

Reconfigurable Flight Control First Year Report

Colin N. Jones
Pembroke College

Supervisor: Dr. J.M.Maciejowski



Control Group
Department of Engineering
University of Cambridge

March 4, 2005

Contents

Contents	iii
List of Tables	vii
List of Figures	ix
List of Acronyms	xi
List of Symbols	xiii
1 Introduction	1
1.1 Motivation	1
1.2 Overview	2
2 Preliminaries	3
2.1 Aircraft Modelling	3
2.1.1 Reference Frames	3
2.1.2 General Equations of Motion	4
2.1.3 Linearized Equations of Motion	8
2.1.4 Fault Models	8
2.2 Model Predictive Control (MPC)	9
3 Reconfigurable Flight Control: A Survey	13
3.1 Introduction	13
3.2 Multiple Model Control	15
3.2.1 Multiple Model Switching and Tuning (MMST)	15
3.2.2 Interacting Multiple Models (IMM)	18
3.2.3 Propulsion Controlled Aircraft (PCA)	19
3.3 Control Allocation (CA)	19
3.4 Adaptive Feedback Linearization via Artificial Neural Network	21
3.4.1 SISO Feedback Linearization	21
	iii

3.4.2	Feedback Linearization for Reconfigurable Control	22
3.5	Sliding Mode Control (SMC)	24
3.5.1	Introduction to Sliding Mode Control	24
3.5.2	Reconfigurable Sliding Mode Control	27
3.6	Eigenstructure Assignment (EA)	28
3.6.1	Introduction to Eigenstructure Assignment	28
3.6.2	Reconfigurable Eigenstructure Assignment	29
3.6.3	Pseudo Inverse Method (PIM)	30
3.7	Model Reference Adaptive Control (MRAC)	30
3.7.1	Indirect Adaptation	31
3.7.2	Direct Adaptation	32
3.8	Model Predictive Control (MPC)	32
3.9	Conclusion	35
4	Sufficient Conditions for Pilot Control	37
4.1	Introduction	37
4.2	Problem Definition	37
4.3	A Motivating Example - Flight EL AL 1862	39
4.3.1	Failure Scenario	39
4.3.2	Control Surfaces	41
4.3.3	Trimming and Linearization	41
4.3.4	Pilot Control via MPC	44
4.3.5	Trajectory Visualization	70
4.3.6	Conclusions	73
5	Research Proposal	75
5.1	Summary of Progress	75
5.2	Research Proposal	75
5.3	Improvements to the Boeing 747 Model	77
5.4	Fault Tolerant GARTEUR Group	78
5.5	Preliminary Schedule	79
	Appendix A Step Responses	81
	Appendix B jmpc Toolbox	91
B.1	Code	92
B.1.1	jmpc Model Creation	92
B.1.2	Control Move Calculation	96

References

99

Appendix C Author Index

106

List of Tables

2.1	Aircraft Failure Modes	9
3.1	Comparison of Reconfigurable Control Methods	34
4.1	Boeing 747 Pilot Controls and Mechanical Limits	41
4.2	Boeing 747 Control Surfaces and Mechanical Limits	42
4.3	Trim Configuration	43
4.4	Comparison of Lateral and Longitudinal Modes for Nominal, Failure and Typical Aircraft	44
4.5	Inner Loop Tuning Parameters: Output Constraints	47
4.6	Inner Loop Tuning Parameters: Input Constraints (Nominal Model)	47
4.7	Inner Loop Tuning Parameters: Input Constraints (Failure Model)	48
4.8	Outer Loop Tuning Parameters: Output Constraints	48
4.9	Outer Loop Tuning Parameters: Input Constraints	48

List of Figures

2.1	Relationship between the body-fixed reference frame F_B , flight-path reference frame F_W and stability reference frame F_S	5
3.1	Classification of Approaches to Reconfigurable Control	14
3.2	Multiple Model Switching and Tuning	16
3.3	Single Model vs. Multiple Model Adaptation	16
3.4	Control Allocation	20
3.5	Nonlinear Adaptive Output Feedback Controller	23
3.6	Block Diagram of the Error Dynamics	24
3.7	Model Reference Adaptive Control	31
4.1	Model Following Controller	38
4.2	MPC Reconfigurable Controller	45
4.3	MPC ‘Pilot’ Controller	45
4.4	Linear Nominal Model Outer Loop Trajectory Tracking	50
4.5	Linear Nominal Model: Pilot Input Commands	51
4.6	Linear Nominal Model: Scaled Actuator Commands and Physical Limits	52
4.7	Linear Nominal Model: Height and Ground Path	53
4.8	Linear Nominal Model: Inner Loop Tracking Performance	54
4.9	Linear Failure Model Outer Loop Trajectory Tracking	55
4.10	Linear Failure Model: Pilot Input Commands	56
4.11	Linear Failure Model: Scaled Actuator Commands and Physical Limits	57
4.12	Linear Failure Model: Height and Ground Path	58
4.13	Linear Failure Model: Inner Loop Tracking Performance	59
4.14	Nonlinear Nominal Model Outer Loop Trajectory Tracking	61
4.15	Nonlinear Nominal Model: Pilot Input Commands	62
4.16	Nonlinear Nominal Model: Scaled Actuator Commands and Physical Limits	63
4.17	Nonlinear Nominal Model: Height and Ground Path	64
4.18	Nonlinear Nominal Model: Inner Loop Tracking Performance	65

4.19	Nonlinear Failure Model Outer Loop Trajectory Tracking	66
4.20	Nonlinear Failure Model: Pilot Input Commands	67
4.21	Nonlinear Failure Model: Scaled Actuator Commands and Physical Limits . .	68
4.22	Nonlinear Failure Model: Height and Ground Path	69
4.23	Nonlinear Failure Model: Inner Loop Tracking Performance	70
4.24	Trajectory Visualization Tool	72
5.1	Preliminary Schedule	80
A.1	Step Response of Linear and Nonlinear Pilot Models	82
A.2	Engine Step Response of Linear, and Nonlinear Models	83
A.3	Aileron Step Response of Linear and Nonlinear Models	84
A.4	Elevator Step Response of Linear and Nonlinear Models	85
A.5	Rudder Step Response of Linear and Nonlinear Models	86
A.6	Engine Step Response of Linear, and Nonlinear Failure Models	87
A.7	Aileron Step Response of Linear and Nonlinear Failure Models	88
A.8	Elevator Step Response of Linear and Nonlinear Failure Models	89
A.9	Rudder Step Response of Linear and Nonlinear Failure Models	90

List of Acronyms

ANN	Artificial Neural Network
CA	Control Allocation
DEA	Direct Eigenstructure Assignment
DMC	Dynamic Matrix Control
EA	Eigenstructure Assignment
FCS	Flight Control System
FTC	Fault Tolerant Control
FDI	Fault Detection and Isolation
FMEA	Failure Modes and Effects Analysis
GWEA	Gain Weighted Eigenspace Assignment
GPC	Generalised Predictive Control
ICE	Innovative Control Effector
ICEM	Integrated Control Effector Management
IMM	Interacting Multiple Model
MIMO	Multiple Input Multiple Output
MPC	Model Predictive Control
MBPC	Model Based Predictive Control
MRAC	Model Reference Adaptive Control
MMST	Multiple Model Switching and Tuning
MMAE	Multiple Model Adaptive Estimation
PCA	Propulsion Controlled Aircraft
PID	Proportional Integral Derivative Controller
PIM	Pseudo Inverse Method
QP	Quadratic Program
RHC	Receding Horizon Control
RG	Reference Governor
SISO	Single Input Single Output
SOLO	Sequential Open-Loop Optimizing Control
SMC	Sliding Mode Control

TAFa Tailless Advanced Fighter Aircraft

List of Symbols

F_B	Body-fixed reference frame
F_S	Stability reference frame
F_W	Flight-path or wind reference frame
F_E	Earth-fixed reference frame
c.g.	Center of gravity
\mathbf{V}_B	Velocity with respect to F_B
\mathbf{V}_E	Velocity with respect to F_E
\mathbf{r}_E	Position of the c.g. with respect to F_E
$\boldsymbol{\omega}_B$	Angular velocity of the airplane relative to F_B
p	Roll rate
q	Pitch rate
r	Yaw rate
\mathbf{I}_B	Inertia matrix
\mathbf{W}_E	Wind velocity with respect to F_E
α	Angle of attack
β	Angle of sideslip
ψ	Roll (with respect to F_E)
θ	Pitch (with respect to F_E)
ϕ	Yaw (with respect to F_E)
L	Rolling moment
M	Pitching moment
N	Yawing moment
X	Axial force component
Y	Lateral force component
Z	Normal force component
δ_c	Control column position
δ_w	Control wheel position
δ_r	Rudder pedal position
δ_{stab}	Stabilizer handle position

δ_{sbh}	Speedbrake handle position
δ_{fh}	Flap handle position

Chapter 1

Introduction

1.1 Motivation

In 1989 a United Airlines DC-10, Flight 232 enroute from Denver to Minneapolis, sustained a catastrophic failure which cut through hydraulic lines from all three independent systems leaving the plane uncontrolled at 37,000 feet. In an unprecedented attempt, Captains Al Haynes and Dennis Fitch improvised a control method using only the throttles of the two wing engines and successfully brought the plane to a crash landing in Sioux City, Iowa, saving the lives of 184 of the 296 passengers. The Sioux City accident brought a great deal of interest to the emerging field of Fault Tolerant Control (FTC), which aims to develop controllers that can handle failures such as this.

FTC refers to methods which attempt to maintain control despite failures. [Bla99] defines FTC as “a system [which] is able to continue operation [after a failure]; a degradation of performance may be accepted.” Reconfigurable control is a class of FTC in which the controller takes explicit account of the failures in order to restore stability and sufficient performance. [Bla99] defines reconfigurable control, or active fault accommodation as “detection and isolation of a fault [which] leads to a change in the control operation to accommodate a fault. Actions can include, but are not limited to re-configuration, controller re-design, change of control path, or change of path or measurements.”

The obvious benefit of reconfigurable control to flight systems is the increase in survivability and safety. A secondary benefit comes as a result of one of the enabling technologies: Fault Detection and Isolation (FDI). FDI reduces the downtime and therefore the maintenance costs by using flight data to compute the most likely failed components. Despite these benefits, there are currently no aircraft, commercial or military, which use a reconfigurable controller as the aircraft industry is in general very conservative and is hesitant to use complex control techniques. Research into the field is active and ongoing, and reconfigurable control systems will likely be seen soon in unmanned vehicles followed by military aircraft

and then passenger jets.

Currently, most research into reconfigurable flight control focuses on autonomous aircraft or assumes that the controller will have full authority over the plane. This is unlikely to be the case in civil aircraft where one, or several pilots will be available. The question then becomes one of determining what is necessary to allow the pilot to continue flying the plane, and when this option is superior to giving full control of the plane to the computer.

1.2 Overview

The purpose of this report is to propose a topic in reconfigurable flight control suitable for a PhD research program. The mathematical details of aircraft and fault modelling will be given as well as a brief overview of Model Predictive Control (MPC) in Chapter 2. Chapter 3 provides a review of the current literature in the area of reconfigurable flight control. Chapter 4 gives an introduction to the problem of maintaining pilot control during aircraft failures. A motivating example of a crash which occurred in Amsterdam is studied, an MPC reconfigurable controller is designed and it is demonstrated that the pilot can continue to fly the plane. Finally, Chapter 5 presents an overview of proposed topics for the remainder of the degree.

Chapter 2

Preliminaries

This chapter provides the relevant background for reconfigurable flight control. Namely, aircraft modelling is reviewed, the basics of MPC are covered and the notation for the remainder of the report is standardized.

2.1 Aircraft Modelling

The first successful study of control and dynamics of an aircraft is credited to Bryan (1911) and Lanchester (1908) [Coo97]. The equations of motion have changed little in the last century, and the original formulations are still used. The following sections give an overview of the various reference frames used to describe an aircraft's state, provide an overview of the general nonlinear and linear equations of motion for an aircraft and review the various ways that failures can be modelled.

2.1.1 Reference Frames

There are a variety of reference frames used to describe aircraft movement and orientation. Some are suited to navigation (F_E) and some to control (F_B, F_S, F_W). The relevant frames are described in the following list: [Rau01]

Body-fixed reference frame F_B : This is a right-handed orthogonal reference system which has its origin, O_B , at the centre of gravity of the aircraft. The $X_B O_B Z_B$ plane coincides with the aircraft's plane of symmetry if it is symmetric, or it is located in a plane, approximating what would be the plane of symmetry if it is not. The X_B -axis is directed towards the nose of the aircraft, the Y_B -axis points to the right wing, and the Z_B -axis points towards the bottom of the aircraft.

Stability reference frame F_S : This is a special body-fixed reference frame, used in the study of small deviations from a nominal flight condition. The reference frames F_B and

F_S differ in the orientation of their respective X -axes. The X_S -axis is chosen parallel to the projection of the true airspeed vector \mathbf{V} on the $O_B X_B Z_B$ -plane (if the aircraft is symmetric this is the plane of symmetry), or parallel to \mathbf{V} itself in case of a symmetrical nominal flight condition. The Y_S -axis coincides with the Y_B -axis.

Flight-path or wind reference frame F_W : This reference frame, also called the wind reference frame, has its origin in the c.g. of the aircraft. The X_W -axis is aligned with the velocity vector of the aircraft and the Z_W -axis coincides with the Z_S -axis.

Earth-fixed reference frame F_E : This reference frame, also called the topodetic reference frame, is a right-handed orthogonal system which is considered to be fixed in space. Its origin can be placed at an arbitrary position, but will be chosen to coincide with the aircraft's centre of gravity at the start of a flight test manoeuvre. The Z_E -axis points downwards, parallel to the local direction of gravity. The X_E -axis is directed to the North, the Y_E -axis to the East.

The relationships between the various frames of reference are shown in Figure 2.1.

2.1.2 General Equations of Motion

Our goal in this section is to describe the position and orientation of the aircraft in appropriate reference frames, as well as the dynamics and relationships between the various frames.

The velocity of the aircraft is defined as follows:

$$\mathbf{V}_B = \begin{bmatrix} u & v & w \end{bmatrix}^T : \text{Velocity with respect to } F_B$$

$$\mathbf{V}_E = \begin{bmatrix} u_E & v_E & w_E \end{bmatrix}^T : \text{Velocity with respect to } F_E$$

$$\mathbf{W}_E = \begin{bmatrix} W_x & W_y & W_z \end{bmatrix}^T : \text{Wind velocity with respect to } F_E$$

The velocity \mathbf{V}_B does not include the effects of wind. If wind is to be explicitly included a superscript W will be used. $\mathbf{V}_B^W = \mathbf{V}_B + L_{BE}\mathbf{W}_E$, where $L_{BE} = L_{EB}^{-1}$ is the coordinate transform from F_E to F_B . L_{EB} is defined as

$$L_{EB} = \begin{bmatrix} \cos \theta \cos \psi & \sin \phi \sin \theta \cos \psi - \cos \phi \sin \psi & \cos \phi \sin \theta \cos \psi + \sin \phi \sin \psi \\ \cos \theta \sin \psi & \sin \phi \sin \theta \sin \psi + \cos \phi \cos \psi & \cos \phi \sin \theta \sin \psi - \sin \phi \cos \psi \\ -\sin \theta & \sin \phi \cos \theta & \cos \phi \cos \theta \end{bmatrix}$$

where θ, ψ, ϕ describes the orientation of the aircraft as defined below.

Two important relationships between the various reference frames can now be defined. As shown in Figure 2.1, the angle of attack, α , is the angle between the body-fixed axis and

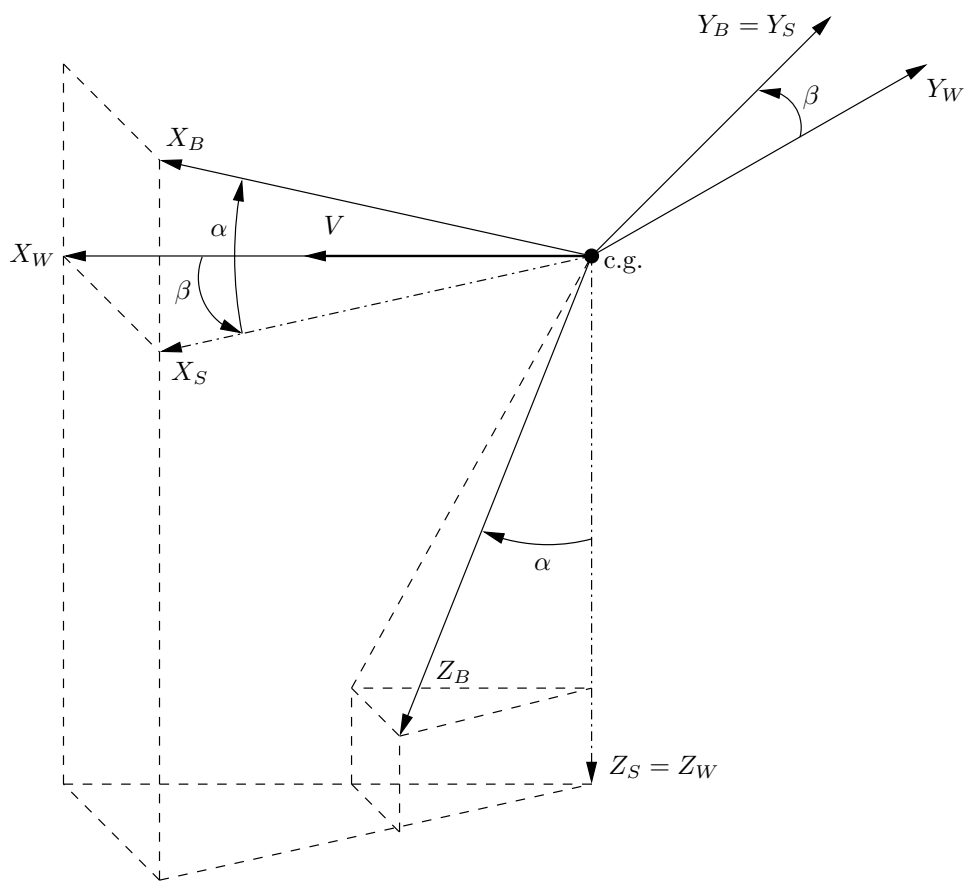


Figure 2.1: Relationship between the body-fixed reference frame F_B , flight-path reference frame F_W and stability reference frame F_S

the stability axis. The sideslip angle, β is the angle between the wind axis and the stability axis. These values can be computed as follows:

$$\alpha = \tan^{-1} \frac{w}{u} \quad (2.1)$$

$$\beta = \sin^{-1} \frac{v}{\sqrt{u^2 + v^2 + w^2}} \quad (2.2)$$

The position of the aircraft is most often used for navigation, and hence its dynamics are given in the earth-frame as follows:

$$\mathbf{r}_E = \begin{bmatrix} \dot{x}_E \\ \dot{y}_E \\ \dot{z}_E \end{bmatrix} = \mathbf{V}_E^W = L_{EB} \mathbf{V}_B^W \quad (2.3)$$

Note that z_E points into the ground and is therefore negative for positive height. As a result, the altitude or height is often used instead of z_E :

$$h = -z_E$$

The orientation of the aeroplane is defined relative to the earth axes and is given by the Euler angles (ψ, θ, ϕ) . The Euler angles define the rotations from the earth-fixed reference to the body-fixed axis. The ordering of the rotations is important and is done as follows:

1. Rotate ψ about $O_E Z_E$
2. Rotate θ about $O_E Y_E$
3. Rotate ϕ about $O_E Z_E$

Euler angles are not unique, and so in order to avoid ambiguities, the following ranges are defined:

$$\begin{aligned} -\pi \leq \psi < \pi \quad \text{or} \quad 0 \leq \psi < 2\pi \\ -\frac{\pi}{2} \leq \theta < \frac{\pi}{2} \\ -\pi \leq \phi < \pi \quad \text{or} \quad 0 \leq \phi < 2\pi \end{aligned} \quad (2.4)$$

The angular rotation rates are defined relative to the body-fixed inertial frame F_B as $\boldsymbol{\omega}_B = [p \ q \ r]^T$, where p is the roll rate, q is the pitch rate, and r is the yaw rate. The angular rates are related to the rate of change of the Euler angles by the following coordinate transforms:

$$\begin{bmatrix} p \\ q \\ r \end{bmatrix} = \begin{bmatrix} 1 & 0 & -\sin \theta \\ 0 & \cos \phi & \sin \phi \cos \theta \\ 0 & -\sin \phi & \cos \phi \cos \theta \end{bmatrix} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \quad (2.5)$$

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 1 & \sin \phi \tan \theta & \cos \phi \tan \theta \\ 0 & \cos \phi & -\sin \phi \\ 0 & \sin \phi \sec \theta & \cos \phi \sec \theta \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix} \quad (2.6)$$

Note that when perturbations are small, such that (ϕ, θ, ψ) may be treated as small angles, Equations 2.5 and 2.6 may be approximated by

$$\begin{bmatrix} p \\ q \\ r \end{bmatrix} = \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix}. \quad (2.7)$$

We are now ready to compute the dynamic equations of the aircraft. The dynamics are built from basic Newtonian dynamics of a rigid body, as given by the general force and moment equations:

$$\begin{aligned} \mathbf{F} &= m \left(\frac{\partial \mathbf{V}}{\partial t} + \boldsymbol{\Omega} \times \mathbf{V} \right) \\ \mathbf{M} &= \frac{\partial (\mathbf{I}\boldsymbol{\Omega})}{\partial t} + \boldsymbol{\Omega} \times (\mathbf{I}\boldsymbol{\Omega}), \end{aligned} \quad (2.8)$$

where \mathbf{V} is linear velocity, $\boldsymbol{\Omega}$ is angular velocity and \mathbf{I} is the moment of inertia. Equation (2.8) written in terms of the variables defined in this section is:

$$\begin{aligned} X - mg \sin \theta &= m(\dot{u}^W + qw^W - rv^W) \\ Y + mg \cos \theta \sin \phi &= m(\dot{v}^W + ru^W - pw^W) \\ Z + mg \cos \theta \cos \phi &= m(\dot{w}^W + pv^W - qu^W) \end{aligned} \quad (2.9)$$

$$\begin{aligned} L &= I_x \dot{p} - I_{yz}(q^2 - r^2) - L_{zx}(\dot{r} + pq) - I_{xy}(\dot{q} - rp) - (I_y - I_z)qr \\ M &= I_y \dot{q} - I_{zx}(r^2 - p^2) - I_{xy}(\dot{p} + qr) - I_{yz}(\dot{r} - pq) - (I_z - I_x)rp \\ N &= I_z \dot{r} - I_{xy}(p^2 - q^2) - I_{yz}(\dot{q} + rp) - I_{zx}(\dot{p} - qr) - (I_x - I_y)pq, \end{aligned} \quad (2.10)$$

where $\mathbf{I}_B = \begin{bmatrix} I_x & -I_{xy} & -I_{xz} \\ -I_{yx} & I_y & -I_{yz} \\ -I_{zx} & -I_{zy} & I_z \end{bmatrix}$ is the inertia, $[X, Y, Z]$ and $[L, M, N]$ are the forces and moments acting on the aircraft respectively. These forces and moments are functions of the control surfaces, thrust, and drag. They can be written as functions of the six linear and angular velocities (u, v, w, p, q, r) and the actuator positions, which is usually given by the aileron, elevator, rudder and throttle positions.

In some cases it is easier to use the true velocity, $V = |\mathbf{V}_B|$, sideslip angle β and angle of attack α rather than the body-fixed velocities (u, v, w) . Equation 2.9 can then be replaced by

$$\begin{aligned} \dot{V} &= \frac{1}{m}(X \cos \alpha \cos \beta + Y \sin \beta + Z \sin \alpha \sin \beta) \\ \dot{\alpha} &= \frac{1}{V \cos \beta} \left\{ \frac{1}{m}(-X \sin \alpha + Z \cos \alpha) \right\} + q - (p \cos \alpha + r \sin \alpha) \tan \beta \\ \dot{\beta} &= \frac{1}{V} \left\{ \frac{1}{m}(-X \cos \alpha \sin \beta + Y \cos \beta - Z \sin \alpha \sin \beta) \right\} + p \sin \alpha - r \cos \alpha. \end{aligned} \quad (2.11)$$

Equation 2.9 or 2.11 can now be combined with Equations 2.10 and 2.3 to form the full nonlinear dynamics of the aeroplane.

Other commonly used variables, which can be computed directly from the aircraft's state, are the azimuth or heading angle, χ , and the flight path angle, γ . These are defined as

$$\begin{aligned}\chi &= \beta + \psi \\ \gamma &= \sin^{-1}\left(\frac{h}{V}\right) = \theta - \alpha.\end{aligned}\tag{2.12}$$

2.1.3 Linearized Equations of Motion

The equations of motion developed in the previous section are nonlinear. If it is assumed that the motion of the aeroplane consists of small deviations from a reference condition of steady flight, then a linearized model around the trim condition can be obtained.

Steady flight can be defined, for example, as one of the following:

$$\begin{aligned}\textit{steady wings-level flight:} & \quad \phi, \dot{\phi}, \dot{\theta}, \dot{\psi} = 0, \\ \textit{steady turning flight:} & \quad \dot{\phi}, \dot{\theta} = 0, \quad \dot{\psi} = \text{turn rate} \\ \textit{steady pull-up:} & \quad \phi, \dot{\phi}, \dot{\psi} = 0, \quad \dot{\theta} = \text{pull-up rate} \\ \textit{steady roll:} & \quad \dot{\theta}, \dot{\psi} = 0, \quad \dot{\phi} = \text{roll rate},\end{aligned}$$

where $\dot{p}, \dot{q}, \dot{r}, \dot{V}, \dot{\alpha}, \dot{\beta} = 0$ and all control surface inputs are zero.

There are three methods of computing a linear model for small perturbations around the trim or steady-state condition. The first is to replace all nonlinearities in the general dynamic equations with their first order Taylor series approximations (see [ER96] for such a development). The second is to compute numerically the effect of small changes in state variables and inputs on the state derivatives. This can be done, for example, by using a Matlab linearization routine such as `linmod` on a nonlinear Simulink model. The final method is to run an identification algorithm using data collected either from a nonlinear model or from the physical aeroplane. See, for example, [AW95, Chapter 2] for details.

The result of the linearization is that we can write the model in standard state-space format:

$$\begin{aligned}\dot{x} &= Ax + Bu \\ y &= Cx.\end{aligned}\tag{2.13}$$

2.1.4 Fault Models

The goal of this section is describe the types of failures that can occur on an aircraft, which are relatively limited when compared to a general plant. Aircraft failures can be grouped into three categories: sensor, actuator and structural failures, or some combination of the three. Table 2.1 below describes the general failures that can occur and their classification into these three categories.

Large planes, and passenger jets in particular, have triply redundant sensor systems and so the likelihood of a failure is extremely small. A sensor failure by itself does not necessarily

pose a threat to a large plane, as there is always a pilot on board with his own set of backup sensors. Sensor failures can become significant in autonomous craft. However, as many authors have shown ([CFS⁺01, May99, WCNF02, CFNS02]), any single failed sensor can be recreated from the remaining working ones, assuming the aerodynamics of the craft have not changed. For these reasons sensor failures are not considered in this work.

Structural or aerodynamic failures affect the equations of motion and are caused by physical damage. The structure of the model remains the same during such a failure, as the equations of motion are completely general. However, the constants governing the behaviour can change. Actuator failures are critical as they change the structure of plant that the controller was designed for. For this reason, the loss of an actuator is the primary focus for many FTC methods as will be seen in Chapter 3.

Sensor	Actuator	Structural	Failure	Effect
✓			Sensor loss	Minor if it is the only failure
	✓		Partial hydraulics loss	Maximum rate decrease on several control surfaces
	✓		Full hydraulics loss	One or more control surfaces become stuck at last position for hydraulic driven aircraft, or float on light aircraft
	✓		Control loss on one or more actuators due to internal fault (not external damage)	One or more control surfaces become stuck at last position
	✓	✓	Loss of part/all of control surface	Effectiveness of control surface is reduced, but rate is not; minor change in the aerodynamics
	✓	✓	Loss of engine	Large change in possible operating region; significant change in the aerodynamics
		✓	Damage to aircraft surface	Possible change in operating region; Significant change in aerodynamics

Table 2.1: Aircraft Failure Modes

2.2 Model Predictive Control (MPC)

Predictive control has been known by many names over the years: Model Based Predictive Control (MBPC), Receding Horizon Control (RHC), Generalised Predictive Control (GPC),

Dynamic Matrix Control (DMC) and Sequential Open-Loop Optimizing Control (SOLO), among others. There are three key aspects of MPC which differentiate it from other methods:

- an explicit internal model is used to make predictions about the future behaviour of the system
- the future control trajectory is chosen to optimise some aspect of the predicted behaviour
- only the first step of the control trajectory is implemented and the cycle of prediction and optimisation is repeated at each time step

A `Matlab` MPC toolbox has been created by the author in order to provide a tool with which to implement the algorithms introduced in Chapter 4. This section will briefly review the implementation and some of the issues in MPC. As the implementation currently accepts only linear models, this development will assume a linear internal model and plant. This Section is heavily based on the text [Mac00].

Consider the linear system

$$\begin{aligned} x(k+1) &= Ax(k) + Bu(k) \\ y(k) &= Cx(k), \end{aligned} \tag{2.14}$$

where $x \in \mathbb{R}^n$, $y \in \mathbb{R}^m$ and $u \in \mathbb{R}^l$. We assume constraints on the inputs, input rates and outputs. Handling input rate constraints requires that the inputs be included in the state vector and so we define a new state $\xi(k) = \begin{bmatrix} x(k) \\ u(k-1) \end{bmatrix}$, which gives the state equations

$$\begin{aligned} \xi(k+1) &= \begin{bmatrix} A & B \\ 0 & I \end{bmatrix} \xi(k) + \begin{bmatrix} B \\ I \end{bmatrix} \Delta u(k) \\ y(k) &= \begin{bmatrix} C & 0 \end{bmatrix} \xi(k), \end{aligned} \tag{2.15}$$

where $\Delta u(k) = u(k) - u(k-1)$.

At each time step, k , the MPC algorithm makes predictions up to the prediction horizon, H_p steps ahead, and chooses control inputs up to the control horizon, H_u steps ahead, such that all constraints are satisfied for all H_p steps. To simplify notation, we define

$$\mathcal{X}(k) = \begin{bmatrix} \hat{\xi}(k+1 | k) \\ \vdots \\ \hat{\xi}(k+H_p | k) \end{bmatrix}, \quad \mathcal{T}(k) = \begin{bmatrix} \hat{r}(k+1 | k) \\ \vdots \\ \hat{r}(k+H_p | k) \end{bmatrix}, \quad \Delta \mathcal{U}(k) = \begin{bmatrix} \Delta \hat{u}(k | k) \\ \vdots \\ \Delta \hat{u}(k+H_u-1 | k) \end{bmatrix}$$

as the estimated future state, reference trajectory and input changes respectively. Assuming full state measurement, the future states \mathcal{X} can be written as a function of the current state and the future inputs

$$\mathcal{X}(k) = \Phi \xi(k) + \Theta \Delta \mathcal{U}(k), \tag{2.16}$$

where Φ and Θ are defined appropriately (see [Mac00, Page 62] for a possible formulation).

