathbf{B}_T$  by decoupling the dynamics into longitudinal states  $\Delta\mathbf{x}_{long}$  and lateral states  $\Delta\mathbf{x}_{lat}$ . Due to the symmetry about the xz-plane and the assumption of small deviations from trim in the states such as roll, the coupling between the longitudinal and lateral states is approximated to zero in the linear state space representation,

$$\Delta \dot{\mathbf{x}} = \mathbf{A}_T \Delta \mathbf{x} + \mathbf{B}_T \Delta \mathbf{u} = \begin{bmatrix} \Delta \dot{\mathbf{x}}_{long} \\ \Delta \dot{\mathbf{x}}_{lat} \end{bmatrix} = \begin{bmatrix} \mathbf{A}_{long} & \mathbf{0} \\ \mathbf{0} & \mathbf{A}_{lat} \end{bmatrix} \begin{bmatrix} \Delta \mathbf{x}_{long} \\ \Delta \mathbf{x}_{lat} \end{bmatrix} + \begin{bmatrix} \mathbf{B}_{long} & \mathbf{0} \\ \mathbf{0} & \mathbf{B}_{lat} \end{bmatrix} \begin{bmatrix} \Delta \mathbf{u}_{long} \\ \Delta \mathbf{u}_{lat} \end{bmatrix} \quad (\text{A.9})$$

where,

$$\Delta \mathbf{x}_{long} = [u \quad w \quad q \quad \theta]^\top, \quad \Delta \mathbf{u}_{long} = [\delta_e \quad \Delta T]^\top \quad (\text{A.10})$$

$$\Delta \mathbf{x}_{lat} = [v \quad p \quad r \quad \phi]^\top, \quad \Delta \mathbf{u}_{lat} = [\delta_a \quad \delta_r]^\top \quad (\text{A.11})$$

The linearisation problem reduces to determining the partial derivatives that form the system and control matrices of the two decoupled linear systems. It is often more meaningful to present the aircraft velocity in terms of polar co-ordinates  $(\bar{V}, \alpha, \beta)$ . The following assumptions can be made for straight and level flight at small angles of attack and sideslip,

$$U = \bar{V} \cos \alpha \cos \beta \approx \bar{V} \quad (\text{A.12})$$

$$V = \bar{V} \sin \beta \approx \bar{V}_T \beta \quad (\text{A.13})$$

$$W = \bar{V} \cos \alpha \cos \beta \approx \bar{V}_T \alpha \quad (\text{A.14})$$

The decoupled four state longitudinal and four state lateral linear dynamics were obtained for the chosen trim condition using the provided linearise function that is included with the GTM model. The longitudinal linear dynamics are given by its state space representation,

$$\Delta \dot{\mathbf{x}}_{long} = \mathbf{A}_{long} \Delta \mathbf{x}_{long} + \mathbf{B}_{long} \Delta \mathbf{u}_{long} \quad (\text{A.15})$$

where the longitudinal state and input matrices are modelled by,

$$\mathbf{A}_{long} = \begin{bmatrix} \frac{\partial \dot{U}}{\partial U} & \bar{V}_T \frac{\partial \dot{U}}{\partial \alpha} & \frac{\partial \dot{U}}{\partial Q} & \frac{\partial \dot{U}}{\partial \Theta} \\ \frac{1}{\bar{V}_T} \frac{\partial \dot{\alpha}}{\partial U} & \frac{\partial \dot{\alpha}}{\partial \alpha} & \frac{1}{\bar{V}_T} \frac{\partial \dot{\alpha}}{\partial Q} & \frac{1}{\bar{V}_T} \frac{\partial \dot{\alpha}}{\partial \Theta} \\ \frac{\partial \dot{Q}}{\partial U} & \bar{V}_T \frac{\partial \dot{Q}}{\partial \alpha} & \frac{\partial \dot{Q}}{\partial Q} & \frac{\partial \dot{Q}}{\partial \Theta} \\ \frac{\partial \dot{\Theta}}{\partial U} & \bar{V}_T \frac{\partial \dot{\Theta}}{\partial \alpha} & \frac{\partial \dot{\Theta}}{\partial Q} & \frac{\partial \dot{\Theta}}{\partial \Theta} \end{bmatrix}, \quad \mathbf{B}_{long} = \begin{bmatrix} \frac{\partial \dot{U}}{\partial \delta_e} & \frac{\partial \dot{U}}{\partial T} \\ \frac{1}{\bar{V}_T} \frac{\partial \dot{\alpha}}{\partial \delta_e} & \frac{1}{\bar{V}_T} \frac{\partial \dot{\alpha}}{\partial T} \\ \frac{\partial \dot{Q}}{\partial \delta_e} & \frac{\partial \dot{Q}}{\partial T} \\ \frac{\partial \dot{\Theta}}{\partial \delta_e} & \frac{\partial \dot{\Theta}}{\partial T} \end{bmatrix} \quad (\text{A.16})$$

and the longitudinal states and input vectors are,

$$\Delta \mathbf{x}_{long} = [\bar{v} \quad \alpha \quad q \quad \theta]^\top, \quad \Delta \mathbf{u}_{long} = [\delta_e \quad \Delta T]^\top \quad (\text{A.17})$$

The lateral linear dynamics are given by its state space representation,

$$\Delta \dot{\mathbf{x}}_{lat} = \mathbf{A}_{lat} \Delta \mathbf{x}_{lat} + \mathbf{B}_{lat} \Delta \mathbf{u}_{lat} \quad (\text{A.18})$$

where the lateral state and input matrices are modelled by,

$$\mathbf{A}_{lat} = \begin{bmatrix} \frac{\partial \dot{V}}{\partial V} & \frac{1}{\bar{V}_T} \frac{\partial \dot{V}}{\partial P} & \frac{1}{\bar{V}_T} \frac{\partial \dot{V}}{\partial R} & \frac{1}{\bar{V}_T} \frac{\partial \dot{V}}{\partial \Phi} \\ \bar{V}_T \frac{\partial \dot{P}}{\partial V} & \frac{\partial \dot{P}}{\partial P} & \frac{\partial \dot{P}}{\partial R} & \frac{\partial \dot{P}}{\partial \Phi} \\ \bar{V}_T \frac{\partial \dot{R}}{\partial V} & \frac{\partial \dot{R}}{\partial P} & \frac{\partial \dot{R}}{\partial R} & \frac{\partial \dot{R}}{\partial \Phi} \\ \bar{V}_T \frac{\partial \dot{\Phi}}{\partial V} & \frac{\partial \dot{\Phi}}{\partial P} & \frac{\partial \dot{\Phi}}{\partial R} & \frac{\partial \dot{\Phi}}{\partial \Phi} \end{bmatrix}, \quad \mathbf{B}_{lat} = \begin{bmatrix} \frac{1}{\bar{V}_T} \frac{\partial \dot{V}}{\partial \delta_a} & \frac{1}{\bar{V}_T} \frac{\partial \dot{V}}{\partial \delta_r} \\ \frac{\partial \dot{P}}{\partial \delta_a} & \frac{\partial \dot{P}}{\partial \delta_r} \\ \frac{\partial \dot{R}}{\partial \delta_a} & \frac{\partial \dot{R}}{\partial \delta_r} \\ \frac{\partial \dot{\Phi}}{\partial \delta_a} & \frac{\partial \dot{\Phi}}{\partial \delta_r} \end{bmatrix} \quad (\text{A.19})$$

and the lateral states and input vectors are,

$$\Delta \mathbf{x}_{lat} = [\beta \quad p \quad r \quad \phi]^T, \quad \Delta \mathbf{u}_{lat} = [\delta_a \quad \delta_r]^T \quad (\text{A.20})$$

#### A.4.2 Linear Analysis of Natural Modes of Motion

The dynamic response of the linearised aircraft dynamics is governed by the system's poles, which are the eigenvalues of the system's state matrix  $\mathbf{A}$ . The poles of the longitudinal dynamics and lateral dynamics are the eigenvalues of the respective linear systems'  $\mathbf{A}_{long}$  and  $\mathbf{A}_{lat}$  state matrices.

##### Longitudinal Modes of Motion

The longitudinal system pole locations are shown in Figure A.10 and consist of two complex pole pairs. The high frequency pole pair is referred to as the short-period mode and the low frequency pair is referred to as the phugoid mode.

The short-period mode describes the aircraft's tendency to realign itself with the velocity vector when disturbed longitudinally. The phugoid mode is a largely a slow kinematic mode of motion and describes the exchange of potential and kinetic energy when the aircraft is disturbed from trimmed flight. The natural frequency and damping characteristics of the longitudinal modes of motion are listed in Table A.2.

Table A.2: Longitudinal characteristics.

Characteristic	Short-Period Mode	Phugoid Mode
$\omega_n$ (rad/s)	8.09	0.247
$\zeta$	0.46	0.076

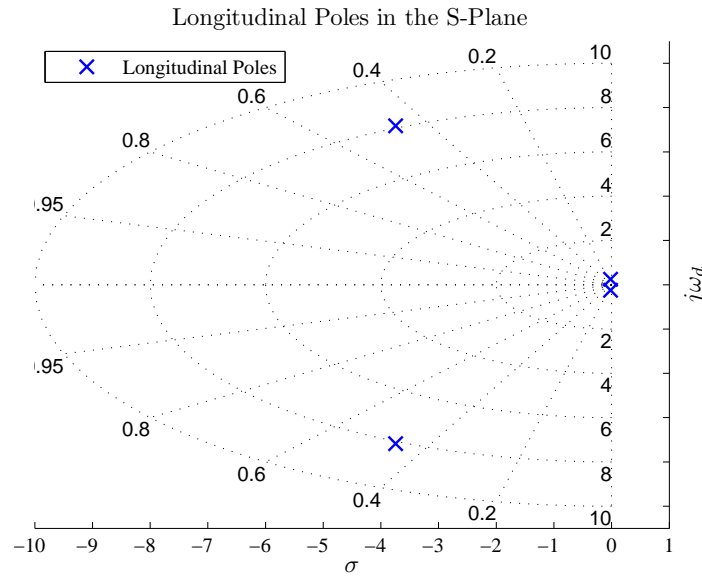


Figure A.10: Longitudinal Poles in the S-Plane.

### Lateral Modes of Motion

The lateral system pole locations are shown in Figure A.11 and consist of two real poles and a complex pole pair. These modes of motion are commonly referred to (from highest to lowest natural frequency) as the roll, dutch roll and spiral modes.

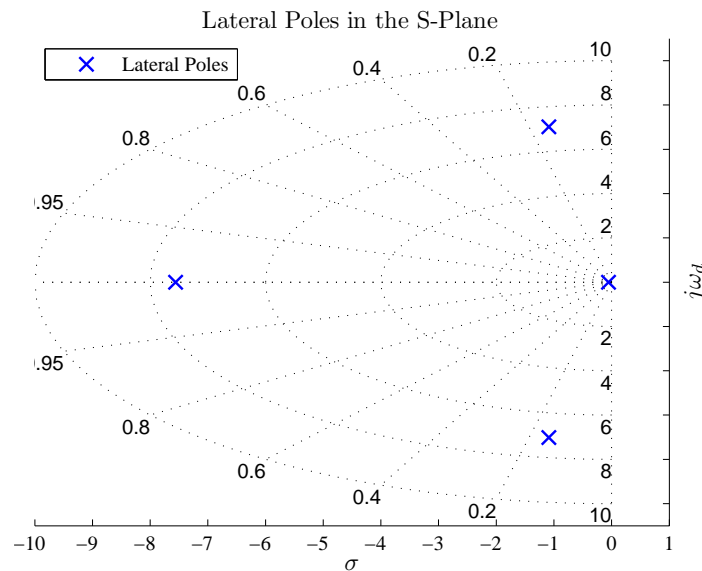


Figure A.11: Lateral Poles in the S-Plane

The fast roll mode describes the roll rate dynamics of an aircraft. When an aircraft experiences a roll moment disturbance, the roll rate will initially start to grow but will quickly be damped by the wing's natural roll damping to a constant roll rate. The fast nature of this mode usually makes aircraft appear to always operate at a constant roll rate. The complex pole pair in Figure A.11 is referred to as the dutch roll mode and is usually very poorly damped. Its similar to the short-period mode, as it describes the tendency of the aircraft to align itself with the oncoming

airflow when disturbed laterally. The spiral mode is usually quite slow and it is also very common for the mode to be slightly unstable. Like the phugoid mode, the spiral mode is a largely kinematic mode of motion and describes the tendency of the aircraft to restore itself to wings level flight or diverge from wings level flight when laterally disturbed. The natural frequency and damping characteristics of the lateral modes of motion are listed in Table A.3.

Table A.3: Longitudinal characteristics.

Characteristic	Roll Mode	Dutch Roll Mode	Spiral Mode
$\omega_n$ (rad/s)	7.56	7.1	0.053
$\zeta$	1	0.153	1

To validate the linear state space models, we investigate the response to control inputs and how it corresponds to the full non-linear model of the GTM Simulink model. For the longitudinal model a doublet input is applied from trim to the elevator while the thrust is held constant with  $\Delta T = 0$ . Figure A.12 shows the linear and non-linear response to an elevator command doublet with 1 degree amplitude and a 0.5 second input pulse duration that starts at 2 seconds.

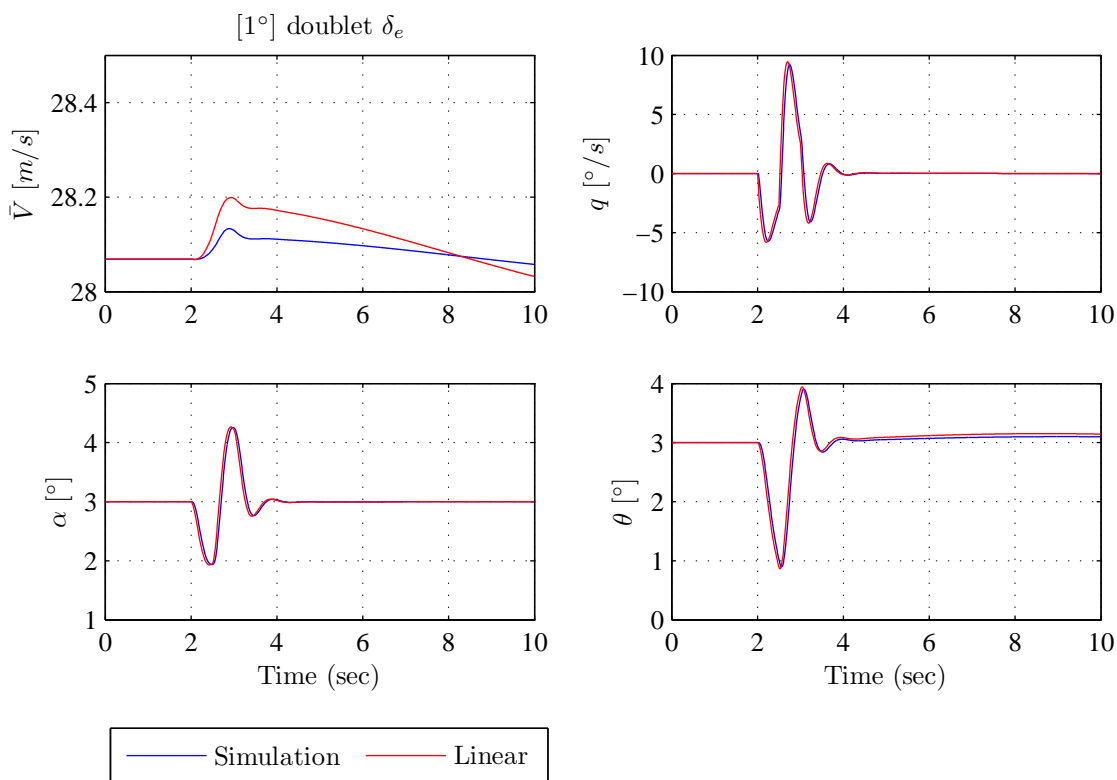


Figure A.12: Non-linear versus linear longitudinal system response for small elevator control input doublet

For the lateral model a doublet input is applied from trim to the aileron and rudder respectively. Figure A.13 shows the linear and non-linear response to an aileron and rudder command doublet with 1 degree amplitude and a 0.5 second input pulse duration that starts at 2 seconds.

The decoupled linear models corresponds quite well with the non-linear simulation model and can be used in the conventional linear aircraft controller design process in Chapter 3.

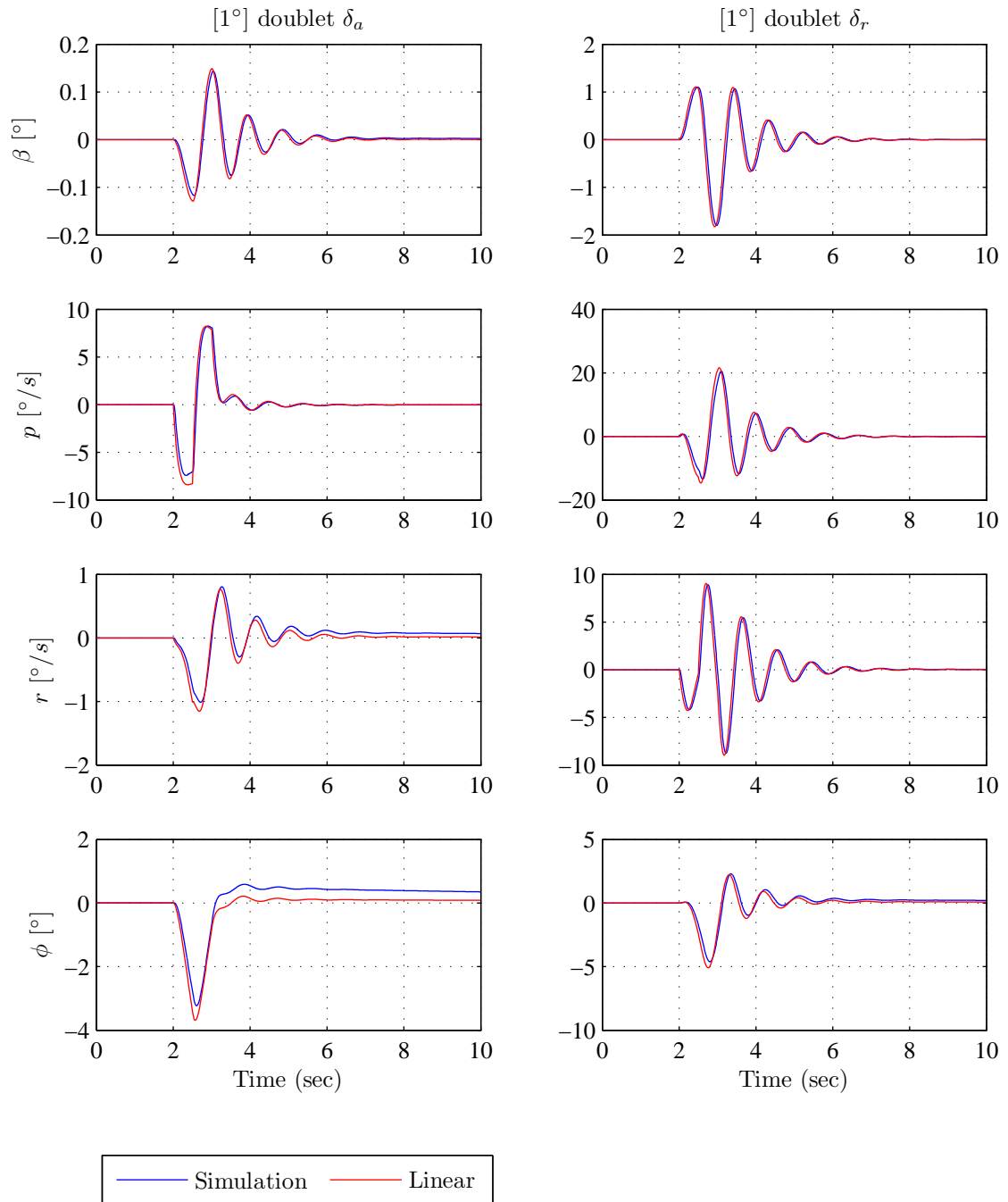


Figure A.13: Non-linear versus linear lateral system response for small aileron and rudder control input doublet respectively

## Appendix B

# Additional Numerical Results

This appendix provides the numerical error estimation plots for the numerical results in Chapter 4 and initial numerical trajectory results of the SQP method.

### B.1 Error Plots of Numerical Results

All the corresponding error plots for all the SQP thrust varying trajectory cases of section §4.5.11 are plotted in this section. The top half of each figure plots the error estimate in the dynamics of each state trajectory in time. The bottom half of each figure plots the total estimated error of the state of each interval of the trajectory. This is the integral over time of the error in state dynamics between each interval.

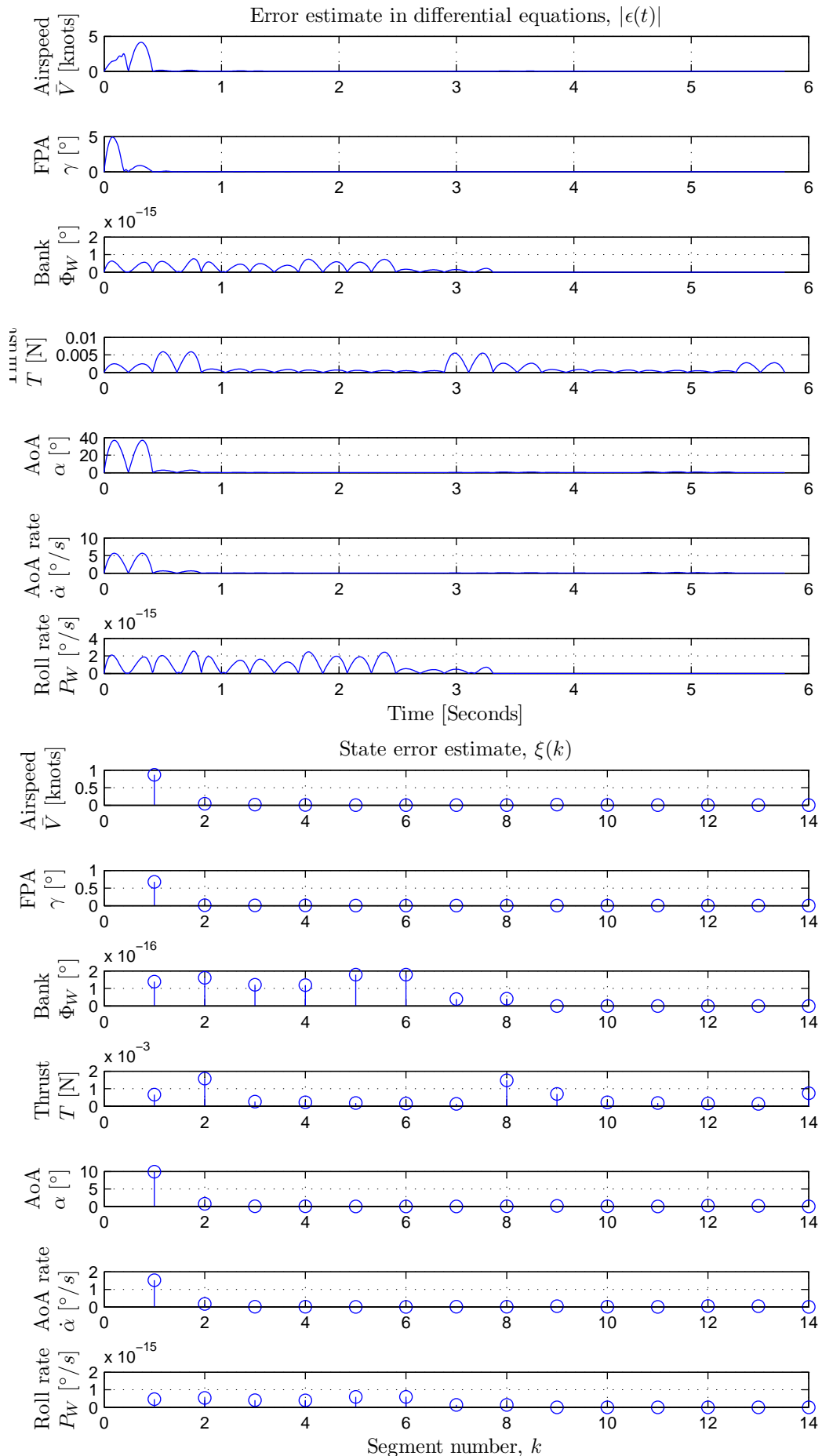


Figure B.1: Estimated error in SQP trajectory for step flight path angle descent upset



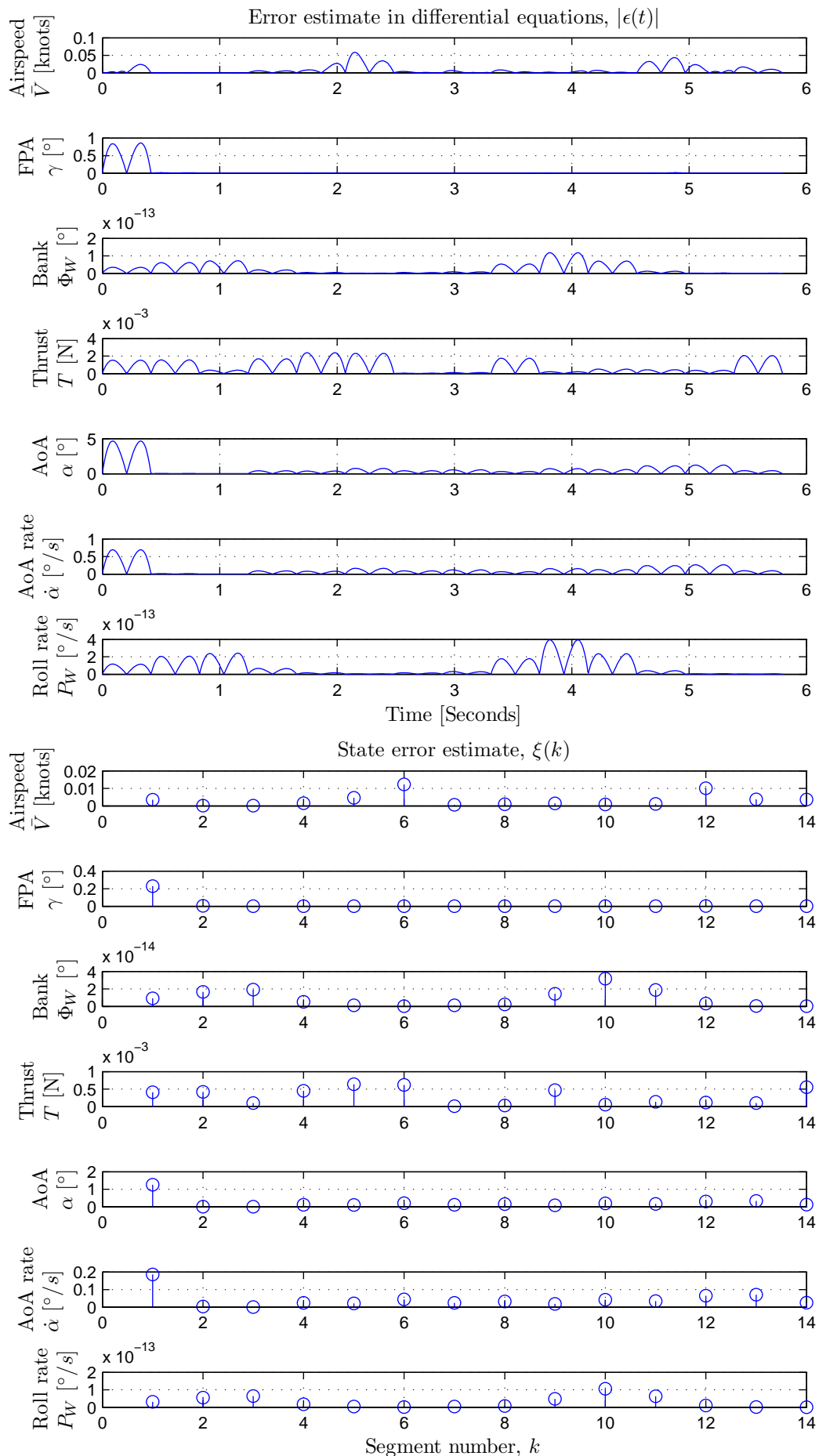


Figure B.2: Estimated error in SQP trajectory for step flight path angle climb upset