We assume the following linear constraints over the system inputs, input rates and outputs for all points up to and including the prediction horizon:

$$\begin{aligned} \Delta u &\in \Delta \mathbf{U} = \{\Delta u \in \mathbb{R}^m \mid E\Delta u \leq e\} \\ u &\in \mathbf{U} = \{u \in \mathbb{R}^m \mid Fu \leq f\} \\ y &\in \mathbf{Y} = \{y \in \mathbb{R}^l \mid Gy \leq g\}. \end{aligned} \quad (2.17)$$

With a slight abuse of notation, Equations (2.17) can be written as

$$E\Delta \mathcal{U} \leq e \quad (2.18)$$

$$F\mathcal{U} \leq f \quad (2.19)$$

$$GC\mathcal{X} \leq g. \quad (2.20)$$

Equations (2.19) and (2.20) can then be written in terms of the current state and the future input changes \mathcal{U} . Let F_i be the i^{th} column of F . Define $\mathcal{F}_i = \sum_{j=i}^{H_u} F_j$ and $\mathcal{F} = [\mathcal{F}_1, \dots, \mathcal{F}_{H_u}]$. Then Equation (2.19) can be written as

$$\mathcal{F}\Delta \mathcal{U} = \begin{bmatrix} 0 & \mathcal{F}_1 \end{bmatrix} \xi - f. \quad (2.21)$$

Using Equation (2.16), (2.20) can be written as

$$\begin{aligned} GC(\Phi\xi + \Theta\Delta \mathcal{U}) &\leq g \\ GC\Theta\Delta \mathcal{U} &\leq g - GC\Phi\xi \end{aligned} \quad (2.22)$$

Define $\Pi(k)$ as the set of sequences

$$\left[\Delta \hat{u}(k \mid k), \dots, \Delta \hat{u}(k + H_u \mid k) \right]^T \subseteq \mathbb{R}^{m \cdot H_u},$$

such that the constraints (2.17) are satisfied for all $k \in [k, k + H_p]$. Combining Equations (2.17), (2.21) and (2.22), the admissible set can be written as a polytope of $\Delta \mathcal{U}$ given the current state:

$$\Pi(k) = \left\{ \Delta U \in \mathbb{R}^{m \cdot H_p} \left[\begin{array}{c} E \\ \mathcal{F} \\ GC\Theta \end{array} \right] \Delta U \leq \begin{bmatrix} e \\ -f \\ g \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & \mathcal{F}_1 \\ GC\Phi & 0 \end{bmatrix} \xi(k) \right\}. \quad (2.23)$$

The set of admissible control actions which can be taken, $\pi(k)$, are now given by projecting onto the first m dimensions in the set $\Pi(k)$:

$$\pi(k) = \left\{ \alpha \in \mathbb{R}^m \mid \exists \beta \in \mathbb{R}^{m \cdot (H_p - 1)}, [\alpha \ \beta] \in \Pi(k) \right\}. \quad (2.24)$$

A particular control input needs to be chosen from the set $\pi(k)$ at each time step. This can be done by computing the solution to the Quadratic Program (QP)

$$V(k) = \|C\mathcal{X} - \mathcal{T}\|_Q^2 + \|\Delta \mathcal{U}\|_R^2 \quad (2.25)$$

subject to $\Delta\mathcal{U} \in \Pi(k)$,

where Q and R are appropriate weighting matrices. Let $\epsilon(k) = \mathcal{T}(k) - C\Phi\xi(k)$. Then the quadratic program (2.25) can be written as

$$\begin{aligned} V(k) &= \|C\Theta\Delta\mathcal{U} - \epsilon(k)\|_Q^2 + \|\Delta\mathcal{U}\|_R^2 \\ &= \Delta\mathcal{U}^T (\Theta^T C^T Q C \Theta + R) \Delta\mathcal{U} - 2\Delta\mathcal{U}^T \Theta^T C^T Q \epsilon + \epsilon^T Q \epsilon, \end{aligned}$$

which is in the standard form for QP solvers.

The algorithm described here has been implemented in `Matlab` and the toolbox is described in Appendix B.

Chapter 3

Reconfigurable Flight Control A Survey

3.1 Introduction

Many methods have been proposed to solve the problem of fault tolerant control as described in Chapter 2. As shown in Figure 3.1 they fall into two main categories: active and passive.

Passive methods are essentially robust control techniques which are suitable for certain types of structural failures that can be modelled as uncertainty regions around a nominal model. Any failure which doesn't push the system outside of the stability radius given by the robust controller will still have satisfactory stability and performance guarantees. However, any controller with a large enough stability radius to encompass most failure situations will likely be unnecessarily conservative and there is no guarantee that unanticipated or multiple failures could be handled or even that such a controller exists. There are also many types of common failures, such as actuator or sensor faults, which cannot be adequately modelled as uncertainty. These problems motivate the need for a controller which more directly addresses the situation.

The active, or reconfigurable methods differentiate themselves from passive approaches in that they take fault information explicitly into account and do not assume a static nominal model. Reconfigurable flight control is for the most part still an academic notion. Although there have been very few controllers implemented on physical systems and none on commercial aircraft, over the last 20 years several research programs have been formed to investigate their potential and as a result there are a variety of active methods. The following sections give an overview of each approach.

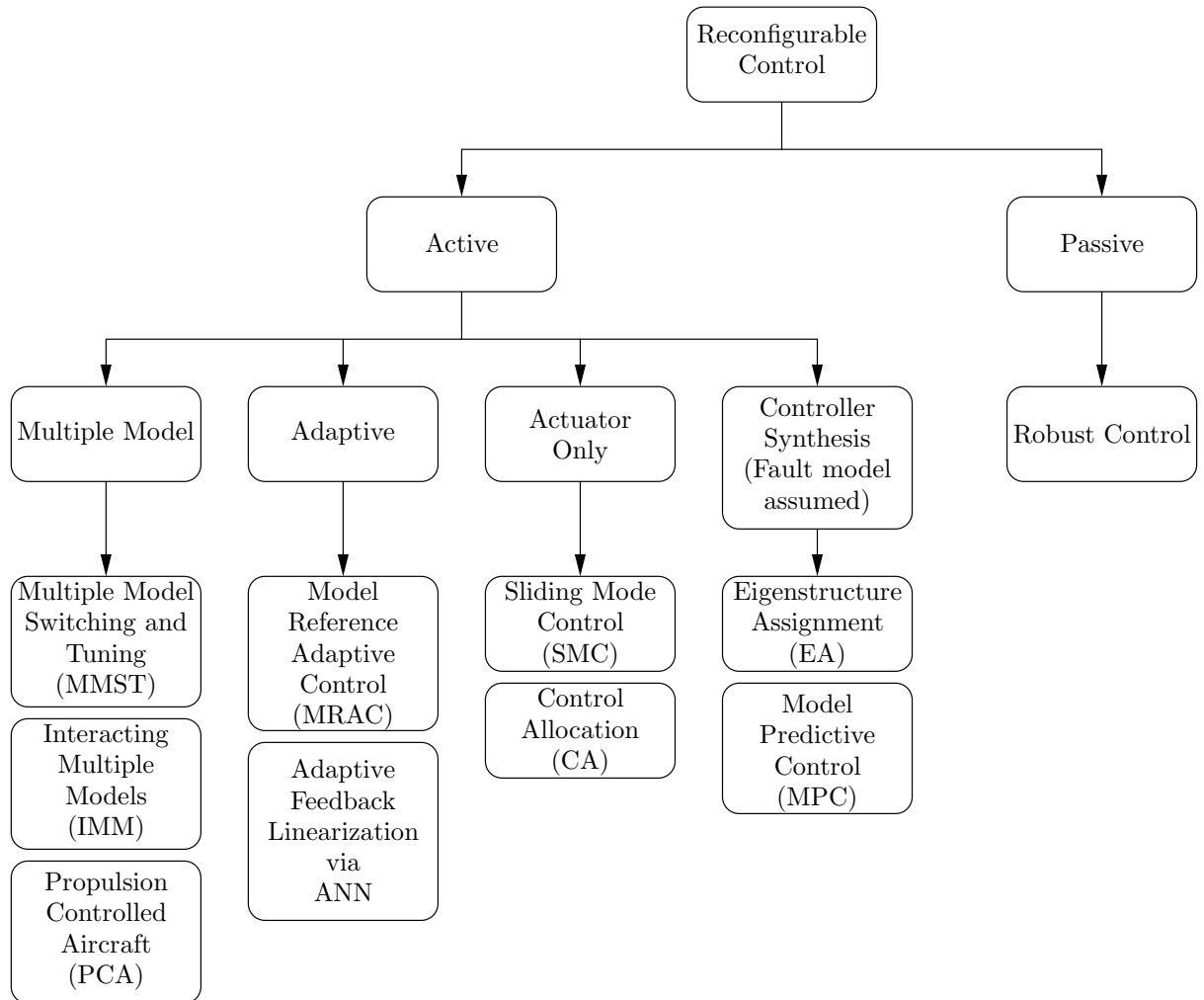


Figure 3.1: Classification of Approaches to Reconfigurable Control

3.2 Multiple Model Control

There are three flavours of reconfigurable control that fall under the heading of multiple model control: Multiple Model Switching and Tuning (MMST), Interacting Multiple Model (IMM) and Propulsion Controlled Aircraft (PCA). In the first two cases all expected failure scenarios are enumerated during a Failure Modes and Effects Analysis (FMEA) and fault models constructed which cover each situation. When a failure occurs MMST switches to a pre-computed control law corresponding to the current failure situation. Rather than using the model which is closest to the current failure scenario, IMM computes a fault model as a convex combination of all pre-computed fault models and then uses this new model to make control decisions. PCA is a special case of MMST, where the only anticipated fault is a total hydraulics failure, and in this case only the engines are used for control. The following sections discuss these three approaches.

3.2.1 Multiple Model Switching and Tuning (MMST)

Although the idea of multiple model control has been around for many years, it has seen some interest in the reconfigurable control literature in the last few years [BM98, GBMR98, BM99, BLM00a, BLM00b, BLM01, KV00a, Dem01]. In MMST, the dynamics of each fault scenario is described by a different model. These models are referred to as the identification models [BM98] and are setup in parallel, with each one having a corresponding controller as shown in Figure 3.2. The problem then becomes one of choosing which model/controller pair to switch to at each time instant.

Figure 3.3 helps to motivate the use of MMST in reconfigurable control systems. During a failure the plant is assumed to move from some nominal model P_0 to a failure model P_f some distance away in parameter space. The top half of the figure shows an adaptive control scheme which is using only a single model, and the lower a MMST method. As can be seen, for certain plants, the MMST would converge to the correct fault model faster than a single model approach.

We are considering systems of the form

$$P = \begin{cases} \dot{x} &= A_0(p(t))x + B_0(p(t))u \\ y &= C_0(p(t))x \end{cases} \quad (3.1)$$

where $x \in \mathbb{R}^n$, $u \in \mathbb{R}^m$, $y \in \mathbb{R}^k$, $A_0 \in \mathbb{R}^{n \times n}$, $B_0 \in \mathbb{R}^{n \times m}$, $C_0 \in \mathbb{R}^{k \times n}$ and $p(t) \in \mathcal{S} \subseteq \mathbb{R}^l$ are the plant parameters. $p(t)$ varies in time in an abrupt fashion and represents the various failure scenarios.

Definition 3.2.1 (Model Set) *The model set \mathcal{M} is a set of N linear models*

$$\mathcal{M} : \{M_1, \dots, M_N\}$$

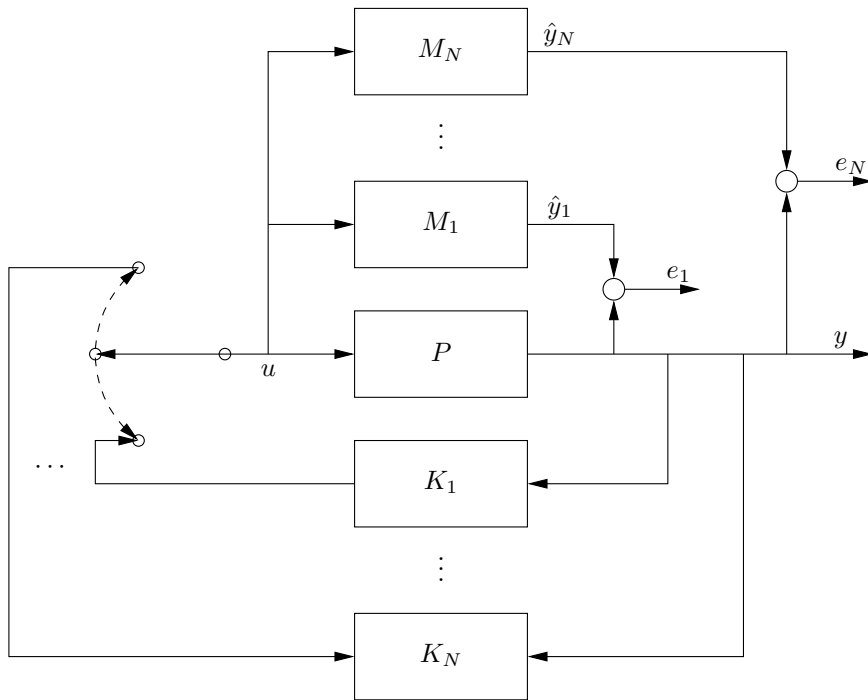


Figure 3.2: Multiple Model Switching and Tuning

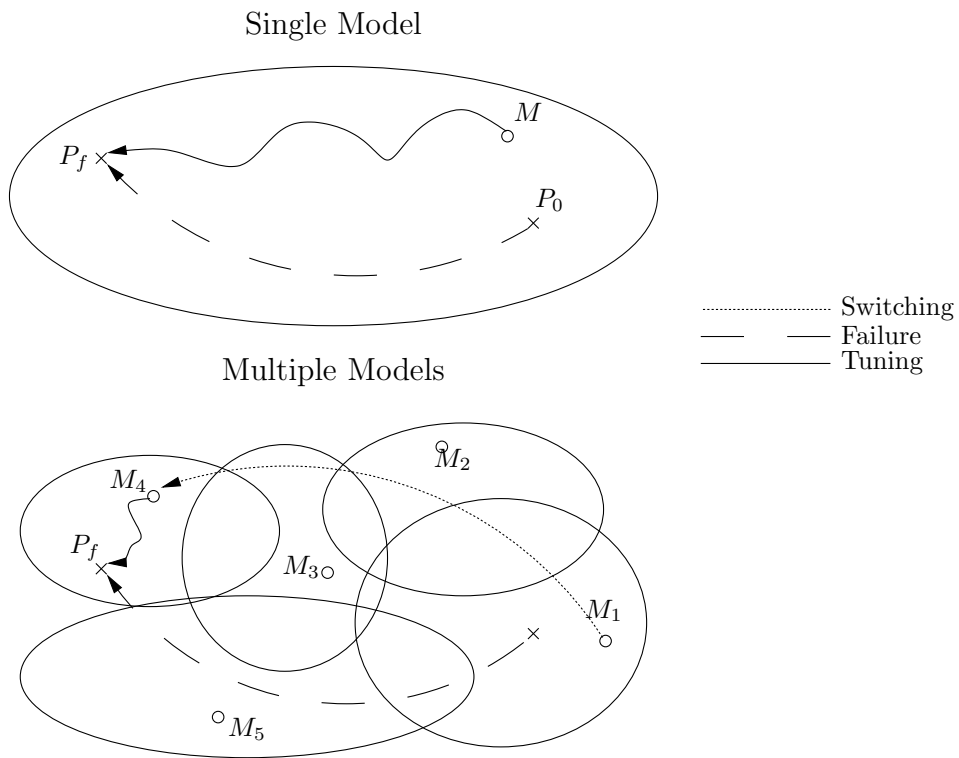


Figure 3.3: Single Model vs. Multiple Model Adaptation

such that

$$M_i : \begin{cases} \dot{x}_i &= A_i x_i + B_i u \\ y_i &= C_i x_i \end{cases}$$

where model M_i corresponds to a particular set of parameters $p_i \in \mathcal{S}$.

A stabilizing controller K_i is designed for each model $M_i \in \mathcal{M}$.

The control law proceeds as follows. At each time step, the model which is closest to the current system is determined by computing a performance index $J_i(t)$, which is a function of the errors $e_i(t)$ between the estimated outputs of model M_i and the measurements at time t . A commonly used index is [NB97]

$$J_i(t) = \alpha e_i^2(t) + \beta \int_0^t e^{-\lambda(t-\tau)} e_i^2(\tau) d\tau$$

$$\alpha \geq 0, \beta > 0, \lambda > 0$$

where α and β are chosen to give a desired combination of instantaneous and long-term accuracy measures. The forgetting factor λ ensures the boundedness of $J_i(t)$ for bounded e_i . The model/controller, M_i/K_i with the smallest index is switched to and a waiting period of $T_{min} > 0$ is allowed to pass in order to prevent arbitrarily fast switching. Most MMST algorithms include a ‘tuning’ part which occurs during the period while a controller K_i is active, during which time the parameters of the corresponding model, and only the corresponding model M_i , are being updated using an appropriate identification technique (e.g. [AW95, Chapter 2]).

Recent interest in this approach arises from the following stability result:

Theorem 3.2.2 [NB97] *Consider the switching and tuning system described above, where the N models are all fixed and the proposed switching scheme is used with $\beta, \lambda, T_{min} > 0$, and $\alpha \geq 0$. Then, for each plant with parameter vector $p \in \mathcal{S}$, there is a positive number T_S and a function $\mu_{\mathcal{S}}(p, T_{min}) > 0$, such that if:*

- the waiting time $T_{min} \in (0, T_S)$
- there is at least one model M_i with parameter error $\|\hat{p}_i - p\| < \mu_{\mathcal{S}}(p, T_{min})$

then all the signals in the overall system, as well as the performance indexes $\{J_i(t)\}$, are uniformly bounded. Here T_S depends only upon \mathcal{S} , and $\mu_{\mathcal{S}}$ also depends upon α, β, λ and \mathcal{S} .

In essence, Theorem 3.2.2 states that the MMST system is stable if the set of models M_i is dense enough in the parameter space \mathcal{S} and the sampling rate T_{min} is fast enough. How dense and how fast depend on the particular system and Theorem 3.2.2 gives no insight into the selection of \mathcal{M} or T_{min} .

Despite the limitations of Theorem 3.2.2, there are several papers which have applied these methods. In [BM98, BLM00a, BLM00b, BLM01] a MMST controller is developed for the

highly over-actuated tailless advanced fighter aircraft (TAFAs). 11 fault models are required to cover the scenario of right wing damage ranging from 0% to 100% and a switching interval of 25ms is needed for stability. Clearly, this approach will not scale well to the situation where more than one failure, or multiple failures are considered. [BM99] describes a MMST scheme which can handle locked, floating, hard-over or loss of effectiveness actuator failures for an F-18 aircraft carrier landing manoeuvre. Only five models are needed for satisfactory performance, but again, multiple failures cannot be accommodated. [BM98] introduces a new method of failure parameterizations for jammed actuators, enabling multiple complete failures of control surfaces for an F-18 to be handled using a large number of simple models.

For systems with relatively few and well understood failure modes, multiple model switching and tuning has advantages in being fast and provably stable. However, the main limitation is that there may be failure scenarios that were not modelled, which would likely be the case for multiple or structural failures. A severe limitation for larger systems is that the number of models required increases exponentially with the number of simultaneous failures considered.

3.2.2 Interacting Multiple Models (IMM)

The method of interacting multiple models (IMM) attempts to deal with the key limitation of MMST, namely that every fault scenario must be modelled, by considering fault models which are convex combinations of models in a model set.

The primary assumption of IMM is that every possible failure can be modelled as a convex combination of models in a pre-determined model set \mathcal{M} as defined above in Definition 3.2.1

$$M_f = \sum_{i=1}^N \mu_i M_i = \mu^T \begin{bmatrix} M_1 \\ \vdots \\ M_N \end{bmatrix}, \quad M_i \in \mathcal{M}, \quad \mu_i > 0 \in \mathbb{R}, \quad \sum_{i=1}^N \mu_i = 1, \quad (3.2)$$

Then M_f is the system:

$$M_f : \begin{cases} \dot{x} = \begin{bmatrix} A_1 & 0 & \dots & 0 \\ 0 & A_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & A_n \end{bmatrix} x + \begin{bmatrix} B_1 \\ B_2 \\ \vdots \\ B_N \end{bmatrix} u \\ y = \begin{bmatrix} \mu_1 C_1 & \mu_2 C_2 & \dots & \mu_N C_N \end{bmatrix} x \end{cases} \quad (3.3)$$

It is still an open question how to choose this model set or when the assumption that the failure model can be written as a convex combination of the models in the set is valid.

Fault detection and modelling is then done online by identifying the variables μ_i in Equation (3.2). Two proposed methods exist for computing the coefficients μ . In the first, a Kalman filter is designed for each $M_i \in \mathcal{M}$ and all filters are run in parallel. The probability

that each of these models represents the true state of the system can be computed and the coefficients μ are set to these probabilities. This method is named Multiple Model Adaptive Estimation (MMAE) and is used in [May99, ZJ99]. In the second approach, the previous k_f time instants are considered and the estimated output at each point is computed as a function of μ , which is then selected to minimize this difference. This approach is advocated in [KV00b, KVN01].

Once a fault model has been identified, there are a variety of methods for control law calculation. [KV00b] and [KVN01] suggest a MPC scheme where the minimization of the past tracking error, and therefore of μ , is included in the cost function. [ZJ99] proposes an Eigenstructure Assignment (EA) (see Section 3.6) method and [May99] uses a fixed controller, using the fault model M_f only for state estimation.

IMM is attractive in its ability to handle multiple failure scenarios by combining single failure models. However, the requirement of finding the coefficients μ after a failure makes this an adaptive algorithm and not a model-switching one. As a result it loses some of the speed of the MMST approach. The formulation of IMM as an MPC problem given in [KVN01] also offers the potential of handling actuator constraints naturally as discussed in Section 3.8.

3.2.3 Propulsion Controlled Aircraft (PCA)

After the possibility of control using only the engine throttles was demonstrated by the Sioux City accident, and following a recommendation from the National Transportation Safety Board of America, the PCA problem was taken up by the NASA Dryden Flight Research Center [BBMB97, BBMF97] in order to provide a backup in case of total hydraulics failure. PCA is a specific instance of a multi-model approach where the fault model is identical to the nominal one, but in which all control surfaces are free floating. In 1995, a demonstration was made during which a MD-11 and a F-15 recovered from a complete hydraulic failure and landed successfully under propulsion-only control [BB97]. PCA is a useful and important idea and solves a very practical problem. However, it clearly is not sufficient to solve the general reconfigurable control problem.

3.3 Control Allocation (CA)

Control allocation is the problem of producing a desired set of forces and moments from a (usually large) set of actuators. For example, as shown in Figure 3.4, the output of the control law can be a set of desired moments and the job of the control allocation block is then to select appropriate setpoints for the actuators which will produce those moments.

The purpose of control allocation in reconfigurable control is to allow the design of control

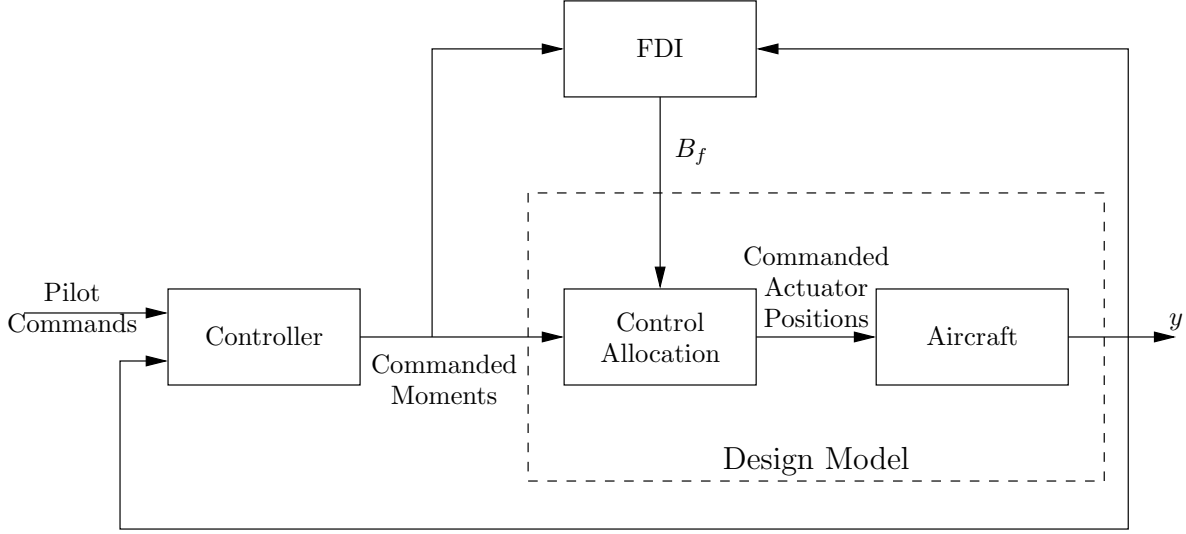


Figure 3.4: Control Allocation

laws which do not consider actuator failures. By assuming the existence of a control allocation algorithm, a control law can be designed for a set of *virtual actuators* which produce the desired forces and moments, regardless of the failure state of the system.

The control allocation algorithm takes as inputs the desired moments and an estimation of the input derivatives (B_f matrix) from either a FDI or a system identification algorithm. The goal is then to produce the desired moments u_d by selecting the appropriate inputs to the system u . Whether this can be done depends on the difference between the size of $u_d \in \mathbb{R}^m$ and the column rank of $B_f \in \mathbb{R}^{n \times k}$. There are three cases to consider:

$m < k$: The moments can be selected exactly and the remaining degrees of freedom can be used (for example) to drive the actuators towards a desired position u_p by minimizing [WBC⁺99, BW01, CLS98]:

$$\frac{1}{2} \|u - u_p\|_{W_p} = \frac{1}{2} (u - u_p)^T W_p (u - u_p) \text{ where } W_p = W_p^T > 0$$

subject to $Bu = u_d$

where W_p is a weighting matrix prioritizing critical actuators.

$m = k$: There is only one solution which places the moments exactly

$$u = B^{-1}u_d$$

$m > k$: There are not enough degrees of freedom to achieve u_d and so a compromise must be made by (for example) minimizing the weighted norm

$$\frac{1}{2} \|Bu - u_d\|_{W_d}$$

Control allocation has been heavily studied in relation to over-actuated systems (see [Enn98] for a survey) and has received a great deal of attention in the literature for reconfigurable systems as it allows actuator failures to be handled without the need to modify the control law. However, there are two major limitations to this approach to reconfiguration. First, the system will not necessarily be stable, even with a stabilizing control law, when $m > k$, as the input seen by the system may not be equal to that intended by the controller. Second, the dynamics and limitations of the actuators after a failure are not taken into account in the control law. This means that the controller will still be attempting to achieve the original system performance even though the actuators are not capable of achieving it.

Extensions to the simple control allocation problem presented here have been considered in the literature. In [BD95] and [DB96] the problem of control allocation with magnitude and rate limits on the actuators is considered, [DLB01] develops a Control Allocation (CA) controller for the extremely over-actuated Innovative Control Effector (ICE) aircraft and [ZHZ99] looks at restoring as much of the performance of the original B matrix as possible after an actuator failure.

3.4 Adaptive Feedback Linearization via Artificial Neural Network (ANN)

This section examines a method primarily developed by Calise *et al* [IJCK01, JC01, IJC01, CHI01, CLS00, WBC⁺99, CLS98] involving a Model Reference Adaptive Control (MRAC) scheme through adaptive feedback linearization augmented by an Artificial Neural Network (ANN). This approach has been successfully demonstrated via simulation on the Tailless Advanced Fighter Aircraft (TAFA) [WBC⁺99, CLS98] and the X-36 [CLS00]. The approach presented here splits the dynamics of the plane into three SISO subsystems, each of which has a model reference adaptive controller: roll, pitch and yaw. The output of each controller is a command specifying a desired roll, pitch or yaw moment and it is then the job of the Integrated Control Effector Management (ICEM) [BW01, WBC⁺99], a form of control allocation, to generate these moments using the available control surfaces. In the next three sections, we first give a brief overview of the principles of feedback linearization on SISO systems, review the particulars and benefits of its use in reconfiguration and finally go over the ICEM and its role in the proposed method.

3.4.1 SISO Feedback Linearization

Consider the SISO nonlinear system:

$$\begin{aligned} \dot{x} &= f(x, u) \\ y &= h(x) \end{aligned} \quad x \in \mathbb{R}^n, u, y \in \mathbb{R} \quad (3.4)$$

In feedback linearization the goal is to design a control law for the SISO nonlinear system given in Equation 3.4 such that the closed loop system is linear and controllable. Assuming the relative degree of h is $r = n$, the r^{th} derivative of the output is the first derivative that is directly affected by the control. As a result, we can write the system dynamics in the normal form [Isi89, Section 4.2]:

$$\begin{aligned}
\Phi_1(x) &= h(x) = z_1 = y \\
\Phi_2(x) &= \frac{dh(x)}{dt} = \dot{z}_1 = z_2 \\
\Phi_3(x) &= \frac{d^2h(x)}{dt^2} = \dot{z}_2 = z_3 \\
&\vdots && \vdots && \vdots \\
\Phi_r(x) &= \frac{d^r h(x)}{dt^r} = \dot{z}_{r-1} = z_r \\
&&& \dot{z}_r = h_r(z, u)
\end{aligned} \tag{3.5}$$

where $\Phi(x) = z = [z_1, \dots, z_r]'$.

We now define the ‘pseudo control signal’ ν

$$\nu = \hat{h}_r(\Phi(x), u)$$

where $\hat{h}_r(\Phi(x), u)$ is an invertible estimate of $h_r(z, u)$. Then the system dynamics can be expressed as

$$\begin{aligned}
\dot{z}_i &= z_{i+1}, \quad 1 \leq i \leq r-1 \\
\dot{z}_r &= \nu + \Delta \\
y &= z_1
\end{aligned} \tag{3.6}$$

where

$$\Delta = \Delta(z, u) = h_r(z, u) - \hat{h}_r(y, u)$$

In effect, the transformation places r integrators between the pseudo control ν and the system output y , with the error Δ acting as a disturbance signal. This is now a linear and controllable system, which was our target.

3.4.2 Feedback Linearization for Reconfigurable Control

Feedback linearization can be used in a model-following configuration by choosing the pseudo control to have the form [CHI01]

$$\nu = y_c^r + \nu_{dc} - \nu_{ad},$$

where ν_{dc} is the output of a stabilizing linear compensator for the linearized system given by Equation (3.6) with $\Delta = 0$. ν_{ad} is an adaptive signal designed to cancel Δ and y_c^r is the r^{th} derivative of the signal to be tracked. y_c^r can be obtained from an (at least) r^{th} order reference model which defines the desired dynamics.

If the model of the system is perfect, we would have $\Delta = 0$ and we could simply apply the input $u = \hat{h}_r^{-1}(x, \nu) = h_r^{-1}(x, y_c^r + \nu_{dc})$ and the system would track the reference trajectory. However, as there will always be modelling errors, the error Δ needs to be compensated for online and for this an ANN can be used. Neural networks can be trained to approximate any function with an arbitrary precision. As a result, the ANN can estimate the modelling error and hence cancel it. The benefit of this approach is that no model structure needs to be assumed in order to estimate the error. Figure 3.5 shows the structure of the full controller, and Figure 3.6, that of the linear compensator.

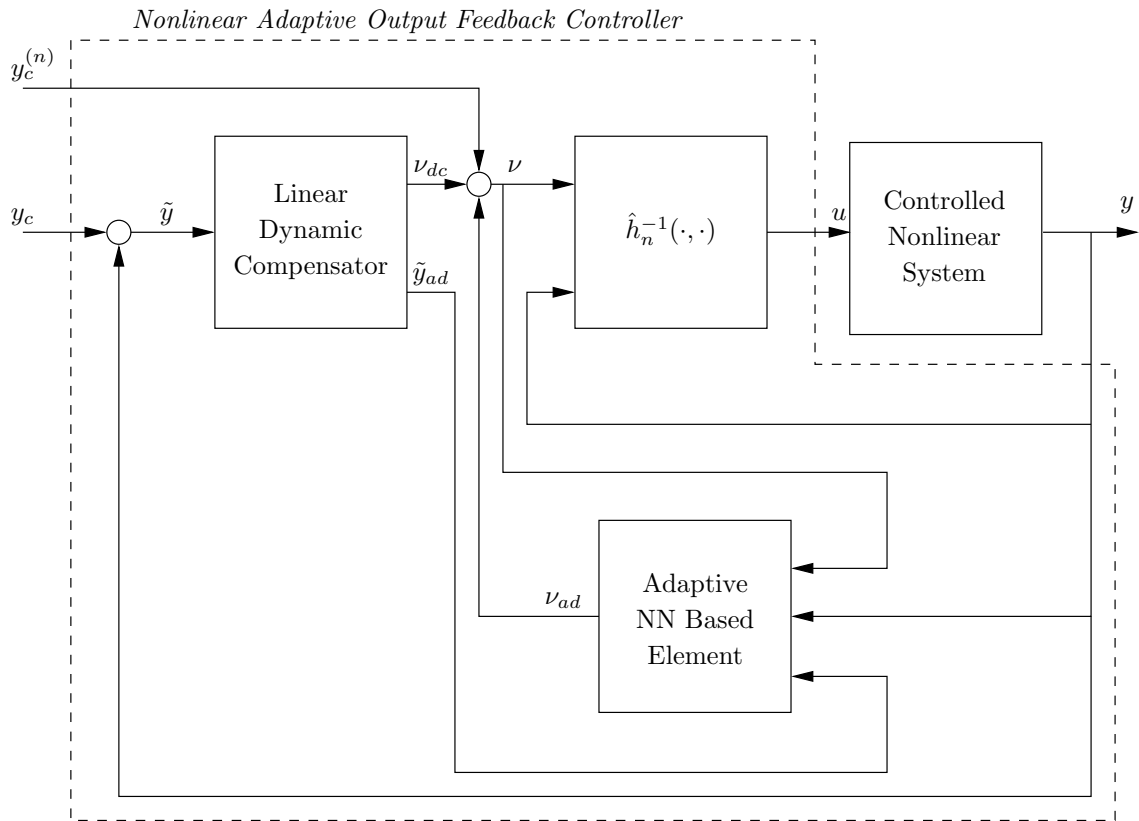


Figure 3.5: Nonlinear Adaptive Output Feedback Controller

This control technique was proposed as a method of reconfigurable control in combination with Wise’s ICEM [BW01] (see Section 3.3). This scheme is suited to reconfigurable control, as the adaptation makes no assumptions about the structure of the system after the failure. Since the ANN can approximate any nonlinear function, it can track and cancel any structural failures which may occur under the assumption of sufficient control authority and excitation for adaptation. The techniques presented in this section have been developed and expanded upon in several publications: Single Input Single Output (SISO) stability proofs [CHI01], input saturation [JC01], combined aero/engine control [IJCK01] and highly over-actuated

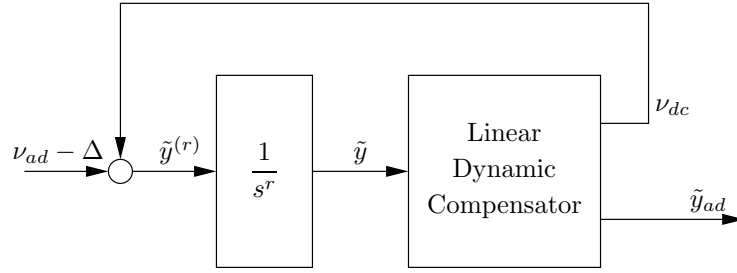


Figure 3.6: Block Diagram of the Error Dynamics

systems [CLS00].

3.5 Sliding Mode Control (SMC)

The method presented in this section, developed by Shtessel, Buffington and Banda [SB99, SS01], uses Sliding Mode Control (SMC) to develop a robust controller which adaptively handles input magnitude and rate constraints. We first give an overview of SMC before going through the particulars of its use in reconfiguration.

3.5.1 Introduction to Sliding Mode Control

Consider a nonlinear MIMO system of the form

$$x_i^{(n_i)} = \frac{d^{(n_i)}x_i}{dt^{(n_i)}} = f_i(x) + \sum_{j=1}^m b_{ij}(x)u_j \quad i = 1, \dots, m \quad j = 1, \dots, m$$

$$\dot{x} = f(x) + B(x)u \quad (3.7)$$

where $u = [u_1, \dots, u_m]'$ is the control input and the state x is composed of the x_i 's and their first $(n_i - 1)$ derivatives, in a similar fashion to the change of coordinates shown in Section 3.4.1.

The system is modelled by estimates \hat{f} and \hat{B} of f and B respectively, which are assumed to be bounded:

$$\begin{aligned} f &= \hat{f} + \Delta_f, & \|\Delta_f\|_\infty &\leq C_f, & C_f > 0 \\ B &= (I + \Delta_B)\hat{B}, & \|\Delta_B\|_\infty &\leq C_B, & C_B > 0 \end{aligned}$$

We make the assumption that \hat{B} is square, everywhere invertible, continuously dependent on parametric uncertainty, and such that $\hat{B} = B$ in the absence of uncertainty [SL91, Chapter 7].

The goal is to cause the state x to track a desired trajectory $x_d = [x_{1d}, \dots, x_{md}]'$. This is done by defining a sliding surface $S(t) = \{x \in \mathbb{R}^n \mid |s_i(x; t)| = 0, \forall i = 1, \dots, m\}$, such that

if $x \in S(t)$ then $x = x_d$. The controller can be split into two phases: reaching and sliding. During the reaching mode $s_i(x; t) \neq 0$ and so a stabilizing control law is needed such that $\lim_{t \rightarrow \infty} s_i(x; t) = 0$. A possible definition of $s(x; t)$ from [SL91, Chapter 7] is

$$\begin{aligned}\tilde{x} &= x - x_d \\ s_i(x; t) &= \left(\frac{d}{dt} + \lambda \right)^{n_i-1} \left(\int_0^t \tilde{x}_i dr \right) \\ s(x; t) &= [s_1(x; t), \dots, s_m(x; t)]^T\end{aligned}\tag{3.8}$$

For example, if $n_i = 3$, then we get a PID format

$$s_i(x; t) = \frac{d^2 \int_0^t \tilde{x}_i dr}{dt^2} + 2\lambda \frac{d \int_0^t \tilde{x}_i dr}{dt} + \lambda^2 \int_0^t \tilde{x}_i dr\tag{3.9}$$

$$= \ddot{\tilde{x}}_i + 2\lambda \dot{\tilde{x}}_i + \lambda^2 \int_0^t \tilde{x}_i dr.\tag{3.10}$$

Once the sliding surface has been reached, the goal becomes keeping the state on the surface. For this a control law is needed which makes $S(t)$ an invariant set. This can be done by selecting a control law such that $\dot{s}(x; t) = 0, \forall x \in S(t)$. To simplify notation, we define $\tilde{x}_r = [\tilde{x}_{1r}, \dots, \tilde{x}_{mr}]^T$ where $\tilde{x}_{ir} = C_i^1 \tilde{x}_i^{(n_i-1)} + \dots + C_i^{n_i} \tilde{x}_i - x_{id}^{(n_i)}$ where C_j^i are appropriate constants computed from Equation (3.8). From Equation (3.8) we have

$$\begin{aligned}\dot{s}_i(x; t) &= x_i^{(n_i)} - x_{id}^{(n_i)} + C_i^1 \tilde{x}_i^{(n_i-1)} + \dots + C_i^{n_i} \tilde{x}_i \\ &= x_i^{(n_i)} + \tilde{x}_{ir} \\ &= f_i(x) + \sum_{j=1}^m b_{ij}(x) u_j + \tilde{x}_{ir}\end{aligned}\tag{3.11}$$

Therefore the best estimate we have for a control law that will maintain $\dot{s}(x; t) = 0$ is

$$\begin{aligned}\dot{s}_i(x; t) &= 0 \\ &\Downarrow \\ \sum_{j=1}^m \hat{b}_{ij}(x) \hat{u}_j &= -\hat{f}_i(x) - \tilde{x}_{ir} \\ &\Downarrow \\ \hat{u} &= -\hat{B}^{-1} [\hat{f}(x) + \tilde{x}_r]\end{aligned}\tag{3.12}$$

Next, we want to augment the control law such that $s(x; t) \rightarrow 0$ in finite time for all x . This can be done using a Lyapunov function such as

$$V(s) = \frac{1}{2} s(x; t)^T s(x; t) > 0\tag{3.13}$$

Our goal is then to augment the control law such that the derivative of V is negative everywhere outside of $S(t)$, while still maintaining the property $\dot{s}(x; t) = 0, \forall x \in S(t)$. If we can

find such an augmentation, we can use Lyapunov's Theorem [SL91, Page 62] to show that $S(t)$ is globally asymptotically stable. Consider the control law [SL91, Chapter 7]

$$u = \hat{u} + \begin{bmatrix} k_1 \text{sgn}(s_1) \\ \vdots \\ k_m \text{sgn}(s_m) \end{bmatrix} = \hat{u} + K \text{sgn}(s) \quad (3.14)$$

From Equations (3.11) and (3.14) we have

$$\dot{V} = s^T \dot{s} = \sum_{i=1}^m s_i \dot{s}_i \leq 0 \quad \text{if} \quad s_i \dot{s}_i \leq 0, \quad \forall i \quad (3.15)$$

$$\begin{aligned} s_i \dot{s}_i &= s_i \left(\dot{f}_i + \sum_{j=1}^m b_{ij} u_j + \tilde{x}_{ir} \right) \\ &= s_i \left(\hat{f}_i + \Delta_{f_i} - \sum_{j=1}^m (1 + \Delta_{B_{ij}}) (\hat{f}_{ij} + \tilde{x}_{rij} + k_i \text{sgn}(s_i)) + \tilde{x}_{ir} \right) \\ &= s_i \left(\Delta_{f_i} - \sum_{j=1}^m \Delta_{B_{ij}} (\hat{f}_{ij} + \tilde{x}_{rij} + k_i \text{sgn}(s_i)) - k_i \text{sgn}(s_i) \right) \\ &= |s_i| \text{sgn}(s_i) \left(\Delta_{f_i} - \sum_{j=1}^m \Delta_{B_{ij}} (\hat{f}_{ij} + \tilde{x}_{rij} + k_i \text{sgn}(s_i)) - k_i \text{sgn}(s_i) \right) \\ &= |s_i| \left(\text{sgn}(s_i) \left(\Delta_{f_i} - \sum_{j=1}^m \Delta_{B_{ij}} (\hat{f}_{ij} + \tilde{x}_{rij} + k_i \text{sgn}(s_i)) \right) - k_i \right) \end{aligned}$$

Therefore $\dot{V} \leq 0$ and the system is globally asymptotically stable if K is chosen such that

$$k_i \geq \frac{C_f + C_B \left\| \hat{f}_i + \tilde{x}_{ir} \right\|_1}{1 + C_B m \text{sgn}(s_i)}$$

This control law will cause a large amount of chattering due to the $\text{sgn}(\cdot)$ term. This problem can be removed by introducing a boundary layer around the switching surface:

$$S(t) = \{x \mid |s(x; t)| \leq \Phi\}, \quad \Phi > 0$$

To maintain the properties of the Lyapunov function the $\text{sgn}(\cdot)$ term needs to be changed to a saturation function [SL91]

$$\begin{aligned} \text{sat}(y) &= y, & \text{if } |y| \leq 1 \\ \text{sat}(y) &= \text{sgn}(y), & \text{otherwise} \end{aligned}$$

The control law then becomes

$$u = \hat{u} + K \text{sat}(s/\Phi)$$

Sliding mode control is a nonlinear control method which allows an intuitive tradeoff between tracking performance and parametric uncertainty. This tradeoff can be adjusted online by modifying the boundary layer. As a result, it can be used to adaptively reduce tracking performance, while simultaneously increasing insensitivity to uncertainty during failures, as will be shown in the next section.

3.5.2 Reconfigurable Sliding Mode Control

This section is a review of the paper [SB99] and the presentation [Sht01]. The proposed controller is setup in a two-loop cascade configuration, with the ultimate goal of tracking a trajectory given by roll, pitch and yaw angle setpoints. The outer-loop takes roll, pitch and yaw setpoints and provides angular rate commands to the inner-loop, which is assumed to track the commands using the inputs to the actuators.

The outer-loop is designed using standard robust SMC techniques as described in Section 3.5.1. The inner-loop is also a robust sliding mode controller but has an adaptive feature to handle actuator magnitude and rate limitations. In [SB99] it is shown that modifying the size of the boundary layer online can ensure that integrators do not wind up, as well as ensuring that actuator magnitude and rate limits are satisfied. There is a direct trade-off between the size of the boundary layer and tracking performance. Therefore, this procedure provides an intuitive method of maximizing tracking while ensuring actuator limits.

The benefits of this controller to reconfigurable control are two-fold. First, being a robust control technique, it can handle all structural failures which modify the dynamics of the plant less than the assumed uncertainty. Second, the online adaptation of the boundary layer can handle partial loss of actuator surfaces, while avoiding limits and integrator windup by reducing the tracking performance. Although this technique provides benefits to aircraft control, there are limitations due to the use of SMC when it is presented with the full reconfigurable problem.

1. In SMC, the assumption must be made that the input function B (see Equation (3.7)) is square and invertible. For aircraft control this forces two requirements on the system: first, there must be one and only one control surface for every controlled variable and second, none of the control surfaces can ever be lost. This is handled in [SB99] by only considering failures which cause a partial loss of effectiveness of the control surfaces, which is not realistic as floating or jammed actuators are certainly possible failure scenarios. This problem could be addressed by placing a CA algorithm (see Section 3.3) between the requested outputs and the physical actuators, although to the author's knowledge this has not been proposed.
2. The method proposes to use robust control to handle all structural failures. This re-

quires a de-tuning of the controller to the point that it can handle uncertainties including all possible structural failures, which may well result in an excessively conservative controller in the non-failure situation.

3.6 Eigenstructure Assignment (EA)

EA was made popular in the 1980s primarily by Andry, Shapiro and Chung in their paper [ASC83] where the method of Direct Eigenstructure Assignment (DEA) was introduced. The idea behind the method is to place the eigenvalues of a linear system using state feedback and then use any remaining degrees of freedom to align the eigenvectors as accurately as is possible. The eigenvalues determine the natural frequency and damping of each mode while the eigenvectors control how much each mode contributes to a given output. The following sections first give a brief overview of the theory behind EA and then a review of its use in reconfigurable control.

3.6.1 Introduction to Eigenstructure Assignment

This section, which is summarized primarily from [DA94, KA96], motivates the use of EA. Consider the observable and controllable linear system

$$\begin{aligned}\dot{x} &= Ax + Bu \\ y &= Cx\end{aligned}\tag{3.16}$$

where $x \in \mathbb{R}^n$, $u \in \mathbb{R}^m$, $y \in \mathbb{R}^k$, $A \in \mathbb{R}^{n \times n}$, $B \in \mathbb{R}^{n \times m}$, and $C \in \mathbb{R}^{k \times n}$.

The system response is [Nis92, Page 189]

$$y(t) = Ce^{At}x(0) + \int_0^t Ce^{A(t-\tau)}Bu(\tau)d\tau\tag{3.17}$$

Assuming A is full rank and has distinct eigenvalues, we can write by virtue of the spectral decomposition

$$e^{At} = e^{V\Lambda V^{-1}t} = Ve^{\Lambda t}L = \sum_{j=1}^n v_j e^{\lambda_j t} l_j\tag{3.18}$$

where $L = V^{-1}$, λ_j is the j^{th} system eigenvalue, v_j is the j^{th} column of V and l_j is the j^{th} row of L .

Combining Equations (3.17) and (3.18) gives

$$y_i(t) = \sum_{j=1}^n c_i v_j e^{\lambda_j(t)} l_j x(0) + \sum_{j=1}^n \sum_{k=1}^m c_i v_j l_j b_k \int_0^t e^{\lambda_j(t-\tau)} u_k(\tau) d\tau$$

where b_k is the k^{th} column of B , c_i is the i^{th} column of C and u_k is the k^{th} input. Given an impulse in the k^{th} input the response of the i^{th} output is

$$y_i(t) = \sum_{j=1}^n \sum_{k=1}^m R_{ijk} e^{\lambda_j(t)}$$

where $R_{ijk} = c_i v_j l_j b_k$ is the modal residue for output i associated with eigenvalue j and due to input k .

The conclusion of this analysis is that the effect of each system mode on each output depends on the eigenvectors of the system matrix. For this reason EA tries to use state feedback to position the eigenvalues and then use any remaining degrees of freedom to place the eigenvectors as close as is possible to a set of desired positions, in the least squares sense.

3.6.2 Reconfigurable Eigenstructure Assignment

Although a method for choosing appropriate eigenvectors and eigenvalues is not immediately obvious for aircraft, some studies have been made on the effects of the eigenstructure (eigenvalues and eigenvectors) on flying qualities [DA96]. Methods which propose EA for use in reconfigurable flight control systems [KA96, BS00, ZJ00] first assume a linear fault model which has been given to the controller by a FDI system.

$$\begin{aligned}\dot{x} &= A_f x + B_f u \\ y &= C_f x\end{aligned}$$

The goal is then to design a stabilizing output feedback law K_f

$$u = K_f C_f x \tag{3.19}$$

such that the new eigenstructure closed-loop system $A_f + B_f K_f C_f$ is as close as possible to that of the original closed-loop system $A + BKC$.

The choice of K_f can be made in a variety of ways, but the placement of the eigenspace is limited by Theorem 3.6.1. Generally the eigenvalues of the failed system, λ_i^f are ordered from most important to least and then the top $\max(m, k)$ are made to exactly match those of the non-failed system λ , while the remainder are kept stable. Similarly, the most important $\max(m, k)$ eigenvectors of the failed system, v_i^f , are made close to those of the original system v_i in the least squares sense.

Theorem 3.6.1 [DA96] *Consider the controllable and observable system (3.16) with the output feedback law of (3.19) and the assumption that the matrices B and C are full rank. Then, there exists a matrix $K \in \mathbb{R}^{m \times k}$ such that*

1. $\max(m, k)$ closed-loop eigenvalues can be assigned
2. $\max(m, k)$ eigenvectors can be partially assigned with $\min(m, k)$ entries in each vector arbitrarily chosen

There are several limitations to this approach when applied to reconfiguration. First, only linear systems have been considered and actuator limitations have not been taken into

account. Second, a perfect fault model is assumed and the effects of uncertainty have not been extensively studied, although these issues have been considered in [KA95]. Finally, the effect of the eigenvectors in the failed system not being exactly equal to those in the nominal system is not well understood. The result of these significant limitations is that only a few researchers have proposed this approach.

3.6.3 Pseudo Inverse Method (PIM)

The Pseudo Inverse Method (PIM) is very similar to the EA approach presented above, which was very popular in the literature in the late 80s and early 90s. It has fallen out of grace recently due to difficulties in ensuring stability. The goal in PIM is to recover the closed-loop behaviour by computing a controller gain K_f which minimizes the difference between the closed-loop dynamics of the nominal and faulty systems.

Considering state feedback, the original PIM problem chose K_f such that the closed-loop matrix defining the failed system dynamics was close to the nominal one in the least-squares sense [Huz97].

$$A_f + B_f K_f = A + BK \Rightarrow K_f = B_f^\dagger (A - A_f + BK)$$

where B_f^\dagger denotes the pseudo-inverse of B_f . There is no guarantee, however, that this system will be stable. Several researchers have looked at making this procedure stable by posing it as an optimization problem where the eigenvalues of the faulty system are constrained to be stable (see [Pat97] for a survey).

3.7 Model Reference Adaptive Control (MRAC)

Aström defines an adaptive controller as “a controller with adjustable parameters and a mechanism for adjusting those parameters” [AW95, Page 1]. Clearly, all methods presented in this survey are adaptive to some degree (save for robust control techniques) as they require the identification of a fault model in order to compute a control law. The approach we consider here is model reference adaptive control (MRAC) which can be effective for many types of structural failures and is often used as a final stage in other algorithms.

The goal of adaptive model-following is to force the plant output to track a reference model. We consider linear plants of the form

$$\begin{aligned} \dot{x} &= Ax + Bu + d \\ y &= Cx \end{aligned} \tag{3.20}$$

where $x \in \mathbb{R}^n$, $u \in \mathbb{R}^m$, $y \in \mathbb{R}^k$ and a reference model of the form

$$\dot{y}_d = A_d y_d + B_d r \tag{3.21}$$

where $y_d \in \mathbb{R}^k$ and $r \in \mathbb{R}^k$. A_d and B_d are arbitrary square matrices with A_d stable. State feedback of the form shown in Figure 3.7 is considered.

$$u = C_0 r + G_0 x + v$$

where $C_0 \in \mathbb{R}^{k \times k}$, $G_0 \in \mathbb{R}^{k \times n}$ and $v \in \mathbb{R}^k$ are free controller parameters. The closed loop dynamics are then

$$\dot{y} = (CA + CBG_0)x + CBC_0r + CBv + Cd \quad (3.22)$$

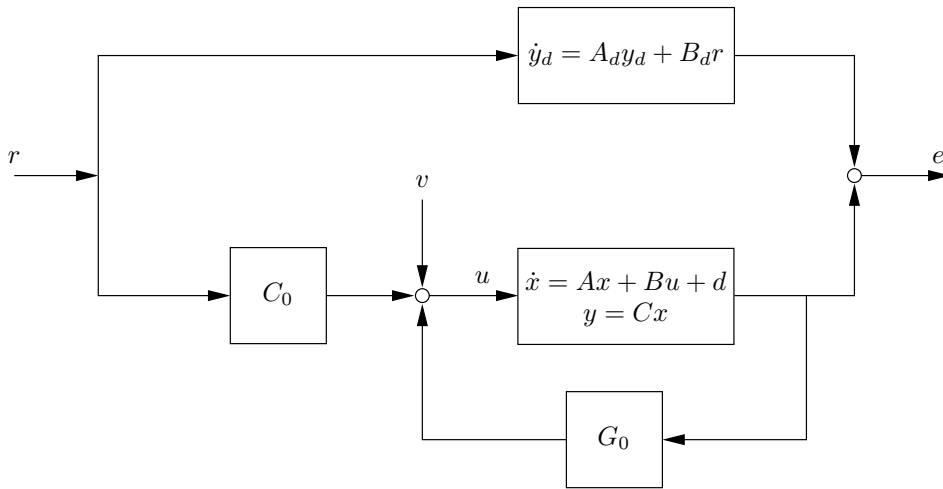


Figure 3.7: Model Reference Adaptive Control

The goal is now to make the closed loop dynamics given by Equation (3.22) match the desired dynamics of Equation (3.21). If the model shown in Equation (3.20) was known exactly, the controller parameters C_0, G_0 and v could be computed to achieve this. However, since we do not know the model (3.20) exactly as a failure may have occurred, we need to adapt the controller parameters. There are two methods to achieve this: direct and indirect adaptation.

3.7.1 Indirect Adaptation

There are two stages in indirect adaptive control. First the matrices A, B and d are estimated and then under the assumption that these estimates are correct the control parameters G_0, C_0 and v are computed such that the closed-loop system matches the desired dynamics.

A least squares algorithm can be used to compute the estimates \hat{A}, \hat{B} and \hat{d} [AW95, Chapter 2], which can then be used to compute the controller parameters such that the

closed loop dynamics (3.22) match the desired ones (3.21).

$$\begin{aligned} C_0 &= (C\hat{B})^{-1}B_d \\ G_0 &= (C\hat{B})^{-1}(A_dC - C\hat{A}) \\ v &= (C\hat{B})^{-1}(Cd) \end{aligned}$$

where we must assume that $\det(C\hat{B}) \neq 0$.

The idea of identifying the model online and then computing a control law under the assumption that the estimated model is perfect is common in the reconfigurable control literature. For example, the EA algorithms of Section 3.6 and the IMM algorithms of Section 3.2.2 assume this type of structure.

3.7.2 Direct Adaptation

Direct adaptive control attempts to estimate the controller parameters G_0, C_0 and v directly rather than first computing the model parameters. We define G_0^*, C_0^* and v^* as the ‘correct’ values of the controller parameters which will force the plant to track the reference model. A problem can then be designed such that a least squares routine can be used to estimate the correct controller parameters [BG97]. The idea of direct adaptation is seen in algorithms such as the adaptive feedback linearization presented in Section 3.4.

The basic model-reference adaptive control techniques described here are not by themselves suitable for reconfigurable control for two main reasons. First, in order for these approaches to work a model structure must be assumed. However, the types of failures addressed in reconfigurable control may well cause the plant structure to change drastically. Second, adaptive control requires the system parameters to change slowly enough for the estimation algorithm to track them. Faults may well cause abrupt and drastic changes in the parameters moving the system instantaneously to a new region of the parameter space. There is no guarantee that the system will be stable during the transient period in which the adaptive algorithm is identifying the faulty plant. Despite the limitations of adaptive control for reconfiguration, some researchers are attempting to apply it in slightly modified forms [Bod94, GB95, BG97]. As a result adaptive control on its own is not enough to handle the general problem, but may well be an important part of a reconfigurable algorithm.

3.8 Model Predictive Control (MPC)

Model predictive control has been proposed as a method for reconfiguration due to its ability to handle constraints and changing model dynamics systematically. See Section 2.2 for an introduction to MPC.

MPC relies on an internal model of the system and so, like many of the approaches presented in this survey, a fault model is required and is assumed here. As discussed in

Section 2.1.4 there are two general types of aircraft failures which we are considering in this report; namely actuator and structural failures. As noted in [Mig02], these failures can be handled naturally in a MPC framework via changes in the input constraints and internal model. Actuator limit and rate constraints can be written as:

$$\begin{aligned} u_i^l &\leq u_i(t) \leq u_i^u \\ du_i^l &\leq \dot{u}_i(t) \leq du_i^u \end{aligned}$$

for actuator inputs u_1 through u_m . If actuator i becomes jammed at position u_i^* the MPC controller can be made to compensate by simply changing the constraints on input i to

$$\begin{aligned} u_i^* &\leq u_i(t) \leq u_i^* \\ 0 &\leq \dot{u}_i(t) \leq 0 \end{aligned}$$

The result will be similar to the control allocation approach where other input channels are used to create the same effect. As noted in [Mac98], an MPC controller can be designed so that it has an intrinsic ability to handle jammed actuators without the need to explicitly model the failure.

Structural failures can also be handled in a natural fashion by changing the internal model used to make prediction in either an adaptive fashion [KV00b], a multi-model switching scheme [BM98] or by assuming an FDI scheme which provides a fault model [HM98a, HM98b, Ker98].

Although the MPC approach to reconfiguration has a lot of useful properties, there are still several issues that need to be examined. First, it is not clear how to adjust the weights in the cost function or the prediction/control horizons for an arbitrary fault model. Second, choosing performance targets is not a simple question. Finally, MPC requires an online optimization at every time step, making it difficult to implement as an aircraft controller, where the optimization must occur at high sampling rates and terminate in fixed time. There is also no guarantee that there exists a solution to the optimization problem for all time.

Method	Failures		Robust	Adaptive	Fault Model		Constraints	Model Type	
	Actuator	Structural			FDI	Assumed		Linear	Nonlinear
Multiple Model Switching and Tuning (MMST)		•		•	•			•	
Interacting Multiple Model (IMM)		•		•	•		○	•	
Propulsion Controlled Aircraft (PCA)	•		○			•		•	•
Control Allocation (CA)	•					•	○	•	
Feedback Linearization	•	•		•	•				•
Sliding Mode Control (SMC)	○ ¹	•	• ²				•		•
Eigenstructure Assignment (EA)		•				•		•	
Pseudo Inverse Method (PIM)		•				•		•	
Model Reference Adaptive Control (MRAC)		•		•	•			•	○
Model Predictive Control (MPC)	•	•	○	○	•	•	•	•	•

Table 3.1: Comparison of Reconfigurable Control Methods

3.9 Conclusion

Table 3.1 presents a comparison of the methods considered in this survey. Filled circles mean that the method has the property while empty circles imply that an author has suggested that the approach could be modified to incorporate the property. The columns are explained as follows:

- Failures: Types of failures that the method can handle
- Robust: The method uses robust control techniques
- Adaptive: The method uses adaptive control techniques
- Fault Model:
 - FDI: A FDI algorithm is incorporated into the method
 - Assumed: The method assumes a FDI algorithm which provides a fault model
- Constraints: The method can handle actuator constraints
- Model Type: The type of internal model used

As can be seen from Table 3.1, there have been many methods suggested for reconfigurable control, but most are designed to only deal with a small aspect of the problem. To date, only MPC has the potential for solving the general reconfigurable control problem and it is for this reason that this report focuses on the use of MPC in aircraft reconfigurable control.

¹SMC can handle partial loss of effectiveness of actuators, but not complete loss

²SMC assumes robust control can handle all forms of structural failures

Chapter 4

Sufficient Conditions for Pilot Control

4.1 Introduction

Of the 421 fatal accidents involving large jet aircraft in the UK between 1990 and 1999, not one was caused by loss of pilot [Civ00]. Because of this fact, we make the assumption that for large passenger planes there will always be a pilot (or several) available to fly the plane. In a failure situation the pilot is a valuable resource who must, if possible, be utilised. Removing him from the loop is a last resort and should only be done if it can be shown that he cannot fly the plane¹. To this end we have been looking at finding methods with which it can be proven that the pilot will be able to continue flying the plane for all time and at control algorithms which will allow him to do so.

As seen in Chapter 3, virtually all methods of FTC currently require the assumption that there exists ‘sufficient control authority’ to recover a given level of performance. However, in a failure situation, the plane will undoubtedly have less control authority than it was designed for and as a result will likely not be able to regain an arbitrary level of performance. The question that we intend to address in this work is how to design a controller that will provide sufficient performance such that the pilot can continue to fly the plane.

4.2 Problem Definition

A sufficient condition for the pilot to fly the plane is for it to operate in a fashion that is close to its original specification, i.e., from the pilot’s point of view, the column, wheel, pedal and throttle inputs cause close to the same effect as they do on the working plane. This can be cast as a constrained model-following problem as shown in Figure 4.1. The goal is

¹Defining when a pilot can or cannot fly the plane is a goal of this work

to compute a controller such that the damaged aircraft tracks the state trajectory provided by the model $M(x_d, u_p; p)$. The pilot gives commands to the model $M(x_d, u_p; p)$ and the goal of the controller is to cause the plane to track the resultant trajectory. $M(x_d, u_p; p)$ is a model of the desired performance of the plane where $p \in \mathbb{R}^n$ is a performance parameter such that $\|p\| = 1$ implies that $M(x_d, u_p; p)$ is the nominal model of the plane (i.e., maximum performance).

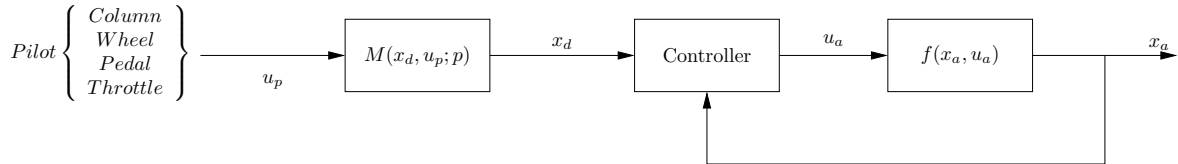


Figure 4.1: Model Following Controller

The following notation introduced in Figure 4.1 will be used throughout the remainder of this chapter: $f(x_a, u_a)$ is the faulty plane, where $x_a \in \mathbb{R}^n$. $M(x_d, u_p; p)$ is the model to be followed where $p \in \mathbb{R}^q$ is a performance parameter. $u_p \in \mathbb{R}^l$ are the pilot commands (column, wheel, pedal, throttle). $u_a \in \mathbb{R}^m$ are the commands to the control surfaces and $x_a \in \mathbb{R}^n$ is the measured state of the aircraft.