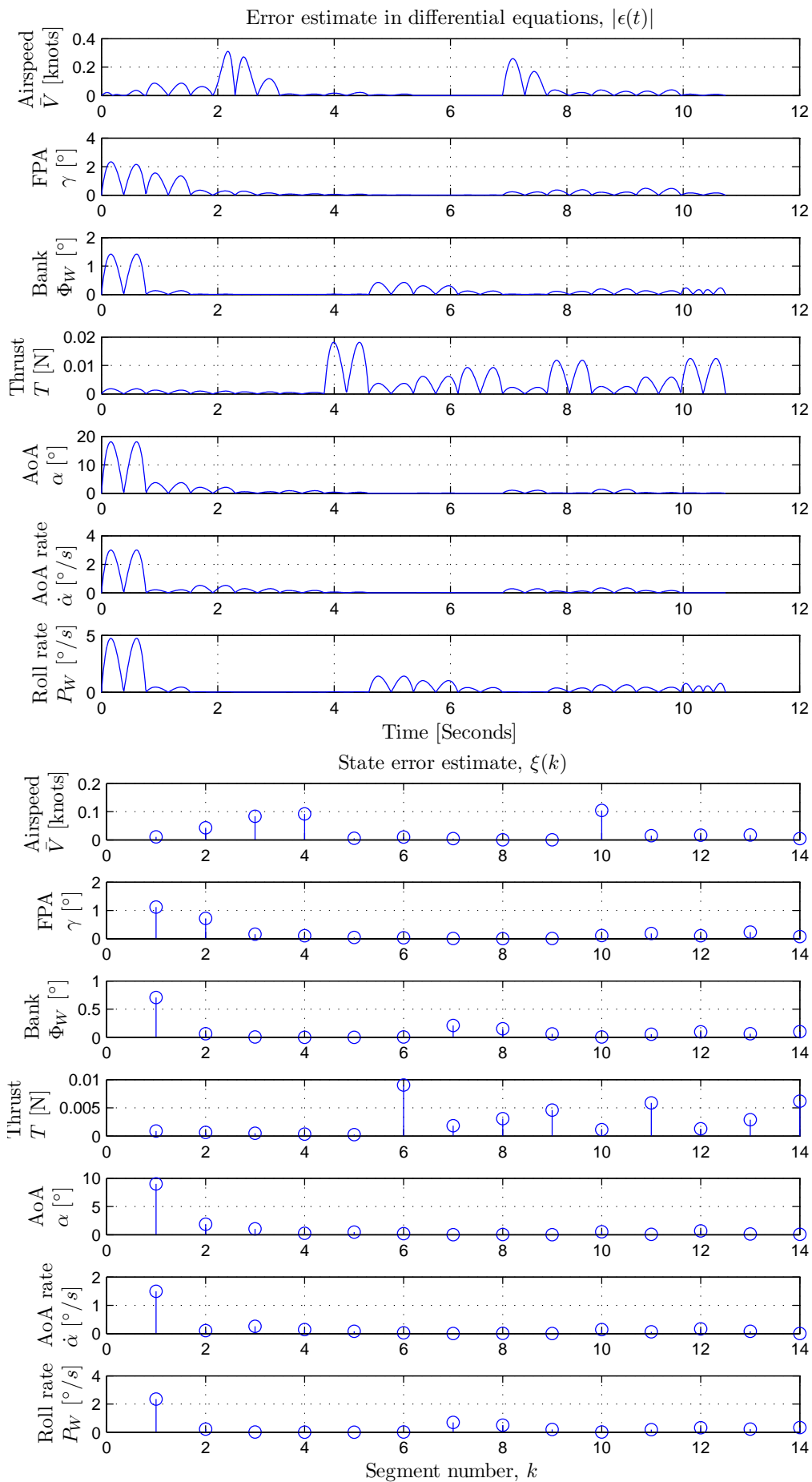


Figure B.3: Estimated error in SQP trajectory for bank angle upset

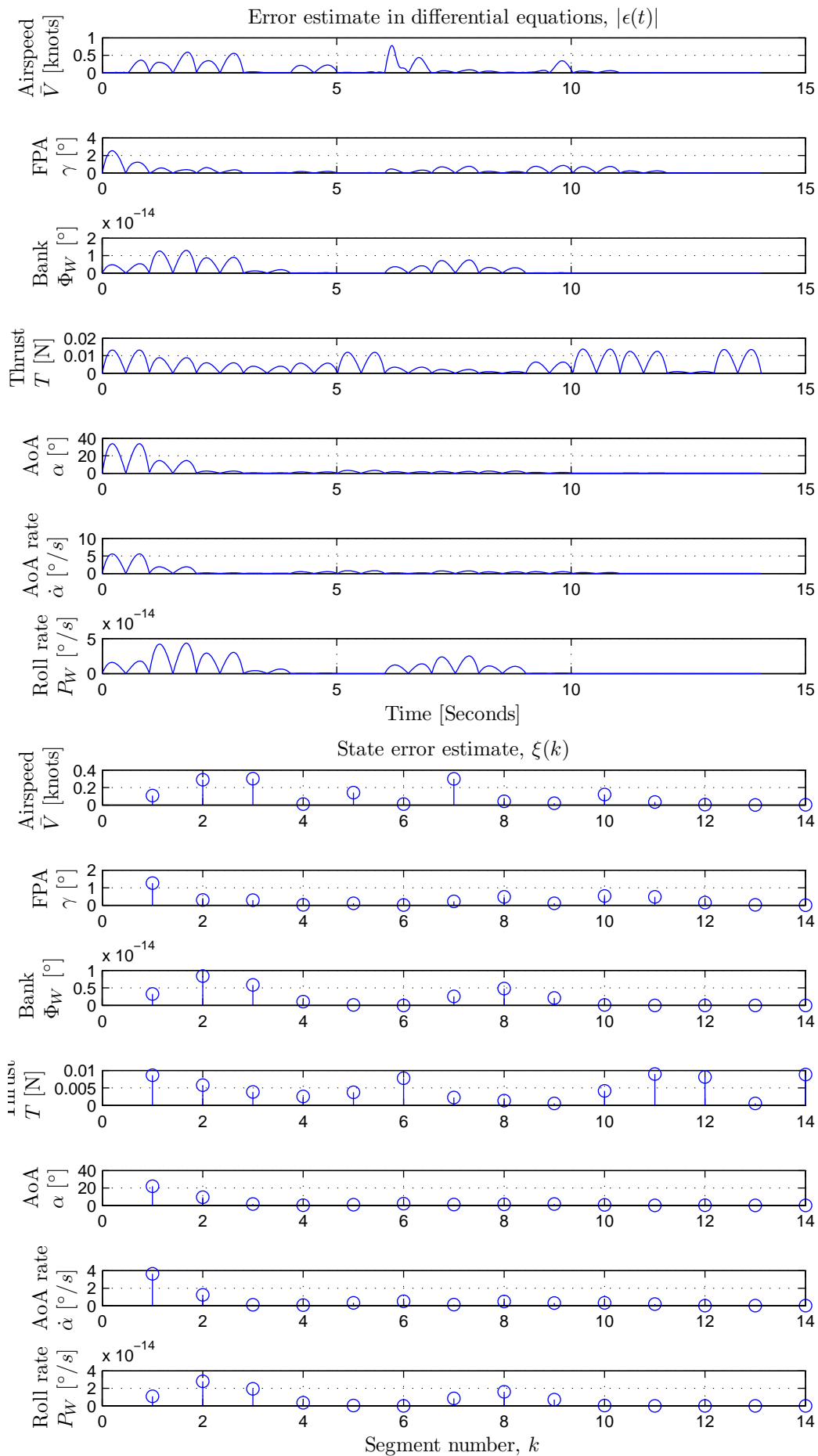


Figure B.4: Estimated error in SQP trajectory for underspeed upset

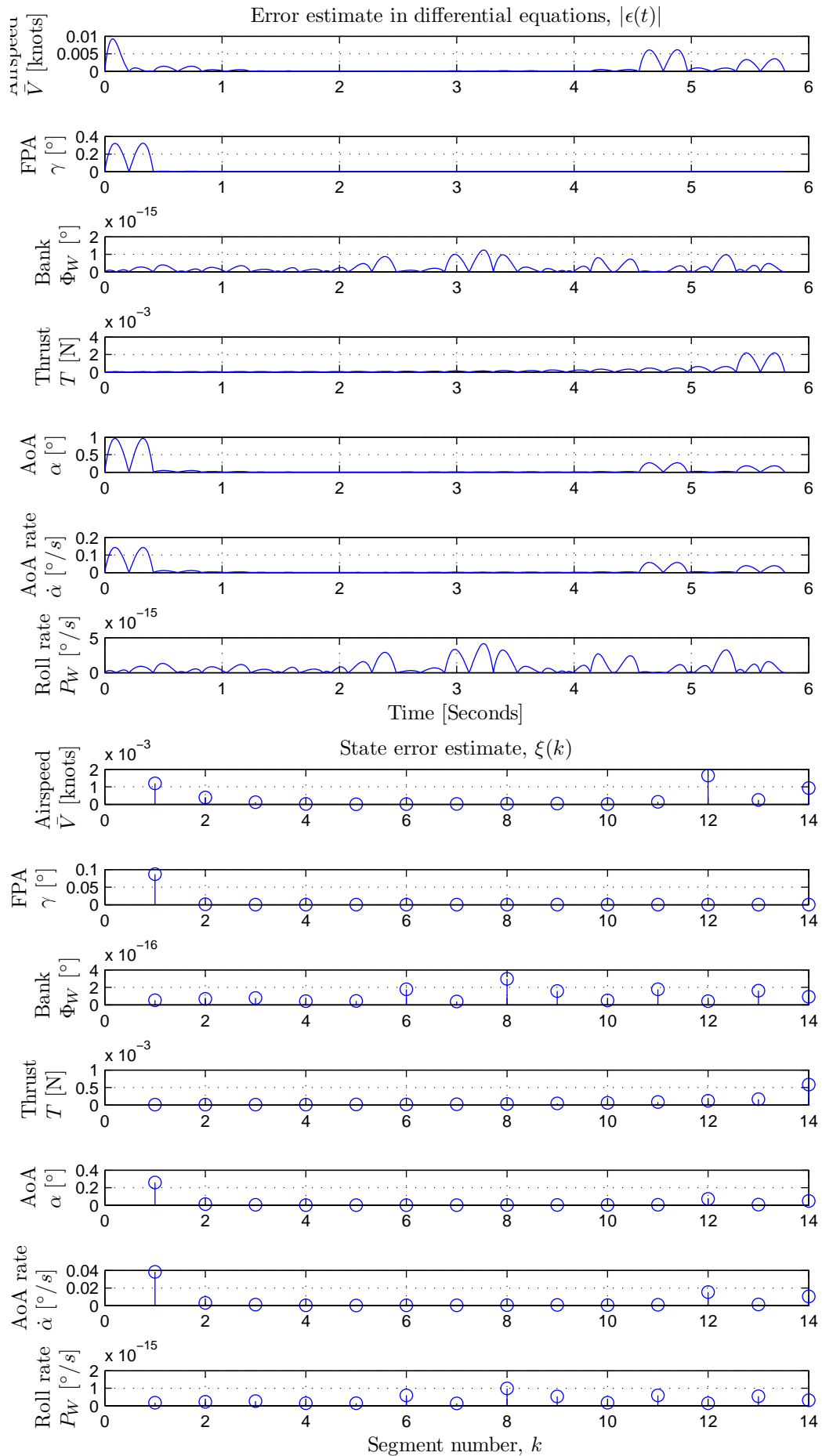


Figure B.5: Estimated error in SQP trajectory for overspeed upset

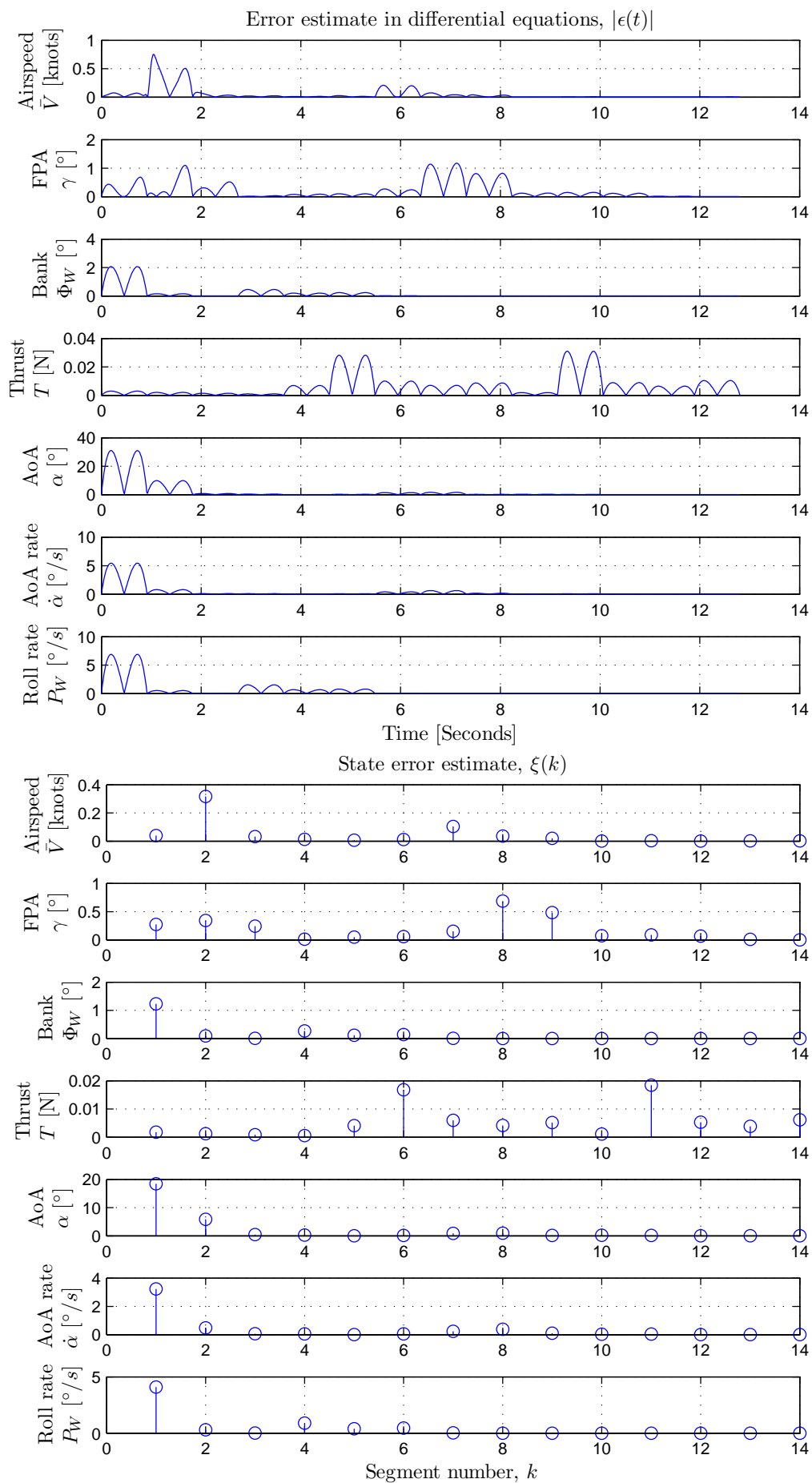


Figure B.6: Estimated error in SQP trajectory for flight path angle and bank angle upset

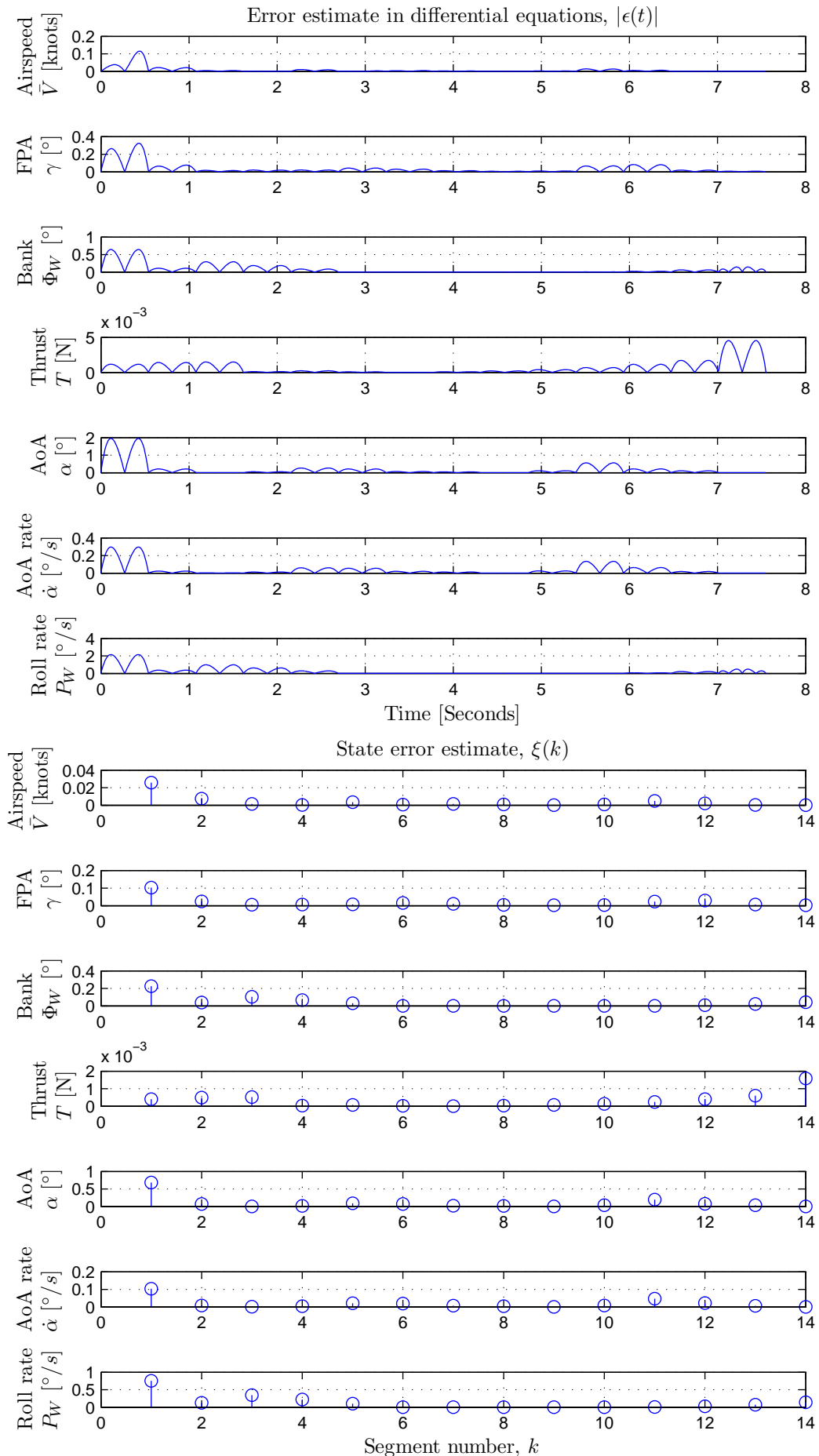


Figure B.7: Estimated error in SQP trajectory for FPA, bank angle and overspeed upset

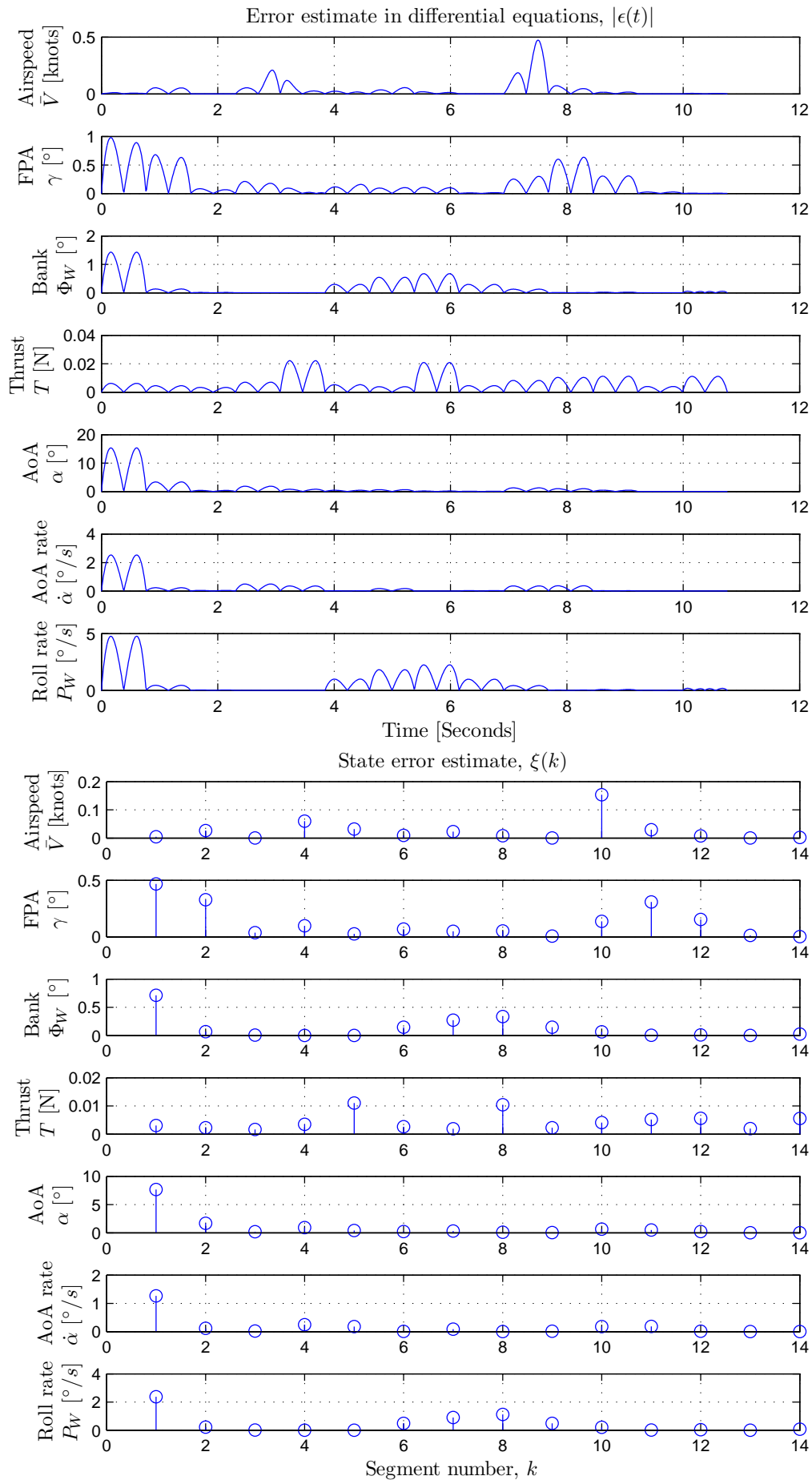


Figure B.8: Estimated error in SQP trajectory for FPA, bank angle and underspeed upset

## B.2 Initial SQP Trajectory Results

This section provides results from the initial SQP implementation that used successively higher dimensions of the system dynamics in a 'stepping-stone' approach. The SQP routine was used to generate trajectory solutions from initial upset conditions for each of these lower dimensional cases. For these initial implementations, the initial guess of the design variable values provided to the SQP routine was obtained from a dynamic programming solution for the specific initial upset condition. Thus the SQP was 'seeded' by the dynamic programming solution instead of using independent guesses.

### B.2.1 System Dynamics

The problem is divided into a set of cases of increasing dimensionality, solving the lower dimensional problem before moving to the next. The different cases are as follows.

**Two-Dimensional case** The two dimensional case of the directly transcribed aircraft system is described by the following equations,

$$\bar{V}(k+1) = \bar{V}(k) + \frac{1}{m} \left[ T(k) - \frac{1}{2} \rho \bar{V}(k)^2 SC_D(\alpha(k)) - mg \sin(\gamma(k)) \right] \Delta t \quad (\text{B.1})$$

$$\gamma(k+1) = \gamma(k) + \frac{1}{\bar{V}(k)m} \left[ \frac{1}{2} \rho \bar{V}(k)^2 SC_L(\alpha(k)) - mg \cos(\gamma(k)) \right] \Delta t \quad (\text{B.2})$$

The system state vector for this case is then,

$$\mathbf{x}_k = \left[ \bar{V}_k \quad \gamma_k \right]^T \quad (k = 0, 1, \dots, K) \quad (\text{B.3})$$

and the input vector is,

$$\mathbf{u}_k = \left[ \alpha_k \quad T_k \right]^T \quad (k = 0, 1, \dots, K) \quad (\text{B.4})$$

**Three-Dimensional Thrust Dynamic Case** The two dimensional case is expanded by including a first order thrust lag state into the system. The directly transcribed aircraft system now becomes a three dimensional system described by,

$$\bar{V}(k+1) = \bar{V}(k) + \frac{1}{m} \left[ T(k) - \frac{1}{2} \rho \bar{V}(k)^2 SC_D(\alpha(k)) - mg \sin(\gamma(k)) \right] \Delta t \quad (\text{B.5})$$

$$\gamma(k+1) = \gamma(k) + \frac{1}{\bar{V}(k)m} \left[ \frac{1}{2} \rho \bar{V}(k)^2 SC_L(\alpha(k)) - mg \cos(\gamma(k)) \right] \Delta t \quad (\text{B.6})$$

$$T(k+1) = e^{-\Delta t/\tau} T(k) + (1 - e^{-\Delta t/\tau}) T_c(k) \quad (\text{B.7})$$

where  $T_c$  is the Thrust command and  $T$  is now a thrust output state. Equation B.7 is derived from the z-transform of the first order model. The system state vector for this case is then,

$$\mathbf{x}_k = \left[ \bar{V}_k \quad \gamma_k \quad T_k \right]^T \quad (k = 0, 1, \dots, K) \quad (\text{B.8})$$

and the input vector is,



$$\mathbf{u}_k = [\alpha_k \quad T_{ck}]^T \quad (k = 0, 1, \dots, K) \quad (\text{B.9})$$

**Three-Dimensional bank angle case** Another three dimensional case is were we expand the two dimensional case with the bank angle dynamics. The directly transcribed aircraft system now becomes a three dimensional system described by,

$$\bar{V}(k+1) = \bar{V}(k) + \frac{1}{m} \left[ T(k) - \frac{1}{2} \rho \bar{V}(k)^2 SC_D(\alpha(k)) - mg \sin(\gamma(k)) \right] \Delta t \quad (\text{B.10})$$

$$\gamma(k+1) = \gamma(k) + \frac{1}{\bar{V}(k)m} \left[ \frac{1}{2} \rho \bar{V}(k)^2 SC_L(\alpha(k)) \cos(\Phi(k)) - mg \cos(\gamma(k)) \right] \Delta t \quad (\text{B.11})$$

$$\Phi(k+1) = \Phi(k) + P_{W_c}(k) \quad (\text{B.12})$$

where  $P_{W_c}$  is the wind axis roll rate command and  $\Phi$  is the bank angle. The system state vector for this case is then,

$$\mathbf{x}_k = [\bar{V}_k \quad \gamma_k \quad \Phi_k]^T \quad (k = 0, 1, \dots, K) \quad (\text{B.13})$$

and the input vector is,

$$\mathbf{u}_k = [\alpha_k \quad T_k \quad P_{W_{ck}}]^T \quad (k = 0, 1, \dots, K) \quad (\text{B.14})$$

**Four-Dimensional Case** The four dimensional case includes all the previously defined dynamics resulting in a four dimensional system. The directly transcribed aircraft system now becomes a four dimensional system described by,

$$\bar{V}(k+1) = \bar{V}(k) + \frac{1}{m} \left[ T(k) - \frac{1}{2} \rho \bar{V}(k)^2 SC_D(\alpha(k)) - mg \sin(\gamma(k)) \right] \Delta t \quad (\text{B.15})$$

$$\gamma(k+1) = \gamma(k) + \frac{1}{\bar{V}(k)m} \left[ \frac{1}{2} \rho \bar{V}(k)^2 SC_L(\alpha(k)) \cos(\Phi(k)) - mg \cos(\gamma(k)) \right] \Delta t \quad (\text{B.16})$$

$$\Phi(k+1) = \Phi(k) + P_{W_c}(k) \quad (\text{B.17})$$

$$T(k+1) = e^{-\Delta t/\tau} T(k) + (1 - e^{-\Delta t/\tau}) T_c(k) \quad (\text{B.18})$$

The system state vector for this case is then,

$$\mathbf{x}_k = [\bar{V}_k \quad \gamma_k \quad \Phi_k \quad T_k]^T \quad (k = 0, 1, \dots, K) \quad (\text{B.19})$$

and the input vector is,

$$\mathbf{u}_k = [\alpha_k \quad T_{ck} \quad P_{W_{ck}}]^T \quad (k = 0, 1, \dots, K) \quad (\text{B.20})$$

### B.2.2 Problem Setup

The problem's variables were set to the values detailed in Table B.1. The maximum  $\bar{V}$  is arbitrarily chosen to be above the normal trim speed but below the structural integrity envelope. The minimum  $\bar{V}$  is chosen to be well within the stall region on the aircraft. The final time instant  $\bar{V}$  bounds are chosen to be within safe cruising speed limits where no extreme inputs are needed to trim the aircraft for these speeds. The maximum angle of attack is chosen with the motivation that the inner loop AoA controller cannot follow AoA commands above 21 degrees. The limit on roll rate is artificially chosen such that passenger aircraft are not capable of excessive roll rates due to safety factors. Note that the bank angle is limited to positive angles, while the roll rate is limited to negative rates. This is because the bank angle solution can be mirrored for negative bank angles and it reduces the search space. Another reason for these limits is to reduce numerical 'oscillation' around  $0^\circ$  as there should be no incentive to bank away from level flight.