The actuator inputs are constrained in both magnitude and rate:

$$\begin{aligned} W_a u_a(k) &\leq w_a, \\ W_{ad}(u_a(k) - u_a(k-1)) &\leq w_{ad}, \end{aligned} \quad (4.1)$$

where W_a and W_{ad} are appropriate weighting matrices.

The goal is now to maximise the performance parameter p , while ensuring that the reference trajectory can be tracked. Some of the key questions that we would like to answer for this system are as follows:

- Can offset-free tracking be achieved? In what area of the state space? For what maximum p ? Which states require offset-free tracking?
- What are the maximum transients? Are these significant?
- How much uncertainty can be handled?
- How is suitable performance defined? Can suitable performance be achieved at all points in the flight path up to and including the runway?

There are several methods of looking at the problem of reference tracking with constrained inputs and states. Mosca and Angeli [AM99, CMA00, Bem97] advocate a Reference Governor (RG) approach, in which the input reference is modified to satisfy the constraints.

By assuming a ‘well-designed’ controller which will track references when there are no constraints, it has been shown that by modifying the reference command the system will remain stable, avoid the constraints and, when possible, track the reference. Saberi and Stoorvogel [SLT94, LS94, SHS01, SHS02] have for some years been working on conditions under which controllers can achieve reference trajectory tracking in constrained systems. Blanchini and Miani [BM00] consider constrained tracking problems for linear systems and have shown that if the reference signal converges to a constant, constraint-admissible value, then any domain of attraction around the origin is also a domain of attraction for reference tracking.

It is not clear which, if any, of these methods is the most suitable for this problem. Constrained reference tracking is an important and open problem in control and research in this direction will be proposed in Chapter 5.

4.3 A Motivating Example - Flight EL AL 1862

On 4 October 1992, a Boeing 747-200F freighter flying out of Amsterdam International Airport crashed into an eleven-floor apartment building after suffering multiple engine separations on the right wing. Despite this failure, the crew continued flying for almost 15 minutes, giving ample time for identification of the failure and for reconfiguration. This particular crash is a good test case for studying reconfigurable control for several reasons. First, the plane was clearly controllable, as the crew stayed in the air for almost a quarter hour after the failure. Second, despite the fact that sufficient time was available, current emergency procedures could not handle the situation as the plane did eventually crash. Finally, there was sufficient damage to the plane to make it difficult to fly, thus making it a challenging and interesting problem for reconfiguration. A detailed Simulink model has been made available by M.H.Smaili and all details of the crash and/or simulation model used in this section are taken from [Sma97].

The following sections will first describe the failure scenario and the resulting damage. They will go through the control surfaces available on the Boeing 747-200F and the trimming and the linearization routines used. Finally, an MPC controller will be described which has been used to reconfigure the aircraft and which attempts to restore nominal performance to the pilot, despite the significant failures.

4.3.1 Failure Scenario

The damage to the aircraft resulting from the loss of two engines on the right wing was severe:

- Right-hand wing leading edge severely damaged
- Right-hand wing leading edge flaps partially lost

- Right-hand outboard aileron floating at a zero hinge moment
- Limited roll control due to no outboard aileron available caused by outboard trailing edge flaps failure
- Limited roll control due to partly lost spoiler system
- Limited directional control due to lagging behind of the lower rudder for unknown reasons
- Right-hand inboard aileron less effective due to disturbed airflow caused by right-wing damage and loss of pylon no. 3
- Engines no. 1 and 2 at high thrust settings

Despite the substantial damage to the plane, modern craft are designed to be controllable in the instance of multiple single-wing engine failures without separation. However, due to the damage to the right wing caused by the removal of the engines, there was a significant loss of both lateral and directional control. In the worst case, wheel deflections of up to 60° and full pedal deflections were needed to maintain level flight, indicating that actuator constraints will need to be considered in any reconfiguration attempt.

The final loss of the plane was due to a hard-over right roll to 104° and an unrecoverable down pitch of 70° . As the pilot took a final right turn to line up with the runway, the airspeed was decreased and the angle of attack increased in preparation for landing. The post-crash reconstruction and simulation of the flight showed that although at low angles of attack, the distribution of lift along both left and right wings is essentially the same, high angles of attack will cause a large differential between lift and will result in a rolling moment. A FDI system warning the pilot not to get into such a configuration may have prevented this final outcome.

This situation is well known and is likely the result of a rudder stall. The rudder was in a hard-over position in order to compensate for the asymmetric thrust. Therefore, it had a much higher stall speed than normal, which resulted in a stall when the velocity was decreased in preparation for landing. According to [Den96]:

If the rudder stalls, it will be about as unpleasant as anything you can imagine. There will be a sudden uncontrollable yawing motion. Because of the yawing motion, the wingtip on the side with the good engine will have a higher airspeed than the wingtip on the other side. Because of the difference in airspeed (plus the difference in prop [or jet] wash patterns) the good-side wing will produce much more lift, so you will get an uncontrollable roll. As the inside wing drops, it will probably stall (since you were already at a low airspeed). You are now in

a spin. There is no guarantee that it will be possible to recover from the spin; multi-engine aeroplane certification regulations do not require spin recoveries.

The recommended solution to this problem is to keep the airspeed above the minimum control speed, V_{MC} , which is defined as “the calibrated airspeed at which, when the critical engine is suddenly made inoperative, it is possible to maintain control of the aeroplane with that engine still inoperative and then maintain straight flight at the same speed with an angle of bank of not more than 5° .”

4.3.2 Control Surfaces

The Boeing 747 has a large number of control effectors, although it is controlled by the pilot in the same fashion as much smaller planes. The inputs from the pilot are given in Table 4.1.

Control	Symbol	Lower Limit	Upper Limit
Control column position	δ_c	-12.5°	12.5°
Control wheel position	δ_w	-88°	88°
Rudder pedal position	δ_r	-14°	14°
Stabilizer handle position	δ_{stab}	0°	15°
Speedbrake handle position	δ_{sbh}	0°	37°
Flap handle position	δ_{fh}	0°	30°

Table 4.1: Boeing 747 Pilot Controls and Mechanical Limits

The control surfaces and limits are shown in Table 4.2. The mapping from pilot inputs to control surfaces is handled by a hydraulic Flight Control System (FCS), which also modifies the dynamics of the craft by adding yaw damping to reduce yaw oscillations (Dutch Roll) and to provide turn coordination when the flaps are down. The Simulink model has been modified to remove the existing control system and to allow direct access to the individual control surfaces.

4.3.3 Trimming and Linearization

Trimming and linearization routines were included with the Simulink model. These have been modified to handle the model with all actuators available as inputs. All inputs have been scaled such that -1 corresponds to the most ‘down’ position and 1 to the most ‘up’ given in Table 4.2.

The aircraft has been trimmed in straight and level flight at an altitude of $5000ft$ ($1524m$), an airspeed velocity of $250KTS$ ($128m/s$), an angle of attack of 8° and with flaps up, which was the approximate configuration of the plane during the pre-crash manoeuvres. Table 4.3 shows the trim conditions for the nominal and failed models.

Control Surface	Mechanical Limit (deg)	Normal Operation Rate: (deg/sec)	One Hydraulic System Failed Rate: (deg/sec)
2 Inboard Elevators	+17	37 down	30 down
	-23	37 up	26 up
2 Outboard Elevators	+17	37 down	30 down
	-23	37 up	26 up
Stabilizer Angle	+3	0.5 to 0.2	0.25 to 0.1
	-12		
2 Inboard Ailerons	+20	40 down	27 down
	-20	45 up	35 up
2 Outboard Ailerons	+15	45 down	22 down
	-25	55 up	45 up
8 Flight Spoilers	45	75	
2 Flight Spoilers	20	75	
2 Ground Spoilers	20	25	
Upper Rudder Segment	± 25	50	40
Lower Rudder Segment	± 25	50	40
2 Inboard Flaps			
2 Outboard Flaps			

Table 4.2: Boeing 747 Control Surfaces and Mechanical Limits

Figure A.1 shows the response of the nominal linear model with the working nonlinear plant to a 10% step in each pilot control, as well as the response of the nonlinear model in the failure configuration. Note that the linear model is quite accurate. Figures A.2, A.3, A.4 and A.5 show the response of the working aircraft to 10% steps on all of the available control surfaces. Figures A.6, A.7, A.8 and A.9 show the response of the nonlinear and linear models under the failure scenario. Also shown for comparison is the step response of the working linear model. Observe that the linear response of the failure model for actuators which are functioning is very similar to that of the linear working model. This implies that it may be valid to model the failed system by setting the columns of the B matrix corresponding to the failed actuators to zero. It would also be necessary to reduce the rate constraints of the operational actuators that have lost one of their two hydraulic systems according to Table 4.2. Note that although the aircraft is not statically stable while trimmed in the failure scenario, it is dynamically stable. Therefore, the observed responses will drift even with no movement of the control surfaces and hence the linear model is not a particularly good match over the 15 second period. Investigation is ongoing into this problem.

Table 4.4 displays the periods and damping ratios of the various modes for the nominal, failed and typical models. The typical values are for a passenger version of the Boeing 747 and are taken from [ER96, Pages 167, 188]. As can be seen from Table 4.4, the lateral

State/Control Surface		Nominal Model	Failure Model
Roll Rate	p	0	0
Pitch Rate	q	0	0
Yaw Rate	r	0	0
Velocity	v	128.6m/s	128.6m/s
Angle of Attack	α	8°	8°
Sideslip Angle	β	0	0
Roll	ϕ	0	0
Pitch	θ	8°	8°
Yaw	ψ	0	0
Height	h_e	1524.4m	1524.4m
Left Outer Engine		62657N(20.2%)	16297N(70.3%)
Left Inner Engine		63117N(20.4%)	17418N(75.9%)
Right Inner Engine		59967N(18.8%)	Failed (0N)
Right Outer Engine		61374N(19.5%)	Failed (0N)
Right Inner Aileron		0.1726°	2.6461°
Left Inner Aileron		0.0580°	-0.3317°
Right Outer Aileron		-5.5620°	Failed (0°)
Left Outer Aileron		-5.6268°	-1.1000°
Right Inner Elevator		-1.2540°	3.8892°
Left Inner Elevator		-1.2540°	Failed (0°)
Right Outer Elevator		-1.3297°	Failed (0°)
Left Outer Elevator		-1.3297°	3.9215°
Upper Rudder		0.1954°	14.9325°
Lower Rudder		0.0564°	16.2407°

Table 4.3: Trim Configuration

modes do not change significantly after the failure although the longitudinal ones do. As the longitudinal modes are highly sensitive to the c.g. [ER96, Page 182], a possible cause could be the movement of the c.g., due both to the loss of the engines and to the dumping of the fuel. This result shows again that it may be valid to model the failed aircraft with changes only to the actuator magnitude and rate limits.

Mode Name		Nominal Model		Failure Model		Typical	
		Period (s)	t_{half} (s)	Period (s)	t_{half} (s)	Period (s)	t_{half} (s)
Lat	Spiral	—	67.51	—	66.7	—	95
	Rolling Convergence	—	0.78	—	0.76	—	1.23
	Dutch Roll	6.85	7.28	6.79	6.91	6.64	21
Long	Phugoid	66.0	285.6	64.6	159.7	93.4	211
	Short-period	6.23	1.23	6.36	1.24	7.08	1.86

Table 4.4: Comparison of Lateral and Longitudinal Modes for Nominal, Failure and Typical Aircraft

4.3.4 Pilot Control via MPC

The goal of this section is to demonstrate that it is possible to recover enough of the nominal performance that a pilot could continue to fly the aircraft after the damage sustained during Flight EL AL 1862. To this end, a MPC controller has been developed which will track a reference trajectory provided by the nominal reference model. As shown in Figure 4.2, the pilot provides wheel, column, rudder and throttle inputs to the nominal reference model, which then produces a state trajectory that the functioning, or nominal, plane would achieve at the next time step. The purpose of the reconfigurable MPC controller is then to cause the state of the true plane to track the reference. If there is no failure, this is equivalent to an online control allocation algorithm. However, if provided with a correct failure model, the MPC controller will also function as a reconfigurable controller, causing the damaged aircraft to respond, if possible, to the pilot’s input as expected.

For the purposes of demonstration, a ‘pilot’ is needed to control the plane. Another MPC controller is used for this purpose, which has the reference plane as its internal model, in order to track forward velocity (v), angle of attack (α), sideslip (β), pitch (θ), and the heading derivative ($\dot{\chi}$). This configuration is shown as Figure 4.3.

The inner loop runs at a frequency of $10Hz$, which is chosen to be faster than the fastest mode given in Table 4.4 and the outer loop runs at $\frac{1}{10}^{th}$ the frequency of the inner. The prediction horizon of the inner loop was originally chosen as 10% of the length of the longest period in Table 4.4. However, with experimentation it was found that a much shorter horizon

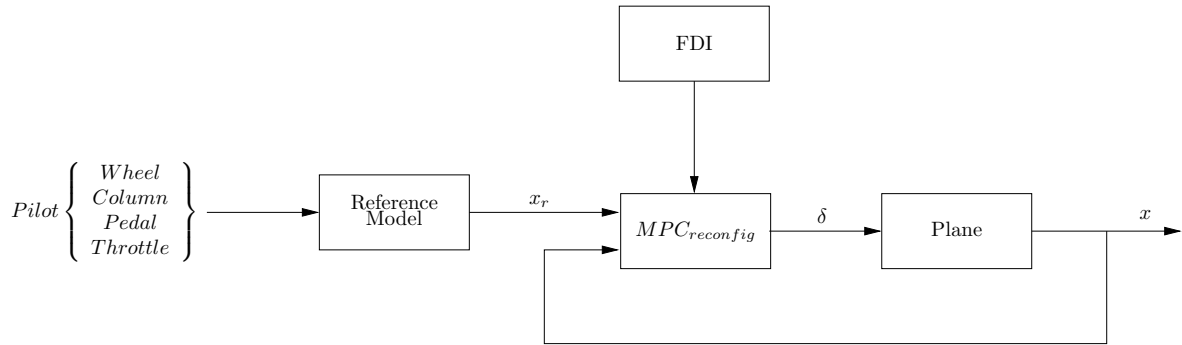


Figure 4.2: MPC Reconfigurable Controller

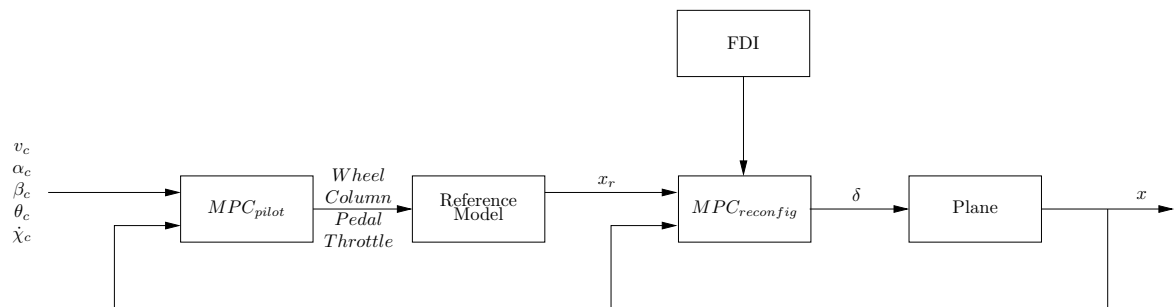


Figure 4.3: MPC 'Pilot' Controller

was sufficient; only 15 samples prediction were needed for the outer loop and 12 for the inner. Control horizons were chosen experimentally to be 15 and five for the outer and inner loops respectively. While the inner loop control horizon was chosen to be as short as possible, the outer loop was chosen to be longer than necessary for stability, since the reduction in speed was not significant due to its slow sampling rate.

For demonstration, the test reference trajectory is similar to that which was required for Flight EL AL 1862 to lineup with the runway, including several right turns. An additional 90° left turn is added at the beginning of the trajectory in order to test the system's response to left turns.

Four cases are considered: linear nominal, linear failed, nonlinear nominal and nonlinear failed aircraft. In all cases the internal MPC models are linear and all controllers have the same tuning parameters. The tuning of the controllers was a difficult and time consuming endeavour, as it is not clear how aircraft performance requirements should be translated into an MPC cost function. The removal of the requirement to tune the cost function of the controller would make a significant difference in development time. The linear models were not particularly difficult to tune as the internal MPC models are identical to the plant. However, the mismatch between the internal MPC model and the 'true' aircraft becomes a difficulty when dealing with the nonlinear model. A longer prediction horizon does not guarantee or necessarily improve stability, as the predictions will get worse over a longer horizon due to mismatch. The horizons chosen are largely the result of a 'guess and check' approach. The controller is tuned for this particular failure scenario and it is unknown if it would work for other failures or flight conditions. Systematic and general methods of choosing the horizons and cost functions will be discussed again in Chapter 5.

The outer loop is tuned such that it will track changes in the reference θ and $\dot{\chi}$. The velocity v , and the angles α and β are not tracked, but are constrained to remain within a given distance of their trim condition. The pilot's inputs are bounded by their physical limitations and a rate limit of $\frac{1}{2}$ their full range per second. The inner loop is tuned such that it will track the velocity (v, α, β) and the orientation (ϕ, θ, ψ) of the plane. The inputs are constrained by their physical limitations and rates. Tables 4.5, 4.6, 4.7, 4.8, and 4.9 display these details numerically. Note that all values refer to the inputs scaled between $[-1, 1]$.

Linear Simulations

The plots discussed in this section are produced using the linear model from Section 4.3.3. Figures 4.4, 4.5, 4.6, 4.7 and 4.8 show the result of running the algorithm on the correctly functioning aircraft, and 4.9, 4.10, 4.11, 4.12 and 4.13 on the failed. As a result, the inner-loop MPC controller is in effect a control allocation scheme. The controller with an working internal model has been used to control the failed aircraft and, as expected, the plane rapidly

Output		Amplitude		Weight
		Minimum	Maximum	
Roll Rate	p	$-\infty$	∞	0
Pitch Rate	q	$-\infty$	∞	0
Yaw Rate	r	$-\infty$	∞	0
Velocity	v	$-\infty$	∞	1
Angle of Attack	α	$-\infty$	∞	10
Sideslip Angle	β	$-\infty$	∞	10
Roll	ϕ	$-\infty$	∞	10
Pitch	θ	$-\infty$	∞	10
Yaw	ψ	$-\infty$	∞	10

Table 4.5: Inner Loop Tuning Parameters: Output Constraints

Input	Amplitude		Rate		Weight
	Minimum	Maximum	Minimum	Maximum	
Left Outer Engine	-0.1817	0.7182	-0.2000	0.2000	10
Left Inner Engine	-0.1838	0.7162	-0.2000	0.2000	10
Right Inner Engine	-0.1696	0.7303	-0.2000	0.2000	10
Right Outer Engine	-0.1759	0.7240	-0.2000	0.2000	10
Right Inner Aileron	-1.0086	0.9913	-2.2500	2.0000	1
Left Inner Aileron	-1.0029	0.9971	-2.2500	2.0000	1
Right Outer Aileron	-0.9719	1.0281	-2.7500	2.2500	1
Left Outer Aileron	-0.9686	1.0313	-2.7500	2.2500	1
Right Inner Elevator	-1.0873	0.9127	-1.8500	1.8500	1
Left Inner Elevator	-1.0873	0.9127	-1.8500	1.8500	1
Right Outer Elevator	-1.0835	0.9164	-1.8500	1.8500	1
Left Outer Elevator	-1.0835	0.9164	-1.8500	1.8500	1
Upper Rudder	-1.0078	0.9921	-2.0000	2.0000	1
Lower Rudder	-1.0023	0.9977	-2.0000	2.0000	1

Table 4.6: Inner Loop Tuning Parameters: Input Constraints (Nominal Model)

Input	Amplitude		Rate		Weight
	Minimum	Maximum	Minimum	Maximum	
Left Outer Engine	-0.6327	0.2672	-0.2000	0.2000	10
Left Inner Engine	-0.6832	0.2168	-0.2000	0.2000	10
Right Inner Engine	0	0	0	0	0
Right Outer Engine	0	0	0	0	0
Right Inner Aileron	-1.1323	0.8676	-1.7500	1.3500	1
Left Inner Aileron	-0.9834	1.0166	-1.7500	1.3500	1
Right Outer Aileron	0	0	0	0	0
Left Outer Aileron	-1.1950	0.8050	-2.2500	1.1000	1
Right Inner Elevator	-1.3445	0.6555	-1.3000	1.5000	1
Left Inner Elevator	0	0	0	0	0
Right Outer Elevator	0	0	0	0	0
Left Outer Elevator	-1.3461	0.6539	-1.3000	1.5000	1
Upper Rudder	-1.5973	0.4027	-1.6000	1.6000	1
Lower Rudder	-1.6496	0.3503	-1.6000	1.6000	1

Table 4.7: Inner Loop Tuning Parameters: Input Constraints (Failure Model)

Output		Minimum	Maximum	Weight
Velocity	v	$-10m/s$	$10m/s$	0
Angle of Attack	α	-1°	1°	0
Sideslip Angle	β	-3°	3°	0
Pitch	θ	$-\infty$	∞	5
Heading Rate	$\dot{\chi}$	$-\infty$	∞	10

Table 4.8: Outer Loop Tuning Parameters: Output Constraints

Input	Amplitude		Rate		Weight
	Minimum	Maximum	Minimum	Maximum	
Engine	-0.1745	0.7237	-0.5	0.5	10
Column	-0.9973	1.0026	-0.5	0.5	1
Wheel	-1.0013	0.9987	-0.5	0.5	1
Pedal	-1.0000	1.0000	-0.5	0.5	1

Table 4.9: Outer Loop Tuning Parameters: Input Constraints

lost control and crashed. The results are discussed below each figure.

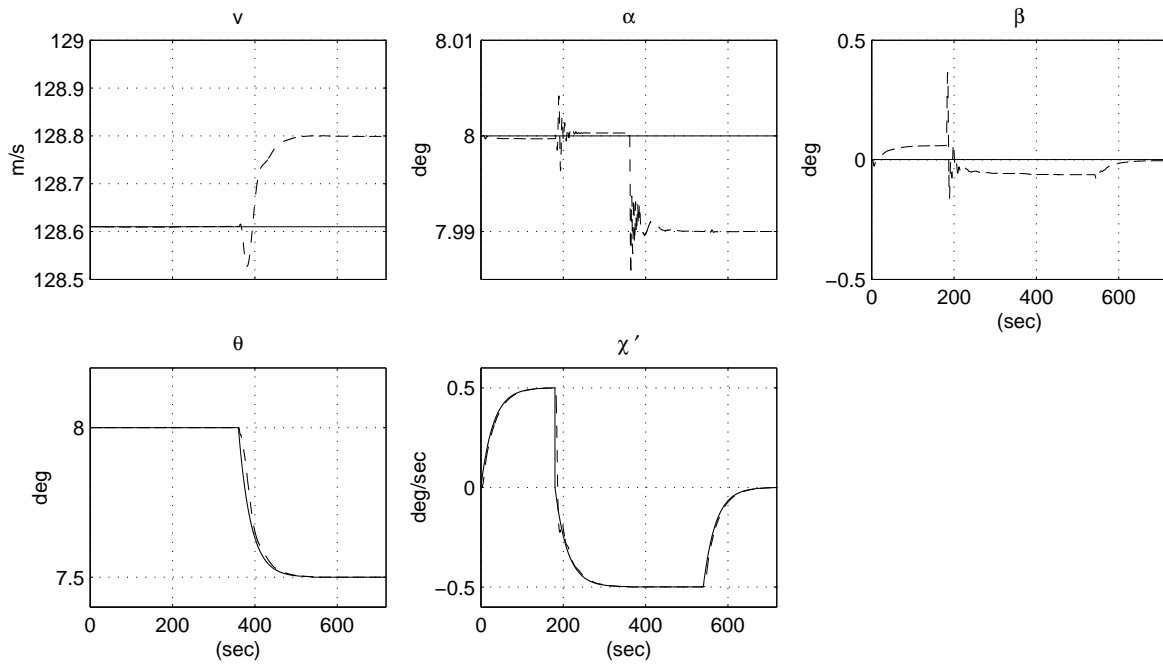


Figure 4.4: Linear Nominal Model: Outer Loop Trajectory Tracking
Desired (solid) and Actual (dotted) Trajectories

This plot shows the ability of the pilot to track the reference trajectory in θ and $\dot{\chi}$ while keeping the velocity v , angle of attack α , and sideslip β within acceptable limits. Note that the performance is very good, as would be expected as the internal model is exact and the plant is linear.

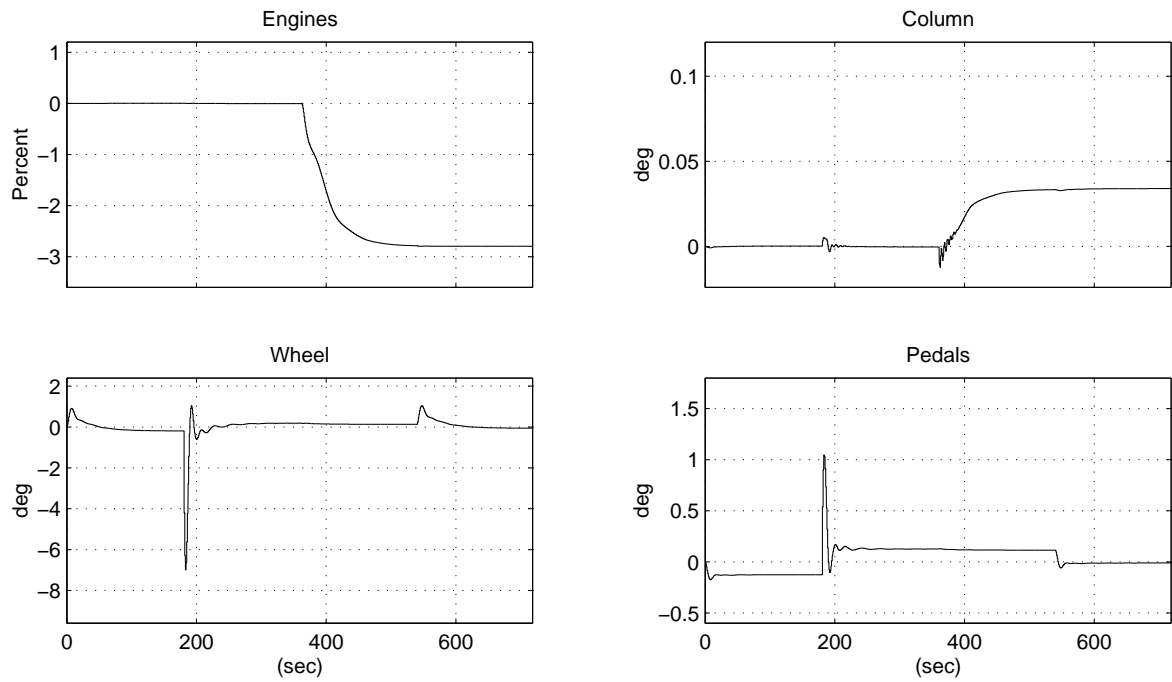


Figure 4.5: Linear Nominal Model: Pilot Input Commands

This plot displays the pilot's input commands. There are two items to note: First, as this is not a very demanding manoeuvre, the controls are not moved more than a degree or two (see Table 4.1 for the mechanical limits). Second, although no knowledge of how aircraft fly has been used in designing the controller, the pilot has correctly executed a coordinated turn by adding right pedal to left wheel in order to prevent sideslip during a turn. This provides some evidence that using a MPC controller in place of a pilot is valid for this example.

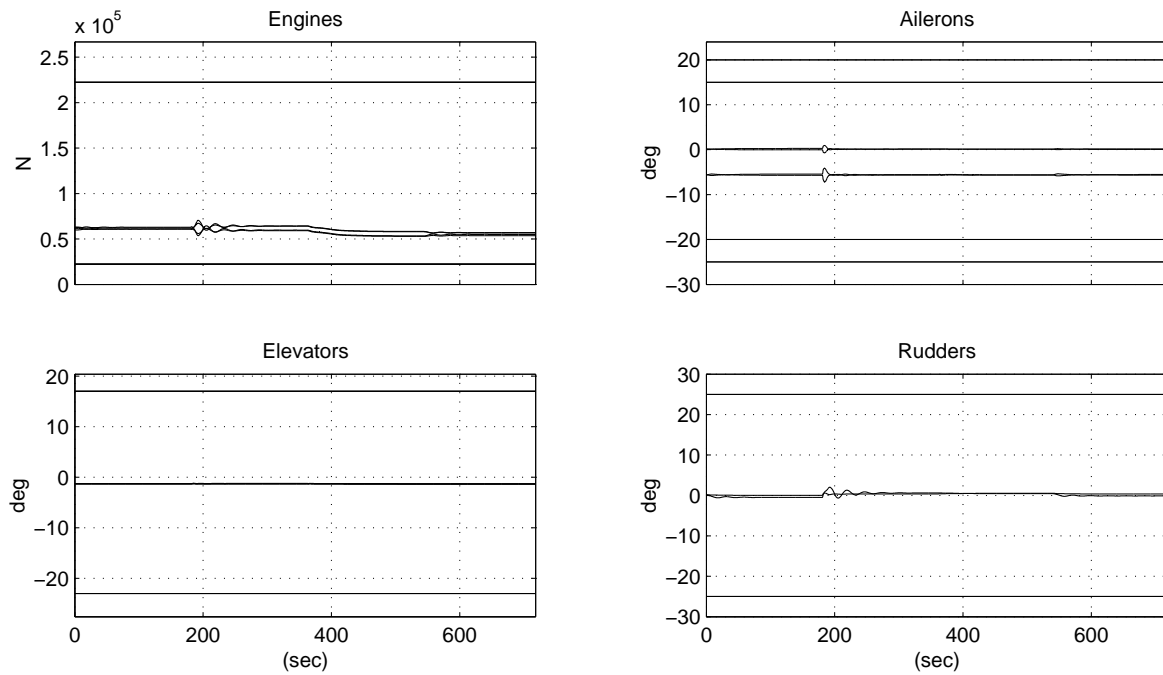


Figure 4.6: Linear Nominal Model: Scaled Actuator Commands and Physical Limits

This plot shows the movements of the physical control surfaces, as well as their mechanical limitations. Note that with no noise present, the nominal model requires very little control authority to execute this manoeuvre.

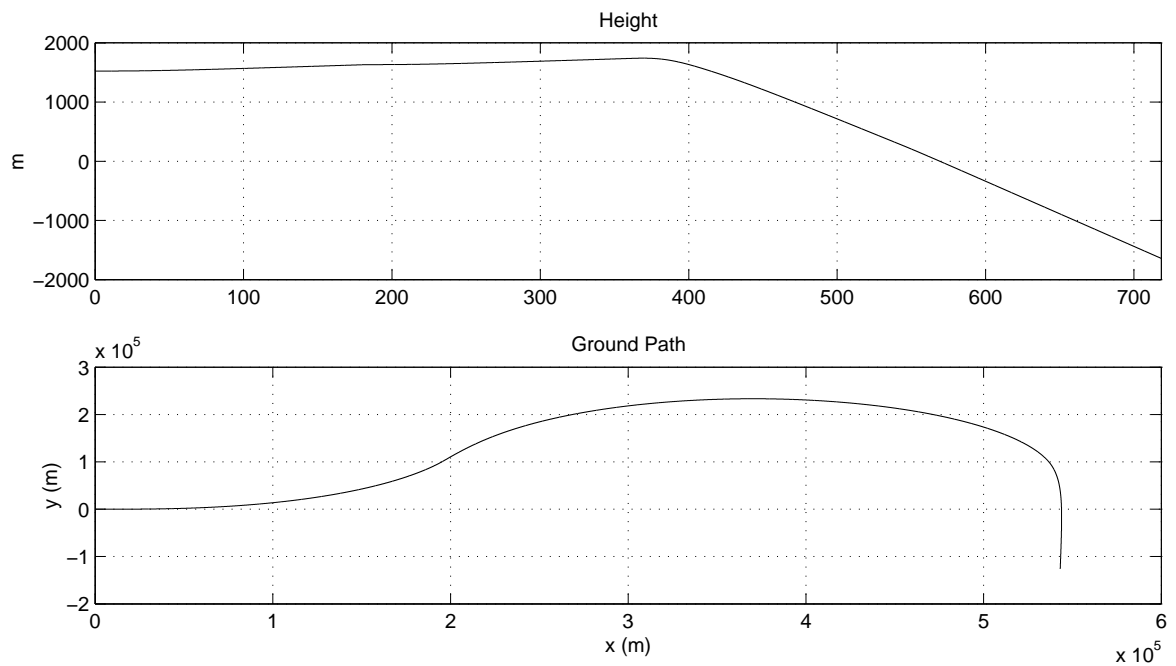


Figure 4.7: Linear Nominal Model: Height and Ground Path

This plot shows the ground path and height of the aircraft through the manoeuvre. The ground path is similar to a portion of the EL AL Flight 1862 during its pre-crash manoeuvres, which required four such right turns to line up with the runway. Note that although the plane passes through the ground at approximately 580 seconds into the flight, this is not a problem, as the states of the linear model are not a function of altitude.