Table B.1: Problem parameter values

Parameter	Symbol	Value	Units
<i>Physical constants</i>			
Mass	$m$	23.59	kg
Gravitational constant	$g$	9.81	m/s <sup>2</sup>
Wing surface area	$S$	0.548	m <sup>2</sup>
Dynamic pressure	$\rho$	1.225	kg/m <sup>3</sup>
Aerodynamic lift Coefficient	$C_L$	Figure B.9	-
Aerodynamic drag Coefficient	$C_D$	Figure B.9	-
Engine lag time constant	$\tau$	2.5	seconds
<i>State bounds</i>			
Velocity	$\bar{V}_U, \bar{V}_L$	145, 20	kn
Final velocity	$\bar{V}_{NU}, \bar{V}_{NL}$	120, 75	kn
Flight path angle	$\gamma_U, \gamma_L$	30, -60	degrees
Bank angle	$\Phi_U, \Phi_L$	180, 0	degrees
Final flight path angle	$\gamma_N$	0	degrees
Final bank angle	$\Phi_N$	0	degrees
<i>Input bounds</i>			
Angle of attack	$\alpha_U, \alpha_L$	21, 0	degrees
Roll rate	$P_{WU}, P_{WL}$	0, -36	degrees/second
Thrust	$T_U, T_L$	67, 0	Newton
<i>Time</i>			
Initial time	$t_0$	0	seconds
Final time	$t_N$	14	seconds
Number of time steps	$N$	14	-
Number of grid points	$N + 1$	15	-
Sample time	$\Delta t$	1	seconds
Dynamic programming sample time	$\Delta t_{DP}$	1	seconds
Number of dynamic programming grid points	$N_{DP} + 1$	15	-

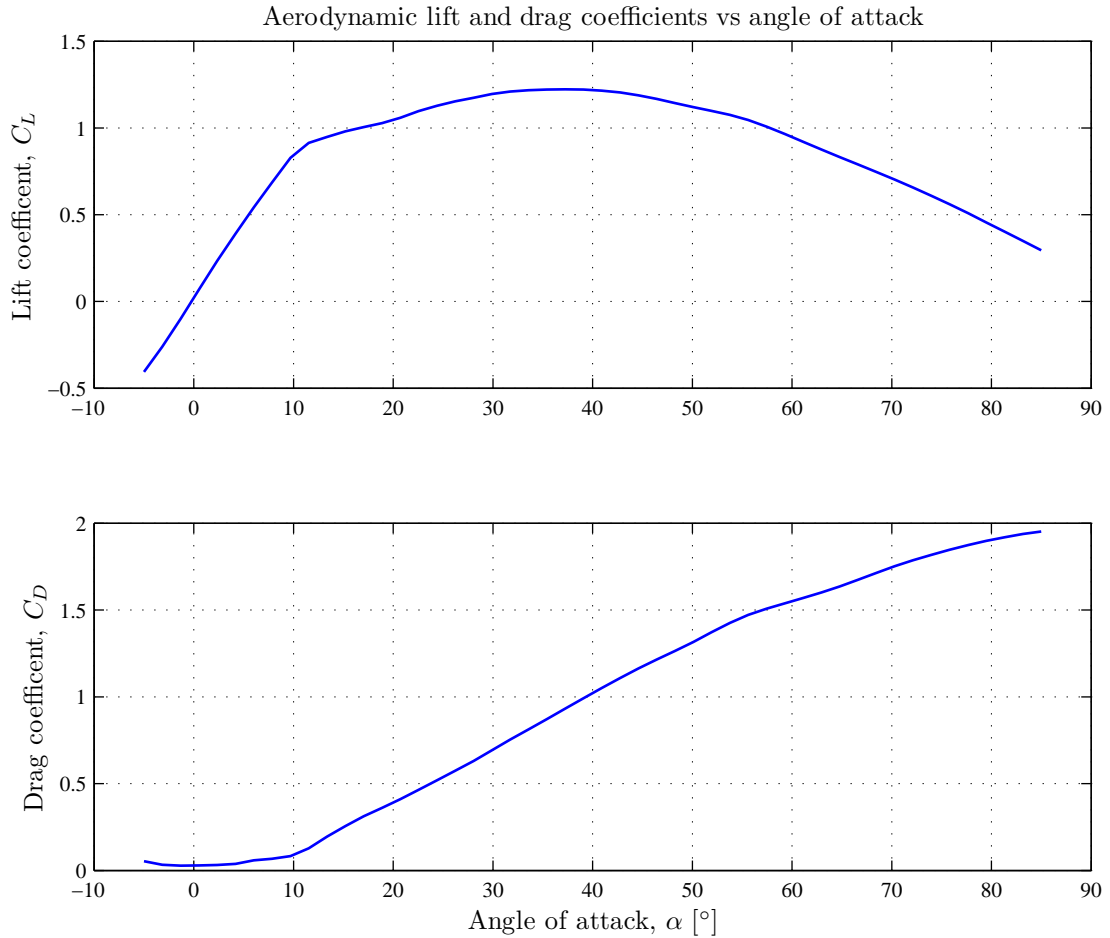


Figure B.9: Aerodynamic lift and drag coefficient functions

### B.2.3 Two-Dimensional case

The two-dimensional case used a two-dimensional dynamic programming seed, of which one had a varying thrust input and another had a constant thrust input. The seed was retrieved from the dynamic programming solution table with the forward grid search method. The initial condition/constraint for the two dimensional case's trajectories were set to an underspeed condition as follows,

Table B.2: Initial state values and dynamic programming setup for two dimensional case

Parameter	Symbol	Value	Units
<i>Initial condition</i>			
Velocity	$\bar{V}_{initial}$	25	kn
Flight path angle	$\gamma_{initial}$	0	degrees
<i>dynamic programming seed quantisation resolution</i>			
Velocity	$\Delta\bar{V}_q$	2.55	kn
Flight path angle	$\Delta\gamma_q$	2.14	degrees

The SQP was seeded with two cases of the two-dimensional dynamic programming solution, namely a constant thrust solution and a variable thrust solution. The variable thrust solution is shown in Figure B.10 and the constant thrust solution is shown in Figure B.11.

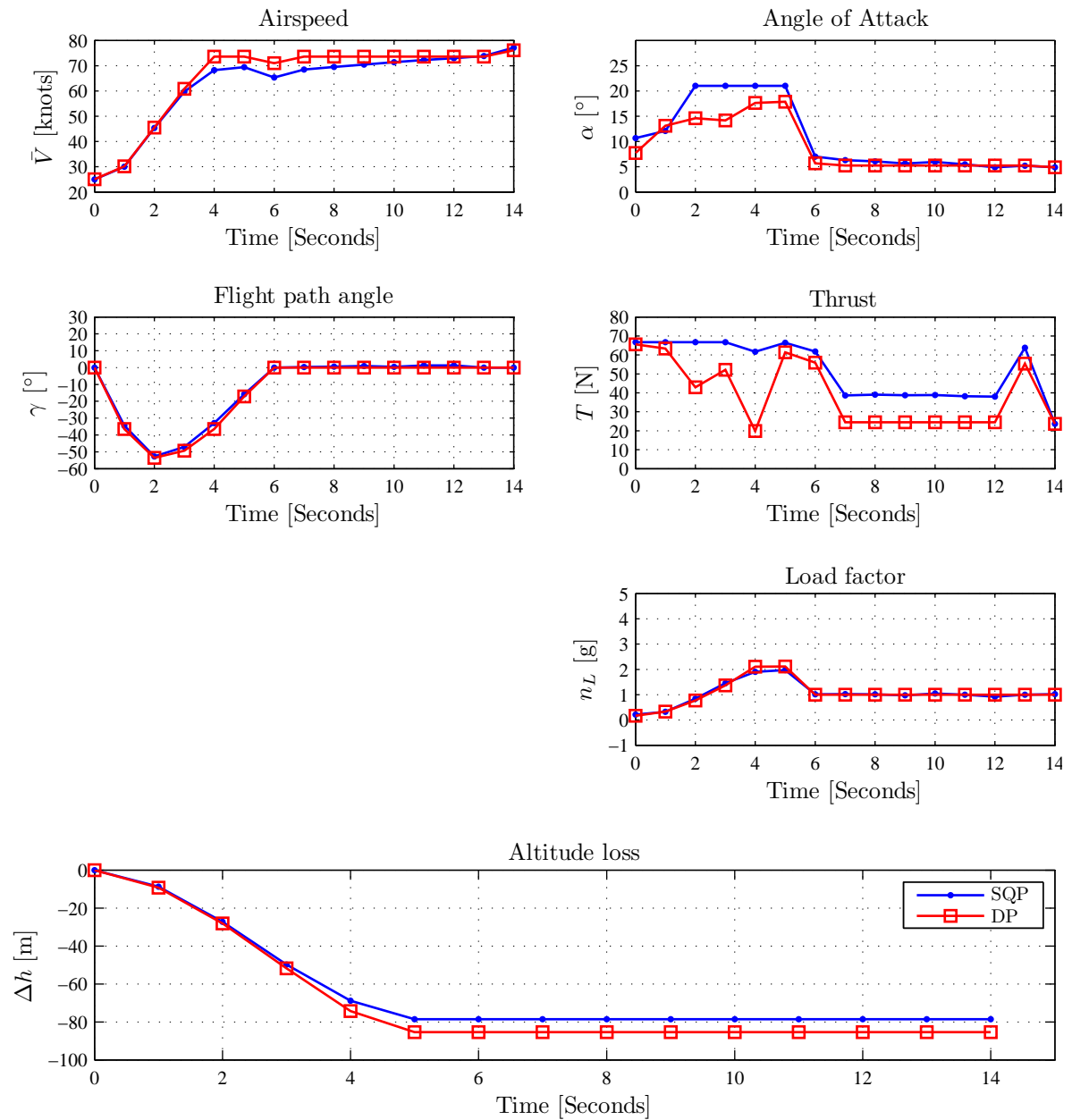


Figure B.10: SQP vs dynamic programming solution with variable thrust dynamic programming seed

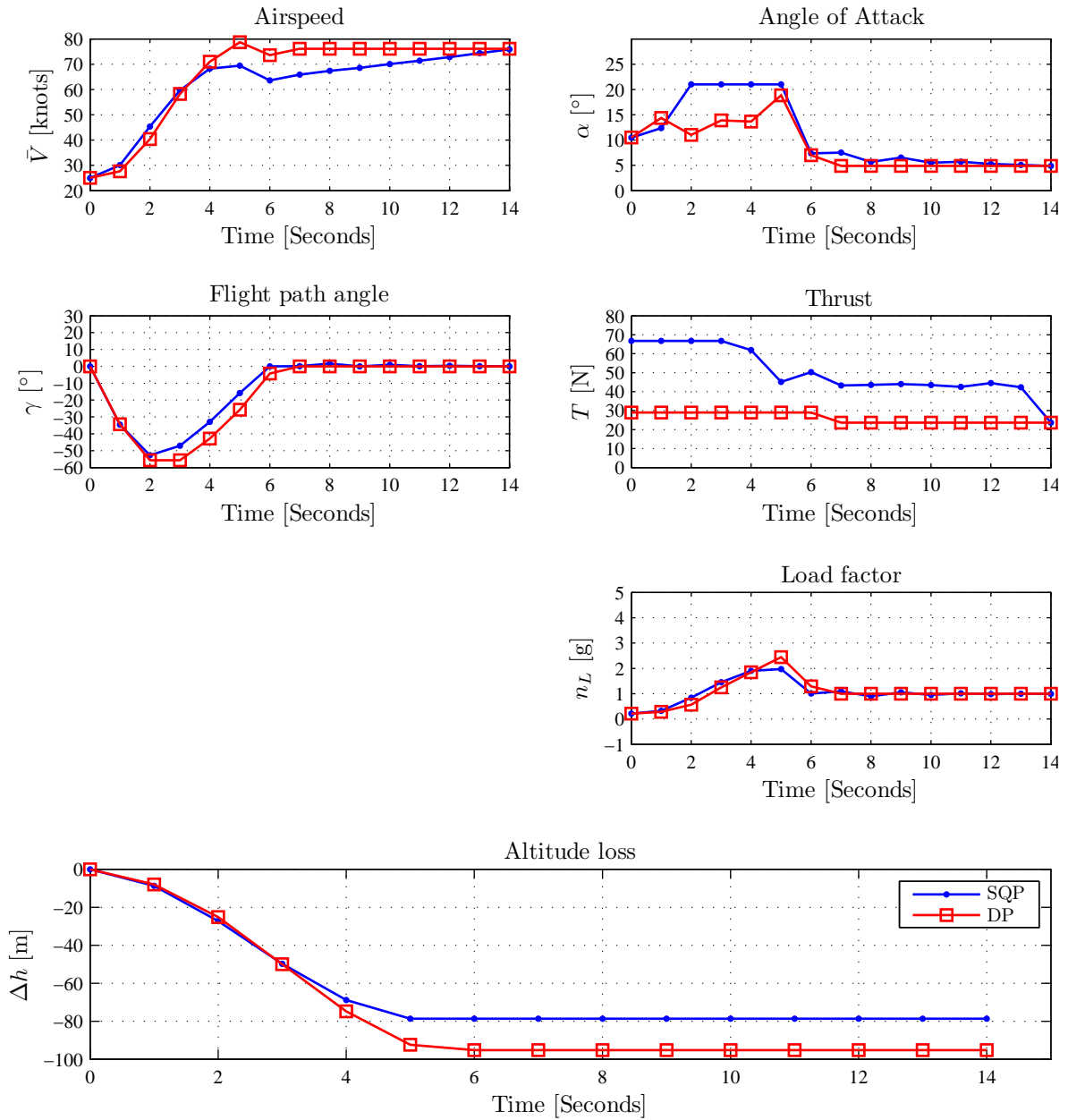


Figure B.11: SQP vs dynamic programming solution with constant thrust dynamic programming seed

The dynamic programming seed was truncated to the time instant the recovery was completed, instead of using the full 14 second fixed time window. This was done to force the SQP to recover within the same time, instead of only reaching the recovered terminal state at the end of the fixed time window. For the truncated cases, the variable thrust solution is shown in Figure B.13 and the constant thrust solution is shown in Figure B.11. From the results it can be seen that the SQP matches the DP solution much closer than in the previous two cases.

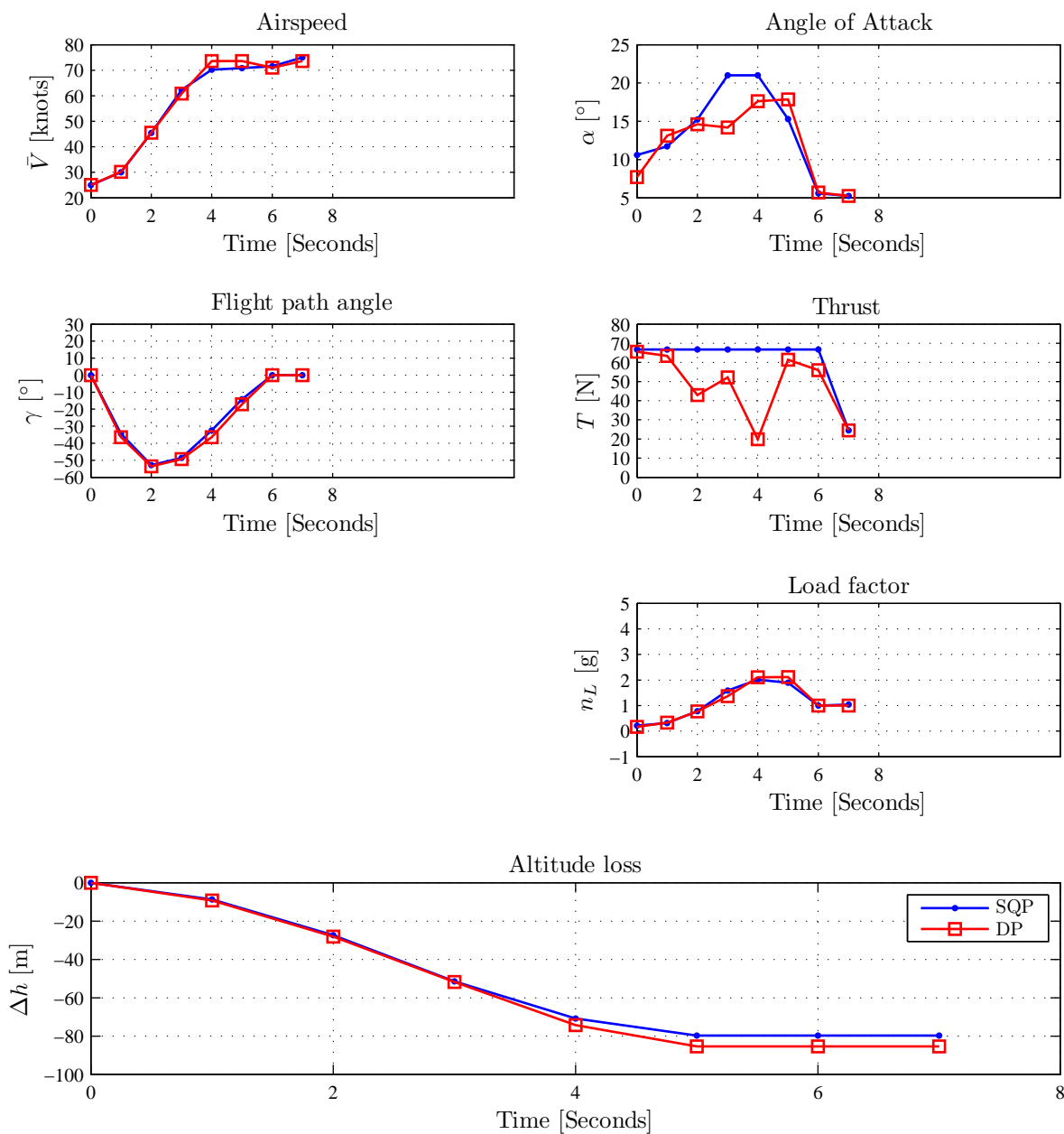


Figure B.12: SQP vs dynamic programming solution with a truncated time window and a variable thrust dynamic programming seed

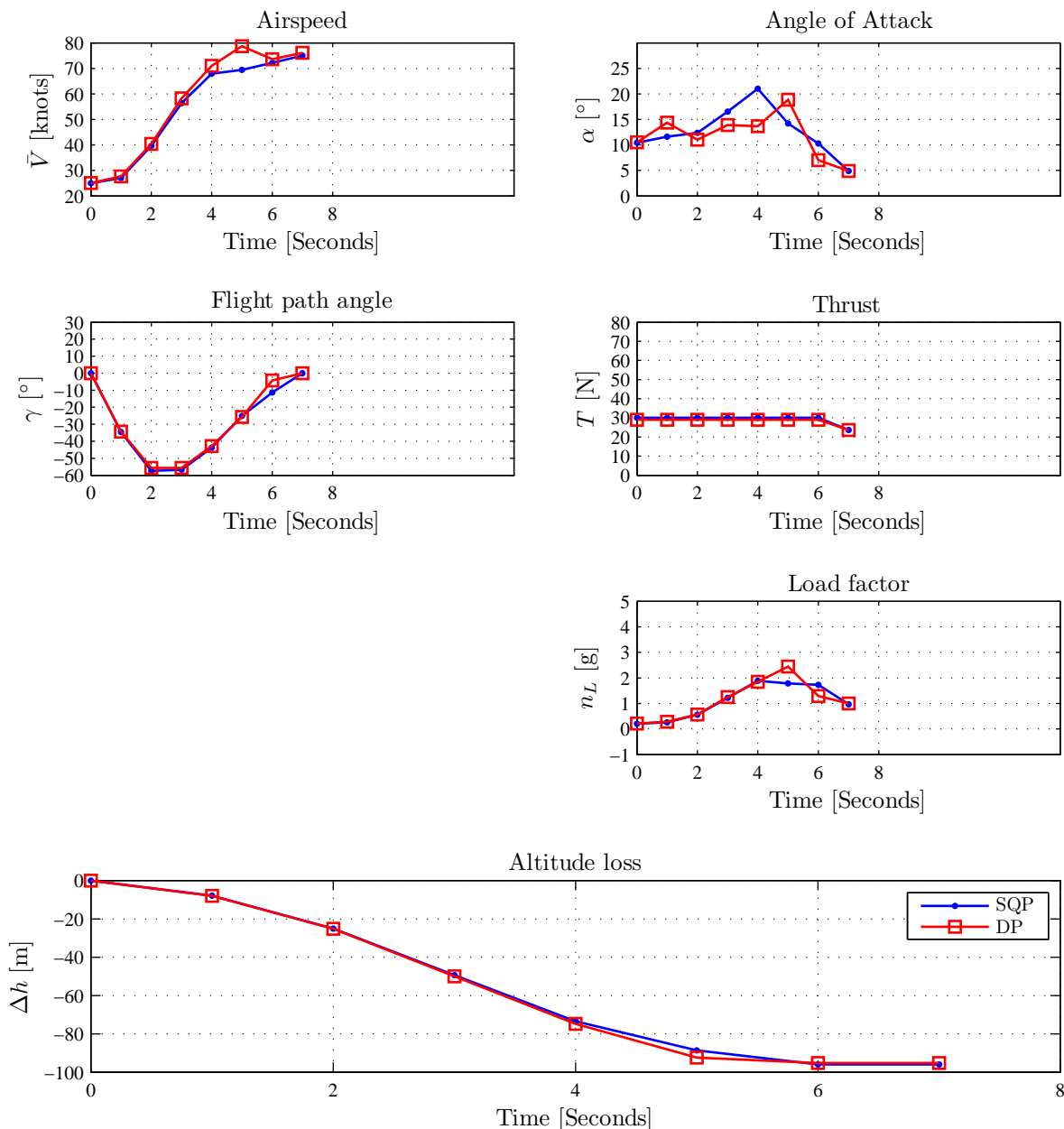


Figure B.13: SQP vs dynamic programming solution with a truncated time window and a constant thrust dynamic programming seed. The SQP's Thrust state has also been bounded to 30 N

### B.2.4 Three-Dimensional Thrust dynamic case

The three-dimensional thrust dynamic case used a three-dimensional thrust lag dynamic programming seed as well as the two dimensional constant thrust seed from the previous case. The three-dimensional seed was retrieved from the dynamic programming solution table with the forward interpolation search method, as the inputs were also quantised for this dynamic programming case. The initial condition for the three dimensional thrust lag case's trajectories were set to an underspeed condition as follows,



Table B.3: Initial state values and dynamic programming setup for three dimensional thrust lag case

Parameter	Symbol	Value	Units
<i>Initial condition</i>			
Velocity	$\bar{V}_{initial}$	25	kn
Flight path angle	$\gamma_{initial}$	0	degrees
Thrust	$T_{initial}$	30	Newton
<i>dynamic programming seed quantisation resolution</i>			
Velocity	$\Delta\bar{V}_q$	3.13	kn
Flight path angle	$\Delta\gamma_q$	3.75	degrees
Thrust	$\Delta T_q$	5.13	Newton
AoA	$\Delta\alpha_q$	1.62	degrees

The SQP was seeded with one case of the three-dimensional thrust lag dynamic programming seed and one case of the two-dimensional constant thrust dynamic programming seed. For a varying thrust seed from the dynamic programming, the SQP trajectory compared to the dynamic programming trajectory can be seen in Figure B.14. For a constant thrust seed from the dynamic programming, the SQP trajectory compared to the dynamic programming trajectory can be seen in Figure B.15.

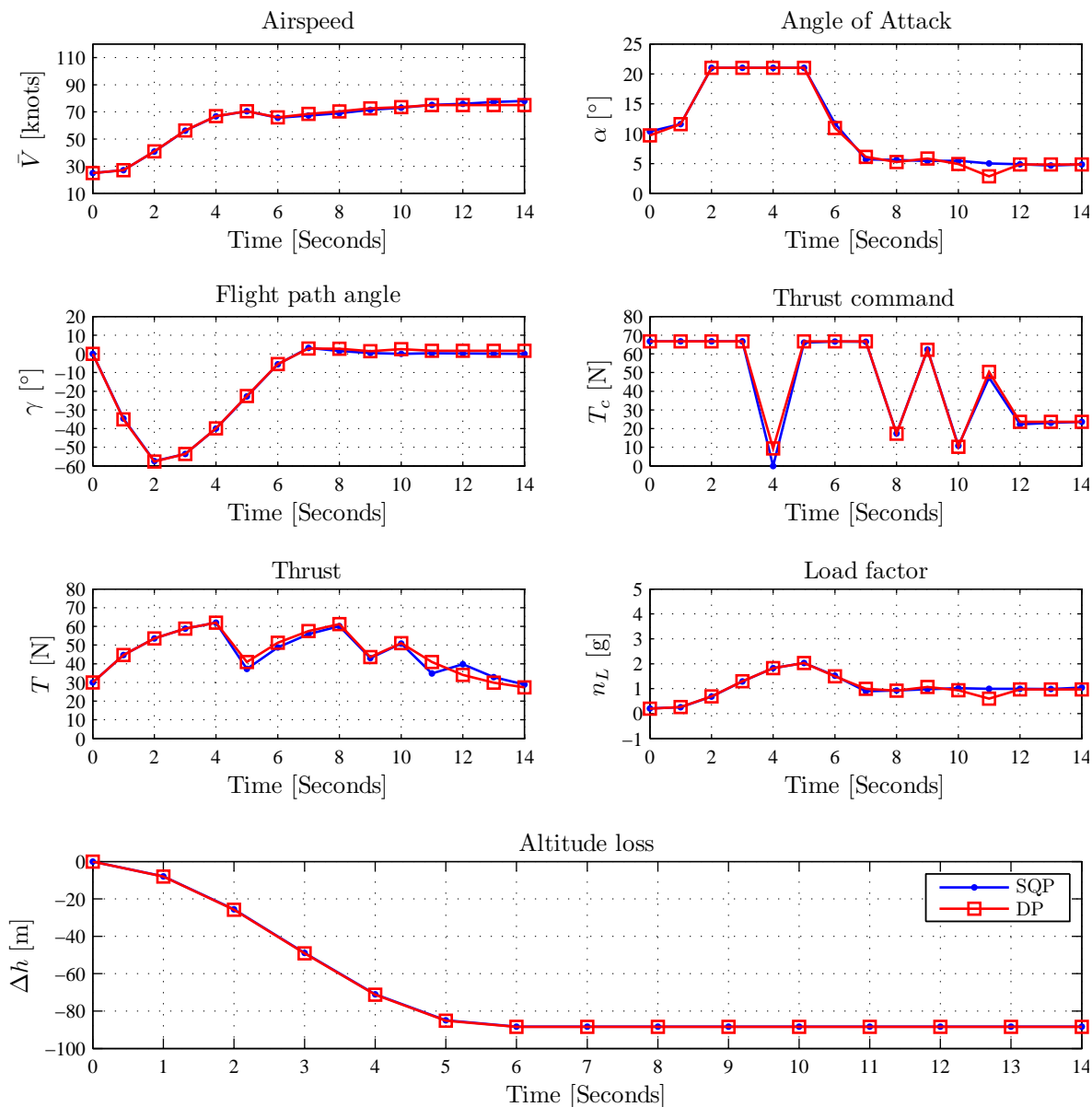


Figure B.14: SQP vs dynamic programming thrust lag solution with variable thrust dynamic programming seed

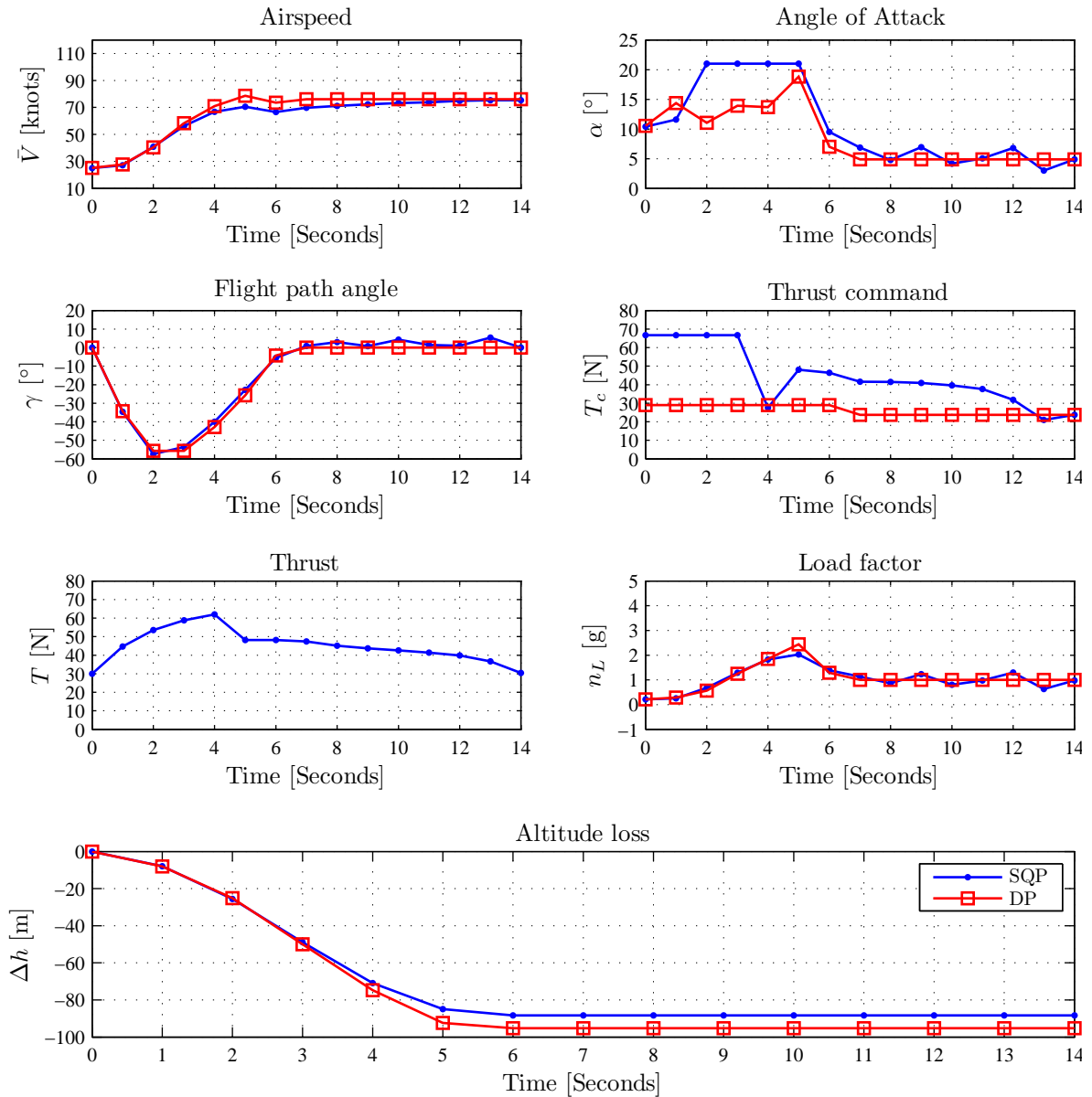


Figure B.15: SQP vs dynamic programming thrust lag solution with constant thrust dynamic programming seed

### B.2.5 Three-Dimensional bank angle case

The three-dimensional bank angle case used a three-dimensional bank angle dynamic programming seed. The seed was retrieved from the dynamic programming solution table with the forward grid search method. The initial condition for the three dimensional bank angle case's trajectories were set to an underspeed, inverted flight condition as follows,

Table B.4: Initial state values and dynamic programming setup for three dimensional bank angle case

Parameter	Symbol	Value	Units
<i>Initial condition</i>			
Velocity	$\bar{V}_{initial}$	45	kn
Flight path angle	$\gamma_{initial}$	0	degrees
Bank angle	$\Phi_{initial}$	130	degrees
<i>dynamic programming seed quantisation resolution</i>			
Velocity	$\Delta\bar{V}_q$	6.58	kn
Flight path angle	$\Delta\gamma_q$	4.29	degrees
Bank angle	$\Delta\Phi_q$	11.25	degrees

The SQP was seeded with a three-dimensional bank angle dynamic programming seed and the trajectory result comparing the DP solution and the SQP solution is shown in Figure B.16.