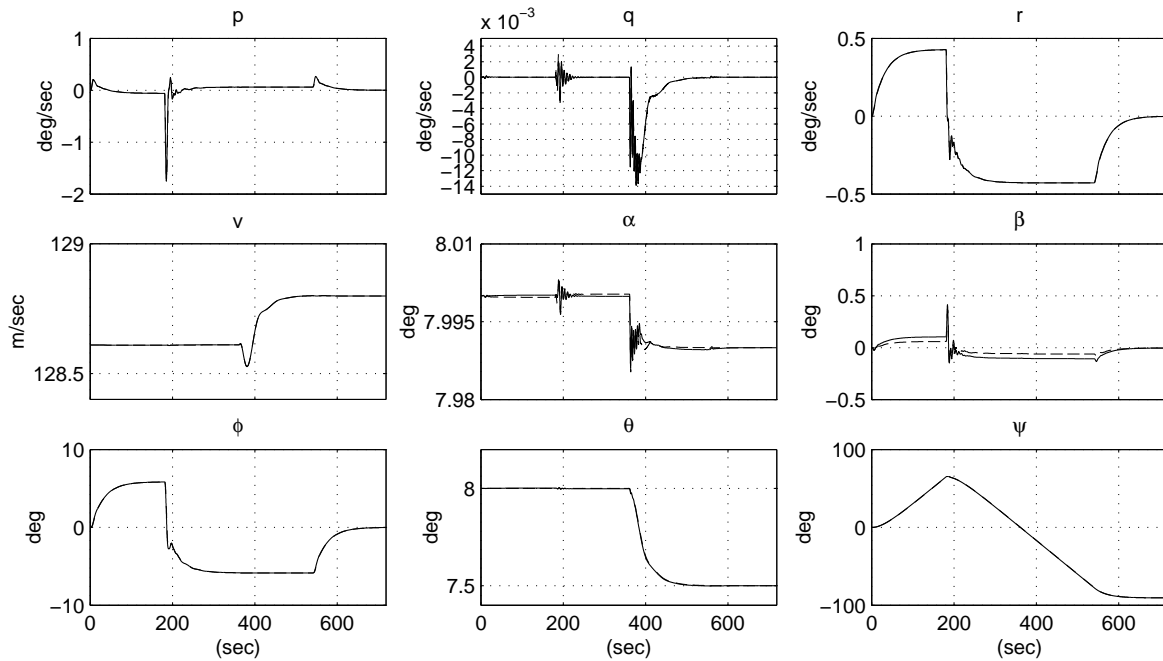


Figure 4.8: Linear Nominal Model: Inner Loop Tracking Performance
Reference (solid) and Actual (dashed)

This plot shows the nine states of interest of the reference model and the aircraft. The goal of the inner loop is to cause the aircraft to track the states v , α , β , ϕ , θ , ψ of the reference model. Note that the tracking is very good, as in most places the reference (dashed) and actual (solid) lines are indistinguishable. The oscillations seen in α and q at 200 and 400 seconds are due to the short period mode; at a frequency of $0.16Hz$ and at their small magnitude they are not significant as they are seen in both the reference and model as well as the actual aircraft. However, it is not known how to properly encode the desire to damp this particular frequency into MPC.

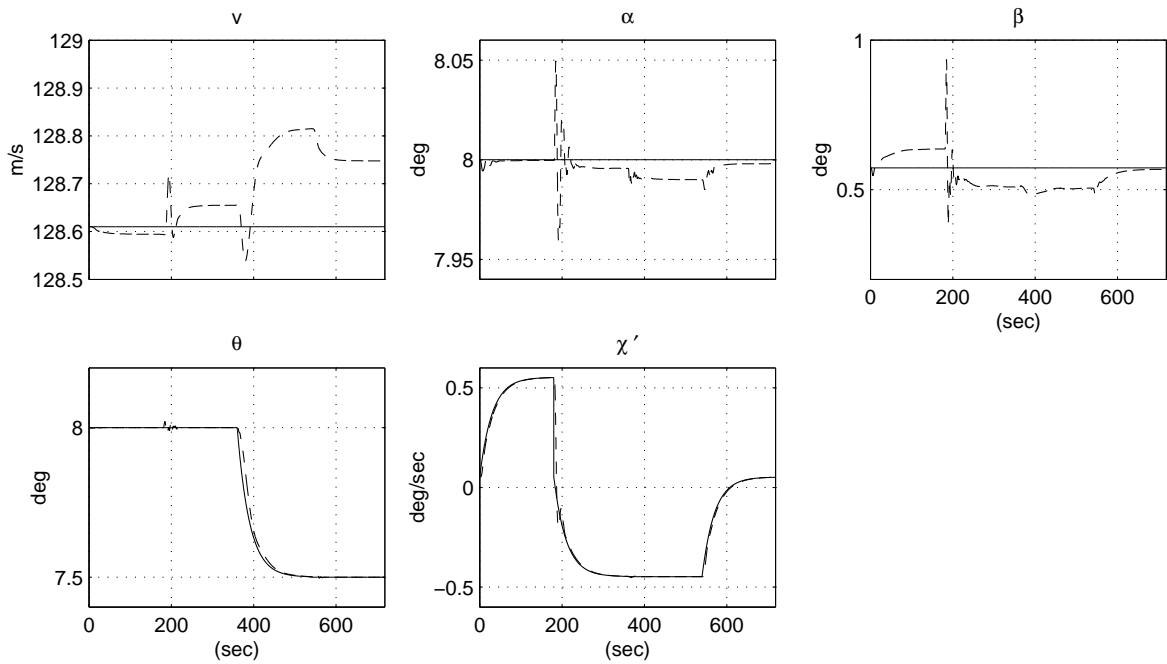


Figure 4.9: Linear Failure Model: Outer Loop Trajectory Tracking
Desired (solid) and Actual (dotted) Trajectories

This plot shows the pilot's ability to track the reference trajectory for the failed linear model. Comparing the plot to Figure 4.4, one can see that the tracking is almost identical for θ and $\dot{\chi}$. The velocity v and the angles α and β drift from their trim values about twice as far as for the nominal model, but are still well within the output constraints.

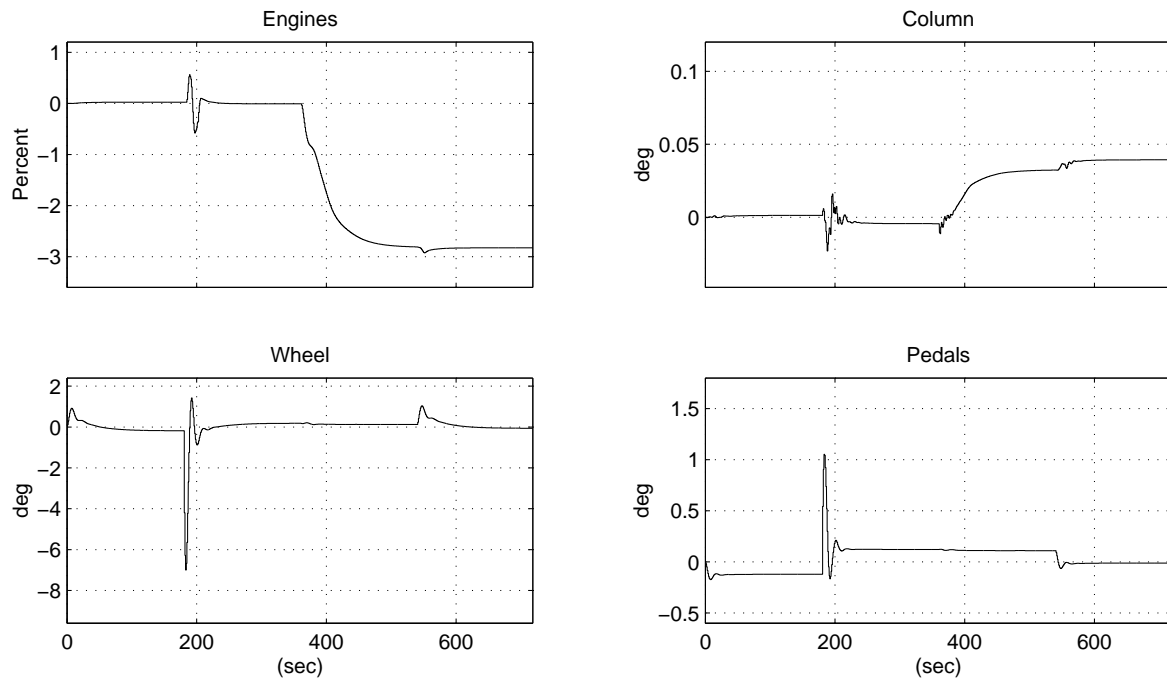


Figure 4.10: Linear Failure Model: Pilot Input Commands

Comparing this plot to Figure 4.5, one can see that the pilot takes almost identical actions while flying both the nominal and failed aircraft. This is evidence that the reconfiguration has worked as desired for the linear models.

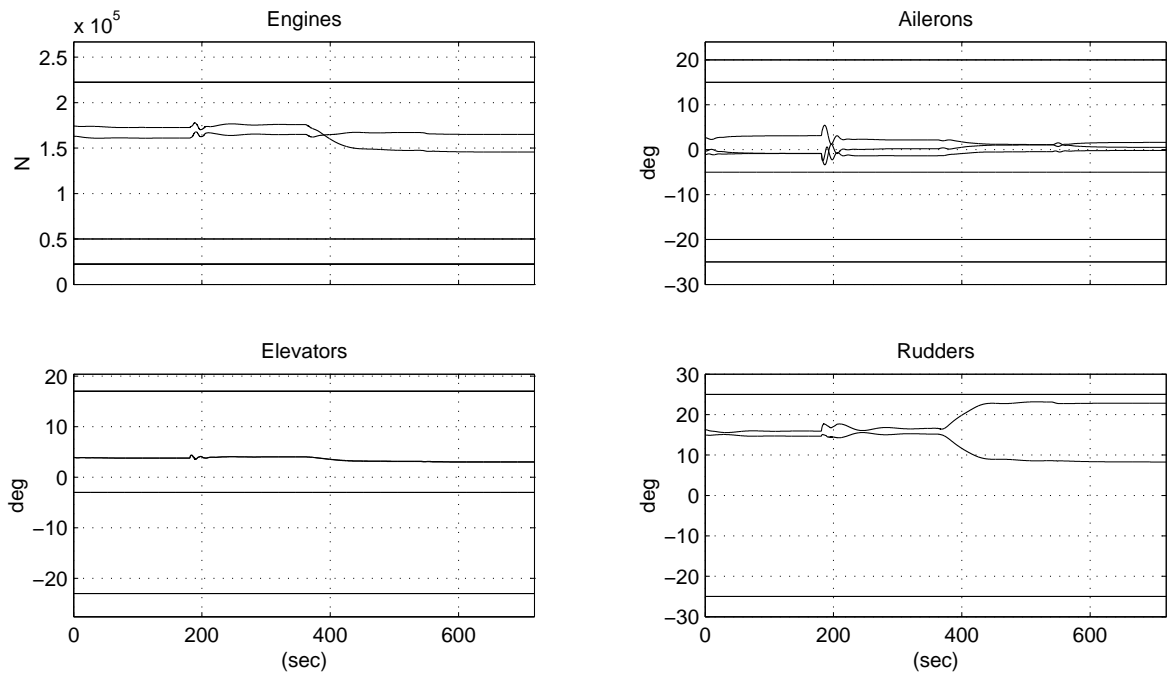


Figure 4.11: Linear Failure Model: Scaled Actuator Commands and Physical Limits

Comparing this plot to Figure 4.6 it can be seen that significantly more control action is required by the remaining control surfaces to control the aircraft. Note that no constraints are active during this manoeuvre and as a result, a linear control law would suffice. However, for this very gentle manoeuvre, with no disturbances, significant control action is required and as a result one would expect that a slightly more demanding manoeuvre, or the presence of wind, would require the consideration of actuator limitations.

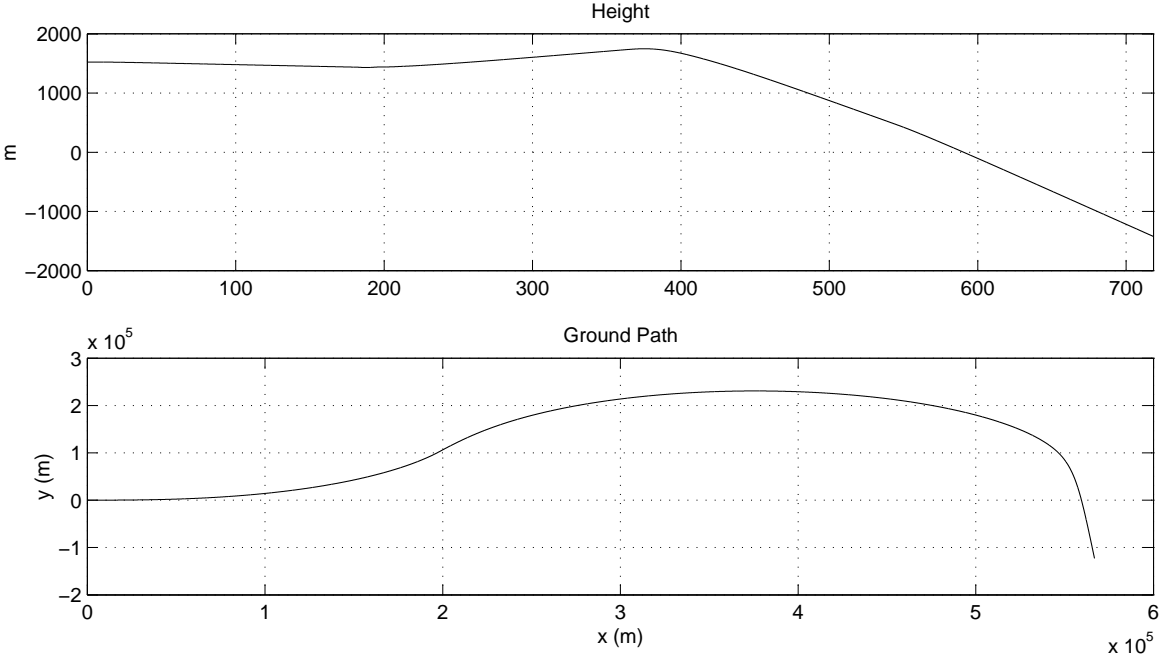


Figure 4.12: Linear Failure Model: Height and Ground Path

This plot shows that the failed model is able to track the desired trajectory in a similar fashion to the nominal one (Figure 4.7).

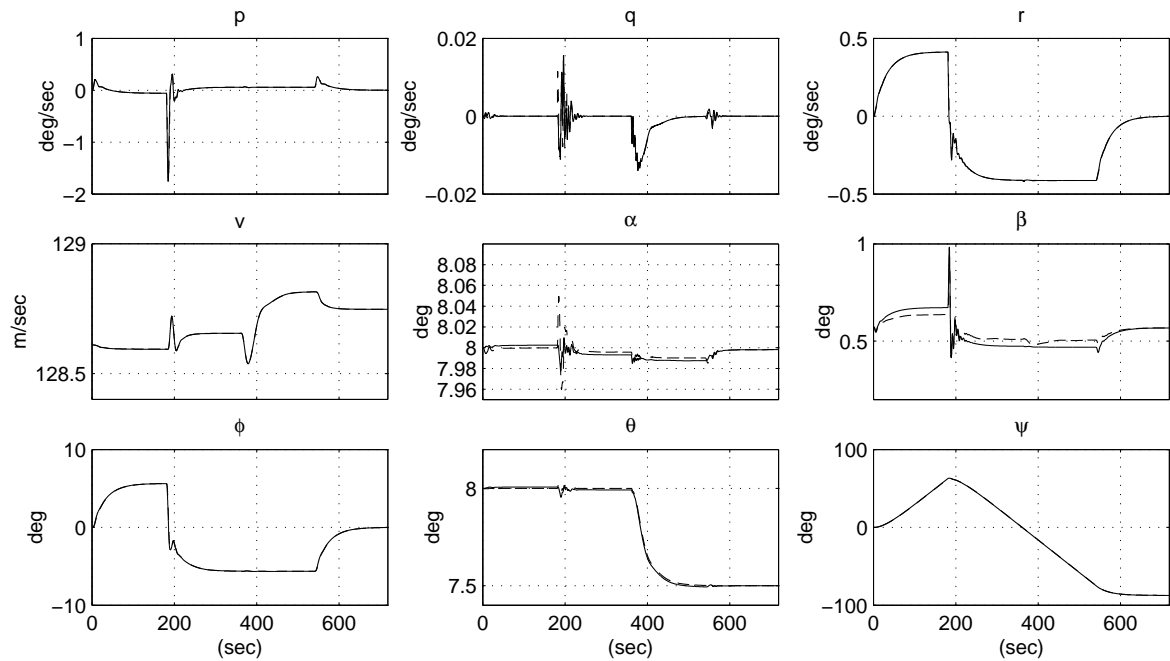


Figure 4.13: Linear Failure Model: Inner Loop Tracking Performance
Reference (solid) and Actual (dashed)

This plot shows that the reconfigurable MPC controller is able to track the nominal reference model almost exactly, despite the failures. This is not unexpected, as the controller's internal model is exact.

Non-linear Simulations

The plots in this section are the result of running the algorithm on the full nonlinear model. The results are discussed below each figure.

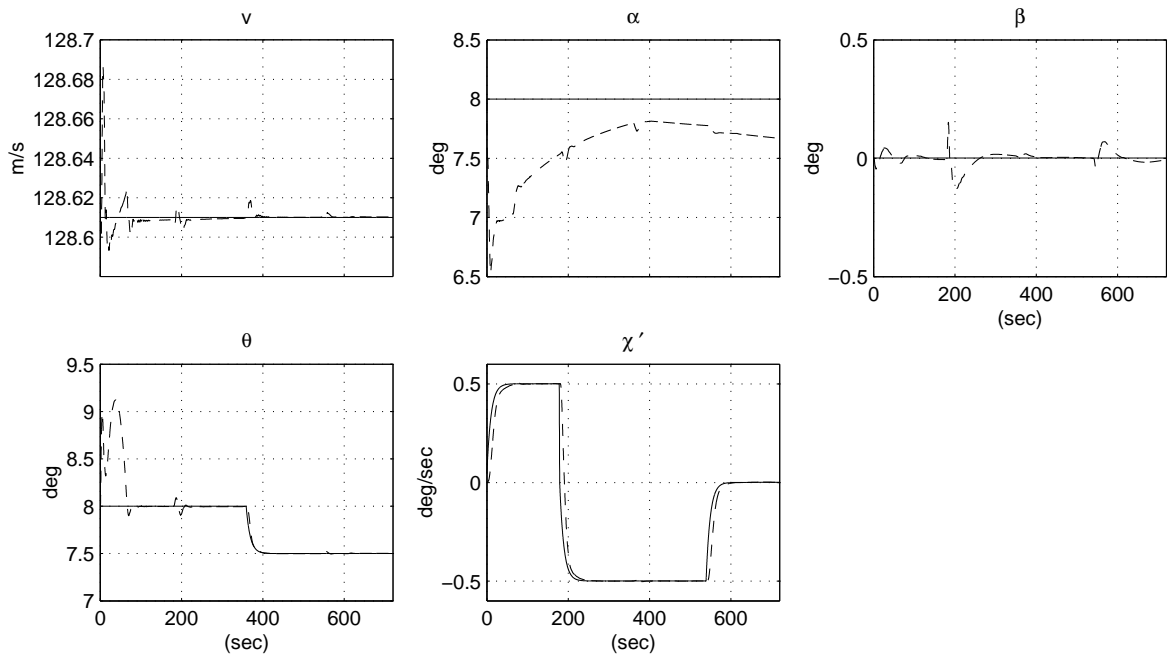


Figure 4.14: Nonlinear Nominal Model: Outer Loop Trajectory Tracking
Desired (solid) and Actual (dotted) Trajectories

This plot shows the ability of the pilot to track the reference trajectory for the nonlinear aircraft model. Note that both the inner and outer loop controllers are identical to those in the linear case, as is the linear reference model. There is one notable difference between the performance shown in this plot and that of the nominal linear model in Figure 4.4: In the first few seconds of the simulation, there is a pitch up of 1° and a corresponding change of -1° in the angle of attack. As the pitch is being controlled, but not the angle of attack (other than to keep it within 1° of trim) the pitch returns to the trim condition much faster than the angle of attack. The result is that there is an increase in the flight path angle γ , and therefore we expect the plane to increase in altitude where it did not in the linear case.

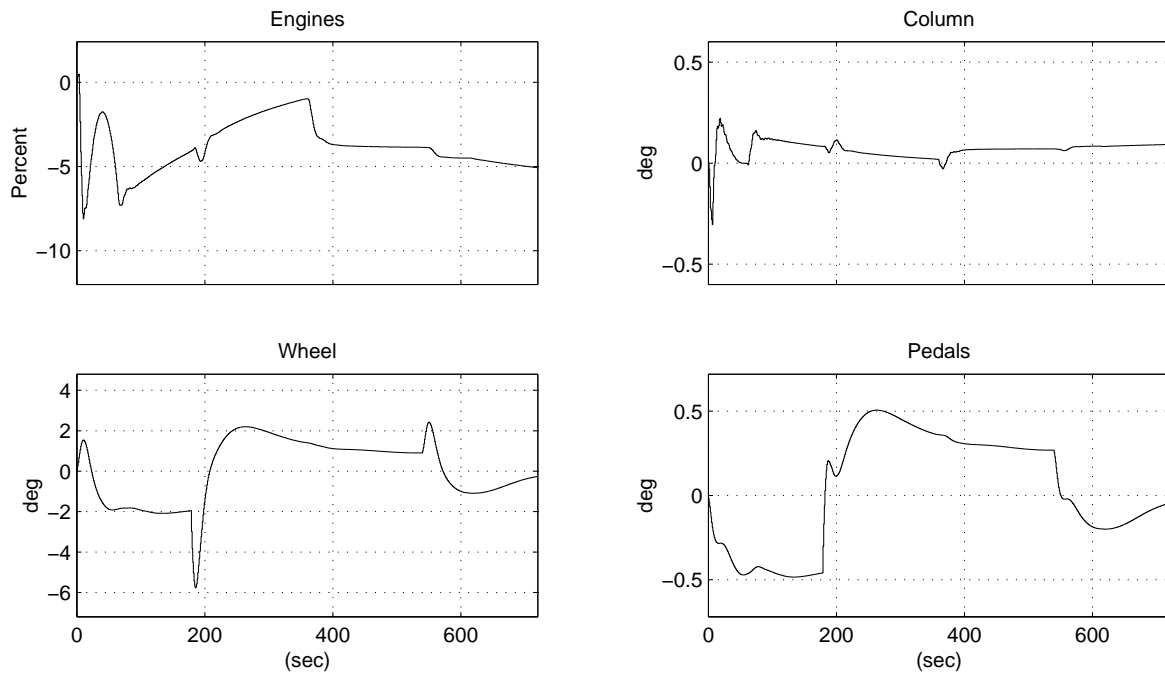


Figure 4.15: Nonlinear Nominal Model: Pilot Input Commands

One can see that significantly more movement is needed by the pilot for the nonlinear case than the linear (Figure 4.5). However, the movements of the column, wheel and pedals are very similar to those of the linear model, albeit scaled. The pulse in the engine during the first minute is due to the attempt to compensate for the initial pitch-up discussed in Figure 4.14.

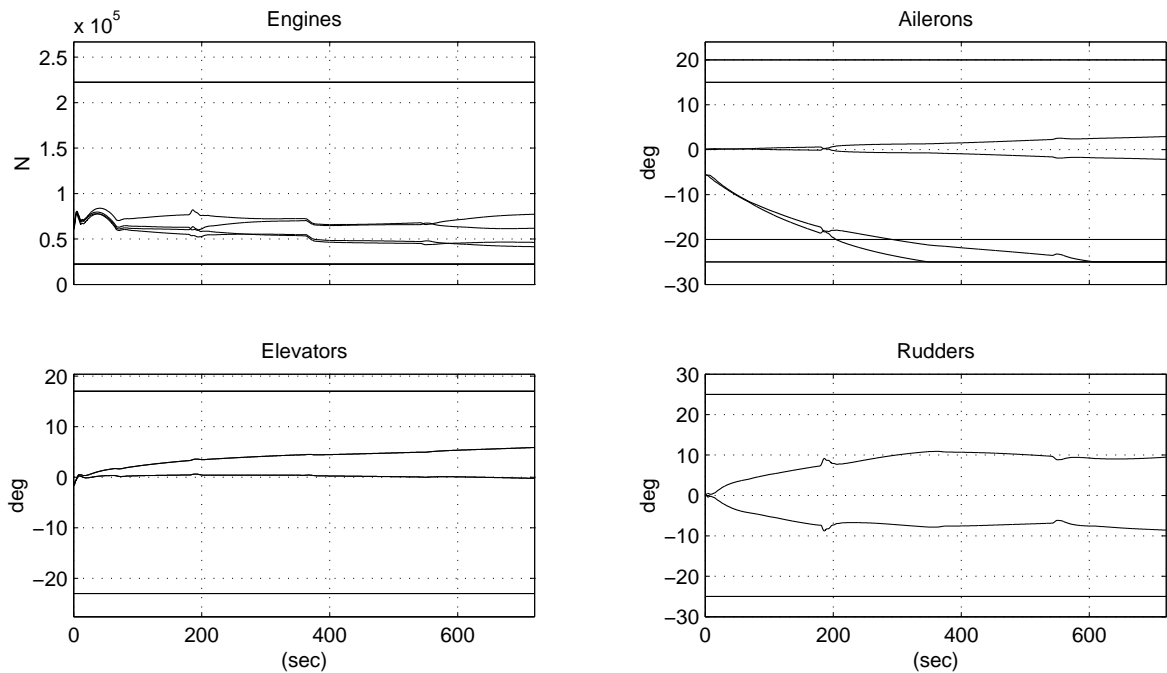


Figure 4.16: Nonlinear Nominal Model: Scaled Actuator Commands and Physical Limits

The main item of note on this plot is that the actuators move significantly more than in the linear case (Figure 4.6). In particular, the outer ailerons wind up and reach their lower mechanical limits at 300 and 600 seconds. The reason for this is not entirely clear, although examining the step responses for the engines (Figure A.2), ailerons (Figure A.3) and elevators (Figure A.4), the following reasoning is proposed: The longitudinal response of the aircraft to change in throttle is poorly modelled, especially in the angle of attack (Figure A.2). This can account for the initial decrease in α seen in Figure 4.14. The controller is now trying to drive α back to trim while holding θ constant, and in effect control the flight path angle, γ . By ganging together the two outer ailerons, a response very similar to a weak elevator can be obtained. The controller then takes advantage of the slight difference between the θ and α responses of the true elevator and this pseudo elevator to control them independently. One can see that while the outer ailerons are decreasing, the elevators are increasing. The controller does not attempt to keep the actuators near their trim conditions as this would prevent offset-free tracking (see Section 2.2). However, the goal of keeping the actuators near their trim conditions does seem necessary to prevent actuator windup, and as a result a more complex cost function or a multi-objective formulation may be needed. A second explanation lies in the linear-nonlinear mismatch discussed below in Figure 4.18.

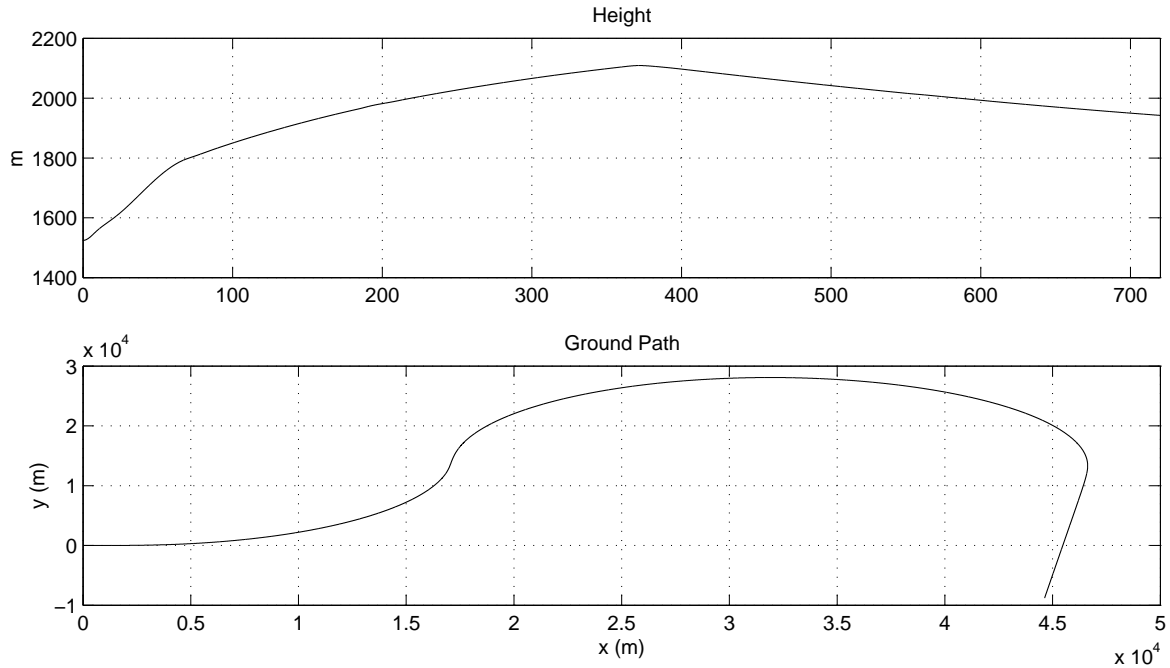


Figure 4.17: Nonlinear Nominal Model: Height and Ground Path

By comparing this plot to Figure 4.8, one can see that the ground path is approximately the same as in the linear case. Note that the outer loop is tracking $\dot{\chi}$ rather than χ , and as a result there is no reason for the ground path to match the linear case exactly. The height trajectory is significantly different than in the linear case. Again, this is due to the choice of tracked variables in the outer loop. The height is not being tracked, but rather the pitch is controlled, while the angle of attack is held within bounds. The result is that the flight path angle (and hence the altitude derivative) is controlled within bounds. The error in α seen in Figure 4.14 causes the offset in the flight path angle. However, as the goal is to study the performance of the inner loop, this is not considered to be a problem.

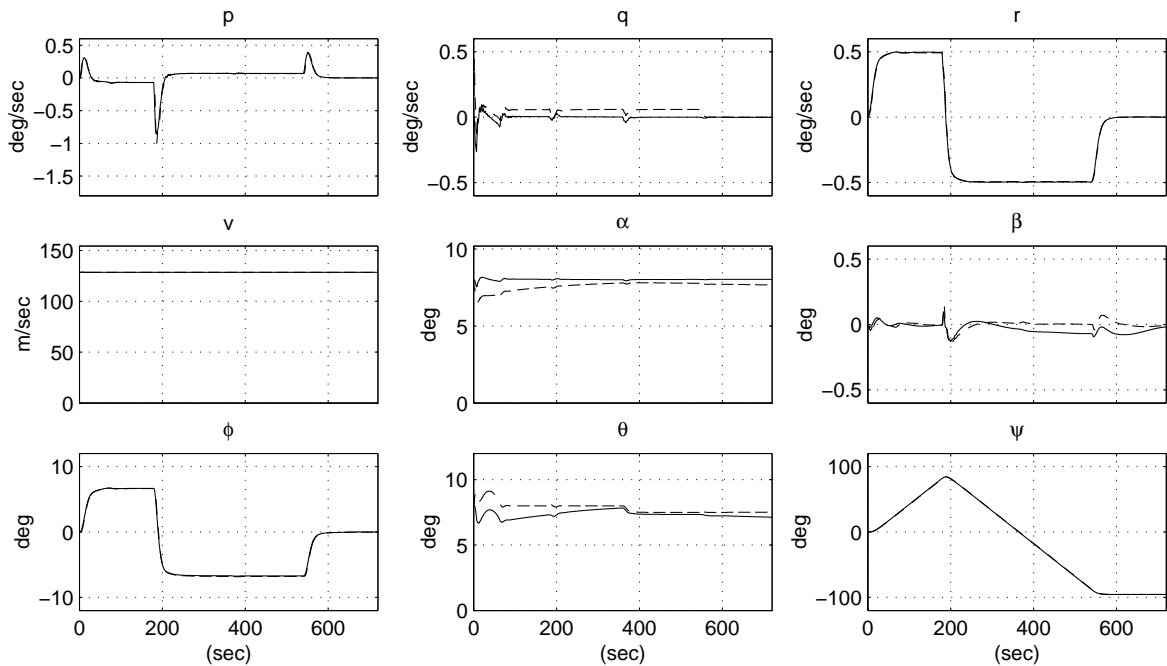


Figure 4.18: Nonlinear Nominal Model: Inner Loop Tracking Performance
Reference (solid) and Actual (dashed)

This is the main plot of interest as it shows the ability of the inner loop to track the reference model. One can see that the lateral states are tracked perfectly, while there is a slight offset in the longitudinal ones. This is for two reasons. Firstly, the model mismatch in the longitudinal direction is far worse than the lateral (the reason for this is still being investigated). Secondly, the linear approximation $\dot{\theta} = q$ is not valid when the roll angle is non-zero. This is the reason for a non-zero q despite the fact that the derivative of θ is clearly zero. This is not a significant problem for a minor manoeuvre, but it becomes more pronounced with larger roll angles. A possible solution to this problem is to use a nonlinear reference model which will be proposed in Chapter 5.

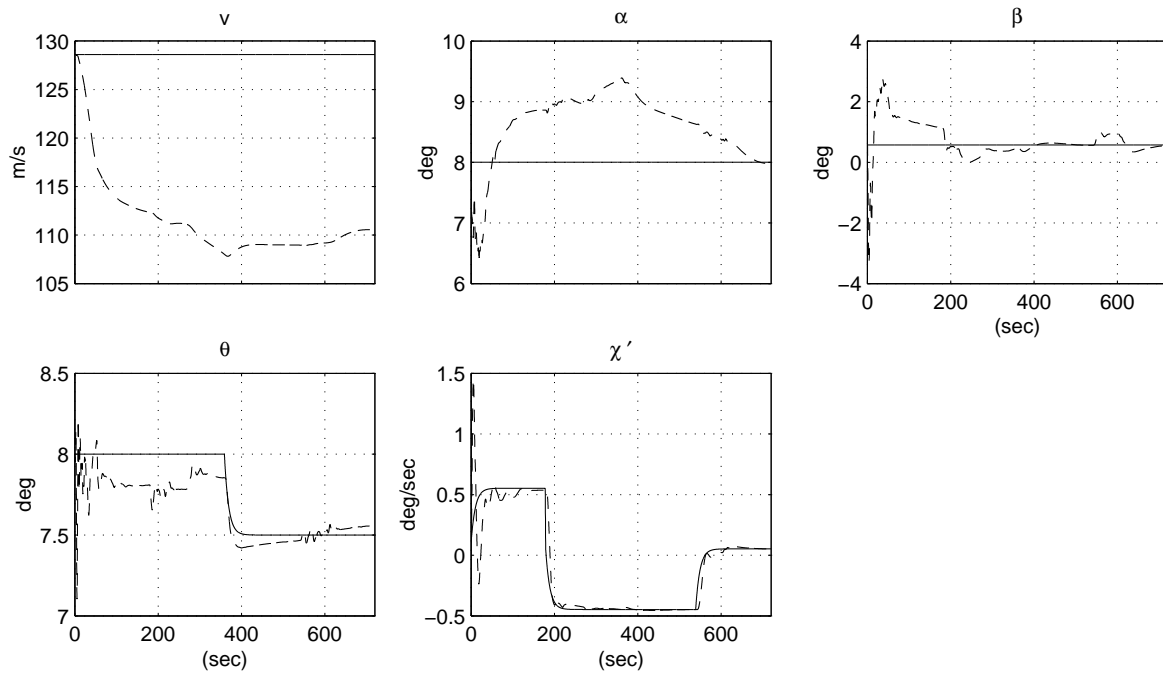


Figure 4.19: Nonlinear Failure Model: Outer Loop Trajectory Tracking
Desired (solid) and Actual (dotted) Trajectories

One can see from this plot that although the tracking performance of the failed nonlinear aircraft is not especially good, it is both stable and able to respond to commands from the pilot. It was found that in order to maintain stability, the velocity must be allowed to decrease, which is the reason for bounding the velocity rather than trying to drive it to the trim condition. Again, the lateral control is superior to the longitudinal and this is still being investigated.

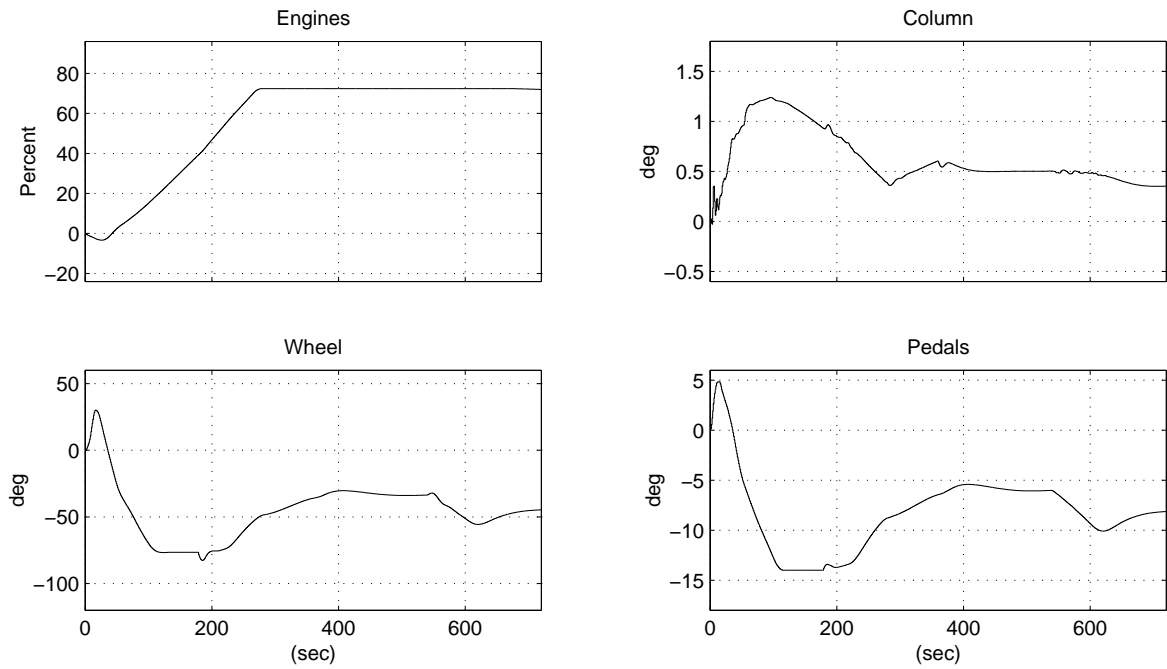


Figure 4.20: Nonlinear Failure Model: Pilot Input Commands

The pilot must now make far more drastic movements to control the plane. Apart from the large increase in throttle, the pilot's control movements are similar in some ways to those in the nominal case. However, as there are in fact differences we have not achieved our goal of full performance restoration in the presence of failure. It is also not clear if the plane is operating in a manner close to that expected by the pilot. A better method for determining this will be discussed in Chapter 5.

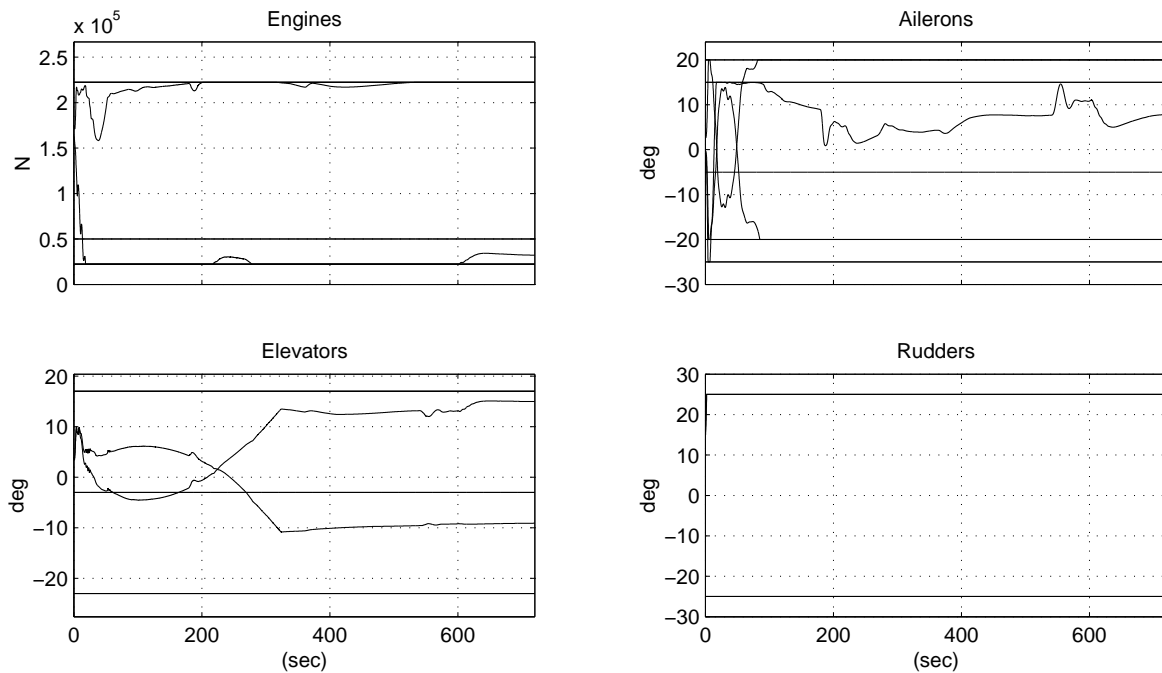


Figure 4.21: Nonlinear Failure Model: Scaled Actuator Commands and Physical Limits

It is clear from this plot that this manoeuvre is near the limitation of what is possible for the damaged aircraft. Maintaining the sideslip angle within limitations is using a great deal of the control authority; both rudders are full right, resulting in the engines being used for yaw control. The Boeing 747 is designed to be able to lose thrust from two engines on the same wing and still have sufficient control authority from the rudders after they have been used to trim the aircraft. This is clearly not the case here, which would suggest an error in either the simulation or the model. However, similar results were seen in the data recordings from flight EL AL 1862, implying that the model is correct and that the loss of the engines or the slow speed accentuates the problem. Much less action would be required of the actuators if the sideslip was not required to be held within tight constraints. However, while this may be acceptable during manoeuvres, it is not during a landing approach. Clearly, taking actuator limitations into account is a requirement to control the aircraft during this manoeuvre and a nonlinear controller is necessary.

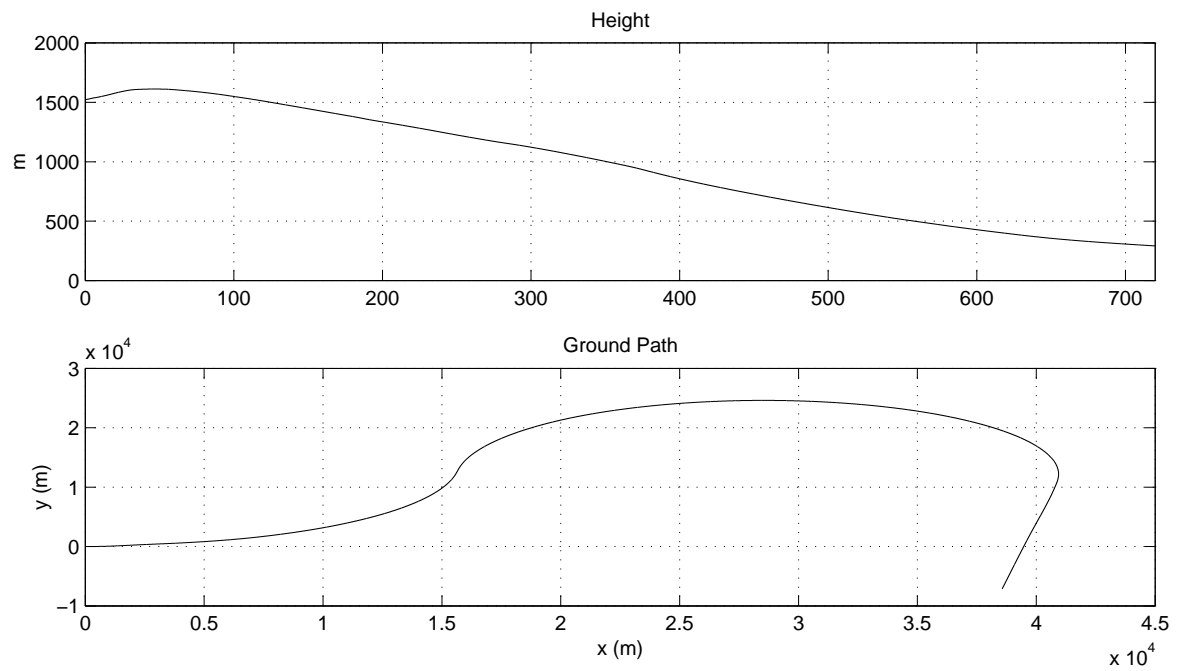


Figure 4.22: Nonlinear Failure Model: Height and Ground Path

It is clear from this plot that the damaged aircraft is flying a similar path as in the other cases.

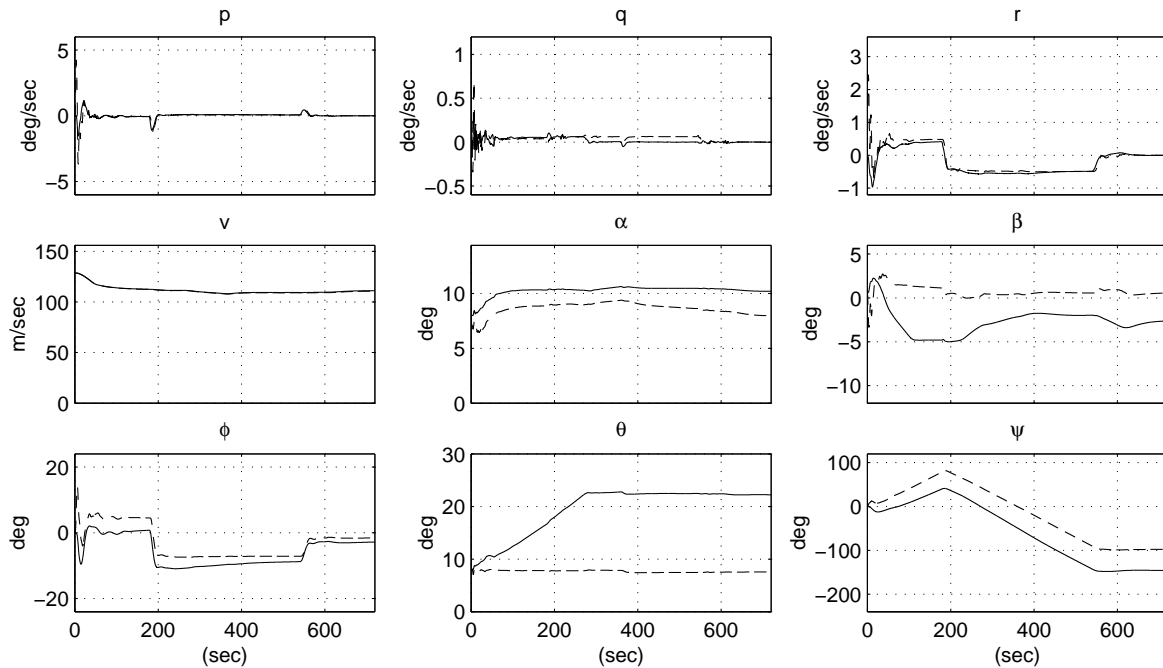


Figure 4.23: Nonlinear Failure Model: Inner Loop Tracking Performance
Reference (solid) and Actual (dashed)

There are three items of note in Figure 4.23: First, the angular rates p , q and r are well tracked. Second, the velocities v , α and β as well as the angles ϕ and ψ are tracked, although there is a constant offset. The inner loop is implemented as discussed in Section 2.2, which should give offset-free tracking for all constant output disturbances. This is clearly not the case here, implying that the disturbance model used in Section 2.2 is not valid for the type of model/plant mismatch seen here. Finally, the reference pitch angle is winding up, while the actual angle θ is not following. The reason for this is not understood, but is likely to lie in the significant model/plant mismatch displayed in Appendix A.

4.3.5 Trajectory Visualization

A tool has been written in C++, using OpenGL for display, which aids in the visualization of an aircraft's trajectory. The progression of the state is saved in ASCII format from Matlab and then read into the program `vtraj`, which is available at `/users/cnj22/work/vtraj`. The program plots the trajectory in 3D space and then places aircraft pictures at regular intervals along its length, in order to show their orientation. Currently, models of the Boeing 747, the F-16, and the SR-71 are available. The user can then move and rotate the viewpoint in order to get a feeling for the trajectory travelled. A screenshot is shown as Figure 4.24, showing

the first portion of the trajectory of the failed aircraft discussed in the previous section.

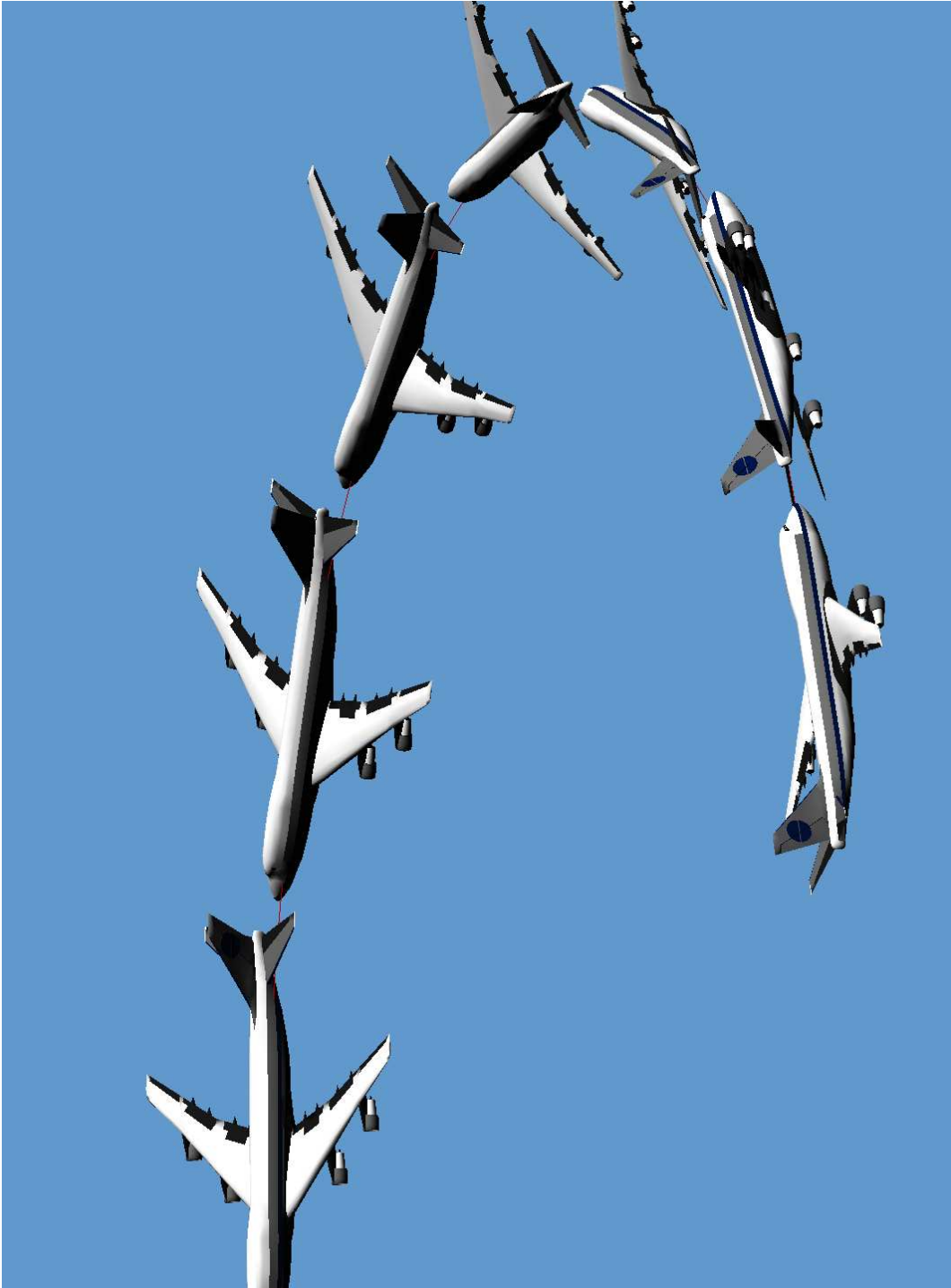


Figure 4.24: Trajectory Visualization Tool

4.3.6 Conclusions

This chapter has introduced the notion of reconfiguration with the purpose of pilot control through the use of MPC model following. The AL EL 1862 crash has been studied and a reconfigurable control system demonstrated which would have allowed the pilot to continue flying the aircraft. This method was able to control both the damaged and working aircraft through maneuvers comparable to those executed before the crash.

However, the solution presented here was tuned to this particular situation and there is no guarantee that it would be effective over the entire flight envelope, or any other scenario. The performance target here was nothing less than the full restoration of nominal control, although it may well not be necessary to restore nominal performance in general, and thus a method of determining how much performance is required, as well as a reasonable measure of performance, is needed.

Comparing the pilot's actions during the failure situation seen in Figure 4.20 to those in the nominal case shown in Figure 4.15, it is obvious that he is not flying the failed plane as if it was working. The question arises: How close does the plane need to be to the nominal case in order that a well-trained pilot will be able to fly it using standard emergency procedures? Answering this question is non-trivial and studying it is proposed in Chapter 5.

The detection of a failure situation by a pilot onboard an aircraft is not nearly as easy as would be expected. For example, in the short term the only effect seen from the loss of two engines during a turn is a slight increase or decrease in turn speed depending on the direction of turn [Den96]. Pilots are trained to detect these signals, and to follow a set of procedures in handling emergency situations. Research into the effect of reconfiguration on a pilot's senses and on the actions he might take in an emergency would help it to find acceptance in the industry.

Consider the following list, which is an excerpt from [Den96] telling pilots of multi-engine planes what to do should one of them fail:

1. Starting from moderate speeds, as you slow down you will need more and more rudder to maintain coordinated flight. This is the coordinated regime. The amount of bank needed to maintain non-turning flight is basically constant.
2. There comes a point where you run out of rudder authority and cannot maintain coordinated flight. As the speed decreases further, the slip angle automatically increases, and more boat turn gets added to the pseudo boat turn. This is the uncoordinated regime. The bank angle must increase as airspeed decreases if you want to maintain non-turning flight.
3. You can maintain control down to V_{MC} (minimum control speed) in the takeoff configuration.

4. If you persist in engine-out flight down to sufficiently low airspeed, at some point the wings and/or rudder will stall and you will be very sorry.
5. If you are below V_{MC} , you should reduce power on the good engine, dive to regain V_{MC} , and then re-open the throttle on the good engine.
6. If you are below V_{YSE} , you should dive to regain V_{YSE} , obstructions permitting.
7. To clear distant obstacles, you want to dive to achieve V_{XSE} as soon as possible. To clear nearby obstacles, you don't want to dive below their altitude, obviously. For a combination of obstacles, you face some tricky tradeoffs. The best solution is to make sure you never get into a low-altitude low-airspeed situation.

This type of expert knowledge is obviously not present in the MPC controller used to simulate the pilot in this chapter and it is not immediately clear how to incorporate it. It is clear that this knowledge could have improved the controllability of the aircraft in the example presented in this chapter. For example, in Figure 4.21 both rudders are full over and the engines are in an asymmetric configuration in order to prevent sideslip. However, as seen in the above list, the recommended solution is to allow a small left roll in order to maintain straight and level flight.

Chapter 5

Research Proposal

This chapter will summarize the work presented in this report and outline the directions for future research in the area of fault tolerant flight control for large aircraft.

5.1 Summary of Progress

A large portion of this year has been spent developing background knowledge in control, and specifically on the problem of fault tolerant aircraft control. To this end, an online database of relevant papers has been created and a survey written.

A result of the survey is the recognition that model based predictive control has many properties that are useful to the fault tolerance problem. MPC has been studied and a `Matlab` toolbox written, through which a familiarity has been gained with the workings and many of the problems of MPC.

A particular crash scenario, namely the Amsterdam EL AL 1862 flight, has been studied and an understanding of the types of failures that can be expected on a large jet aircraft developed. The MPC toolbox has been used to develop a reconfigurable controller which allows the pilot to continue flying despite significant failures. The system has been configured such that it can be used as a benchmark for future ideas.

The survey presented in Chapter 3 deals strictly with aircraft reconfiguration. There is, however, a large collection of literature on general reconfiguration. An extended survey covering these techniques and discussing how the work proposed here fits into this larger field is required.

5.2 Research Proposal

The theme of this work will be the study of reconfigurable control as applied to civil aircraft. As demonstrated in Chapter 4, given an appropriate failure scenario and a correct model,

an MPC controller can restore control to the pilot of a failed aircraft in a model-following framework. The primary questions of interest are then: What if the model is not correct? Is the model-following approach general enough to handle all possible failures?

I believe that these questions can be posed as a matter of achievable performance in a model-following framework. Given a particular model and uncertainty description, what performance can be achieved? Specifically, I am interested in valid methods of defining and computing the achievable performance of an aircraft as it applies to the ability of a pilot to control a plane.

Ensuring reference tracking in the presence of uncertainty is what is required for the sufficient conditions for pilot control as described in Chapter 4. As discussed in Section 4.2, this is still an open problem in the literature when constraints are included. In the short term, investigations will be undertaken into existing work in this area with the long-term goal of applying and extending these ideas to the tracking problem outlined in Section 4.2.

Several interesting sub-problems have been identified and are discussed in the following list:

Planning The effect of aircraft control surfaces varies with altitude and velocity. For example, although the amount of asymmetric thrust in an engine-out scenario does not depend on airspeed, the amount of force the rudder produces depends on the square of the airspeed. Therefore as the aircraft decelerates, progressively more rudder deflection is needed in order to maintain zero side slip. Calculating minimum safe landing speeds (if they exist) or computing the flight envelope (maximum bank angle, etc) would be valuable information for the pilot.

According to [Den96], the correct response to an engine out is to bank slightly in the direction of the working engines. With a damaged aircraft, an important goal would be to compute a given orientation or trajectory which would maximize control authority, or other desirable parameters.

Questions that we are interested in answering for the pilot are:

- Will there be sufficient control authority at the runway?
- What orientations/velocities should be avoided?
- What orientations/velocities should be targeted?

It is expected that sufficient conditions for these questions can be posed as feasibility questions for the model-following problem discussed in Section 4.2.

Aircraft Performance → MPC Cost A key lesson learned from developing the reconfigurable controller in Chapter 4 is that it is very difficult to describe the goals of an

autopilot in terms of a single cost function for MPC, and that tuning to a particular aircraft and failure situation is required for stability and performance. However, it will not be possible to tune a reconfigurable controller *a priori* for a particular failure scenario, and therefore the method must be able to handle general situations. Possible solutions may lie in the direction of non-quadratic costs, multi-objective prioritized cost functions or physical control notions such as Total Energy Control [Lam83].

FDI Modelling Many current reconfiguration techniques assume a model of the failed system. However, fault detection and isolation techniques do not generate fault models instantly, and often they are very uncertain with the level of uncertainty decreasing over time. The types of uncertainty seen in many FDI models does not necessarily lend itself to standard uncertainty description methods. For example, a FDI method might return the statement that the inner *or* the outer aileron is jammed. There is very little on this topic in the reconfiguration literature and it appears to be an open problem.

5.3 Improvements to the Boeing 747 Model

The items in this section follow from the lessons learned while developing the reconfigurable controller for the EL AL 1862 crash. The following list describes some improvements for the aircraft model which will improve its use as a test case for reconfigurable control.

Model Simplification The Boeing 747c model is currently prohibitively complex and as such it is difficult to analyse the effects of failures. The model is constructed from recorded flight data and as a result, there are a large number of lookup tables. It would be preferable, for many applications, to have a simpler, nonlinear model which is fairly accurate. Such a model could also be used as the reference or internal models for the model-following setup discussed in Chapter 4.

Monte-Carlo The complex model does not allow for ready modification of parameters in order to simulate various similar aircraft in order to test for robustness. The simplified model should improve this situation.

Performance Measure A set of criteria needs to be developed which can quantitatively differentiate between two reconfiguration methods.

Flying Qualities Is it reasonable to force the damaged aircraft to track a particular model? Do a set of criteria exist which can be used to determine if the plane is flying in a manner amenable to the pilot.

Human Pilot Interface As mentioned in Chapter 4, designing a controller which simulates a human pilot is difficult and it is not clear if the plane is flying in a manner amenable

to human control. A possible solution would be to create a real-time, online simulator which could be flown by human pilots. Four possibilities exist for this development:

FlightGear FlightGear¹ is an open source, multi-platform, flight simulation program written in C++ and licensed under the GNU General Public License, which means that the source is freely available and modifiable. The graphical interface is highly advanced and virtually every land feature on the earth has been mapped, including Amsterdam airport. The underlying simulation code has its roots at NASA in the form of LaRCsim [Jac95] and has now expanded into JSBSim².

vtraj The existing trajectory visualization tool, **vtraj**, discussed in Section 4.3.5 could be extended into a full simulation package.

Cranfield Simulator Two fixed simulators exist at Cranfield which are available for use. Alternatively, the code has been offered and could be modified for use in the Cambridge Control Lab.

SIMONA A motion simulator is under construction at the SIMONA Institute at Delft in The Netherlands. The simulator sits on a six degree of freedom Stewart platform which is capable of accelerations between $0.02g$ and $1.5g$ at a bandwidth of $15Hz$. The project is looking for collaborators and an application could be made to test reconfiguration ideas on this new platform.