It is interesting to note that the dynamic programming solution table had no feasible solution for underspeed conditions below approximately 45 knots (at the time when these results were initially generated) in airspeed at an initial bank angle of 130 degrees and a level flight path angle, but the SQP routine could find a feasible solution at lower speeds than 45 knots, with no dynamic programming seed. However, refining the dynamic programming's quantisation would most likely solve this issue. Figure B.17 is a trajectory solution with an initial speed of 40 knots instead of 45 knots.

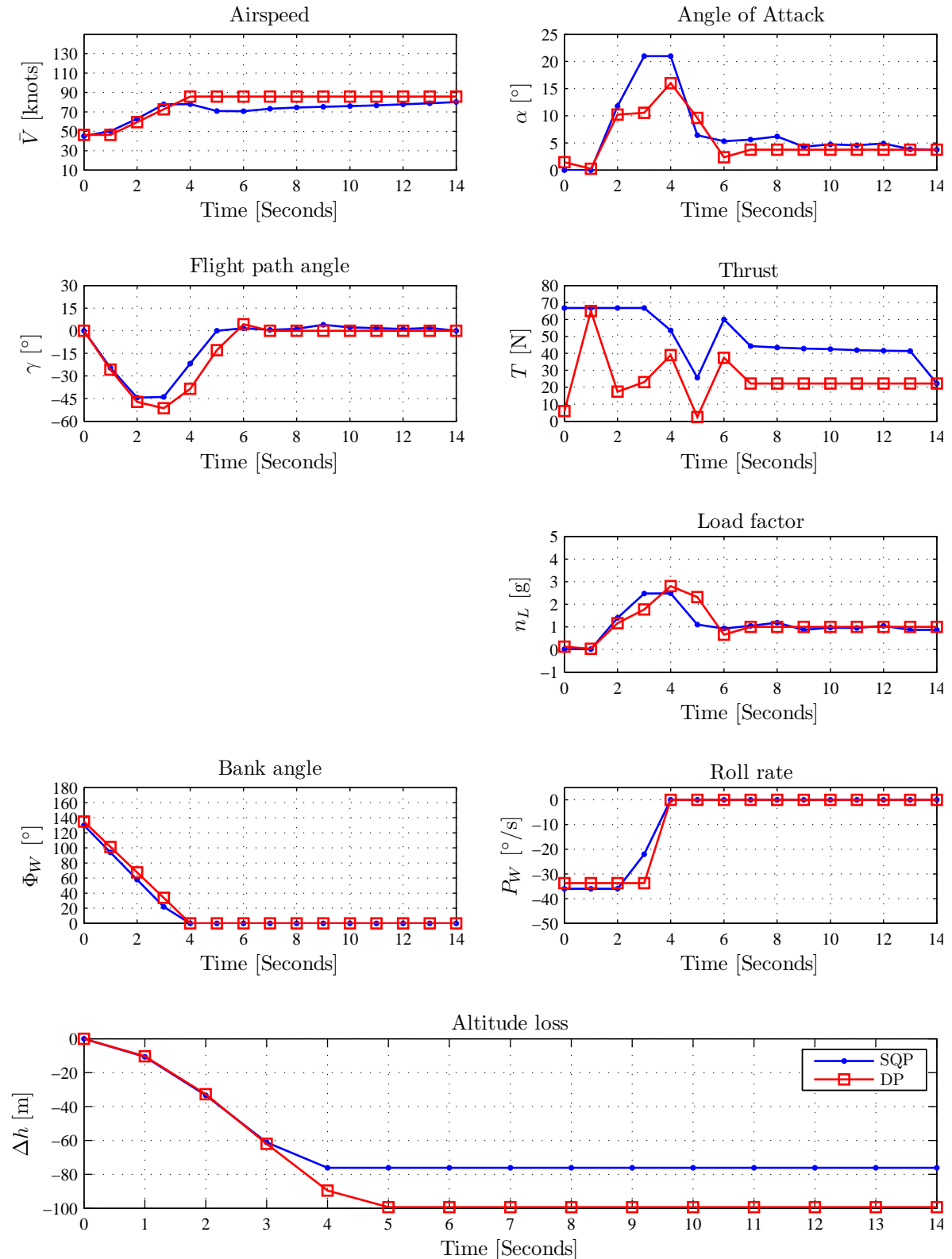


Figure B.16: SQP vs dynamic programming underspeed and bank angle recovery solution with variable thrust dynamic programming seed

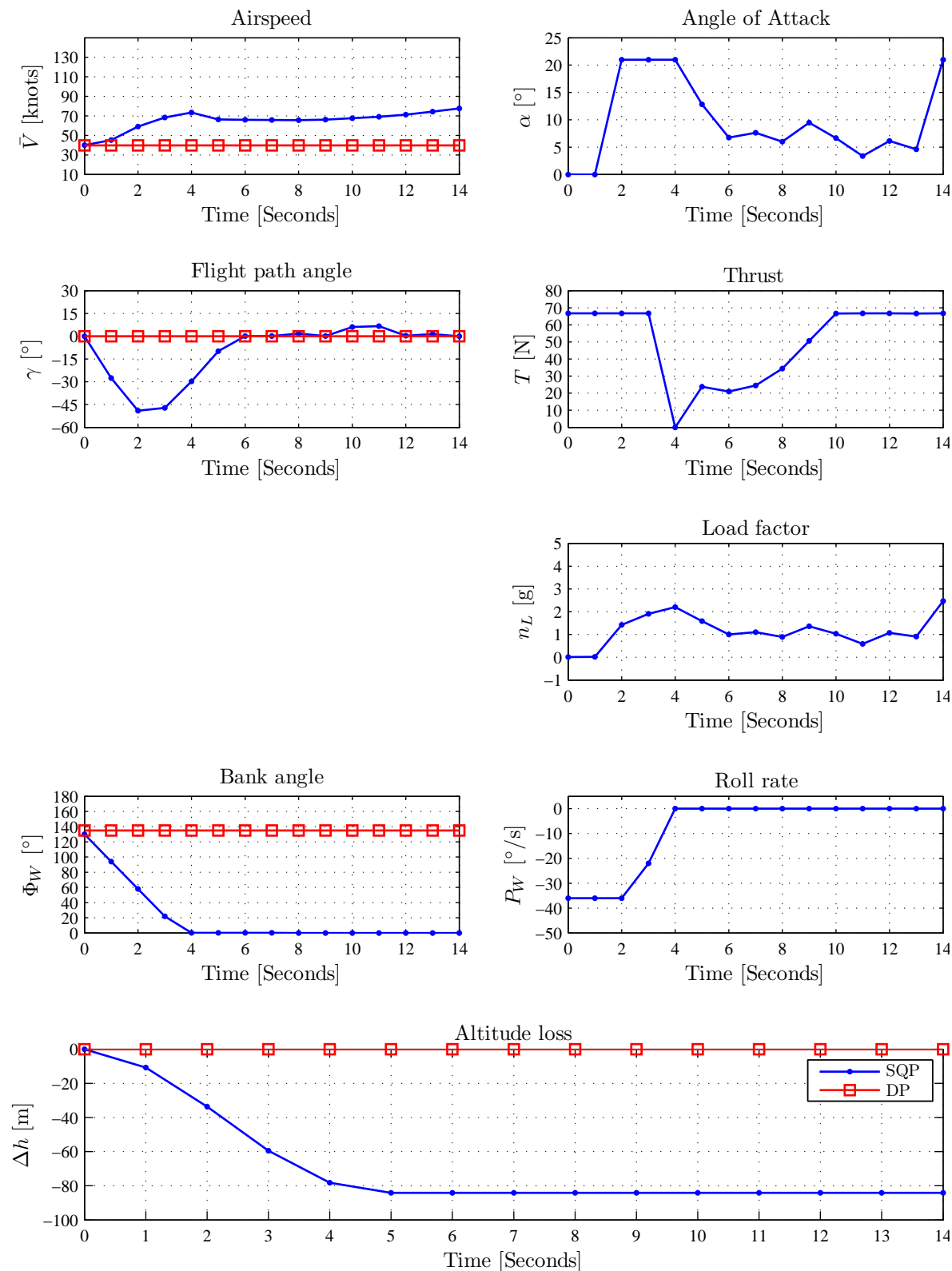


Figure B.17: SQP underspeed and bank angle recovery solution with no solution for dynamic programming seed

## B.2.6 Four-Dimensional bank angle with thrust lag case

### Constant Time Window

The initial condition for the four dimensional case's trajectories were set to an underspeed condition as follows,

Table B.5: Initial state values and dynamic programming setup for four dimensional case

Parameter	Symbol	Value	Units
<i>Initial condition</i>			
Velocity	$\bar{V}_{initial}$	45	kn
Flight path angle	$\gamma_{initial}$	0	degrees
Bank angle	$\Phi_{initial}$	130	degrees
Thrust	$T_{initial}$	30	Newton
<i>dynamic programming seed quantisation resolution</i>			
Velocity	$\Delta\bar{V}_q$	6.58	kn
Flight path angle	$\Delta\gamma_q$	4.29	degrees
Bank angle	$\Delta\Phi_q$	11.25	degrees

The SQP algorithm was seeded with a DP solution from the initial upset condition in Table B.5, and the trajectory result of the SQP and the DP method from this initial upset condition is shown in Figure B.18.

### Variable Time Window

A variable time window case was also tested. By adding  $t_N$  as a design variable for the NLP routine, the sample time  $\Delta t$  is allowed to vary, but the number of grid points is kept the same with the value of  $N$  not changing. This case's time variables were set to the values in Table B.6.

Table B.6: Dynamic programming setup for four dimensional time varying case

Parameter	Symbol	Value	Units
Time window bounds	$t_{KU}, t_{KL}$	10, 0	seconds
Number of grid points	$K + 1$	15	-
Number of dynamic programming grid points	$N_{DP} + 1$	15	-

The same initial condition of Table B.5 was used for this case and the resulting trajectory can be seen in Figure B.19. The dynamic programming seed remains the same with a fixed time window of 14 seconds.

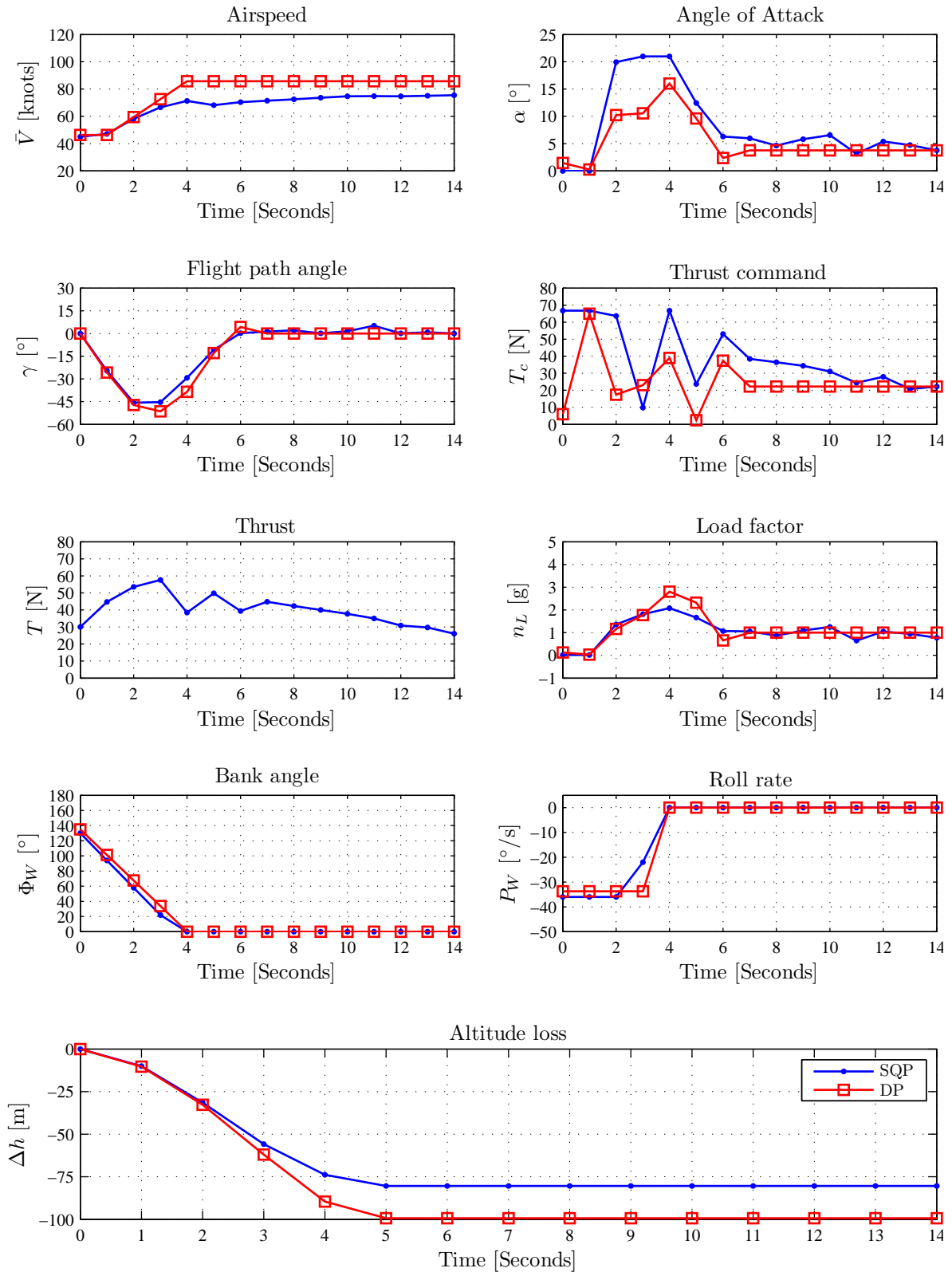


Figure B.18: SQP vs dynamic programming four dimensional solution with variable thrust dynamic programming seed



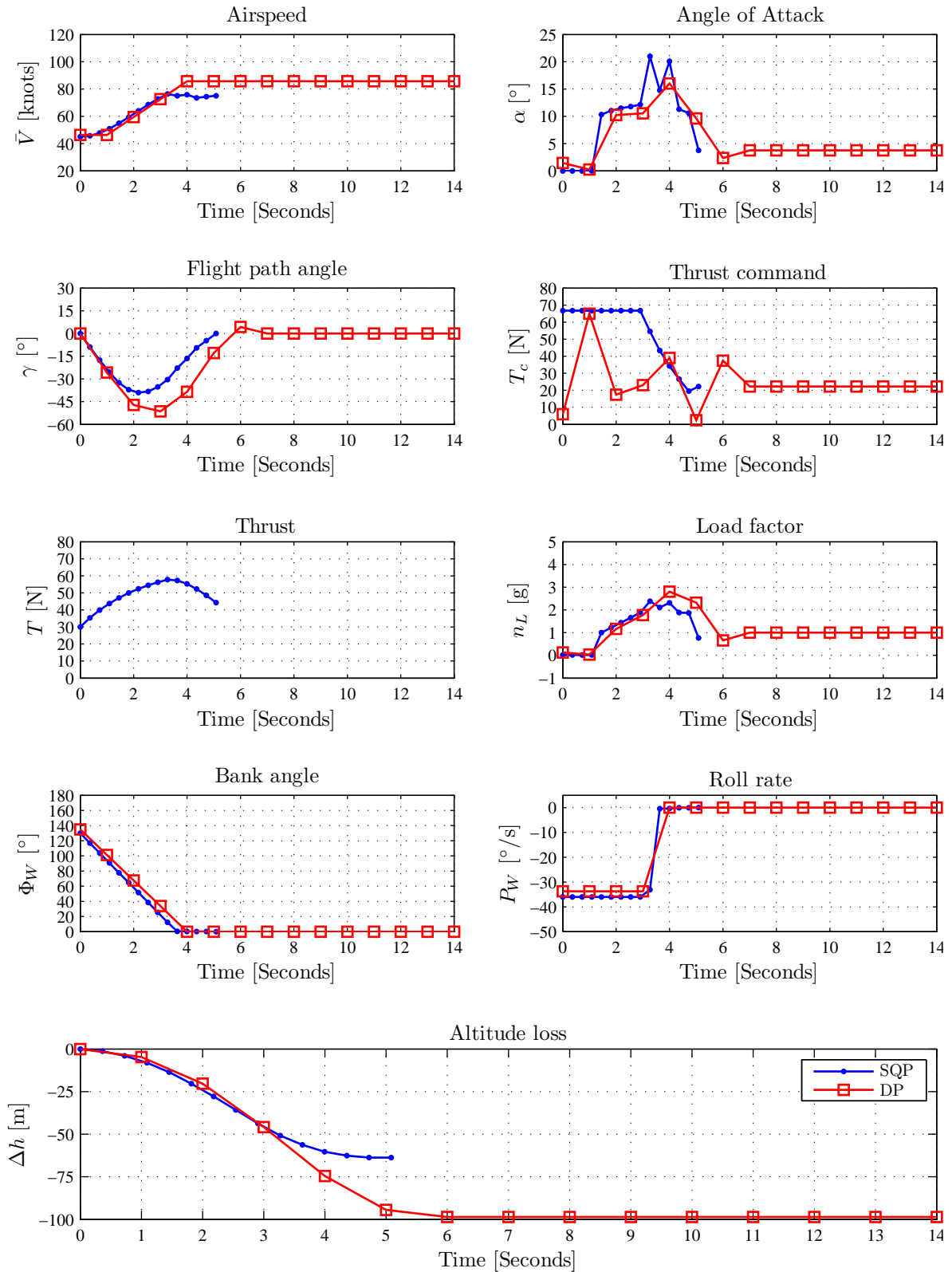


Figure B.19: SQP vs dynamic programming four dimensional time varying solution with constant thrust dynamic programming seed

### B.2.7 Results summary

It is interesting to note that the trajectory solutions generated by both the dynamic programming and SQP routines for bank angle recovery show that almost no AoA should be given while the aircraft is at a bank angle of greater than 90 degrees. As soon as the aircraft has recovered to a bank angle smaller than 90 degrees, AoA commands should be given to recover the flight path angle. The lower the flight path angle, the greater the AoA command should be to recover the flight path angle.

## Appendix C

# DQ Elevator Feed-Forward Closed-Loop Constant

This appendix shows the derivation of the constant term  $N$  used in the calculation of the DQ controller's closed-loop state space model equation in Chapter 3 section §3.2.1 Equation 3.2.20. The derivation is as follows:

For the full-order closed-loop model of the DQ controller the normal acceleration state consists of a state output matrix  $\mathbf{C}_{DQ}$  and an elevator control feed-forward term  $D_{a_z}$ ,

$$a_z = \mathbf{C}_{a_z} \mathbf{x}_{long} + D_{a_z} \delta_e \quad (\text{C.1})$$

The full-order DQ law is then,

$$\delta_e = -\mathbf{K}_{DQ} \mathbf{x}_{DQ} - K_I x_{I_{DQ}} + K_{FF} a_{z_c} \quad (\text{C.2})$$

where

$$\mathbf{x}_{DQ} = \begin{bmatrix} \bar{v} \\ a_z \\ q \\ \theta \end{bmatrix} = \mathbf{C}_{DQ} \mathbf{x}_{long} + \mathbf{D}_{DQ} \delta_e \quad (\text{C.3})$$

Substituting Equation C.3 into Equation C.2 and gathering terms reduces to the elevator command to,

$$\delta_e [1 + \mathbf{K}_{DQ} \mathbf{D}_{DQ}] = -\mathbf{K}_{DQ} \mathbf{C}_{DQ} \mathbf{x}_{long} - K_I x_{I_{DQ}} + K_{FF} a_{z_c} \quad (\text{C.4})$$

$$\delta_e = -N \mathbf{K}_{DQ} \mathbf{C}_{DQ} \mathbf{x}_{long} - N K_I x_{I_{DQ}} + N K_{FF} a_{z_c} \quad (\text{C.5})$$

where  $N$  is the derived control-loop constant that is then used in the rest of the normal closed-loop equations,

$$N = \frac{1}{1 + \mathbf{K}_{DQ} \mathbf{D}_{DQ}} \quad (\text{C.6})$$

## Appendix D

# First Iteration of Trajectory Regulation Control

This appendix provides trajectory execution simulation results of the initial iteration of trajectory execution control schemes, and discusses some observations regarding the results. The appendix then also provides trajectory execution simulation results that compare the baseline angle of attack inner-loop controller provided with the NASA GTM to the designed normal acceleration (DQ) inner-loop controller.

### D.1 Upset Command Tracking

The GTM is initialised for a given upset condition listed in Table D.1 and a combination of the state and input trajectory results of the SQP algorithm is given to the inner and/or middle-loop controllers in different configurations to test the controller performance and also validate the SQP results.

The angle of attack input delay model for these results used a first-order lag model with a time constant of  $\tau_\alpha = 1.5$  seconds, which is not the true time constant of the angle of attack dynamics (which is closer to 0.2 seconds). For this initial implementation an artificially slow angle of attack time constant is used so that we can still assume time scale separation in the rotational dynamics. The roll rate dynamics are also omitted, i.e. the roll rate changes instantly and the roll rate state itself is an input command. This method was not used in the final implementation as we would like to use the SQP's advantage to model the true dynamics of the inner-loop controllers to obtain a more representative optimal solution.

Table D.1: Initial upset condition

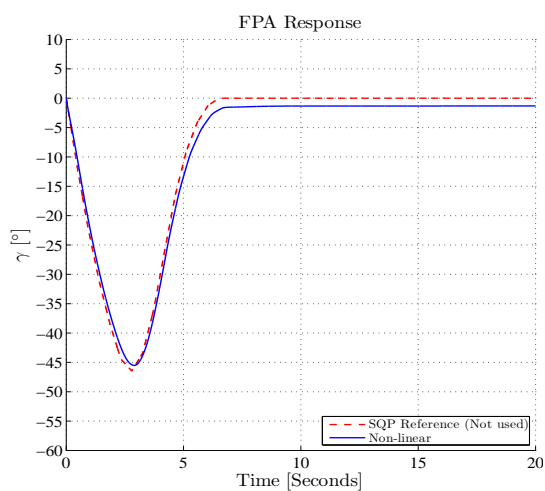
State	Value	Units
$\bar{V}_{initial}$	60	kn
$\gamma_{initial}$	0	degrees
$\alpha_{initial}$	3	degrees
$\beta_{initial}$	0	degrees
$\Phi_{initial}$	160	degrees
$T_{initial}$	25.2	Newton

### D.1.1 Input Reference Only

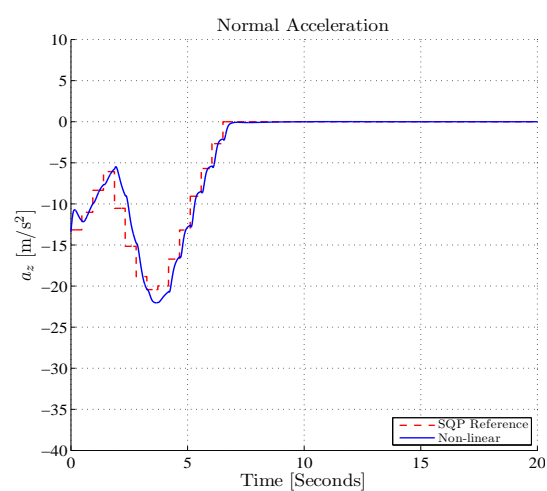
Only the planned input command signals are given to the inner-loop controllers as references and can be seen as giving open-loop commands to the system. It is expected that there will be some steady-state error in the state tracking compared to the predicted state trajectories. The SQP planning algorithm uses angle of attack as an input, whereas the conventional flight controllers use normal acceleration for longitudinal manoeuvring. The angle of attack state trajectory from the SQP algorithm has to be translated into normal acceleration commands to be used with the flight controllers. The following equation is used to convert the angle of attack commands into normal acceleration commands,

$$a_z^* = \frac{1}{m} \left[ \frac{1}{2} \rho \bar{V}^{*2} S C_Z(\alpha^*) + mg \cos(\gamma^* + \alpha^*) \cos \Phi^* \right] \quad (\text{D.1})$$

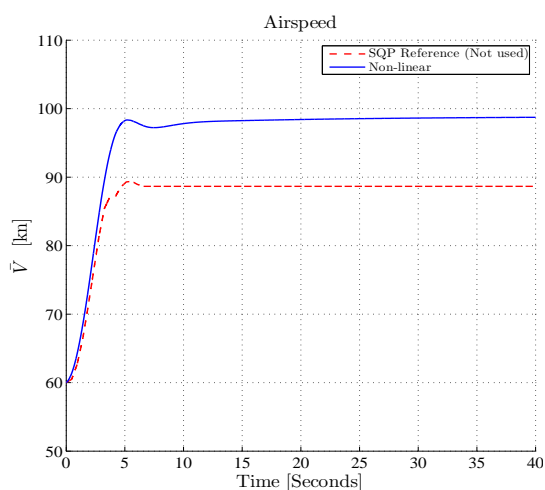
where the notation  $\mathbf{x}^*$ ,  $\mathbf{u}^*$  indicates the state and input trajectory values from the optimisation algorithm. Figures D.1 and D.2 show the GTM state and input response to the open-loop input commands. The overshoot in the speed response in Figure D.1(c) can be attributed to the lower angle of attack (Figure D.1(d)) used during the manoeuvre than expected. The system tracked the translated normal acceleration commands very well (Figure D.1(b)), but because the normal acceleration is insensitive to speed and angle of attack, the normal acceleration could achieve the desired normal acceleration trajectory and not produce the equivalent planned angle of attack. The lower angle of attack produces less drag, causing the speed overshoot.



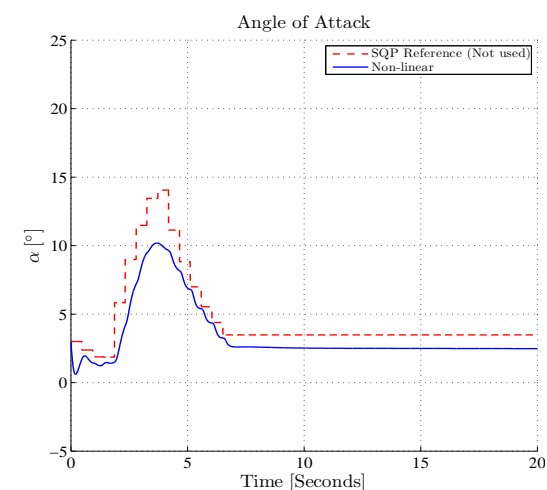
(a) Non-linear flight path angle response



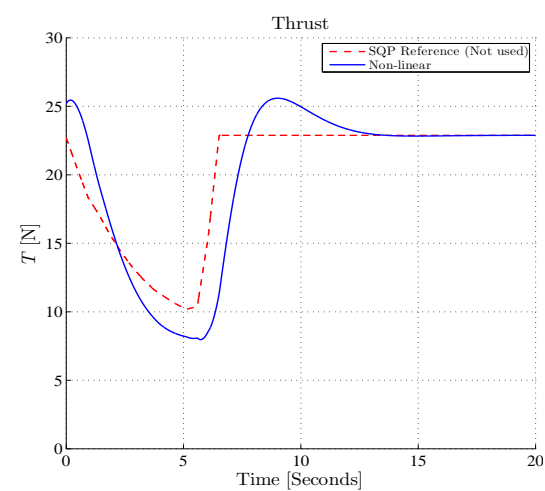
(b) Non-linear normal acceleration response



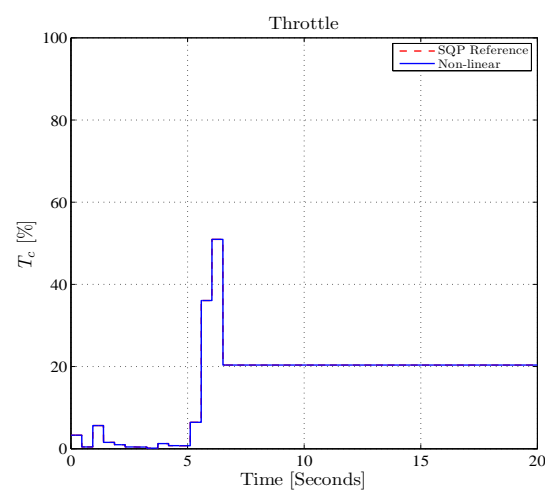
(c) Non-linear airspeed response



(d) Non-linear angle of attack response

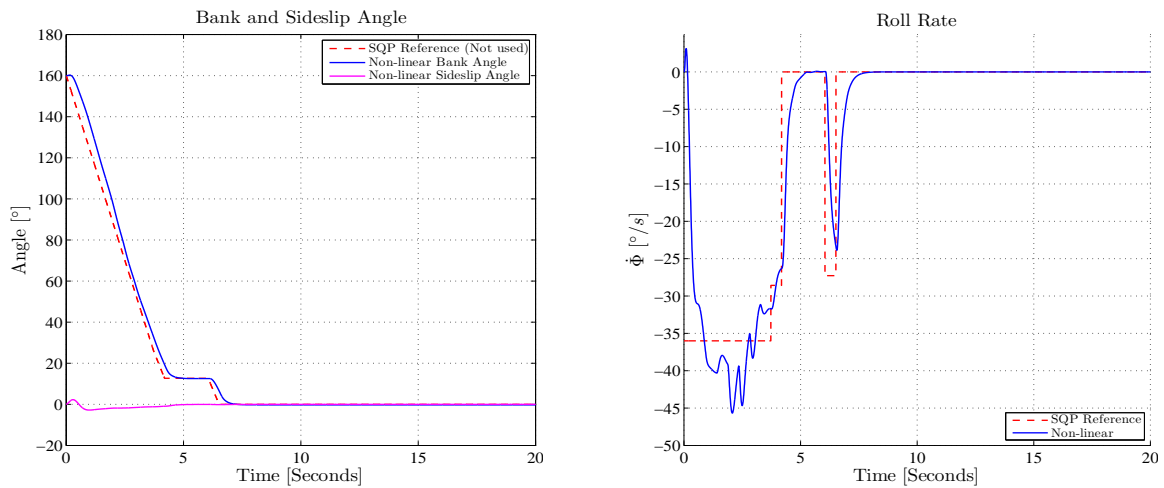


(e) Non-linear thrust response



(f) Non-linear throttle input

Figure D.1: Closed-loop non-linear response to SQP input commands from upset condition.



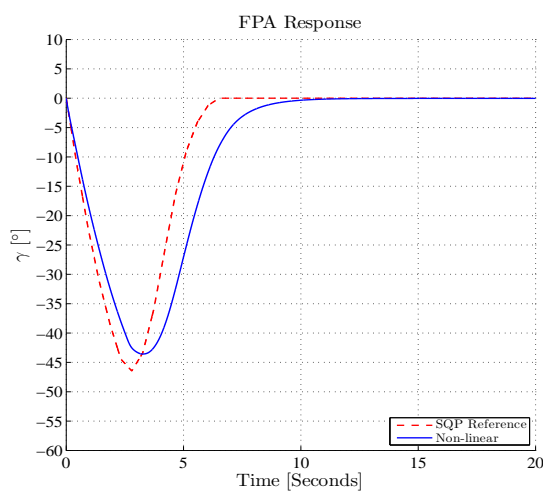
(a) Non-linear bank angle response

(b) Non-linear roll rate response

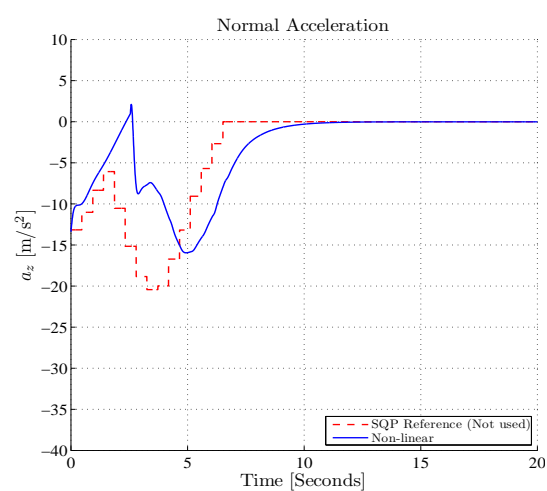
Figure D.2: Closed-loop non-linear response to SQP input commands from upset condition continued.

### D.1.2 State Reference Only with AT off

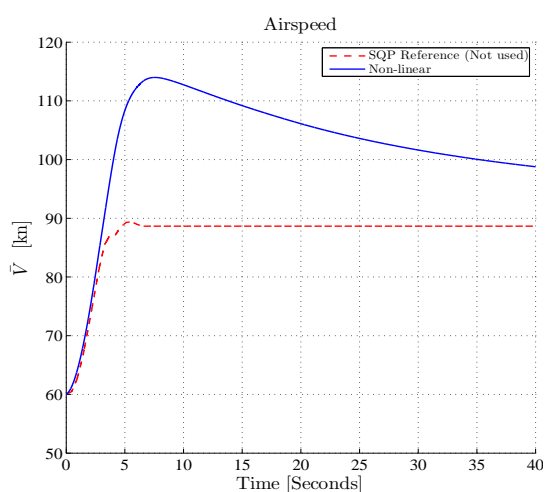
Only the state trajectories are given to the middle-loop controllers as reference commands in a closed-loop configuration. It is expected that the executed trajectories will lag the planned trajectories more than the open-loop input configuration, but there will be minimal steady-state error. With the autothrust off, we expect a steady-state error in the executed airspeed trajectory.



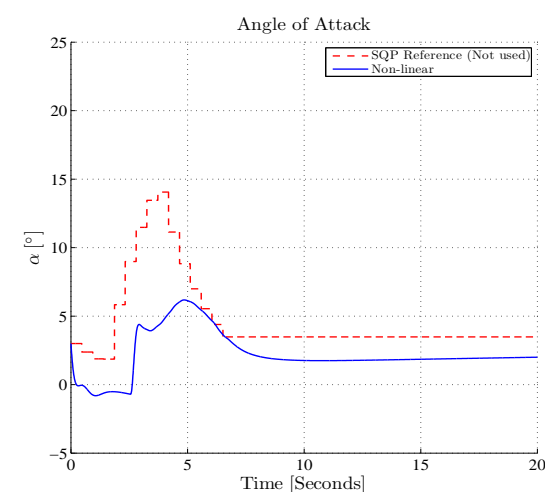
(a) Non-linear flight path angle response



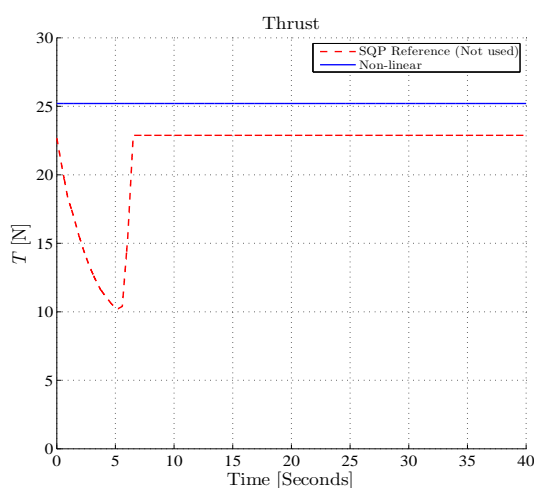
(b) Non-linear normal acceleration response



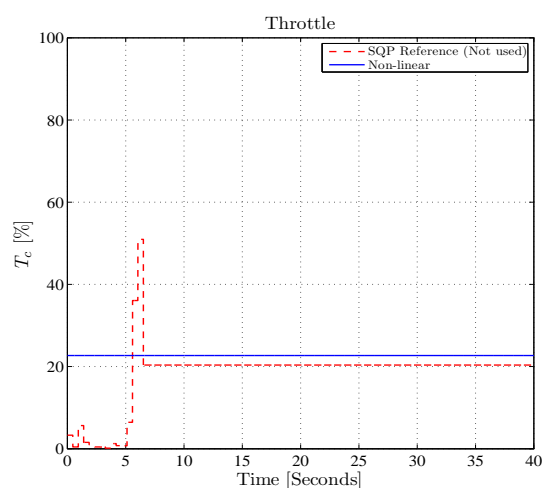
(c) Non-linear airspeed response



(d) Non-linear angle of attack response



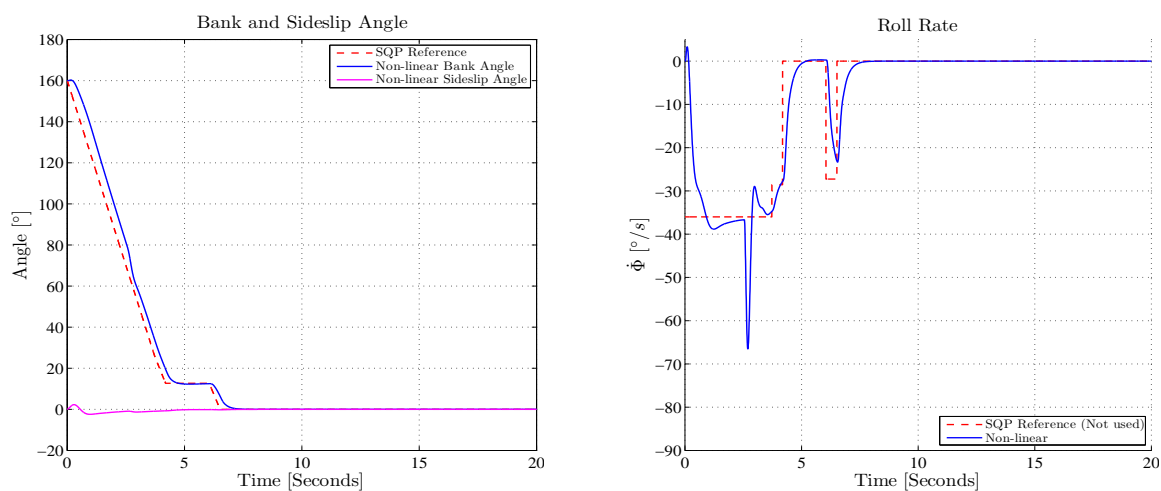
(e) Non-linear thrust response



(f) Non-linear throttle input

Figure D.3: Closed-loop non-linear response to SQP state reference only commands from upset condition with AT off.





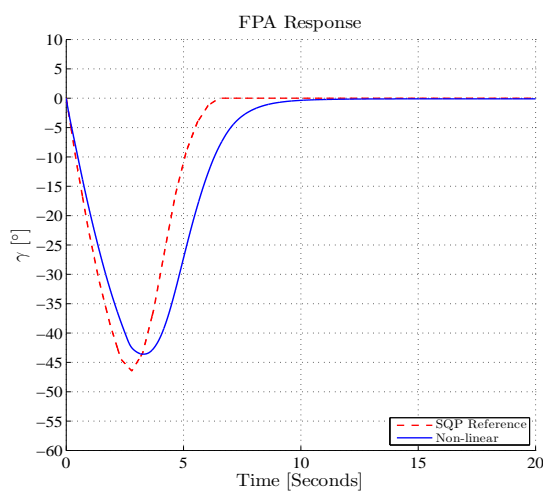
(a) Non-linear bank angle response

(b) Non-linear roll rate response

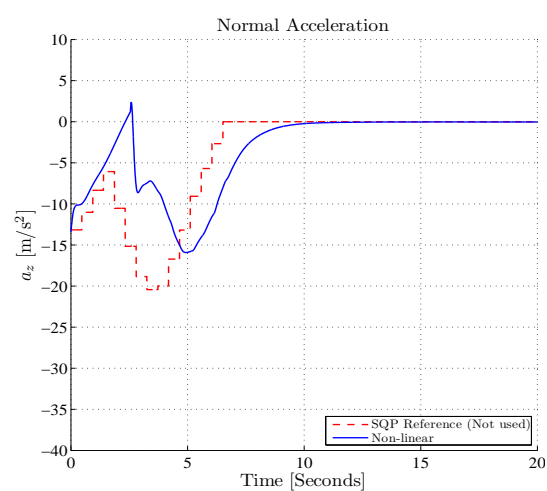
Figure D.4: Closed-loop non-linear response to SQP state reference only commands from upset condition with AT off continued.

### D.1.3 State Reference Only with AT on

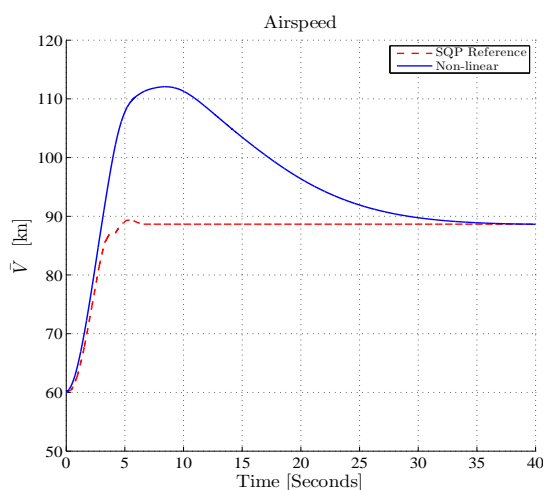
This configuration is similar to the previous one, but with the autothrust on instead. Thus we expect a zero steady-state error in the executed airspeed trajectory.



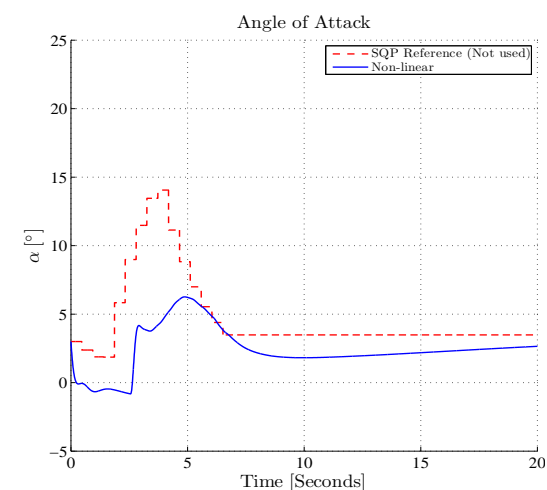
(a) Non-linear flight path angle response



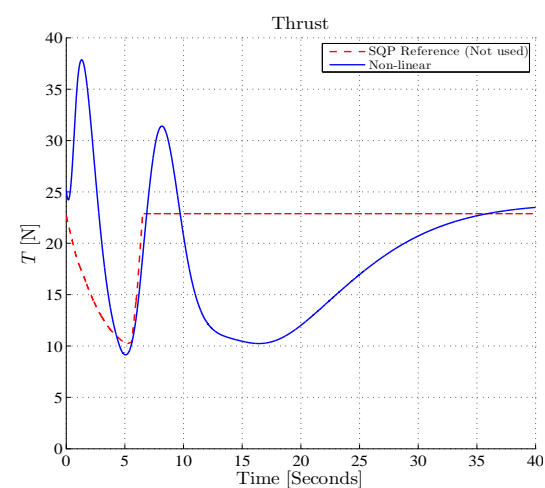
(b) Non-linear normal acceleration response



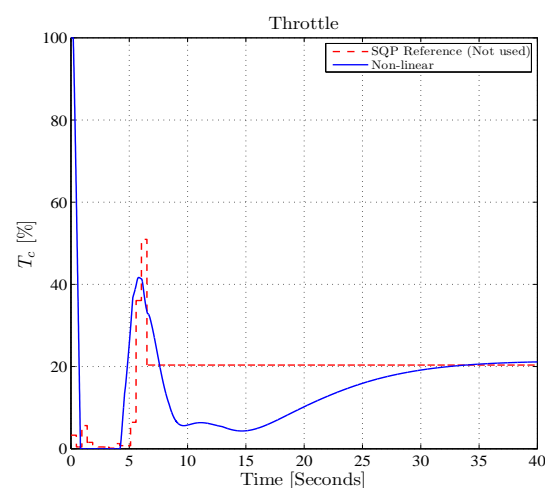
(c) Non-linear airspeed response



(d) Non-linear angle of attack response



(e) Non-linear thrust response



(f) Non-linear throttle input

Figure D.5: Closed-loop non-linear response to SQP state reference only commands from upset condition with AT on.

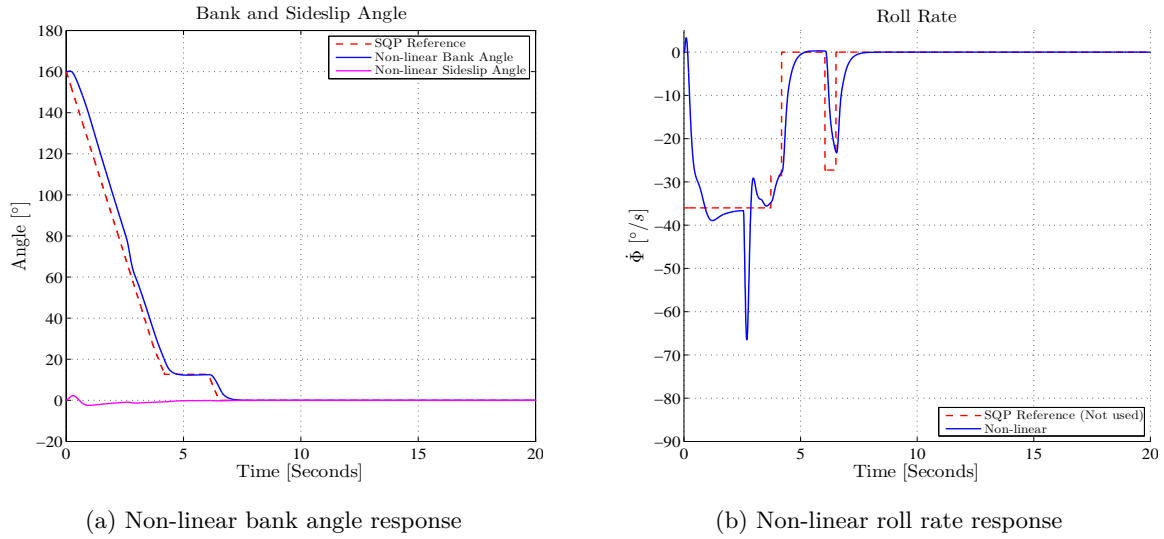


Figure D.6: Closed-loop non-linear response to SQP state reference only commands from upset condition with AT on continued.

#### D.1.4 State and Input Combined Reference

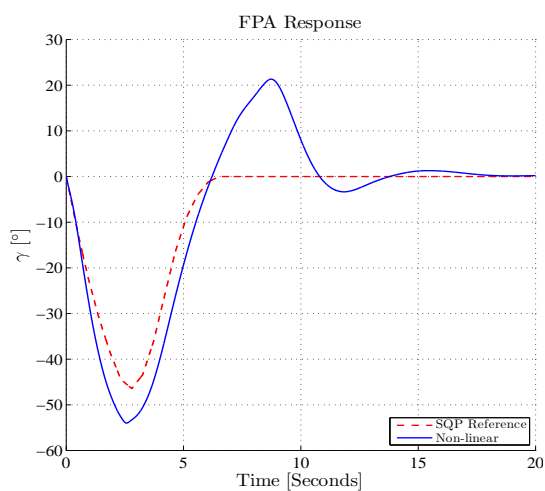
The planned state trajectories and input signals from the SQP planning algorithm are given in combination to the middle and inner-loop controllers so that the executed trajectories matches as close as possible to that of the planned trajectories with minimal delay, while still ensuring stability by giving the state trajectory references to the middle-loop controllers. The planned input signals from the SQP are superimposed with the commands of the middle-loop controllers, and the total commands are given as reference commands to the inner-loop controllers. Note that the airspeed controller was used as an active middle-loop controller during the recovery manoeuvre.

When using the roll angle  $\Phi$  in the flight path angle control law Equation 3.2.25, it was found that superimposing the commands from the middle-loop flight path angle controller with the open-loop normal acceleration commands from the SQP on one another, caused unexpected responses. Several cases were investigated.

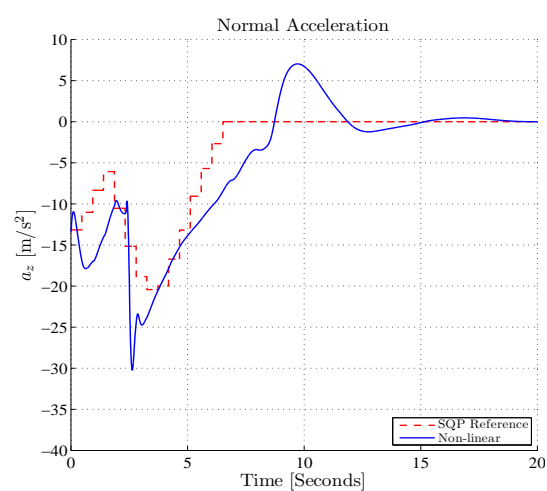
**Case 1** - Case 1 uses the normal flight path angle control law with the 90 degree bank angle handled by setting the normal specific acceleration to zero for bank angles greater than some threshold angle  $\Phi_{lim}$  degrees,

$$a_{z_c} = \begin{cases} \frac{-\bar{V}\dot{\gamma} - a_{\gamma_{gravity}}}{\cos \Phi} + a_{z_{gravity}} + a_z^*, & \text{if } \Phi \leq \Phi_{lim} \\ 0 + a_{z_{gravity}} + a_z^*, & \text{if } \Phi > \Phi_{lim} \end{cases} \quad (D.2)$$

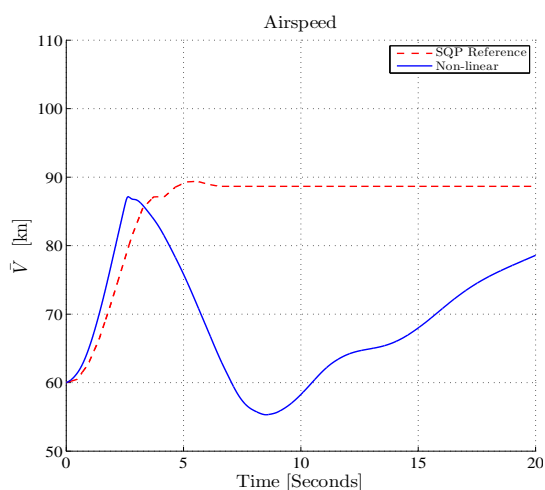
Figures D.7 and D.8 shows the GTM system response from an initial upset condition with  $\Phi_{lim} = 80^\circ$ . There is a large spike in the normal acceleration command at the discontinuity at  $\Phi = 80^\circ$ , resulting in a large sudden increase in angle of attack that causes a large decrease in airspeed. Figures D.9 and D.10 shows the GTM system response from an initial upset condition with  $\Phi_{lim} = 40^\circ$ . The response profile is similar to the previous, but less severe, as the discontinuity in the control occurs at a smaller roll angle, causing a smaller control effort step at  $\Phi = 40^\circ$ . Figures D.11 and D.12 shows the GTM system response from an initial upset condition with  $\Phi_{lim} = 180^\circ$ . With this limit, there is no discontinuity in the control law, but there is a singularity at  $\Phi = 90^\circ$  causing large spikes in angle of attack which only coincidentally results in the close tracking of the speed reference.



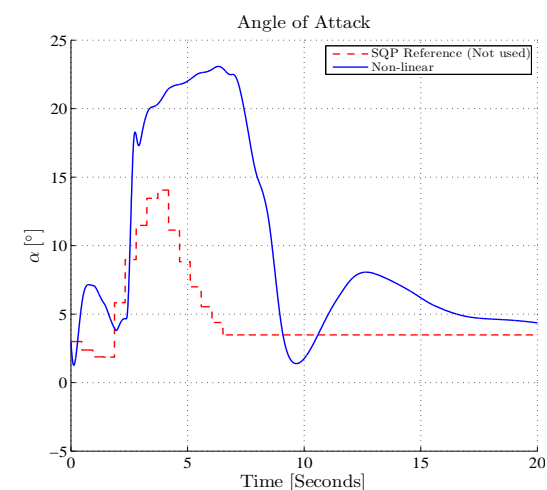
(a) Non-linear flight path angle response



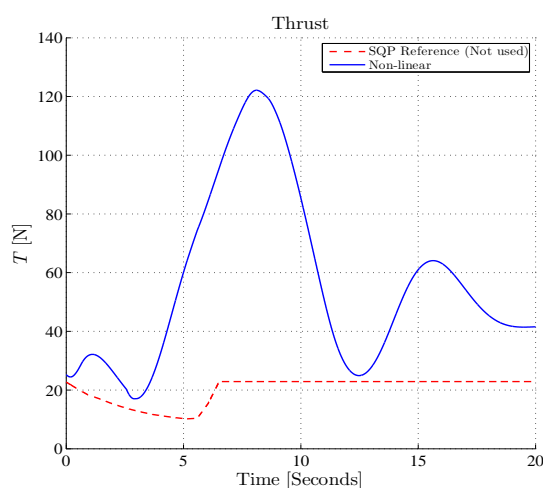
(b) Non-linear normal acceleration response



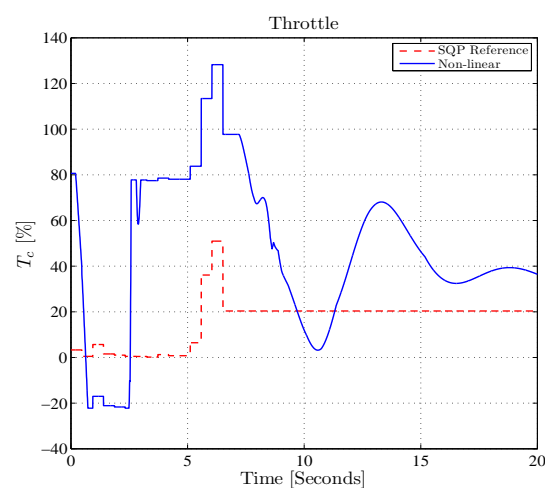
(c) Non-linear airspeed response



(d) Non-linear angle of attack response



(e) Non-linear thrust response



(f) Non-linear throttle input

Figure D.7: Closed-loop non-linear response for Case 1 combined SQP state and input commands with  $\Phi_{lim} = 80^\circ$ .