The benefit of developing a simulator from scratch, or modifying the existing open-source FlightGear code, is a familiarity with all levels of the simulation whereas it may be difficult to work with the closed-source options. However, the time required may not be worth the gains.

5.4 Fault Tolerant GARTEUR Group

A GARTEUR flight mechanics action group has been organized which will study fault tolerant control for aircraft. The purpose of the group will be to investigate the use of new FDI methods and their integration with reconfigurable control techniques. The Boeing 747c model discussed in Chapter 4 has been proposed as a test case for this work. The action group has members spanning Europe from ten universities in six countries as well as Airbus and Kinetic.

Once begun, the project will proceed in three phases, spanning a period of two years. Several documents will be required from this work. A survey/review will be prepared during the first phase, a technical report detailing a method of reconfiguration during the second and

¹FlightGear is available at www.flightgear.org

²JSBSim is available at jsbsim.sourceforge.net

a summary report during the third. The timeline of the project fits well with the timeline of the PhD, assuming that the GARTEUR project begins within the next six months.

5.5 Preliminary Schedule

Figure 5.1 gives a rough idea of the schedule for the proposed research program. Many of the items will be dependent on the scheduling of the GARTEUR action group.

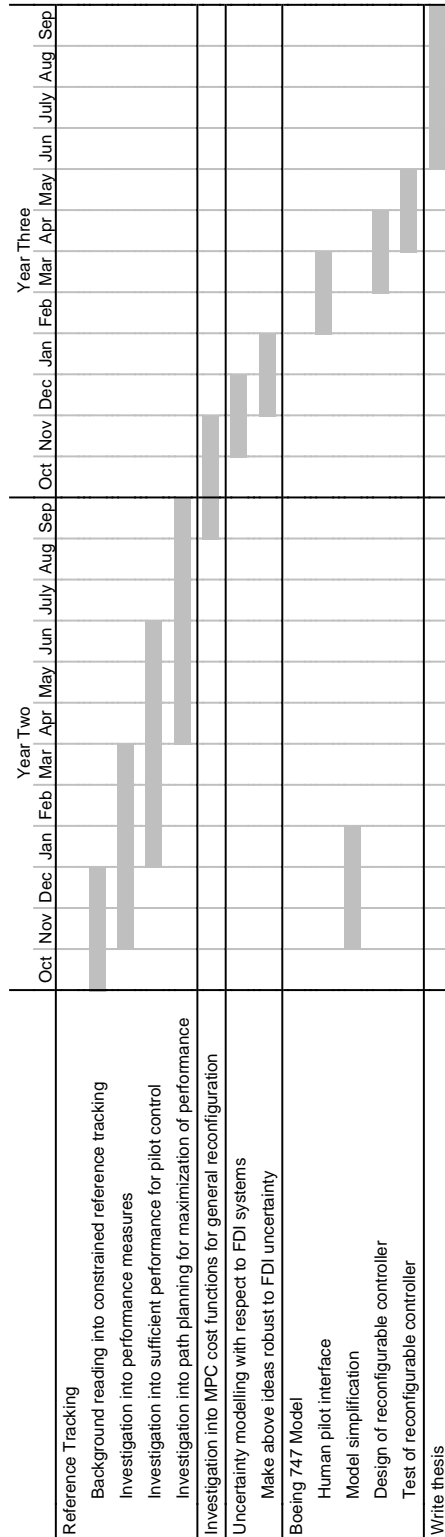


Figure 5.1: Preliminary Schedule

Appendix A

Step Responses

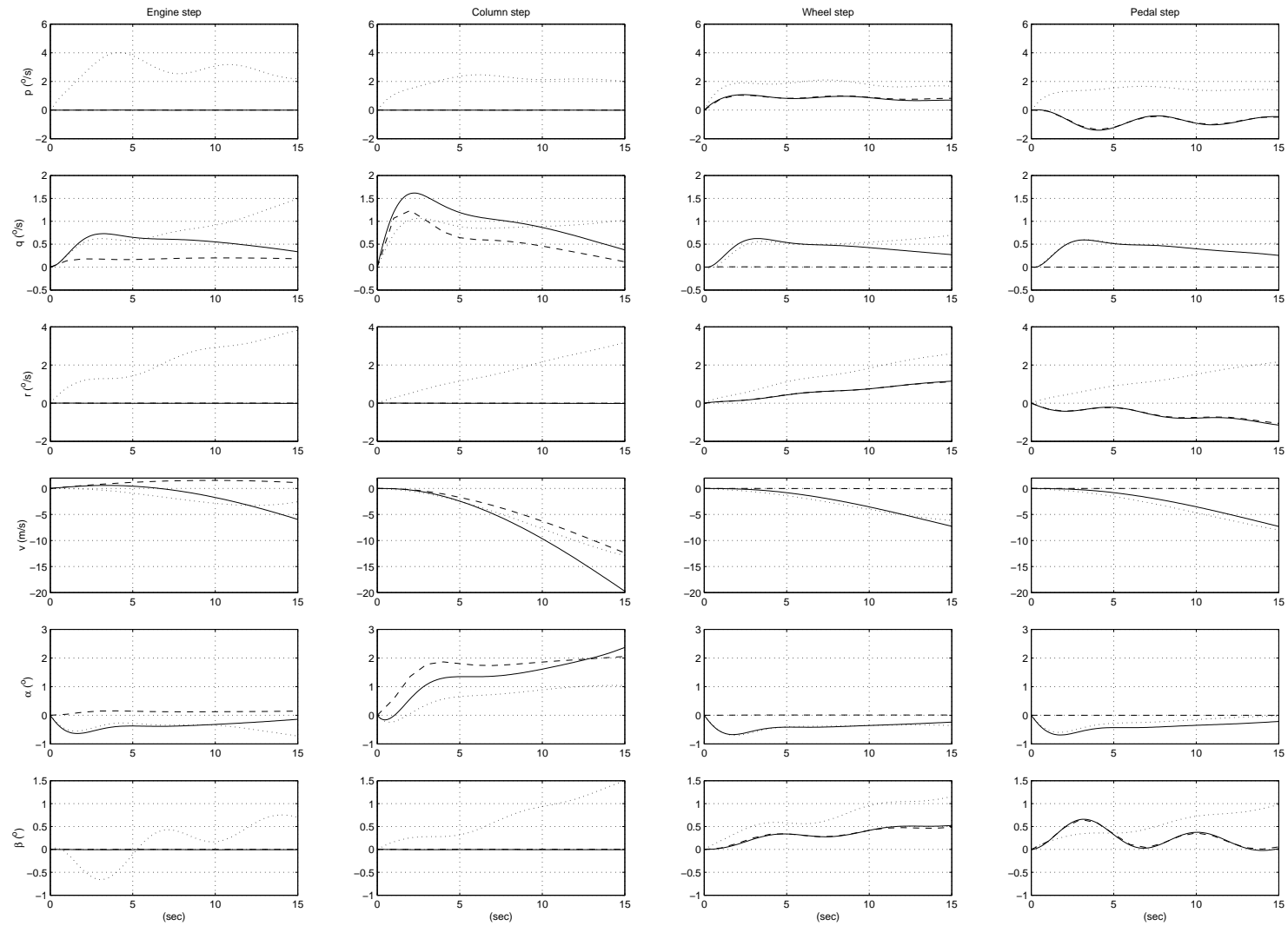


Figure A.1: 10% Step Response of Linear (dashed) and Nonlinear (solid) Models Trimmed at $v = 129\text{m/s}$, $h_e = 5000\text{ft}$ and $\alpha = 8^\circ$

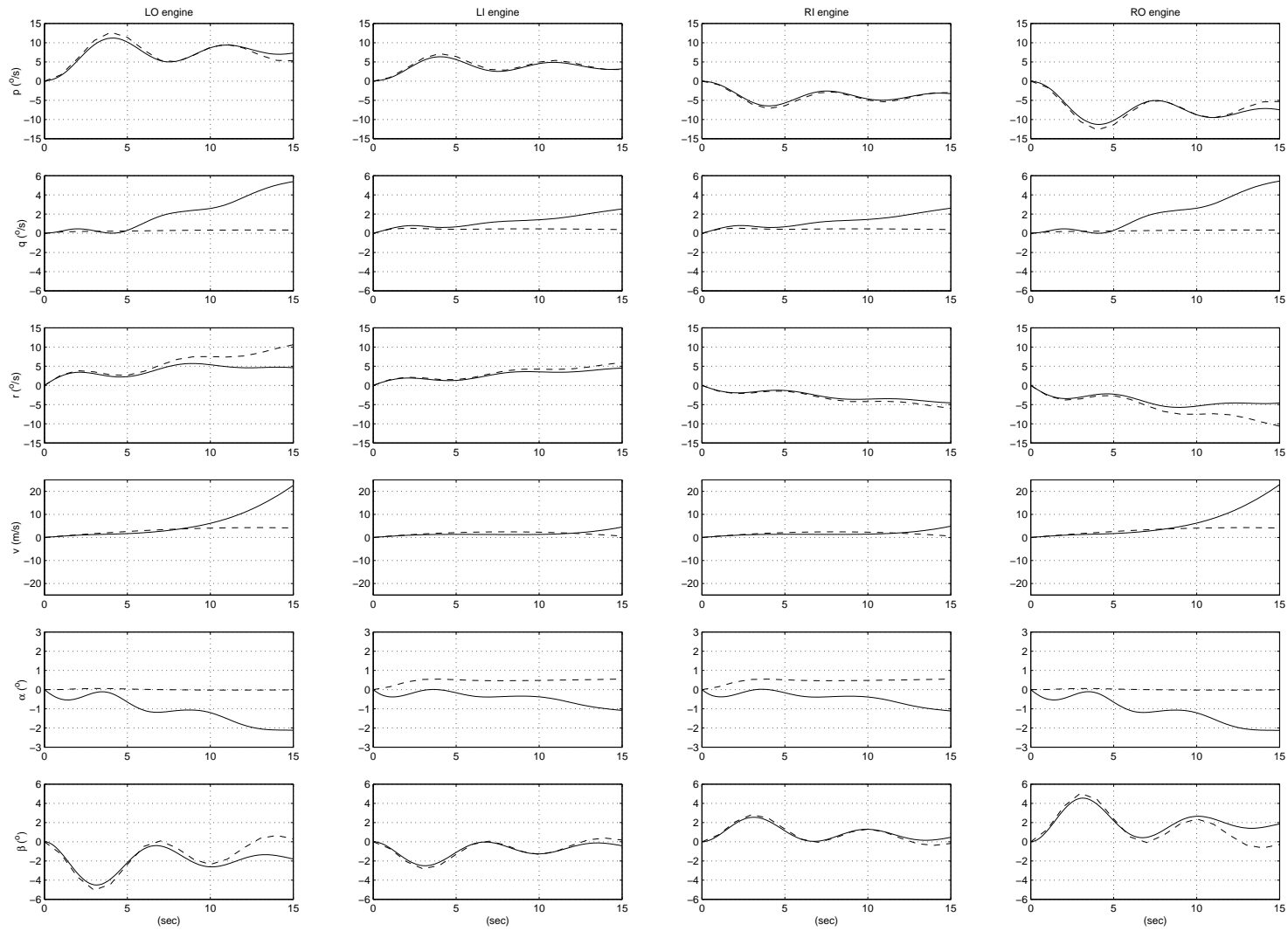


Figure A.2: 10% Engine Step Response of Linear (dashed) and Nonlinear (solid) Models
Trimmed at $v = 129m/s$, $h_e = 5000ft$ and $\alpha = 8^{\circ}$

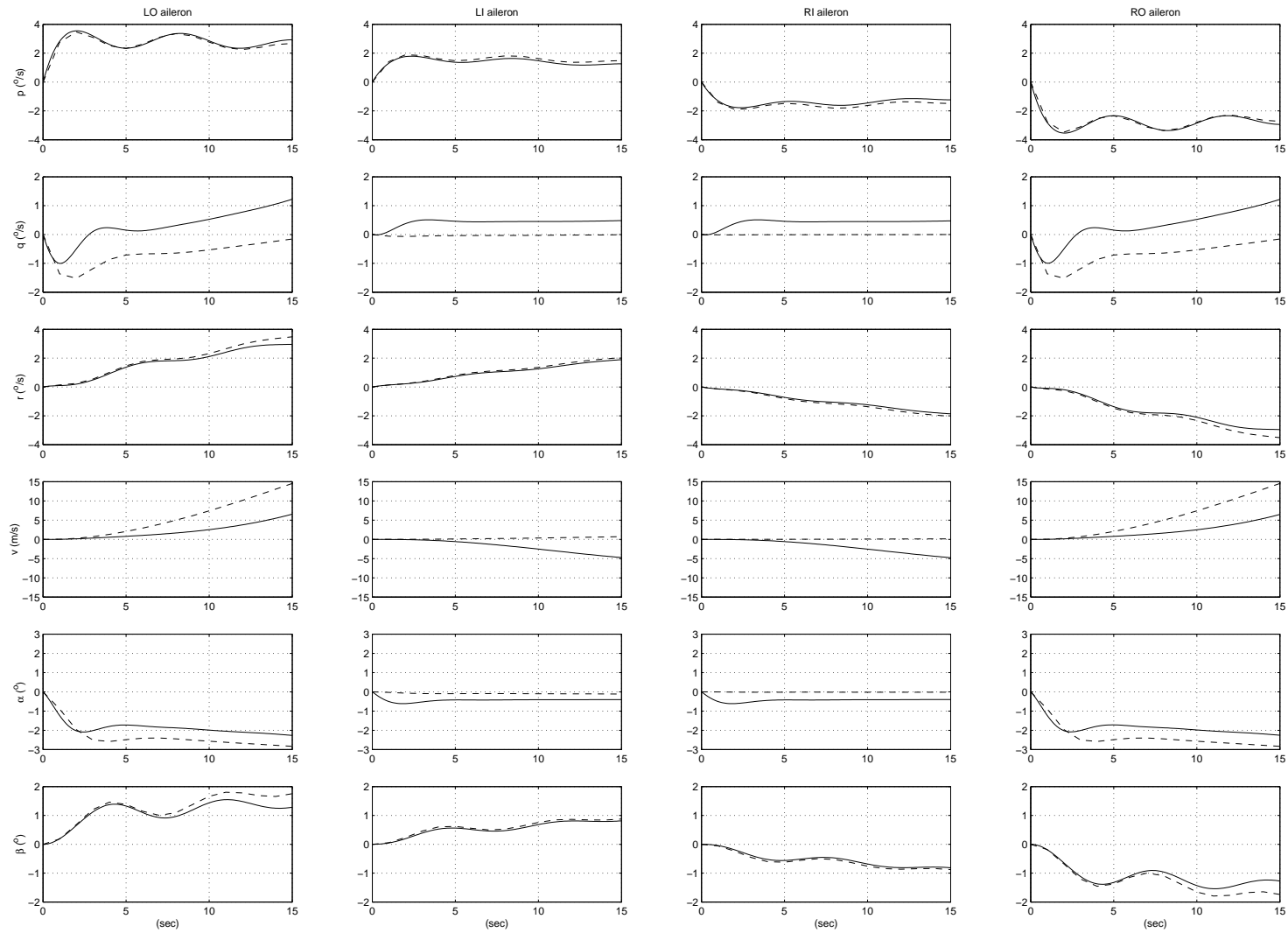


Figure A.3: 10% Aileron Step Response of Linear (dashed) and Nonlinear (solid) Models
Trimmed at $v = 129m/s$, $h_e = 5000ft$ and $\alpha = 8^\circ$

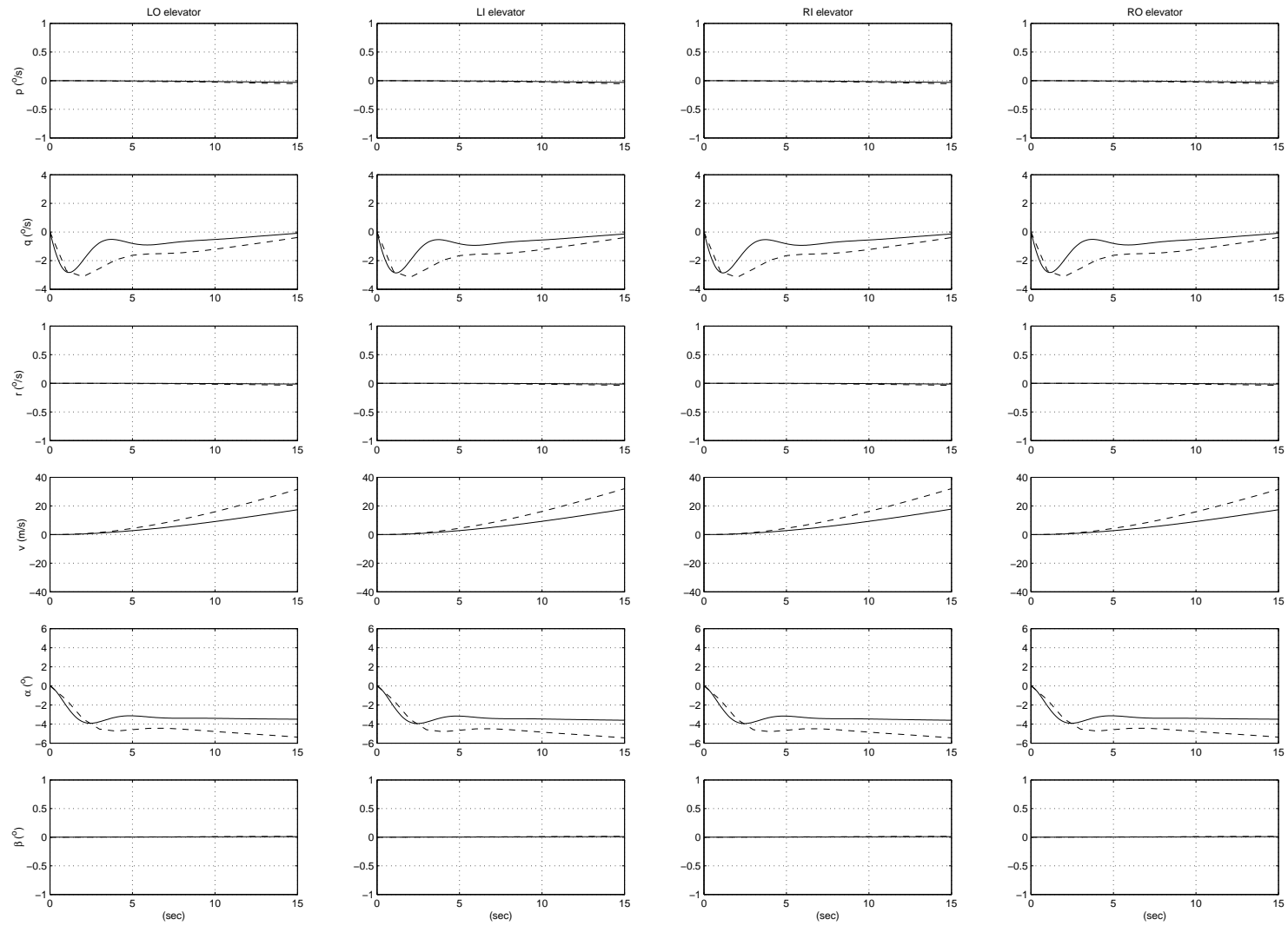


Figure A.4: 10% Elevator Step Response of Linear (dashed) and Nonlinear (solid) Models
Trimmed at $v = 129m/s$, $h_e = 5000ft$ and $\alpha = 8^{\circ}$

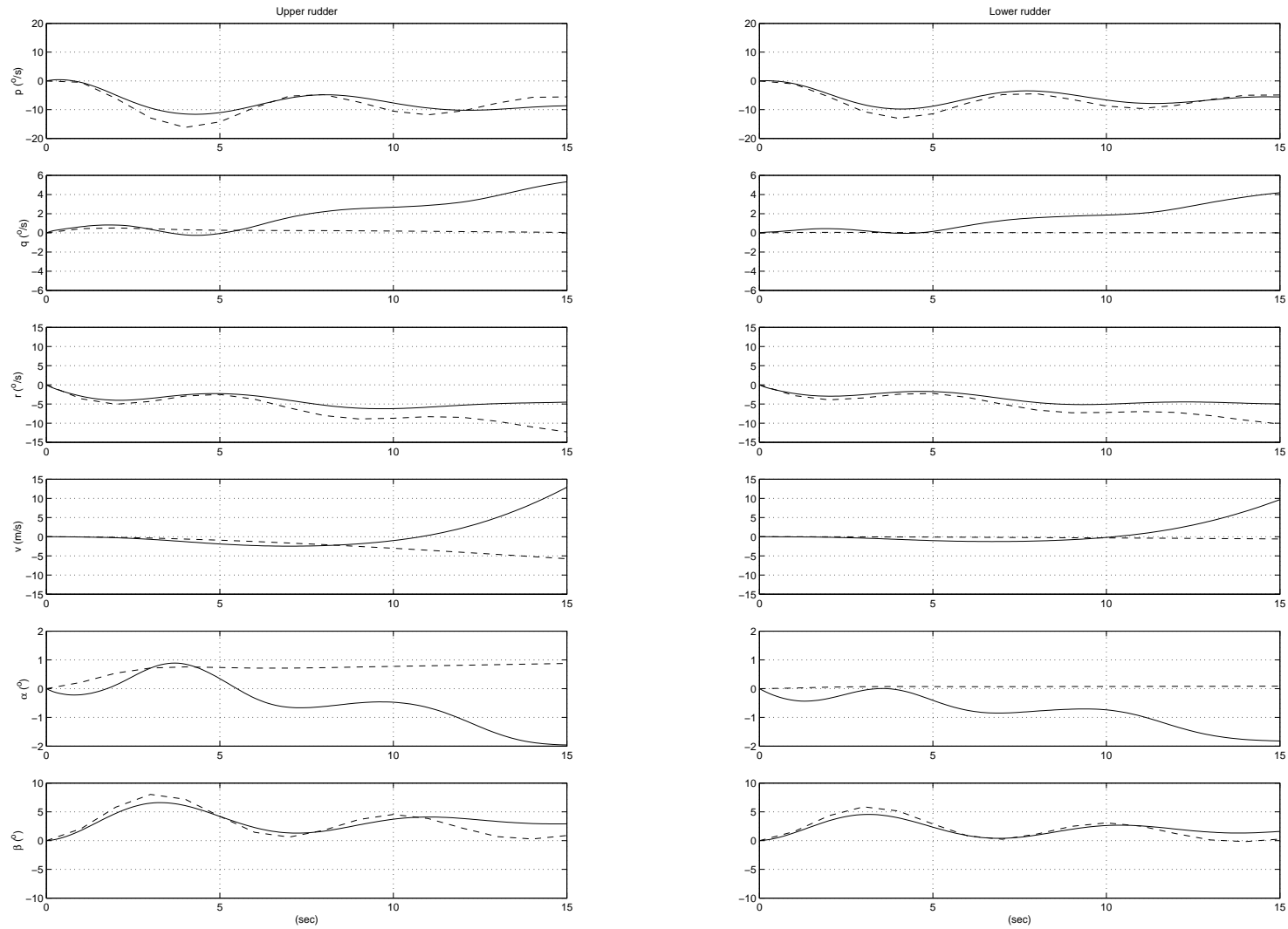


Figure A.5: 10% Rudder Step Response of Linear (dashed) and Nonlinear (solid) Models
 Trimmed at $v = 129m/s$, $h_e = 5000ft$ and $\alpha = 8^{\circ}$

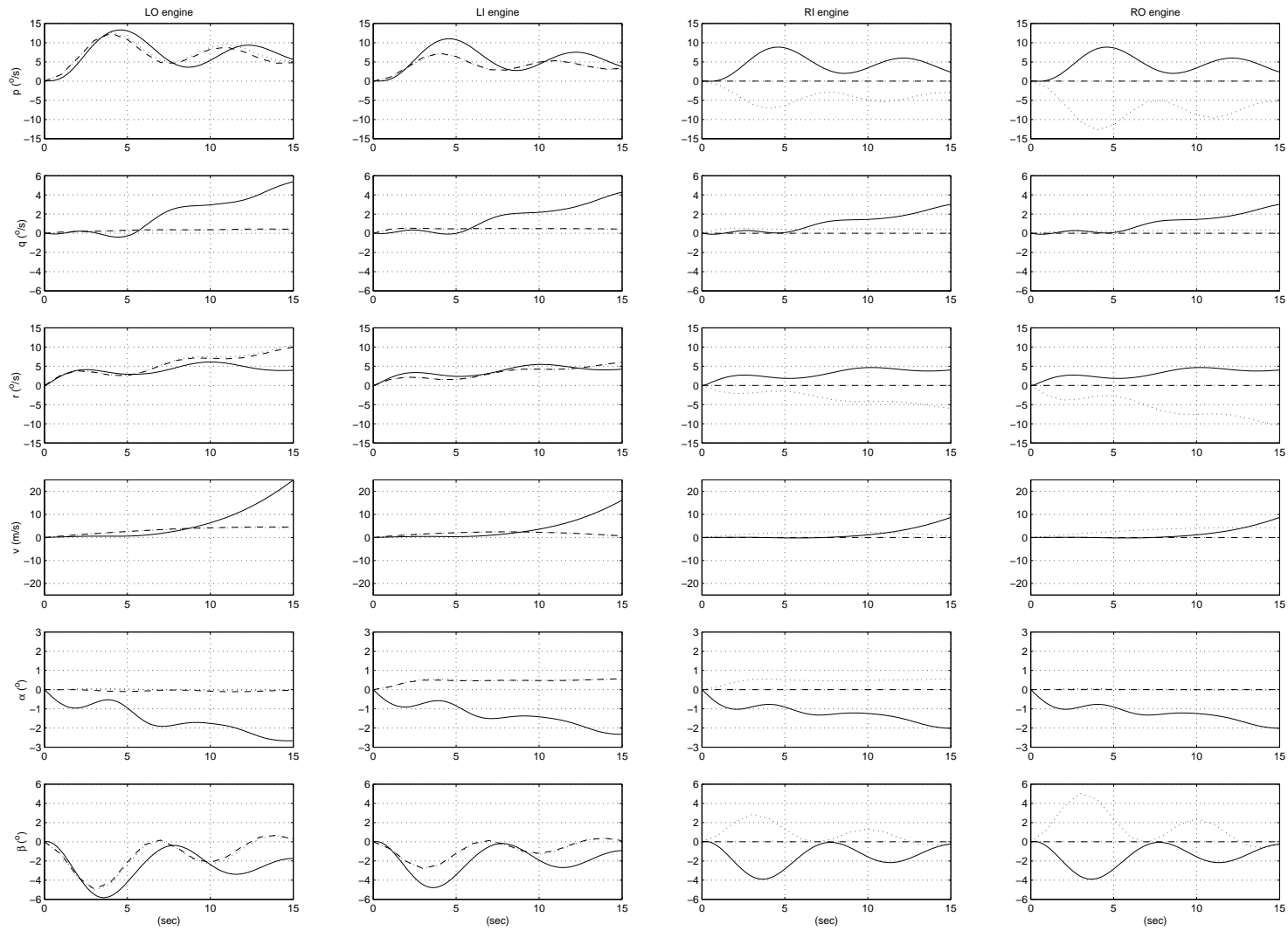


Figure A.6: 10% Engine Step Response of Linear (dashed) and Nonlinear (solid) Failure Models and Nominal Linear Model (dotted) Trimmed at $v = 129m/s$, $h_e = 5000ft$ and $\alpha = 8^{\circ}$

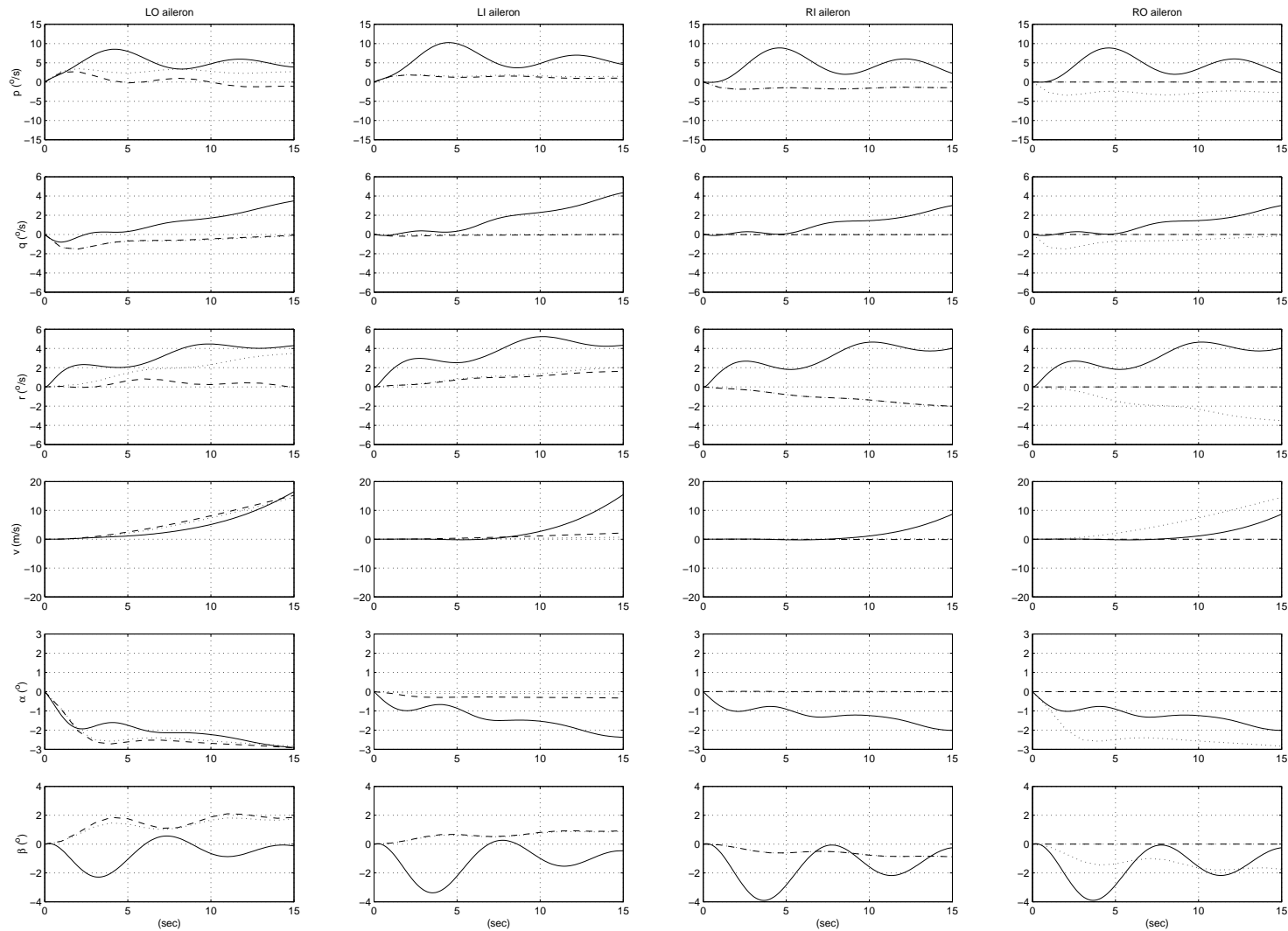


Figure A.7: 10% Aileron Step Response of Linear (dashed) and Nonlinear (solid) Failure Models and Nominal Linear Model (dotted) Trimmed at $v = 129m/s$, $h_e = 5000ft$ and $\alpha = 8^{\circ}$