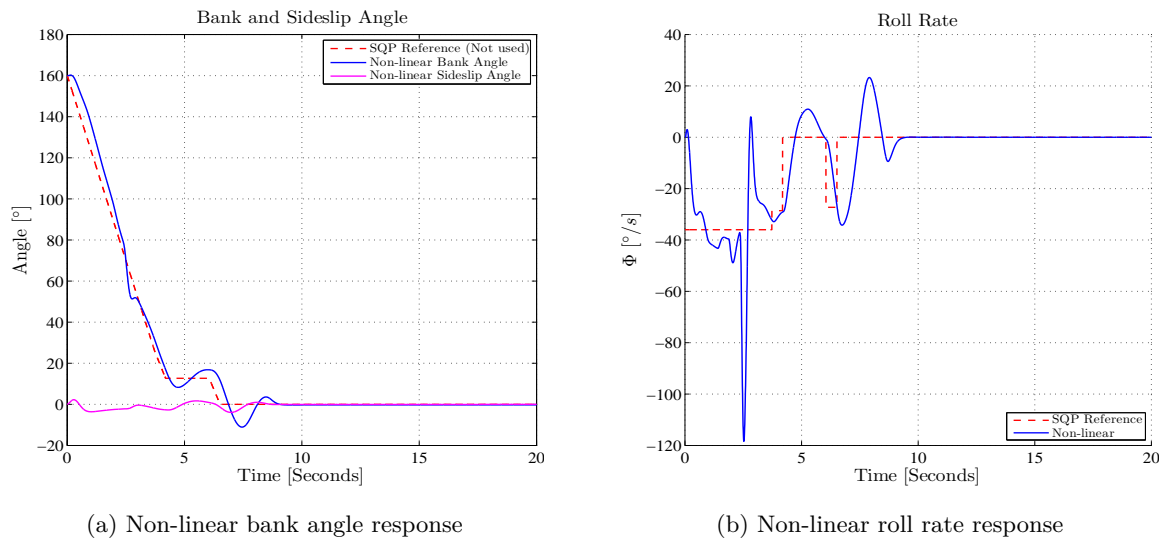
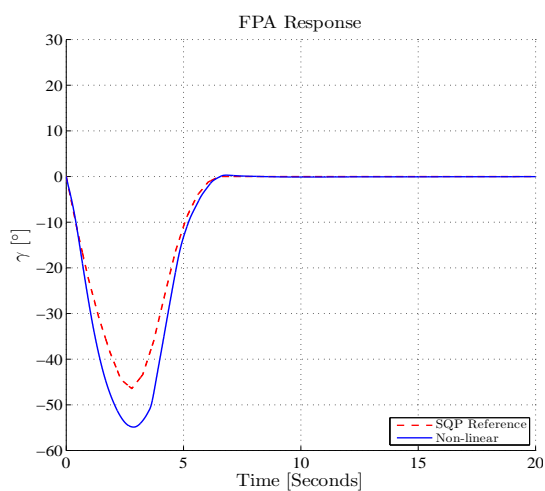
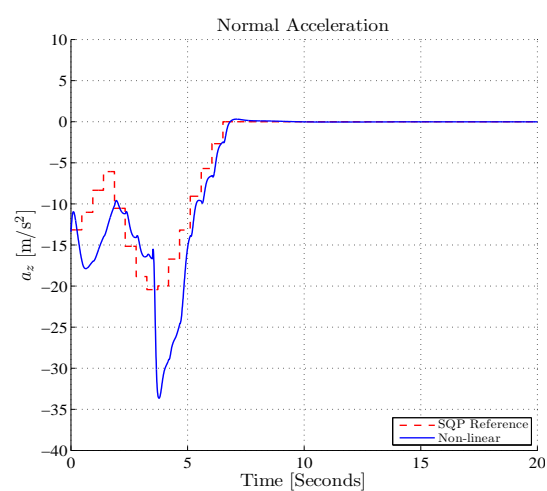


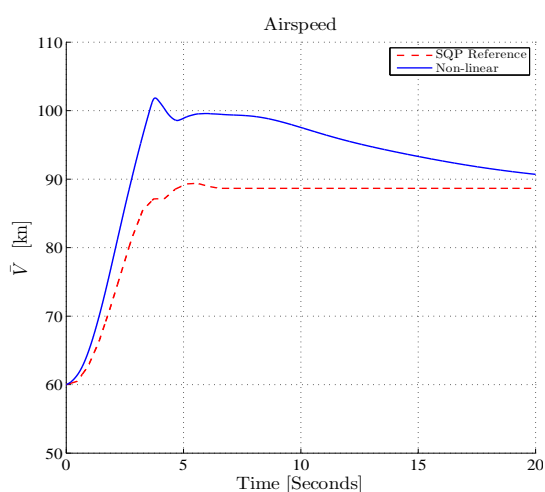
Figure D.8: Closed-loop non-linear response for Case 1 combined SQP state and input commands with  $\Phi_{lim} = 80^\circ$  continued.



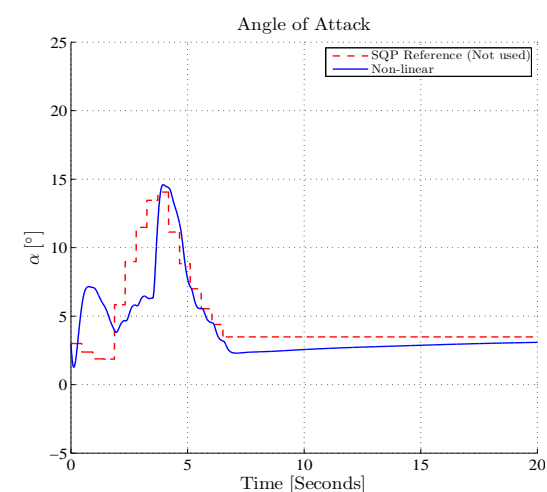
(a) Non-linear flight path angle response



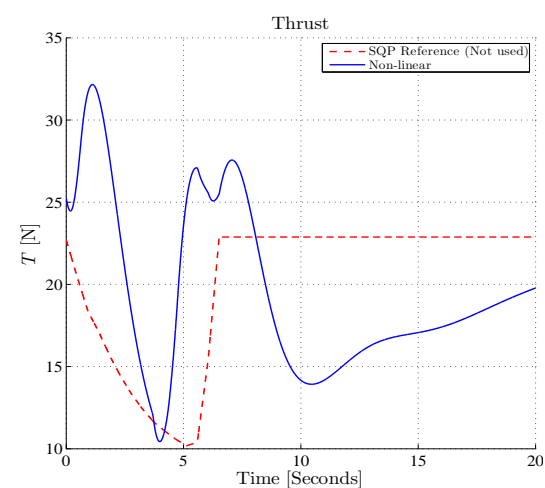
(b) Non-linear normal acceleration response



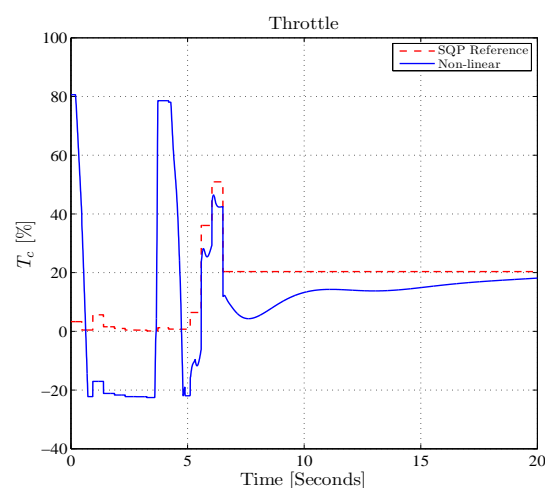
(c) Non-linear airspeed response



(d) Non-linear angle of attack response



(e) Non-linear thrust response



(f) Non-linear throttle input

Figure D.9: Closed-loop non-linear response for Case 1 combined SQP state and input commands with  $\Phi_{lim} = 40^\circ$ .

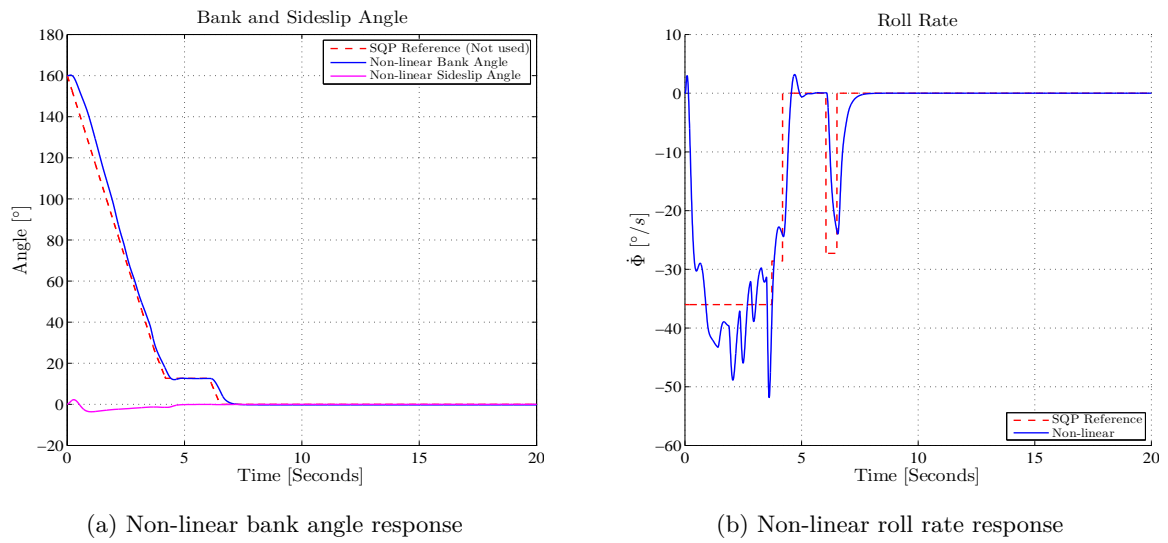
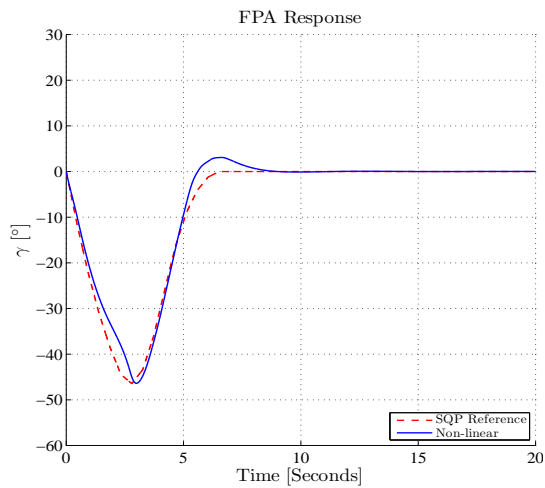
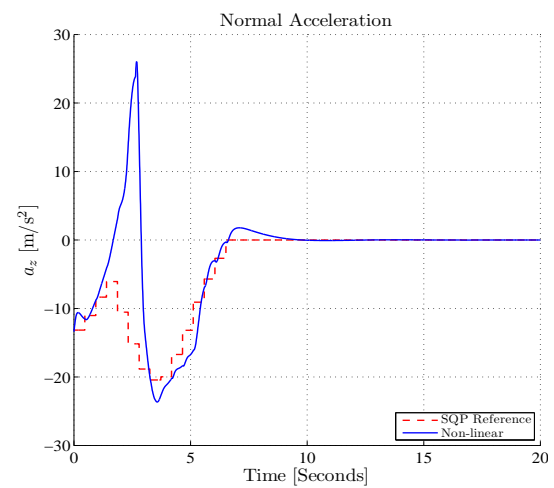


Figure D.10: Closed-loop non-linear response for Case 1 combined SQP state and input commands with  $\Phi_{lim} = 40^\circ$  continued.

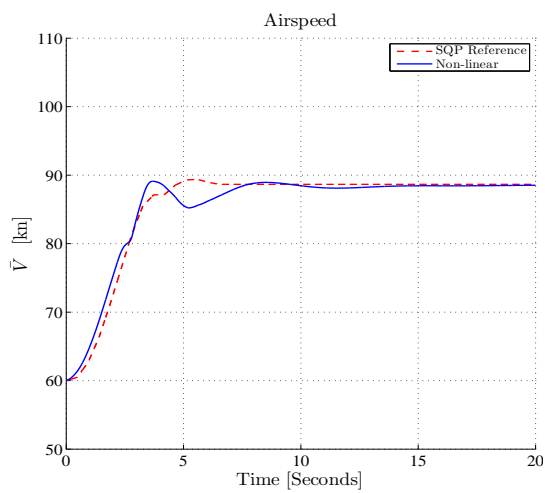




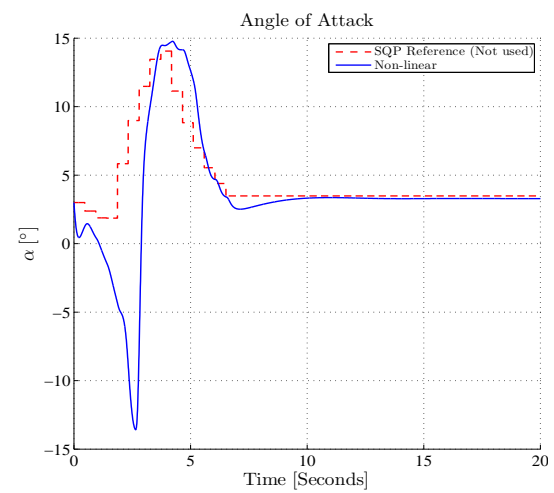
(a) Non-linear flight path angle response



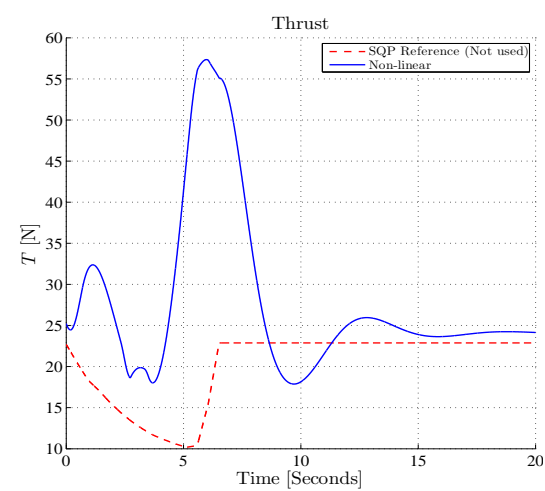
(b) Non-linear normal acceleration response



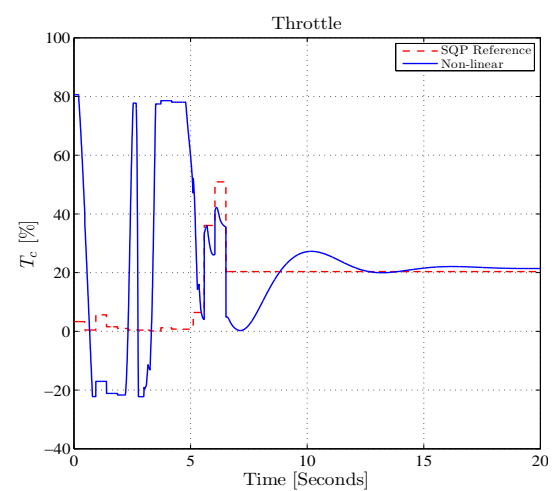
(c) Non-linear airspeed response



(d) Non-linear angle of attack response



(e) Non-linear thrust response



(f) Non-linear throttle input

Figure D.11: Closed-loop non-linear response for Case 1 combined SQP state and input commands with  $\Phi_{lim} = 180^\circ$ .

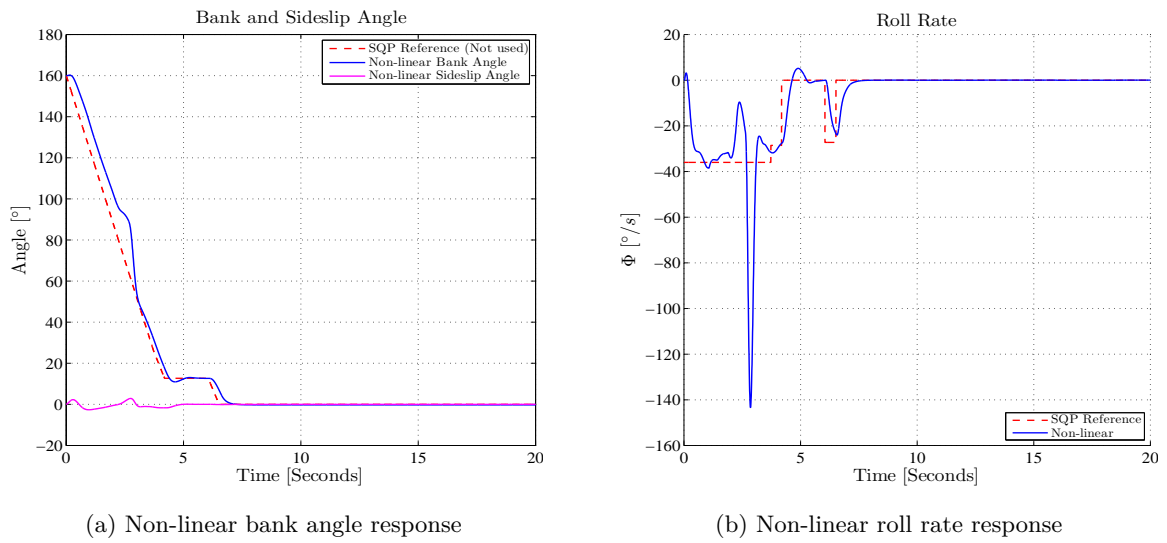
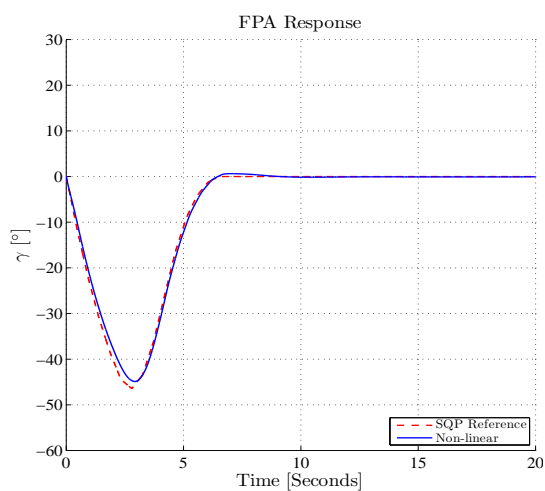


Figure D.12: Closed-loop non-linear response for Case 1 combined SQP state and input commands with  $\Phi_{lim} = 180^\circ$  continued.

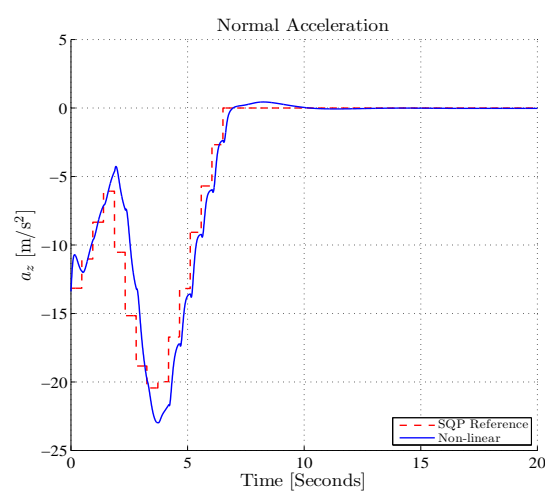
**Case 2** - Case 2 does not take the bank angle into account in the flight path angle control law, with no discontinuity in the command signal. The flight path angle command law then reduces to,

$$a_{z_c} = -\bar{V}\dot{\gamma} + a_z^* \quad (\text{D.3})$$

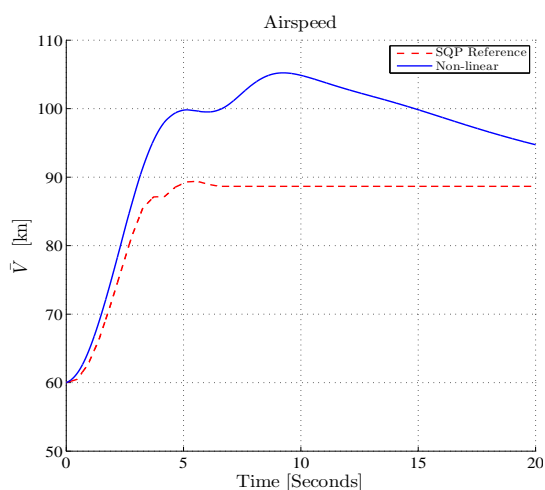
Figures D.13 and D.14 shows the GTM system response for case 2. The lower angle of attack response results in a overshoot in the speed response.



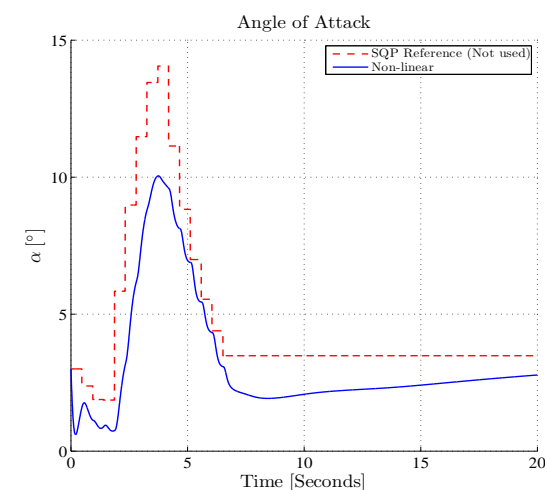
(a) Non-linear flight path angle response



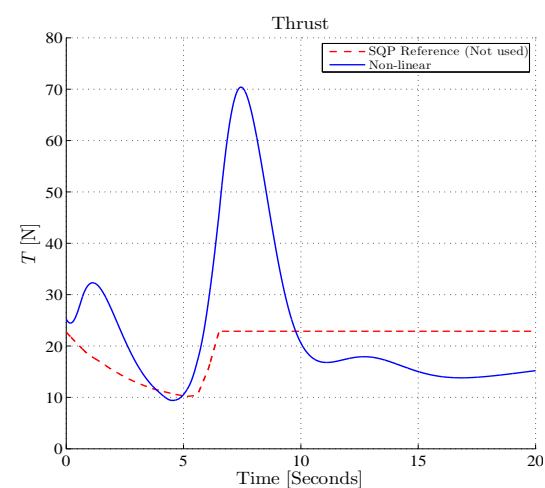
(b) Non-linear normal acceleration response



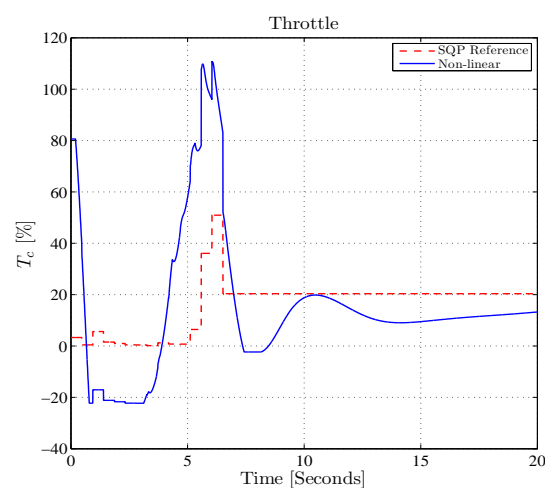
(c) Non-linear airspeed response



(d) Non-linear angle of attack response



(e) Non-linear thrust response



(f) Non-linear throttle input

Figure D.13: Closed-loop non-linear response for Case 2 combined SQP state and input commands.

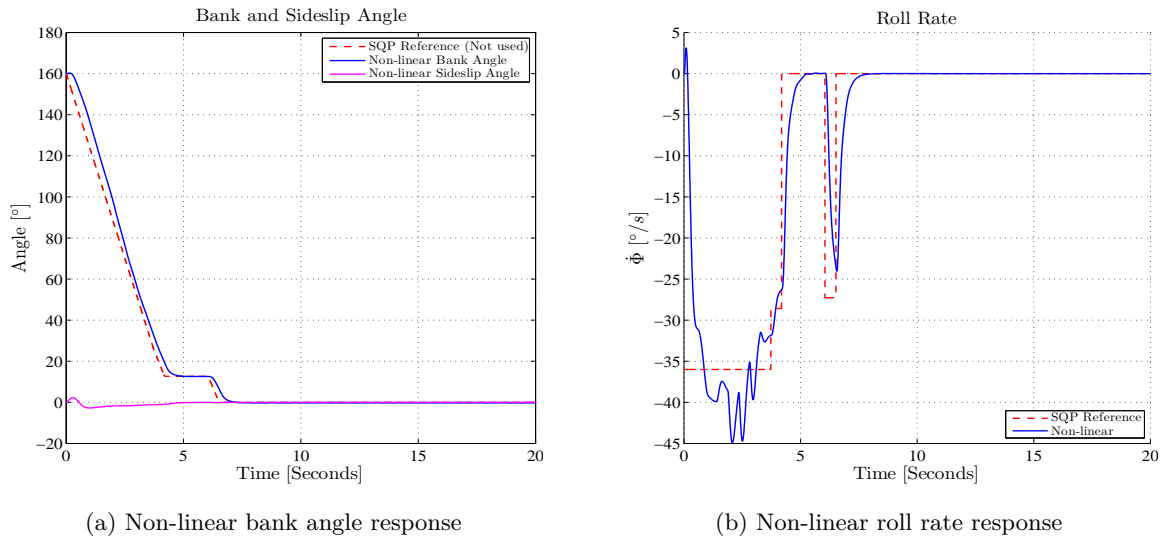
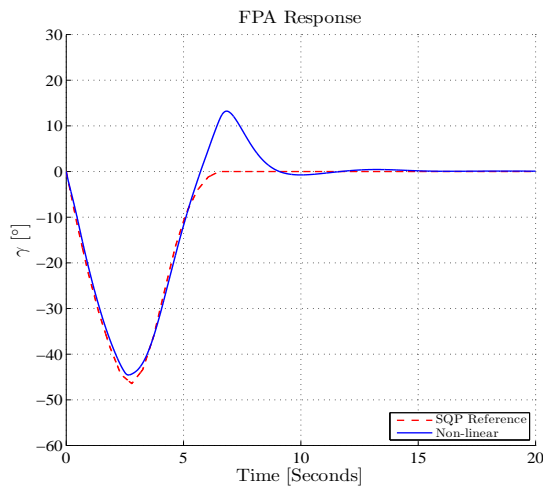


Figure D.14: Closed-loop non-linear response for Case 2 combined SQP state and input commands continued.

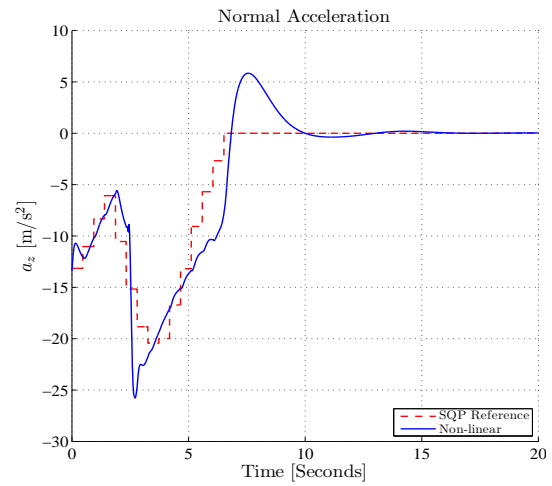
**Case 3** - Case 3 differs from case 1 in that the total normal acceleration is set to zero instead of the normal specific acceleration for bank angles greater than some threshold angle  $\Phi_{lim}$  degrees,

$$a_{z_c} = \begin{cases} \frac{-\bar{V}\dot{\gamma} - a_{\gamma_{gravity}}}{\cos \Phi} + a_{z_{gravity}} + a_z^*, & \text{if } \Phi \leq \Phi_{lim} \\ 0 + a_z^*, & \text{if } \Phi > \Phi_{lim} \end{cases} \quad (D.4)$$

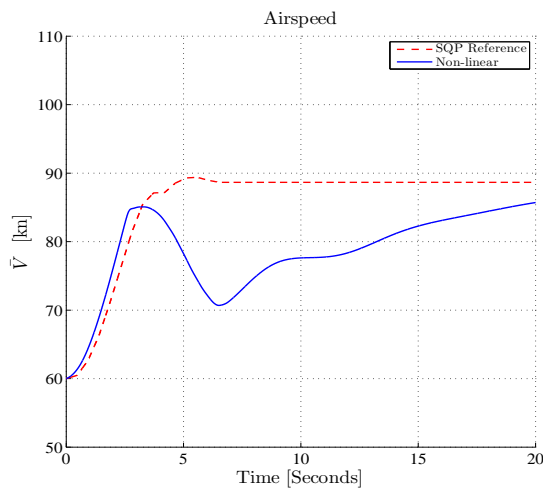
Figures D.15 and D.16 shows the GTM system response for case 3 with  $\Phi_{lim} = 80^\circ$ . Setting the total acceleration from the flight path angle controller to zero results in a smaller jump in control effort at the discontinuity at  $\Phi = 80^\circ$  compared to case 1. Figures D.17 and D.18 shows the GTM system response for case 3 with  $\Phi_{lim} = 60^\circ$ . The jump in control effort is even smaller than the previous case. Both the responses in case 3 also only coincidentally tracks the angle of attack and speed reference due to the jump in normal acceleration command, giving a larger angle of attack.



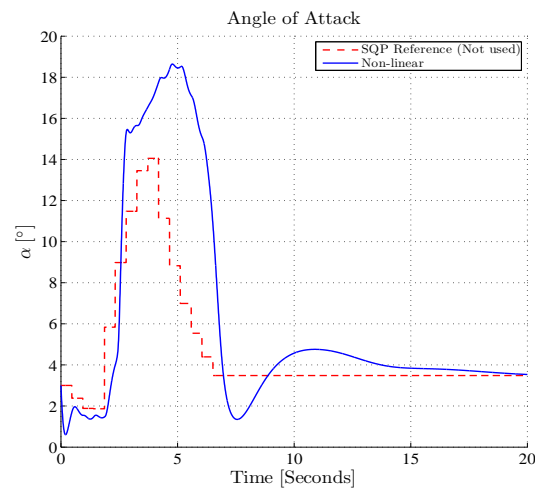
(a) Non-linear flight path angle response



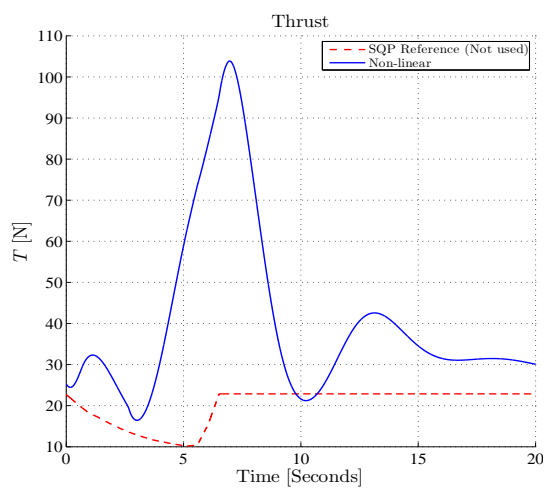
(b) Non-linear normal acceleration response



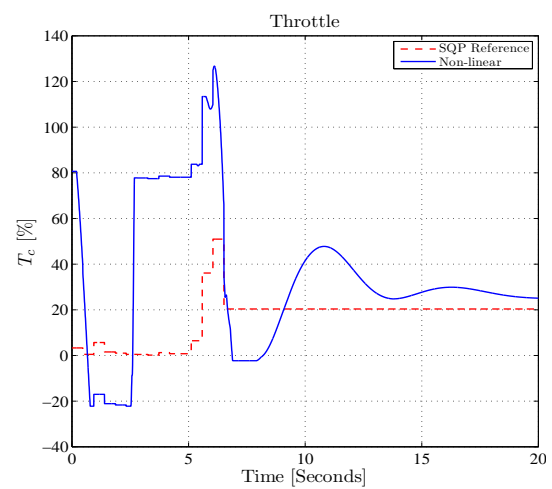
(c) Non-linear airspeed response



(d) Non-linear angle of attack response



(e) Non-linear thrust response



(f) Non-linear throttle input

Figure D.15: Closed-loop non-linear response for Case 3 combined SQP state and input commands with  $\Phi_{lim} = 80^\circ$ .

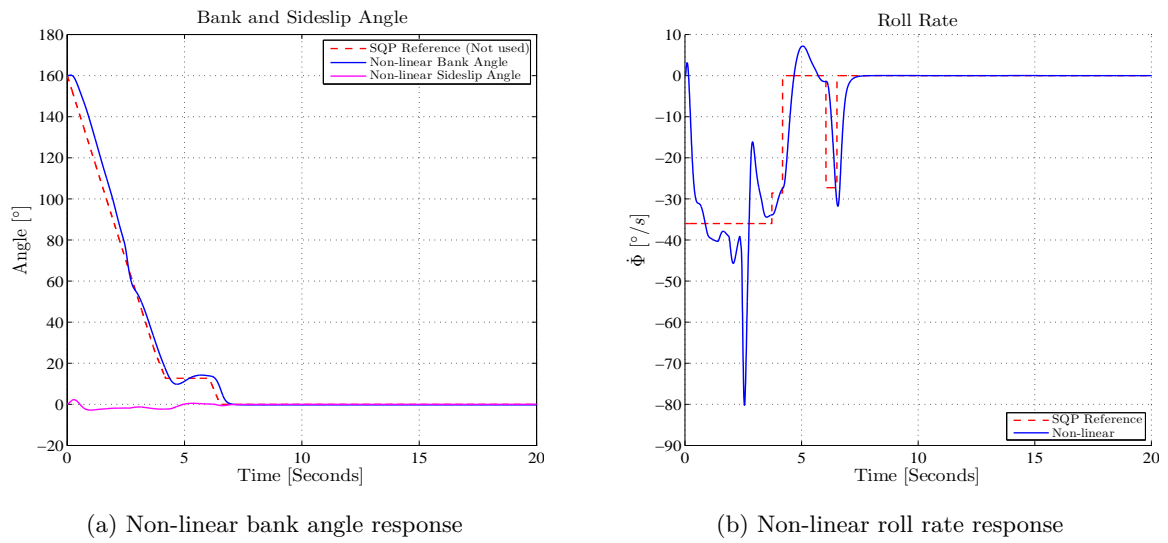
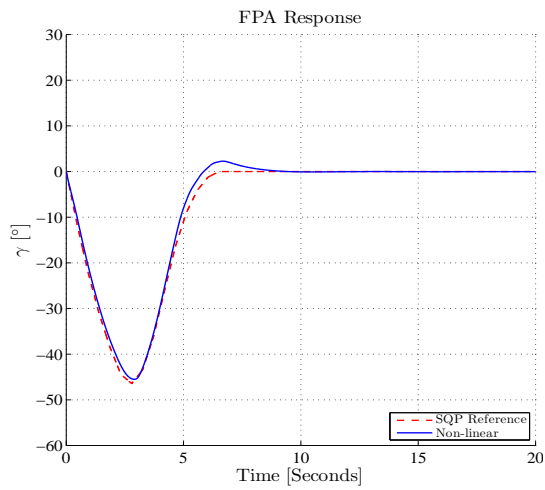
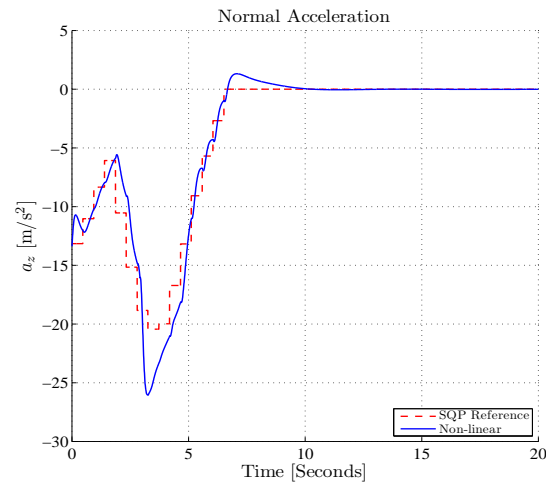


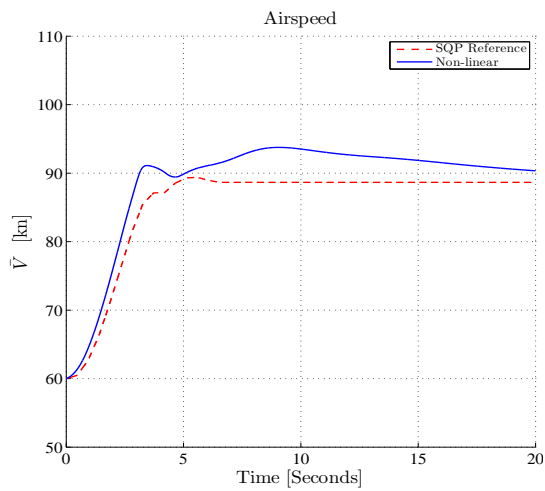
Figure D.16: Closed-loop non-linear response for Case 3 combined SQP state and input commands with  $\Phi_{lim} = 80^\circ$  continued.



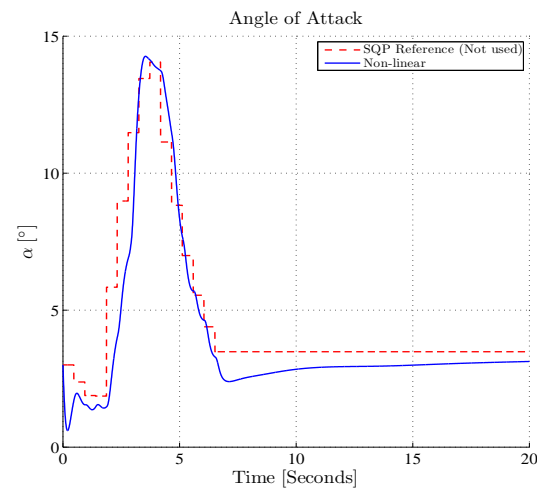
(a) Non-linear flight path angle response



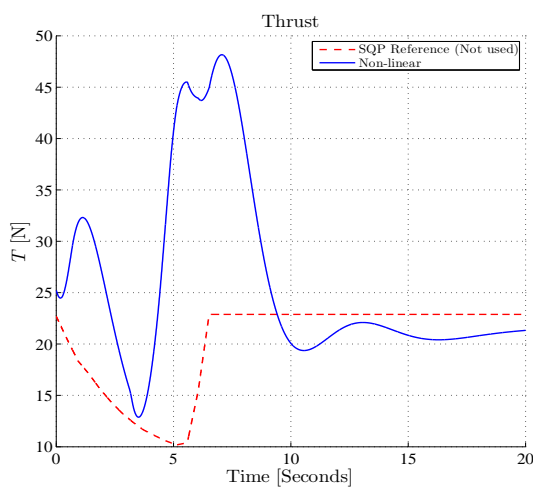
(b) Non-linear normal acceleration response



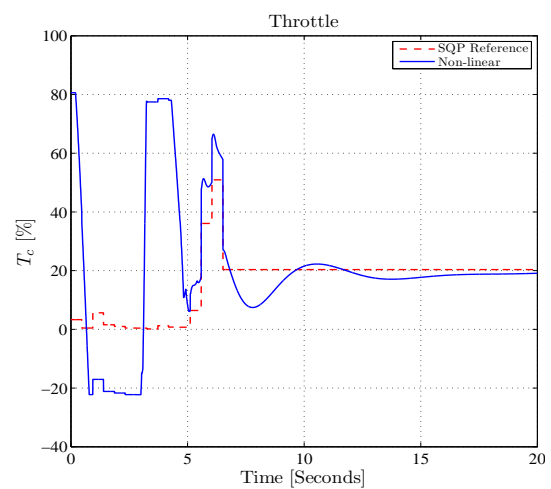
(c) Non-linear airspeed response



(d) Non-linear angle of attack response



(e) Non-linear thrust response



(f) Non-linear throttle input

Figure D.17: Closed-loop non-linear response for Case 3 combined SQP state and input commands with  $\Phi_{lim} = 60^\circ$ .

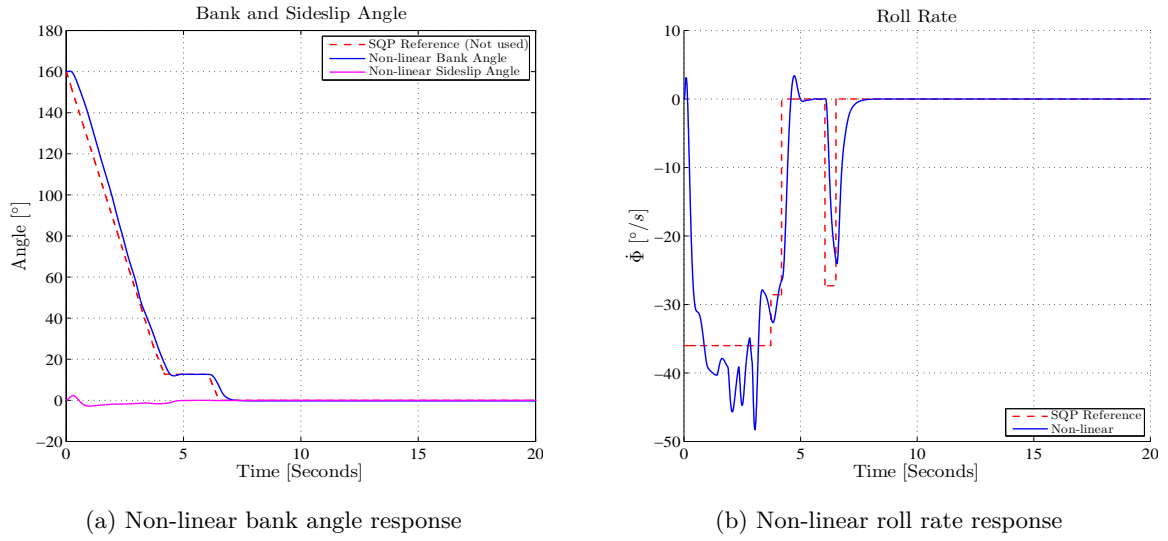


Figure D.18: Closed-loop non-linear response for Case 3 combined SQP state and input commands with  $\Phi_{lim} = 60^\circ$  continued.

## D.2 Angle of Attack vs Normal Acceleration (DQ) Controller

Due to the poor airspeed trajectory tracking, the upset recovery simulations were repeated with a baseline angle of attack controller instead of the normal acceleration (DQ) controller. Furthermore, the translated normal acceleration reference calculation was changed to use a different equation than that of Equation D.1. The Normal acceleration reference is now translated using a wind-axis to body-axis transformation using the angle of attack angle and sideslip angle assumed zero,

$$a_z^* = \sin \alpha^* a_{x_w}^* + \cos \alpha^* a_{z_w}^* \quad (D.1)$$

where  $a_{x_w}^*$  is the axial acceleration in the wind axis and  $a_{z_w}^*$  is the normal acceleration in the wind-axis, i.e. the lift axis acceleration,

$$a_{x_w}^* = \frac{1}{m} \left[ T^* - \frac{1}{2} \rho \bar{V}^{*2} S C_D(\alpha^*) - mg \sin \gamma^* \right] \quad (D.2)$$

$$a_{z_w}^* = \frac{1}{m} \left[ \frac{1}{2} \rho \bar{V}^{*2} S C_L(\alpha^*) + mg \cos \gamma^* \cos \Phi^* \right] \quad (D.3)$$



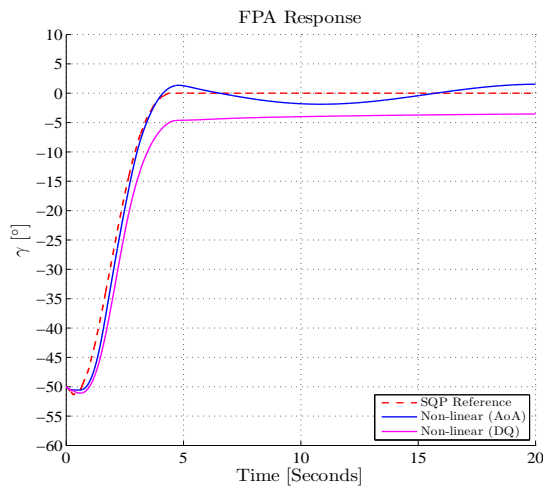
### Input Reference Only

The angle of attack controller variant is tested alongside the normal acceleration (DQ) controller variant for comparison for different upset conditions, using only the planned input signals from the SQP algorithm as references to the inner-loop controllers in an open-loop fashion.

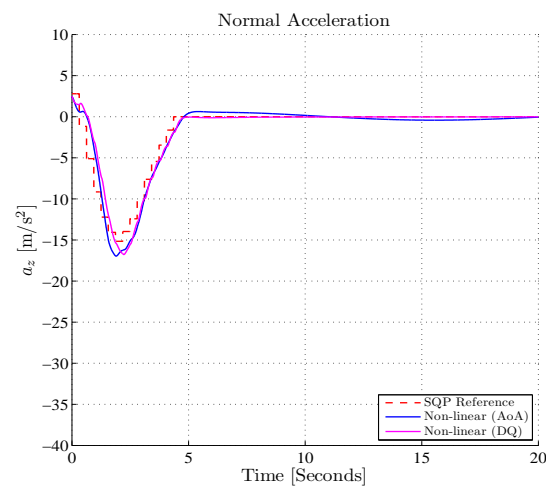
**Case 1** - Case 1 consists of an upset condition of Table D.2 with wings level, descending flight path angle and slight underspeed. The GTM system response from this particular initial upset condition is illustrated in Figures D.19 and D.20. It can be seen that the executed airspeed response of the angle of attack controller scheme variant does not overshoot the planned airspeed trajectory, unlike the DQ controller variant, which exhibits significant airspeed overshoot. The better airspeed tracking of the angle of attack controller scheme variant can be attributed to the fact that the executed angle of attack signal tracks the planned angle of attack signal much closer than the DQ controller variant. This means that the expected drag is produced and that in turn produces the correct airspeed response.

Table D.2: Initial upset condition

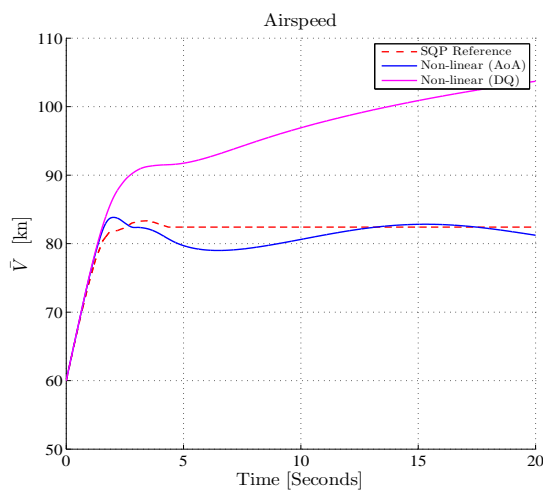
State	Value	Units
$\bar{V}_{initial}$	60	kn
$\gamma_{initial}$	-50	degrees
$\alpha_{initial}$	3	degrees
$\beta_{initial}$	0	degrees
$\Phi_{initial}$	0	degrees
$T_{initial}$	25.2	Newton



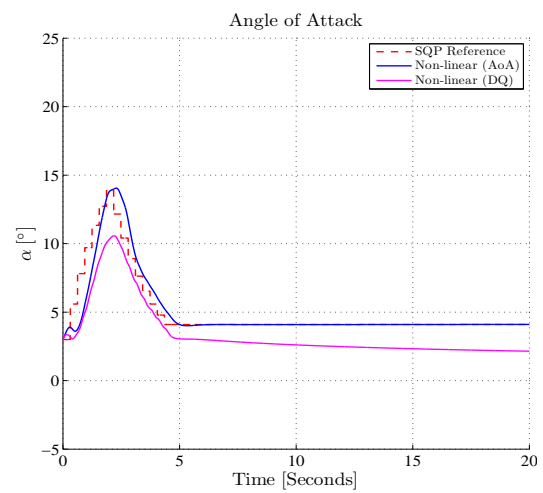
(a) Non-linear flight path angle response



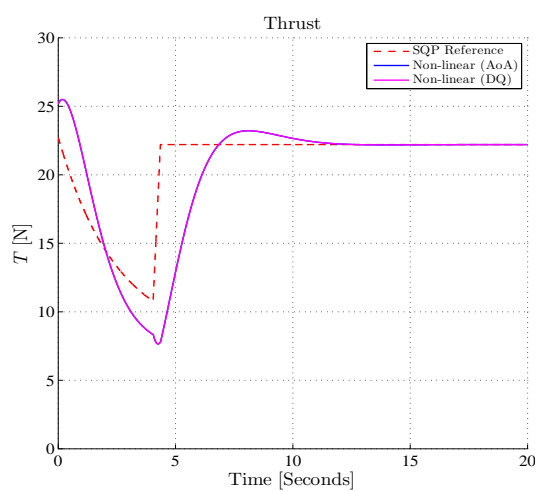
(b) Non-linear normal acceleration response



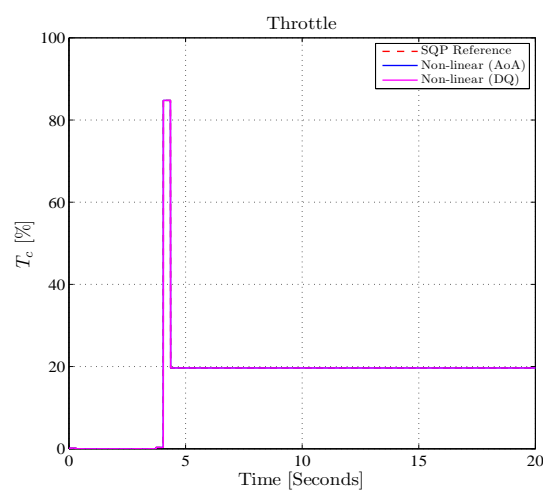
(c) Non-linear airspeed response



(d) Non-linear angle of attack response

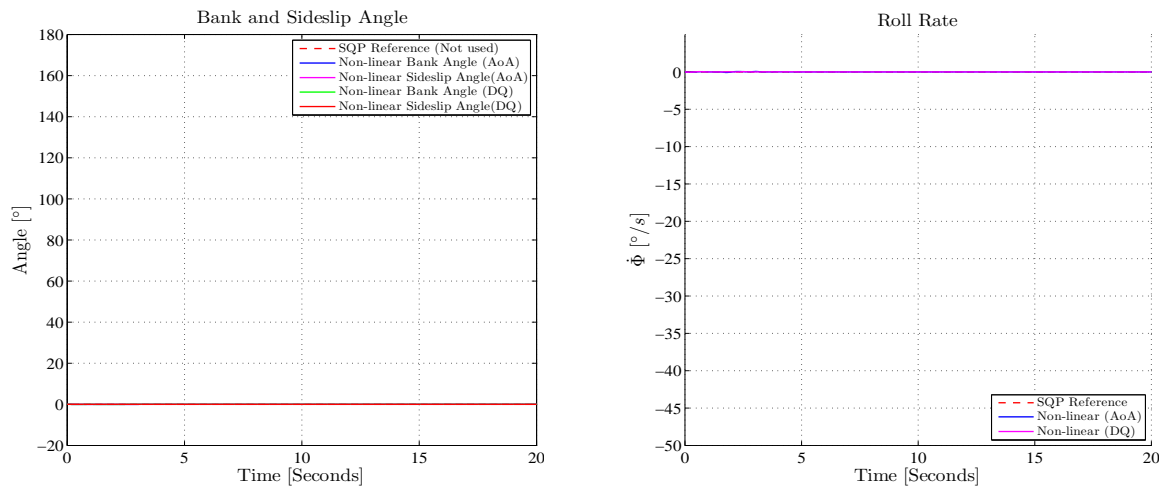


(e) Non-linear thrust response



(f) Non-linear throttle input

Figure D.19: Closed-loop non-linear response to SQP input commands from upset condition.



(a) Non-linear bank angle response

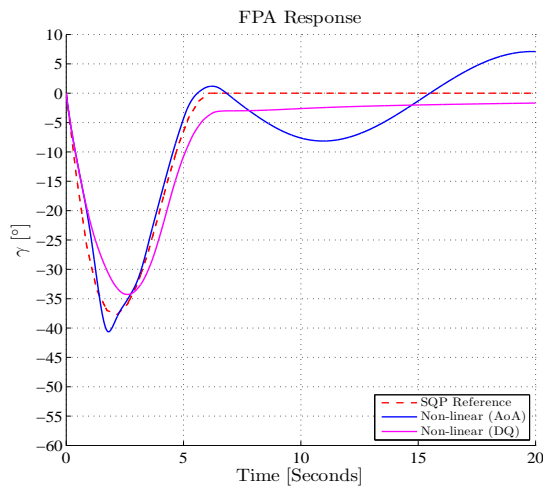
(b) Non-linear roll rate response

Figure D.20: Closed-loop non-linear response to SQP input commands from upset condition continued.

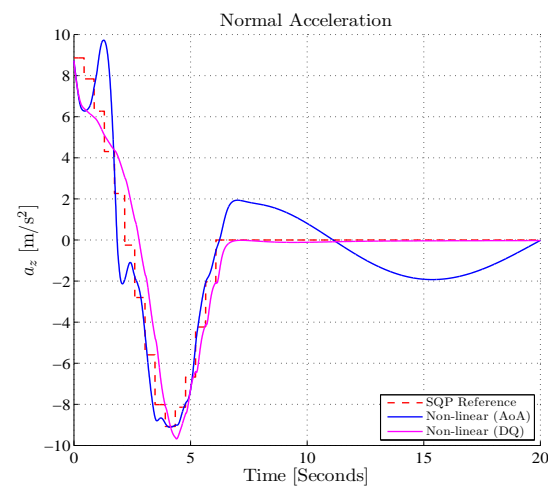
**Case 2** - Case 2 consists of an upset condition of Table D.3 with wings level, severe under-speed and level flight path angle. The GTM system response from this particular initial upset condition is illustrated in Figures D.21 and D.22.

Table D.3: Initial upset condition

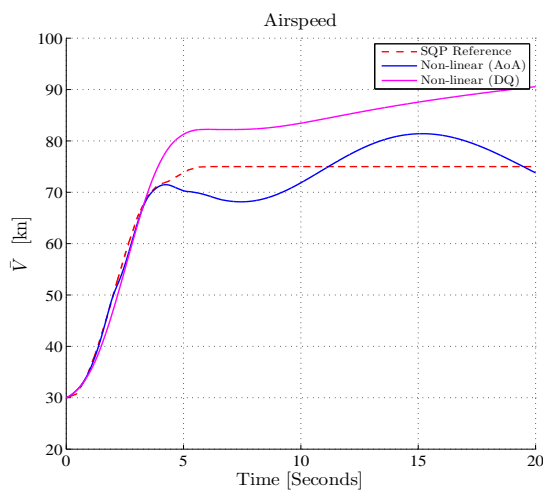
State	Value	Units
$\bar{V}_{initial}$	30	kn
$\gamma_{initial}$	-0	degrees
$\alpha_{initial}$	3	degrees
$\beta_{initial}$	0	degrees
$\Phi_{initial}$	0	degrees
$T_{initial}$	25.2	Newton



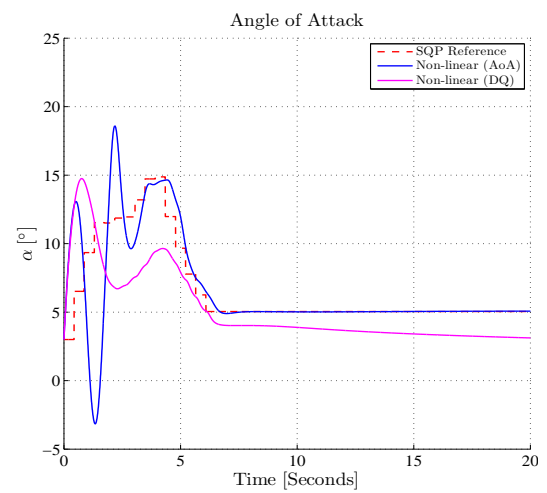
(a) Non-linear flight path angle response



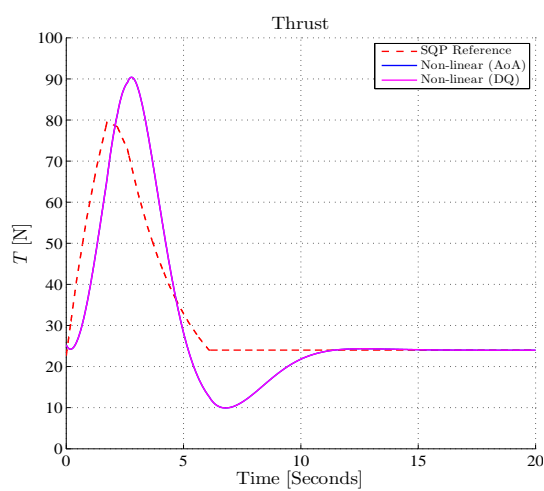
(b) Non-linear normal acceleration response



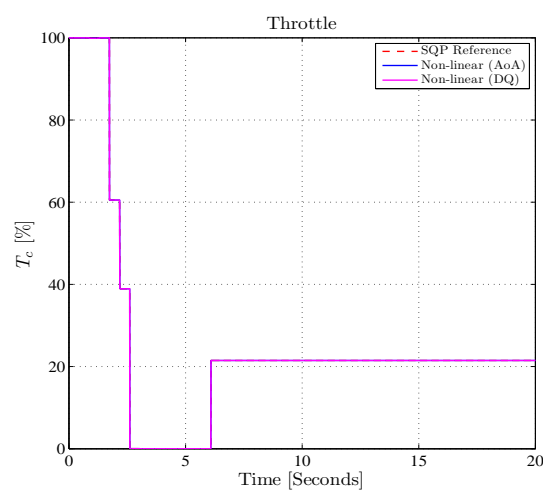
(c) Non-linear airspeed response



(d) Non-linear angle of attack response



(e) Non-linear thrust response



(f) Non-linear throttle input

Figure D.21: Closed-loop non-linear response to SQP input commands from upset condition.

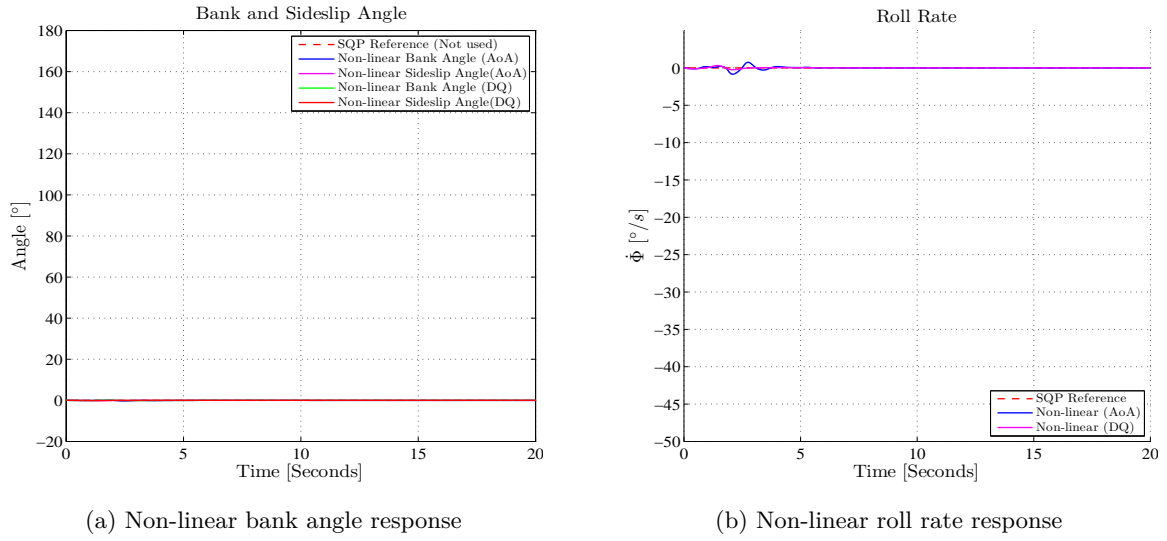


Figure D.22: Closed-loop non-linear response to SQP input commands from upset condition continued.

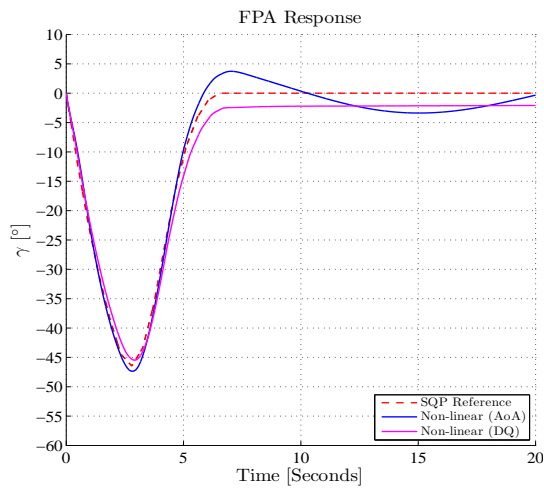
As with the previous case, there is less airspeed overshoot in the case of the angle of attack controller variant, due to the better angle of attack tracking.

It was found that for this case, at low airspeeds the angle of attack controller struggles to regulate the angle of attack and instead oscillates until the airspeed is closer to the trim airspeed.

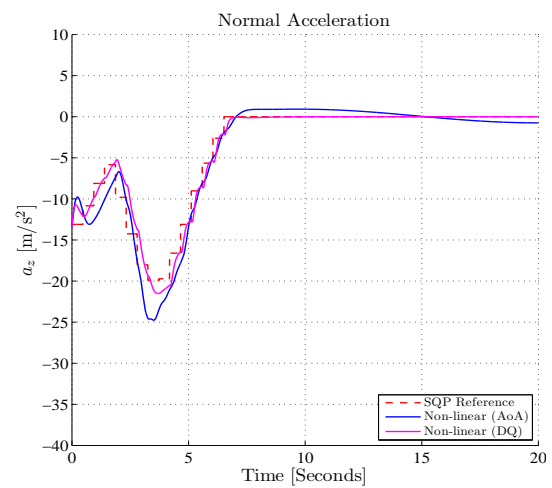
**Case 3** - Case 3 consist of an upset condition of Table D.4 with near inverted bank angle, level flight path angle and slight underspeed. The GTM system response from this particular initial upset condition is illustrated in Figures D.23 and D.24.

Table D.4: Initial upset condition

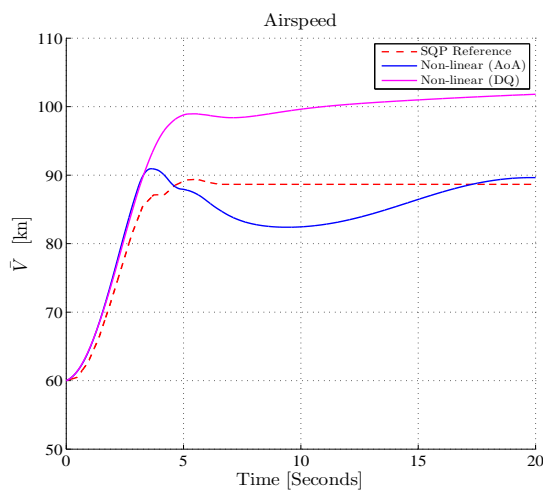
State	Value	Units
$\bar{V}_{initial}$	60	kn
$\gamma_{initial}$	-0	degrees
$\alpha_{initial}$	3	degrees
$\beta_{initial}$	0	degrees
$\Phi_{initial}$	160	degrees
$T_{initial}$	25.2	Newton



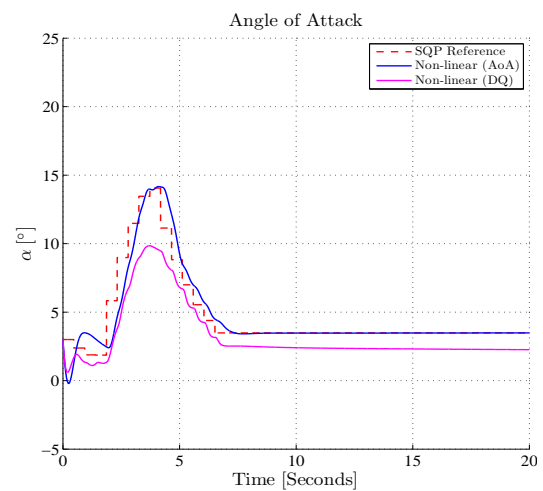
(a) Non-linear flight path angle response



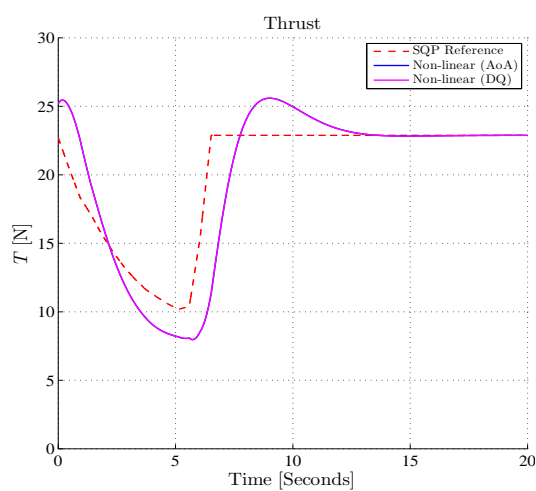
(b) Non-linear normal acceleration response



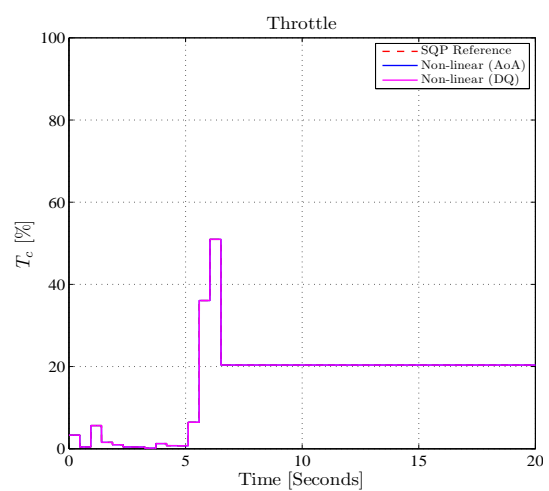
(c) Non-linear airspeed response



(d) Non-linear angle of attack response



(e) Non-linear thrust response



(f) Non-linear throttle input

Figure D.23: Closed-loop non-linear response to SQP input commands from upset condition.

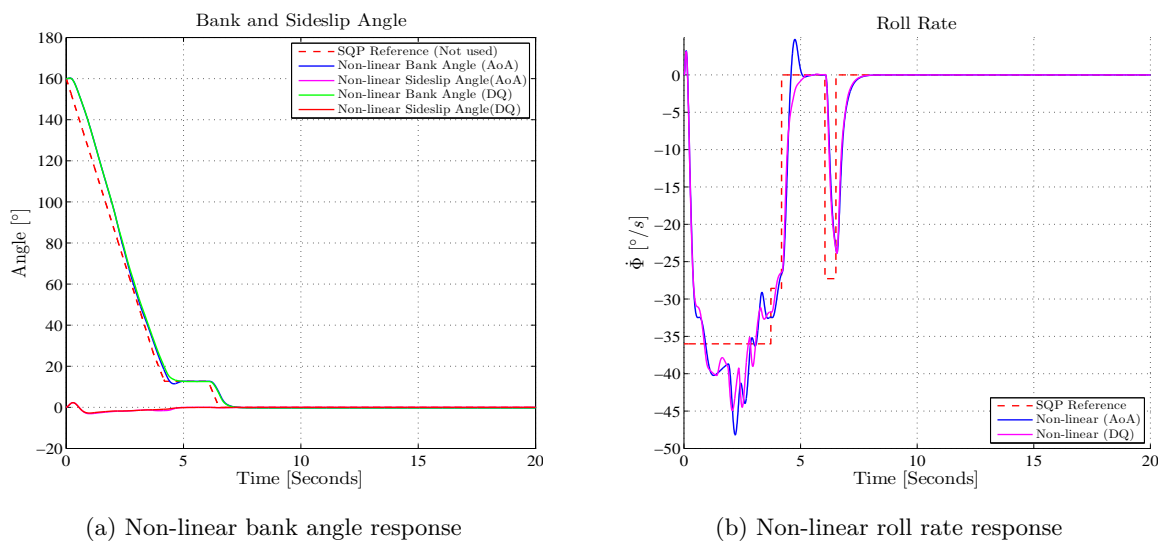


Figure D.24: Closed-loop non-linear response to SQP input commands from upset condition continued.

As with the previous cases, there is less airspeed overshoot in the case of the angle of attack controller variant, due to the better angle of attack tracking.

**Summary** - With the angle of attack strictly regulated, the speed trajectory matches much closer to that of the predicted trajectory. With no middle-loop control, all the cases where the angle of attack controller was used showed an undesirable phugoid response in airspeed and flight path angle at the end of the trajectory.

### Gain Scheduled Angle of Attack Controller

Due to the poor angle of attack state tracking at low airspeeds as can be seen in case 2 Figures D.21 and D.22, a LQI gain scheduled angle of attack controller was designed (detailed in Chapter 3 section §3.4). A feed-forward gain was added to the LQI controller, as was done in the DQ controller design to give similar response times than the DQ controller.

#### D.2.1 Angle of Attack Controller Model Comparison

Figures D.25 and D.26 illustrate the GTM system responses of executed recovery trajectories from the same initial upset condition in order to compare the use of a first-order angle of attack inner-loop controller model compared to using a second-order model in the SQP trajectory planning algorithm.

Figure D.25 shows an example of the executed recovery trajectories by the GTM from an initial longitudinal upset case, using the first-order delay model for the angle of attack controller. The executed angle of attack trajectory shows severe overshoot from the predicted angle of attack trajectory solution at  $t = 0.5$  seconds.

Figure D.26 shows the executed recovery trajectories by the GTM for the same initial longitudinal upset condition, using the second-order angle of attack inner-loop controller model. The overshoot at  $t = 0.5$  seconds has improved in the response between the planned angle of attack and the executed angle of attack trajectory. It can be observed that using the second-order model, that the SQP planning algorithm's angle of attack solution now shows that it expects

more overshoot at  $t = 0.25$  seconds, and thus the relative overshoot from the executed angle of attack vs the planned angle of attack trajectory improved.

Both cases used the Hermite-Simpson collocation method rather than Euler collocation to illustrate the difference in angle of attack model fairly.

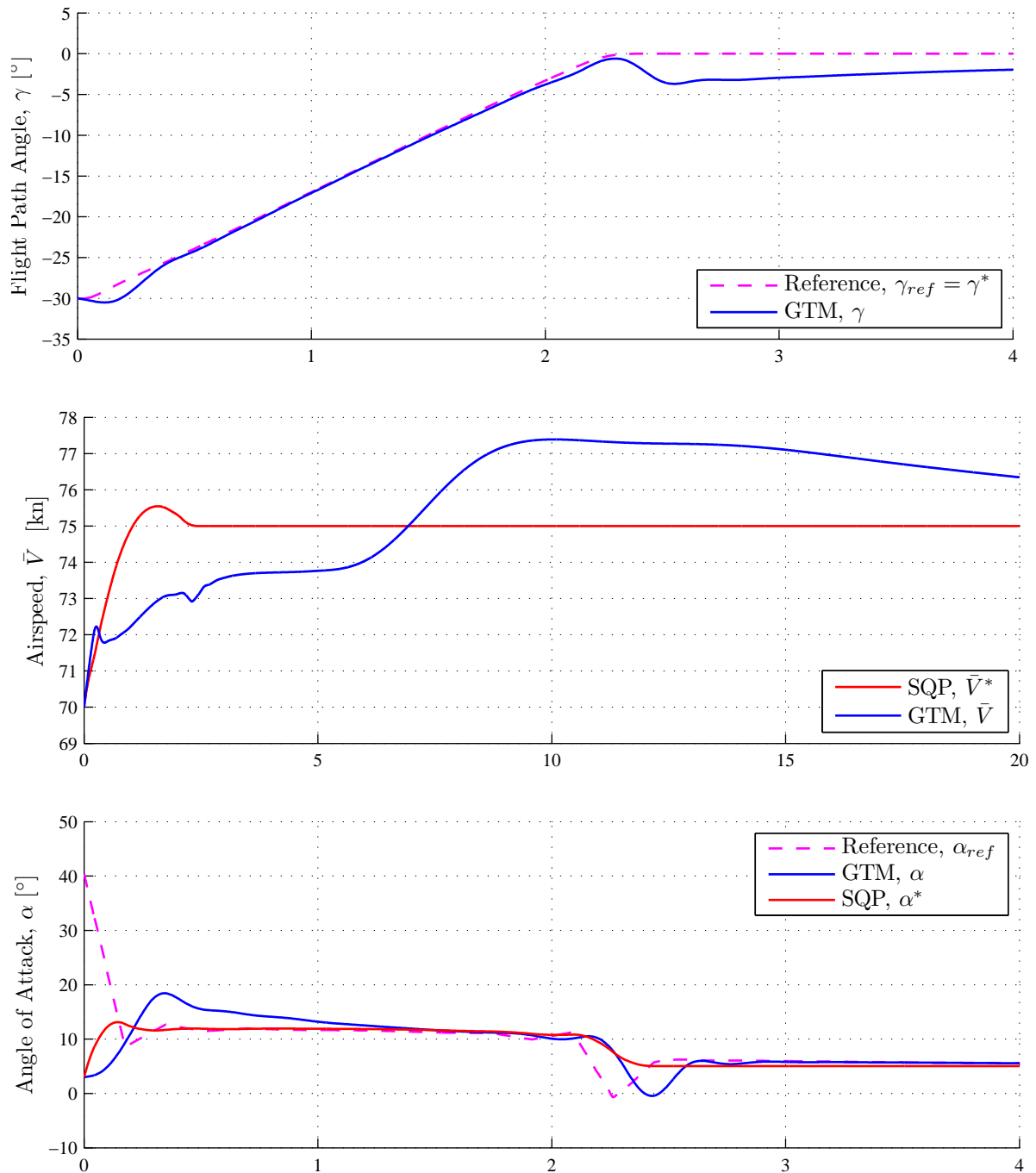


Figure D.25: GTM response to trajectory regulation using a first order angle of attack model with angle of attack inner-loop controller



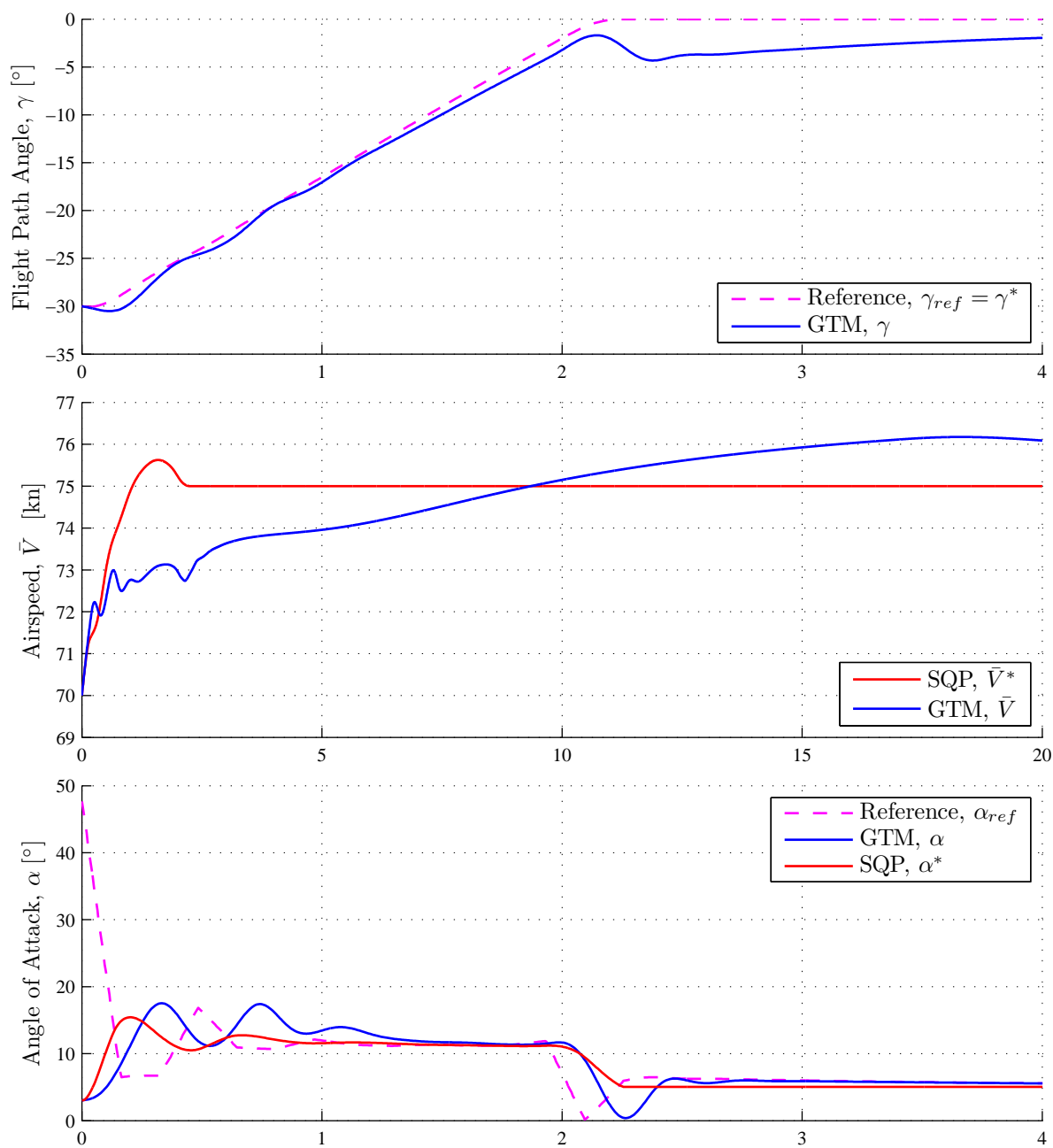


Figure D.26: GTM response to trajectory regulation using a second order angle of attack model with angle of attack inner-loop controller

# Bibliography

- [1] Engelbrecht, J.A.A.: *Automatic Flight Envelope Recovery for Large Transport Aircraft*. PhD Dissertation, University of Stellenbosch, 2016.  
Available at: <http://hdl.handle.net/10019.1/100227>
- [2] Belcastro, C.M.: Loss of Control Prevention and Recovery: Onboard Guidance, Control, and Systems Technologies. *AIAA Guidance, Navigation, and Control Conference, 13-16 August 2012, Minneapolis, MN, USA*, , no. AIAA 2012-4762, pp. 1–15, 2012.
- [3] Kwatny, H.G., Dongmo, J.-E.T., Chang, B.-C., Bajpai, G., Yasar, M. and Belcastro, C.: Aircraft Accident Prevention : Loss-of-Control Analysis. *AIAA Guidance Navigation and Control Conference*, , no. August, pp. 1–14, 2009.  
Available at: <http://www.pages.drexel.edu/~hgk22/papers/AIAA-2009-6256-834.pdf>
- [4] Allen, R.C.: *Safe Set Maneuverability, Restoration, and Protection for Aircraft*. Dissertation, Drexel University, March 2014.
- [5] Crespo, L., Kenny, S., Cox, D. and Murri, D.: Analysis of Control Strategies for Aircraft Flight Upset Recovery. In: *AIAA Guidance, Navigation, and Control Conference*, pp. 1–31. American Institute of Aeronautics and Astronautics, Reston, Virginia, aug 2012. ISBN 978-1-60086-938-9.  
Available at: <http://arc.aiaa.org/doi/10.2514/6.2012-5026>
- [6] Stepanyan, V., Krishnakumar, K.S., Kaneshige, J. and Acosta, D.M.: Stall Recovery Guidance Algorithms Based on Constrained Control Approaches. In: *AIAA Guidance, Navigation, and Control Conference*, pp. 1–26. American Institute of Aeronautics and Astronautics, Reston, Virginia, jan 2016. ISBN 978-1-62410-389-6.  
Available at: <http://arc.aiaa.org/doi/10.2514/6.2016-0878>
- [7] Dongmo, J.-E.: Aircraft Loss-Of-Control Recovery Using Nonlinear Smooth Feedback Regulators. In: *Infotech@Aerospace 2011*, March. American Institute of Aeronautics and Astronautics, Reston, Virginia, mar 2011. ISBN 978-1-60086-944-0.  
Available at: <http://arc.aiaa.org/doi/10.2514/6.2011-1562>
- [8] Engelbrecht, J.J. and Engelbrecht, J.A.: Optimal attitude and flight vector recovery for large transport aircraft using sequential quadratic programming. In: *Pattern Recognition Association of South Africa and Robotics and Mechatronics International Conference (PRASA-RobMech), 2016*, pp. 1–7. IEEE, 2016.
- [9] Carbaugh, D., Rockliff, L. and Vandell, R.: Airplane Upset Recovery Training Aid, Revision 2. *November*, , no. November, 2008.  
Available at: [https://www.faa.gov/other\\_visit/aviation\\_industry/airline\\_operators/training/media/AP\\_UpsetRecovery\\_Book.pdf](https://www.faa.gov/other_visit/aviation_industry/airline_operators/training/media/AP_UpsetRecovery_Book.pdf)

- [10] Wilborn, J.E. and Foster, J.V.: Defining Commercial Transport Loss-of-Control : A Quantitative Approach. *AIAA Atmospheric Flight Mechanics Conference and Exhibit*, pp. 1–11, August 2004.
- [11] Federal Aviation Administration: *Airplane Flying Handbook (FAA-H-8083-3B)*. U.S. Department of Transportation, Federal Aviation Administration, 2016.
- [12] Jordan, T., Langford, W., Belcastro, C.M., Foster, J., Shah, G., Howland, G. and Kidd, R.: Development of a Dynamically Scaled Generic Transport Model Testbed for Flight Research Experiments. *AUVSI's Unmanned Systems North America 2004 Symposium and Exhibition*, p. 16, 2004.
- [13] Murch, A.M.: *Aerodynamic modeling of post-stall and spin dynamics of large transport airplanes*. Ph.D. thesis, Georgia Institute of Technology, 2007.
- [14] Engelbrecht, J.A.A.: Advance Automation 833 Introductory course to aircraft dynamics. 2016.
- [15] Etkin, B. and Reid, L.D.: *Dynamics of Flight: Stability and Control*. 3rd edn. John Wiley & Sons Australia, Limited, 1993. ISBN 978-0-471-03418-6.  
Available at: <http://eu.wiley.com/WileyCDA/WileyTitle/productCd-0471034185.html>
- [16] Trollip, E.F.: *Ride Comfort in Commercial Aircraft During Formation Flight Using Conventional Flight Control*. Master's Thesis, University of Stellenbosch, 2015.
- [17] Peddle, I.K.: *Acceleration based manoeuvre flight control system for unmanned aerial vehicles*. Ph.D. thesis, 2008.  
Available at: <http://scholar.sun.ac.za/handle/10019.1/1172>
- [18] Chambers, J.: *Modeling Flight NASA Latest Version: The role of dynamically scale Free Flight Models in support of NASA aerospace programs*. NASA. US National Aeronautics and Space Admin, 2015. ISBN 9780160846335.  
Available at: <https://books.google.co.za/books?id=uYlUCgAAQBAJ>
- [19] Favre, C.: Fly-by-wire for commercial aircraft: the airbus experience. *International Journal of Control*, vol. 59, no. 1, pp. 139–157, 1994. <http://dx.doi.org/10.1080/00207179408923072>.  
Available at: <http://dx.doi.org/10.1080/00207179408923072>
- [20] Kun, Z., Lixin, W. and Xiangsheng, T.: Flying qualities reduction of fly-by-wire commercial aircraft with reconfiguration flight control laws. *Procedia Engineering*, vol. 17, pp. 179–196, 2011. ISSN 18777058.  
Available at: <http://dx.doi.org/10.1016/j.proeng.2011.10.021>
- [21] Bryson Jr, A.E.: *Control of spacecraft and aircraft*. Princeton university press, 1994.
- [22] Stevens, B., Lewis, F. and Johnson, E.: *Aircraft Control and Simulation: Dynamics, Controls Design, and Autonomous Systems*. Wiley, 2015. ISBN 9781118870983.  
Available at: <https://books.google.co.za/books?id=lvhcGgAAQBAJ>
- [23] Blakelock, J.: *Automatic Control of Aircraft and Missiles*. A Wiley-Interscience publication. Wiley, 1991. ISBN 9780471506515.  
Available at: <https://books.google.co.za/books?id=ubcczZUDcSMC>
- [24] Betts, J.T.: Survey of Numerical Methods for Trajectory Optimization. *Journal of Guidance, Control, and Dynamics*, vol. 21, no. 2, pp. 193–207, 1998. ISSN 0731-5090.

- [25] Rao, A.: A survey of numerical methods for optimal control. *Advances in the Astronautical Sciences*, vol. 135, no. 1, pp. 497–528, 2009. ISSN 1569-3953.  
Available at: <http://vdol.mae.ufl.edu/ConferencePublications/trajectorySurveyAAS.pdf>
- [26] Kelly, M.P.: Transcription Methods for Trajectory Optimization A beginners tutorial. Tech. Rep., 2015.  
Available at: [http://www.matthwepeterkelly.com/research/MattKelly\\_Transcription\\_Methods\\_for\\_Trajectory\\_Optimization.pdf](http://www.matthwepeterkelly.com/research/MattKelly_Transcription_Methods_for_Trajectory_Optimization.pdf)
- [27] Kelly, M.: An introduction to trajectory optimization: how to do your own direct collocation, 2017.  
Available at: [http://www.matthwepeterkelly.com/research/MatthewKelly\\_IntroTrajectoryOptimization\\_SIAM\\_Review\\_2017.pdf](http://www.matthwepeterkelly.com/research/MatthewKelly_IntroTrajectoryOptimization_SIAM_Review_2017.pdf)
- [28] Kirk, D.E.: *Optimal Control Theory: An Introduction*. Dover Publications, Mineola, New York, 2004. ISBN 0486434842.  
Available at: [http://www.goodreads.com/book/show/1020057.Optimal\\_Control\\_Theory](http://www.goodreads.com/book/show/1020057.Optimal_Control_Theory)
- [29] Sparks, D. and Moerder, D.: Optimal aircraft control upset recovery with and without component failures. In: *Proceedings of the 2002 American Control Conference (IEEE Cat. No.CH37301)*, vol. 5, pp. 3644–3649 vol.5. IEEE, 2002. ISBN 0-7803-7298-0. ISSN 07431619.  
Available at: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1024494>
- [30] Engelbrecht, J.J.K.: Flight Control for Upset Recovery of Commercial Aircraft (Unpublished skripsie report). 2015.
- [31] Arora, J.S.: *Introduction to Optimum Design*. 2nd edn. Elsevier Academic Press, Iowa City, Iowa, USA, aug 2004. ISBN 0-12-064155-0.
- [32] Topputo, F. and Zhang, C.: Survey of direct transcription for low-thrust space trajectory optimization with applications. *Abstract and Applied Analysis*, vol. 2014, no. April 2016, 2014. ISSN 16870409.
- [33] Betts, J.T.: *Practical Methods for Optimal Control and Estimation Using Nonlinear Programming*. Society for Industrial and Applied Mathematics, jan 2010. ISBN 978-0-89871-688-7.  
Available at: <http://epubs.siam.org/doi/book/10.1137/1.9780898718577>
- [34] Patterson, M.A., Weinstein, M.J. and Rao, A.V.: An efficient overloaded method for computing derivatives of mathematical functions in MATLAB. *ACM Transactions on Mathematical Software*, vol. 39, no. 3, pp. 17:1–17:36, 2013. ISSN 15564681. 1005.3014.
- [35] Neidinger, R.D.: Introduction to Automatic Differentiation and MATLAB Object-Oriented Programming. *SIAM Review*, vol. 52, no. 3, pp. 545–563, 2010. ISSN 0036-1445.
- [36] Weinstein, M.J. and Rao, A.V.: ADiGator V1.2 User Guide.
- [37] Pardo, D., Möller, L., Neunert, M., Winkler, A.W. and Buchli, J.: Evaluating direct transcription and nonlinear optimization methods for robot motion planning. pp. 1–8, 2015. ISSN 2377-3766. 1504.05803.  
Available at: <http://arxiv.org/abs/1504.05803><http://dx.doi.org/10.1109/LRA.2016.2527062>

- [38] van Wyk, F.: *Optimal Control for Minimum Thrust Demand in Extended Formation Flight*. Master's Thesis, University of Stellenbosch, 2015.