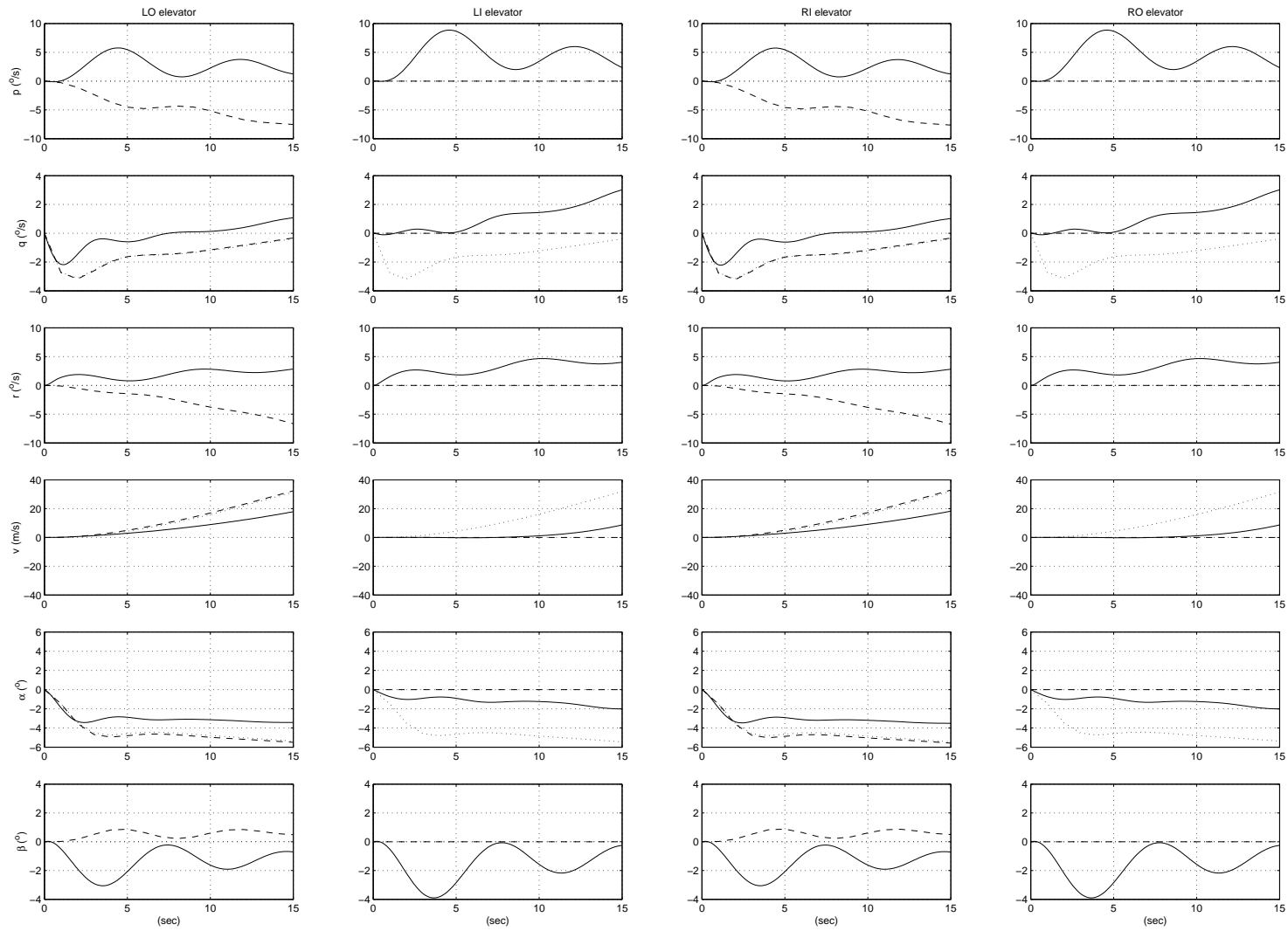


Figure A.8: 10% Elevator Step Response of Linear (dashed) and Nonlinear (solid) Failure Models and Nominal Linear Model (dotted) Trimmed at $v = 129m/s$, $h_e = 5000ft$ and $\alpha = 8^{\circ}$

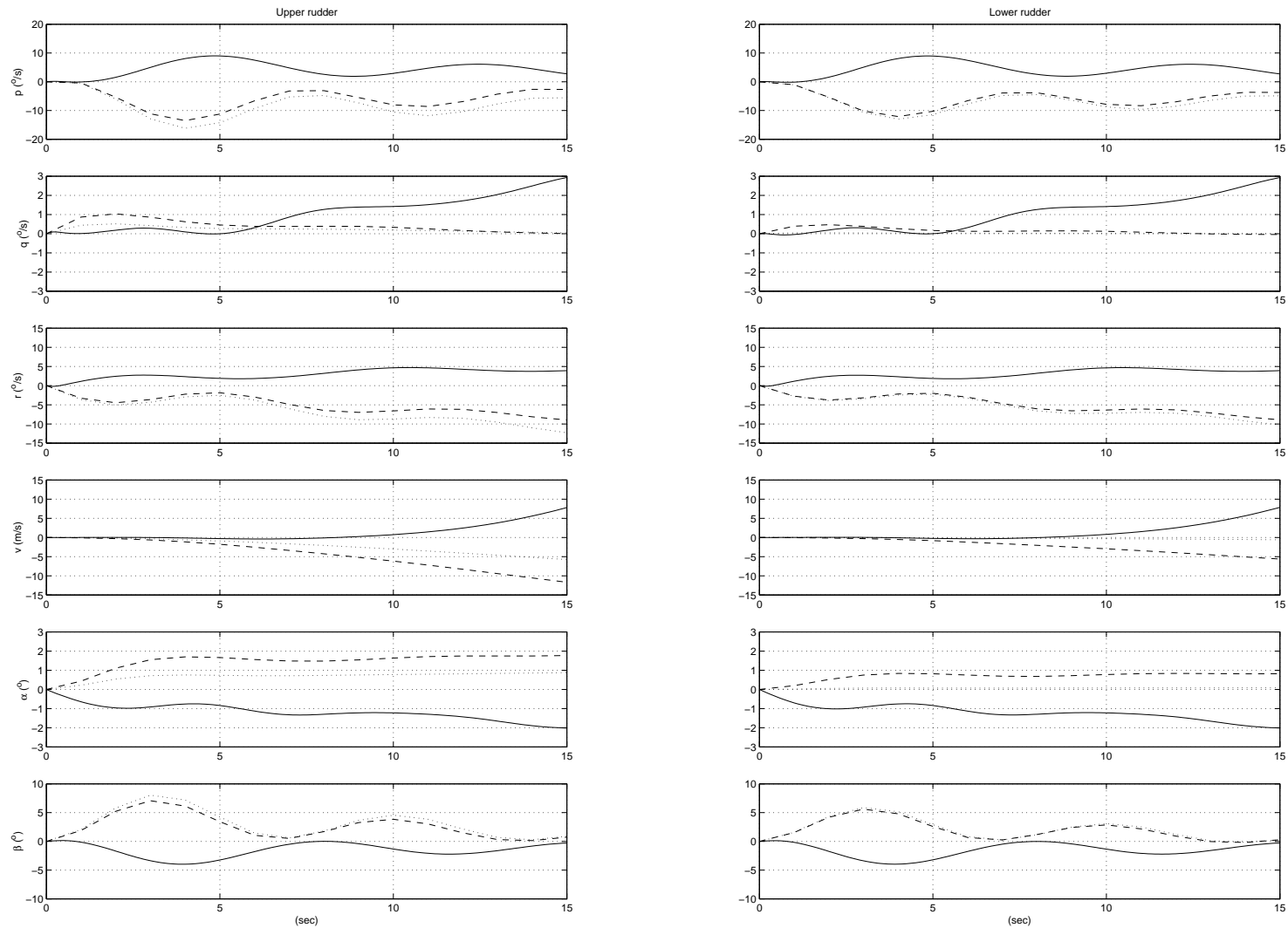


Figure A.9: 10% Rudder Step Response of Linear (dashed) and Nonlinear (solid) Failure Models and Nominal Linear Model (dotted) Trimmed at $v = 129m/s$, $h_e = 5000ft$ and $\alpha = 8^{\circ}$

Appendix B

jmpc Toolbox

A toolbox has been created which implements the MPC algorithm as described in Section 3.8. This Appendix will give a brief overview of the toolbox and provide the code.

The toolbox is used in two steps: during the initialization, a `jmpc` structure needs to be created which stores all of the relevant information about the model. At each step, the function `jmpcMove` is called which takes the structure, current state and measurements and returns the estimated state and current output move.

The following list gives a brief description of the members of the `jmpc` structure:

- `sys`: A state-space `lti` object of the internal model.
- `pred`: Constants used to compute predictions; see [Mac00, Page 62] for details.
- `const`: Constraints in polytope form.
- `horz`: Prediction and control horizons.
- `weights`: Weights for the quadratic cost function.
- `kest`: State-space `lti` object; estimator.
- `Kmpc`: Closed-form, unconstrained MPC solution.
- `Kfull`: Closed-form, unconstrained MPC solution for all steps up to the control horizon.

The remainder of this Appendix gives the `Matlab` code for the toolbox.

B.1 Code

B.1.1 jmpc Model Creation

BuildMPCModel

```

% jMod = BuildMPCModel(sys,Hw,Hp,Hu,Q,R,imag,irate,omag)
%
% INPUTS:
% sys: discrete LTI model of the system
% Hw: first prediction to weight
% Hp: prediction horizon
% Hu: control horizon
% imag: input magnitude limits (default = inf)
% irate: input rate limits (default = inf)
% omag: output magnitude limits (default = inf)
% target: size of target set abs(y^(k+Hp|k) - ref(k+Hp)) < target
% kest : estimator for the augmented model
%
% constraint inputs: (imag,irate,omag)
% imag is 2x1 or 1x1
% if it's 2x1 then the top row are the upper limits and the bottom the lower
% if it's 1x1 then the lower limits are taken as negative the top
%
% OUTPUTS:
% jMod: jMPC model
%

function jMod = BuildMPCModel(sys,Hw,Hp,Hu,Q,R,imag,irate,omag,target,kest)

if(isa(sys,'lti')==0 | isdt(sys)==0)
    error('sys must be a discrete LTI object');
end;

[A,B,C,D,Ts] = ssdata(sys);

n = size(A,1);
m = size(C,1);
l = size(B,2);

[eA,eB,eC,eD,Ts] = ssdata(kest);

n = size(A,1);
m = size(C,1);

l = size(B,2);

if(nargin < 9) omag = repmat([inf -inf],l); end;
if(nargin < 8) irate = repmat([inf -inf],l); end;
if(nargin < 7) imag = repmat([inf -inf],m); end;

if(size(omag,2) == 1) omag(:,2) = -omag(:,1); end;
if(size(irate,2) == 1) irate(:,2) = -irate(:,1); end;
if(size(imag,2) == 1) imag(:,2) = -imag(:,1); end;
if(size(target,2) == 1) target(:,2) = -target(:,1); end;

horz = struct('Hw','Hw','Hp','Hp','Hu','Hu','l',l,'m',m,'n',n);
const = BuildConstraints(imag,irate,omag,horz,target);
weights = BuildWeights(Q,R,horz);

Ai(size(A),size(A),Hp) = 0;
for i=[1:Hp]
    Ai(:,i) = A^i;
end;

Psi = [];
for i=[1:Hp]
    Psi = [Psi;C*Ai(:,i)];
end;

Upsilon = [];
for i=[1:Hp]
    xx(:,i) = C*(B + sum(Ai(:,1:i-1),3)*B);
    Upsilon = [Upsilon;xx(:,i)];
end;

Theta = [];
zz = zeros(size(xx(:,1)));
for c=[1:Hu]
    yy = [];
    for r=[1:c-1]
        yy = [yy;zz];
    end;

```

```

    for r=[1:Hp-c+1]
        yy = [yy;xx(:,r)];
    end;
    Theta = [Theta yy];
end;

if((size(Psi) ~= [m*(Hp-Hw+1),n]) | ...
    (size(Upsilon) ~= [m*(Hp-Hw+1),l]) | ...
    (size(Theta) ~= [m*(Hp-Hw+1),l*Hu]))
    error('Prediction matrix sizes are wrong');
end;

% Build the closed-form unconstrained version
sQ = sqrt(weights.QQ);
sR = sqrt(weights.RR);
Kfull = [sQ*Theta;sR]\[sQ;zeros(size(sR,1),size(sQ,2))];
Kmpc = Kfull(1:l,:);

pred = struct('Psi',Psi,'Upsilon',Upsilon,'Theta',Theta);
jMod = struct('sys',sys,'pred',pred,'const',const,'horz',horz,...
    'weights',weights,'eA',eA,'eB',eB,'eC',eC,'eD',eD,...
    'Kmpc',Kmpc,'Kfull',Kfull);

```

BuildWeights

```
% function weights = BuildWeights(Q,R,horz)
%
% R(1,l,Hu)
% Q(m,m,Hp-Hw+1)

function weights = BuildWeights(Q,R,horz)

m = horz.m;
l = horz.l;

Hw = horz.Hw;
Hp = horz.Hp;
Hu = horz.Hu;

sq= [size(Q) 1];
sr= [size(R) 1];

if(sq(1)~=m | sq(2)~=m | sq(3)~=Hp-Hw+1)
    s = sprintf('Q is the wrong size. Should be %ux%ux%u',m,m,Hp-Hw+1);
    error(s);
end;
if(sr(1)~=l | sr(2)~=1 | sr(3)~=Hu)
    s = sprintf('R is the wrong size. Should be %ux%ux%u',l,l,Hu);
    error(s);
end;

QQ = zeros(m*(Hp-Hw+1));
RR = zeros(l*Hu);

for i=[1:Hp-Hw+1]
    QQ(m*(i-1)+1:m*i,m*(i-1)+1:m*i) = Q(:, :, i);
end;

for i=[1:Hu]
    RR(l*(i-1)+1:l*i,l*(i-1)+1:l*i) = R(:, :, i);
end;

weights = struct('Q',Q,'R',R,'QQ',QQ,'RR',RR);
```

BuildConstraints

```

% BuildConstraints(E,F,G)
%
% imag(1,2) : input magnitude constraints
% irate(1,2) : input rate constraints
% omag(m,2) : output magnitude constraints
%

function const = BuildConstraints(imag,irate,omag,horz,target)

l = horz.l;
m = horz.m;

% test input sizes
[a,b] = size(imag);
if(a ~= l | b~=2)
    error('Size of input magnitude limits is incorrect');
end;
[a,b] = size(irate);
if(a ~= l | b~=2)
    error('Size of input rate limits is incorrect');
end;
[a,b] = size(omag);
if(a ~= m | b~=2)
    error('Size of output rate limits is incorrect');
end;
[a,b] = size(target);
if(a ~= m | b~=2)
    error('Size of target set is incorrect');
end;

Hu = horz.Hu;
Hw = horz.Hw;
Hp = horz.Hp;

% input rate constraints
W = eye(1*Hu);
W = [W;-W];
a = repmat(irate(:,1),Hu,1);
b = -repmat(irate(:,2),Hu,1);
w = [a;b];

% input magnitude constraints
FF = tril(repmat(eye(1),Hu,0));
FF = [FF;-FF];
a = -repmat(imag(:,1),Hu,1);

```

```

b = repmat(imag(:,2),Hu,1);
f = [a;b];
F1 = FF(:,1:1);

% output magnitude constraints
Gamma = [zeros(m,m*(Hp-Hw+1));[zeros(m*(Hp-Hw),m) eye(m*(Hp-Hw))]];
Gamma = [Gamma;-Gamma];
a = [zeros(m,1);-repmat(omag(:,1),Hp-Hw,1)];
b = [zeros(m,1); repmat(omag(:,2),Hp-Hw,1)];
g = [a;b];

% remove all lines which are all zeros
ind = find(sum(abs([Gamma g]),2)>0);
Gamma = Gamma(ind,:);
g = g(ind);

% add target set constraint
Tset_Gamma = [zeros(m,m*(Hp-Hw)) eye(m,m)];
Tset_Gamma = [Tset_Gamma;-Tset_Gamma];
target = [-target(:,1);target(:,2)];

const = struct('FF',FF,'f',f,'F1',F1,'Gamma',Gamma,...
    'g',g,'W',W,'w',w,'imag',imag,'irate',irate,...
    'omag',omag,'Tset_Gamma',Tset_Gamma,'target',target);

```

B.1.2 Control Move Calculation

jmpcMove

```

% [u,flag] = jmpcMove(jMod,r,x,u_1)
%
% Compute the next control move given the current state and reference input
%
% INPUTS:
% jMod      : jMPC model for control - created via BuildMPCModel
% s(m,Hp)   : setpoint trajectory
% xe(n+m,1) : current model state (augmented xe = [x;disturbance])
% u_1(1,1)  : previous input
% y(m,1)    : current measured output
% active    : 1 if online optimization was done in the previous step
%
% OUTPUTS:
% un(1,1)   : next input to apply
% xn(n+m,1) : next model state
% ye(m,1)   : estimated output
% active    : 1 if online optimization was done

function [un,xn,ye,active] = jmpcMove(jMod,s,x,u_1,y,active)

Psi      = jMod.pred.Psi;
Upsilon = jMod.pred.Upsilon;
Theta    = jMod.pred.Theta;

Hp = jMod.horz.Hp;
Hw = jMod.horz.Hw;
Hu = jMod.horz.Hu;

const = jMod.const;
weights = jMod.weights;

[A,B,C,D,Ts] = ssdata(jMod.sys);

[m,N] = size(s);
n = length(xe);
l = length(u_1);

Epsilon(m*(Hp-Hw+1),N) = 0;

% compute the reference trajectory assuming it's constant across the horizon
T = reshape(s, (Hp-Hw+1)*m,1);

% update the state estimate
xn = jMod.eA*xe + jMod.eB*[u_1;y];
xxx = jMod.eC*xe + jMod.eD*[u_1;y];
ye = xxx(1:m);
xe = xxx(m+1:end);

% compute the constraints
Omega = [];
Omega = [Omega;const.FF];
Omega = [Omega;const.Gamma*Theta];
Omega = [Omega;const.W];
Omega = [Omega;const.Tset_Gamma*Theta];

w = [];
if(~isempty(const.F1))
    w = [w;-const.F1*u_1-const.f];
end;
if(~isempty(const.Gamma))
    w = [w;-const.Gamma*(Psi*xe+Upsilon*u_1)+const.Gamma*T-const.g];
end;
if(~isempty(const.w))
    w = [w;const.w];
end;
if(~isempty(const.Tset_Gamma))
    w = [w;-const.Tset_Gamma*(Psi*xe+Upsilon*u_1)+const.Tset_Gamma*T-const.target];
end;

% compute the cost function
HH = Theta'*weights.QQ*Theta + weights.RR;
HH = (HH+HH')/2;

flag = 0;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% compute control move
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% offset-free tracking
d = y-ye;
dd = repmat(d, Hp-Hw+1, 1);

```

```

Epsilon = T - Psi*xe - Upsilon*u1 - dd;

% if the constraints weren't active last time then guess that they won't
% be this time

if(active==0)
    % compute the closed-form solution
    %un = jMod.Kmpc * Epsilon + u1;
    DU = jMod.Kfull * Epsilon;
    un = DU(1:l) + u1;

    %%% NOTE - Hw must be 1 for this to work
    uu = reshape(DU,l,Hu);

    xx(:,1) = xe;
    % make future predictions
    for i=[1:Hu]
        uu(:,i) = uu(:,i) + sum(uu(:,1:i),2) + u1;
        xx(:,i+1) = A*xx(:,i) + B*uu(:,i);
        yy(:,i) = C*xx(:,i) + D*uu(:,i);
        if(~isempty(find(yy(:,i) > jMod.const.omag(:,1)))) flag = 1; end;
        if(~isempty(find(yy(:,i) < jMod.const.omag(:,2)))) flag = 1; end;
    end;

    t1 = ~isempty(find(un > jMod.const.imag(:,1)));
    t2 = ~isempty(find(un < jMod.const.imag(:,2)));
    t5 = ~isempty(find((un-u1) > jMod.const.irate(:,1)));
    t6 = ~isempty(find((un-u1) < jMod.const.irate(:,2)));
    if(t1 | t2 | t5 | t6 | flag==1);
        active = 1;
    end;
end;

% if the constraints are active then run the qp solver
if(active)
    GG = 2*Theta'*weights.QQ*Epsilon;

    [DU,fval,qpflag] = quadprog(2*HH,-GG,Omega,w,[],[],[],[],...
        optimset('Display','off','LargeScale','off'));

    if(qpflag<0)
        warning('Problem is infeasible');
        return;
    end;

    if(isempty(find(Omega*DU==w)))
        active = 0;

```

```

end;

un = u1 + DU(1:l);
end;

```


References

- [AM99] David Angeli and Edoardo Mosca. Command governors for constrained nonlinear systems. *IEEE Transactions on Automatic Control*, 44(4):816–820, April 1999.
- [ASC83] A. N. Andry, E. Y. Shapiro, and J. C. Chung. Eigenstructure assignment for linear systems. *IEEE Transactions on Aerospace Electronic Systems*, 19(5), September 1983.
- [AW95] Karl J. Aström and Björn Wittenmark. *Adaptive Control*. Addison-Wesley Publishing Company, second edition, 1995.
- [BB97] John J. Burken and Frank W. Burcham. Flight-test results of propulsion-only emergency control system on md-11 airplane. *Journal of Guidance, Control and Dynamics*, 20(5), October 1997.
- [BBMB97] Frank W. Burcham, John J. Burken, Trindel A. Maine, and John Bull. Emergency flight control using only engine thrust and lateral center-of-gravity offset: A first look. Technical report, NASA, 1997.
- [BBMF97] Frank W. Burcham, John J. Burken, Trindel A. Maine, and C. Gordon Fullerton. Development and flight test of an emergency flight control system using only engine thrust on an md-11 transport airplane. Technical report, NASA, October 1997.
- [BD95] Kenneth A. Bordignon and Wayne C. Durham. Closed-form solutions to constrained control allocation problem. *Journal of Guidance, Control and Dynamics*, 18(5), September 1995.
- [Bem97] Alberto Bemporad. *Reference Governors: On-Line Set-Point Optimization Techniques for Constraint Fulfillment*. PhD thesis, Università di Firenze, October 1997.
- [BG97] Marc Bodson and Joseph E. Groszkiewicz. Multivariable adaptive algorithms for reconfigurable flight control. *Transactions on Control Systems Technology*, 5(2):217–229, March 1997.

- [Bla99] M. Blanke. Fault-tolerant control systems. In Paul M. Frank, editor, *Advances in Control, Highlights of ECC'99*, pages 171–196. Springer-Verlag, Department of Control Engineering, Aalborg University, Fredrik Bajers Vej 7C, DK 9220, Aalborg, Denmark, 1999.
- [BLM00a] Jovan D Boskovic, Sai-Ming Li, and Raman K. Mehra. Reconfigurable flight control design using multiple switching controllers and on-line estimation of damage-related parameters. In *Proceedings of the 2000 IEEE International Conference on Control Applications*. IEEE, September 2000.
- [BLM00b] Jovan D Boskovic, Sai-Ming Li, and Raman K. Mehra. Study of an adaptive reconfigurable control scheme for tailless advanced fighter aircraft (tafa) in the presence of wing damage. In *Position Location and Navigation Symposium*, pages 341–348. IEEE, 2000.
- [BLM01] Jovan D Boskovic, Sai-Ming Li, and Raman K. Mehra. Robust supervisory fault-tolerant flight control system. In *Proceedings of the American Control Conference*, June 2001.
- [BM98] Jovan D Boskovic and Raman K. Mehra. A multiple model-based reconfigurable flight control system design. In *Proceedings on the 37th IEEE Conference on Decision & Control*. IEEE, December 1998.
- [BM99] Jovan D Boskovic and Raman K. Mehra. Stable multiple model adaptive flight control for accommodation of a large class of control effector failures. In *Proceedings of the American Control Conference*, June 1999.
- [BM00] Franco Blanchini and Stefano Miani. Any domain of attraction for a linear constrained system is a tracking domain of attraction. *SIAM Journal of Optimal Control*, 38(3):971–994, 2000.
- [Bod94] Marc Bodson. Multivariable adaptive algorithms for reconfigurable flight control. In *Proceedings of the 33rd Conference on Decision and Control*. IEEE, December 1994.
- [BS00] Abderrazak I. Belkharraz and Kenneth Sobel. Fault tolerant flight control for a class of control surface failures. In *Proceedings of the American Control Conference*. IEEE, June 2000.
- [BW01] Joseph S. Brinker and Kevin A. Wise. Flight testing of reconfigurable control law on the x-36 tailless aircraft. *Journal of Guidance, Control and Dynamics*, 24(5), September 2001.

- [CFNS02] Giampiero Campa, Mario Luca Fravolini, Marcello Napolitano, and Brad Seanor. Neural networks-based sensor validation for the flight control system of a b777 research model. In *American Control Conference*, 2002.
- [CFS⁺01] Giampiero Campa, Mario Luca Fravolini, Brad Seanor, Marcello Napolitano, Diego Del Gobbo, and Srikanth Gururajan. On-line learning neural networks for sensor validation for the flight control system of a b777 research scale model. *International Journal of Robust and Nonlinear Control*, 2001.
- [CHI01] Anthony J. Calise, Naira Hovakimyan, and Moshe Idan. Adaptive output feedback control of nonlinear systems using neural networks. *Automatica*, 37(8), March 2001.
- [Civ00] Civil Aviation Authority, Aviation Safety Review 1990-1999, October 2000. Available at <http://www.srg.caa.co.uk/publications/cap701.pdf>.
- [CLS98] Anthony J. Calise, Seungjae Lee, and Manu Sharma. Direct adaptive reconfigurable control of a tailless fighter aircraft. In *AIAA Guidance, Navigation and Control Conference*, Boston, MA, aug 1998.
- [CLS00] Anthony J. Calise, Seungjae Lee, and Manu Sharma. Development of a reconfigurable flight control law for the x-36 tailless fighter aircraft. *AIAA Guidance, Navigation, and Control Conference*, August 2000.
- [CMA00] Alessandro Casavola, Edoardo Mosca, and David Angeli. Robust command governors for constrained linear systems. *IEEE Transactions on Automatic Control*, 45(11):2071–2077, November 2000.
- [Coo97] M. V. Cook. *Flight Dynamics Principles*. John Wiley & Sons Inc., 1997.
- [DA94] John B. Davidson and Dominick Andrisani. Gain weighted eigenspace assignment. Technical report, NASA, May 1994.
- [DA96] John B. Davidson and Dominick Andrisani. Lateral-directional eigenvector flying qualities guidelines for high performance aircraft. Technical report, NASA, December 1996.
- [DB96] Wayne C. Durham and Kenneth A. Bordignon. Multiple control effector rate limiting. *Journal of Guidance, Control and Dynamics*, 19(1), February 1996.
- [Dem01] Michael A. Demetriou. Adaptive reorganization of switched systems with faulty actuators. In *Proceedings of the 40th IEEE Conference on Decision and Control*, December 2001.

- [Den96] John S. Denker. See how it flies, a new spin on the perceptions, procedures, and principles of flight. <http://www.monmouth.com/~jsd/how/>, 1996.
- [DLB01] John B. Davidson, Frederick J. Lallman, and W. Thomas Bundick. Real-time adaptive control allocation applied to a high performance aircraft. In *5th SIAM Conference on Control & Its Applications*, 2001. Available at <http://dcb.larc.nasa.gov/DCBStaff/jbd/reports/rtacat01vf.pdf>.
- [Enn98] D. F. Enns. Control allocation approaches. In *Proceedings of AIAA GNC Conference*, August 1998.
- [ER96] Bernard Etkin and Lloyd Duff Reid. *Dynamics of Flight, Stability and Control*. John Wiley & Sons, Inc., third edition, 1996.
- [GB95] Joseph E. Groszkiewicz and Marc Bodson. Flight control reconfiguration using adaptive methods. In *Proceedings of the 34th Conference on Decision & Control*. IEEE, December 1995.
- [GBMR98] Murali Gopinathan, Jovan D Boskovic, Raman K. Mehra, and Constantino Rago. A multiple model predictive scheme for fault-tolerant flight control design. In *Proceedings of the 37th IEEE Conference on Decision & Control*. IEEE, December 1998.
- [HM98a] Mihai Huzmezan and Jan M. Maciejowski. Reconfigurable flight control of a high incidence research model using predictive control. *UKACC International Conference on CONTROL*, September 1998. Available at <http://www.ece.ubc.ca/huzmezan/pcopies.html>.
- [HM98b] Mihai Huzmezan and Jan M. Maciejowski. Reconfiguration and scheduling in flight using quasi-lpv high-fidelity models and mbpc control. In *Proceedings of the American Control Conference*, June 1998.
- [Huz97] Mihai Huzmezan. Reconfigurable flight control methods and related issues - a survey. Technical report, University of Cambridge, September 1997.
- [IJC01] Moshe Idan, Matthew Johnson, and Anthony J. Calise. A hierarchical approach to adaptive control for improved flight safety. *AIAA Journal on Guidance, Control and Dynamics*, July 2001. Available at <http://citeseer.nj.nec.com/idan01hierarchical.html>.
- [IJCK01] Moshe Idan, Matthew Johnson, Anthony J. Calise, and John Kaneshige. Intelligent aerodynamic/propulsion flight control for flight safety: A nonlinear adaptive approach. In *American Control Conference (ACC)*, 2001.

- [Isi89] Alberto Isidori. *Nonlinear Control Systems*. Springer-Verlag, second edition, 1989.
- [Jac95] E. Bruce Jackson. *Manual for a Workstation-based Generic Flight Simulation Program (LaRCsim)*. NASA, Langley Research Center, Hampton, Virginia, 1.4 edition, April 1995.
- [JC01] Eric N. Johnson and Anthony J. Calise. Neural network adaptive control of systems with input saturation. In *American Control Conference (ACC)*, Arlington, Virginia, June 2001.
- [KA95] Ioannis K. Konstantopoulos and Panos J. Antsaklis. An optimization strategy for reconfigurable control systems. Technical report, Interdisciplinary Studies of Intelligent Systems, September 1995. Available at <http://citeseer.nj.nec.com/20177.html>.
- [KA96] Ioannis K. Konstantopoulos and Panos J. Antsaklis. Eigenstructure assignment in reconfigurable control systems. Technical report, Interdisciplinary Studies of Intelligent Systems, January 1996. Available at <http://citeseer.nj.nec.com/152208.html>.
- [Ker98] Eric Kerrigan. Fault-tolerant control of the cosy ship propulsion benchmark using model predictive control. Technical report, University of Cambridge, November 1998.
- [KV00a] S. Kanev and M. Verhaegen. A bank of reconfigurable lqg controllers for linear systems subjected to failures. In *39th IEEE Conference on Decision and Control*, December 2000. Available at <http://citeseer.nj.nec.com/420934.html>.
- [KV00b] S. Kanev and M. Verhaegen. Controller reconfiguration for non-linear systems. *Control Engineering Practice*, 8:1223–1235, October 2000.
- [KVN01] S. Kanev, M. Verhaegen, and G. Nijssen. A method for the design of fault-tolerant systems in case of sensor and actuator faults. In *European Control Conference ECC*, September 2001.
- [Lam83] A.A. Lambregts. Integrated system design for flight and propulsion control using total energy principles. In *AIAA Aircraft design, systems and technology meeting, paper no. AIAA-83-2561*, Fort Worth, Texas, October 1983.
- [LS94] Zongli Lin and Ali Saberi. Semi-global exponential stabilization of linear discrete-time systems subject to input saturation via linear feedbacks. *Systems & Control Letters*, 1994.

- [Mac98] Jan M. Maciejowski. The implicit daisy-chaining property of constrained predictive control. *Applied Math and Computer Science*, 8(4):695–711, 1998.
- [Mac00] J. M. Maciejowski. *Predictive Control with Constraints*. 1 edition, 2000.
- [May99] Peter S. Maybeck. Multiple model adaptive algorithms for detecting and compensating sensor and actuator/surface failures in aircraft flight control systems. *International Journal of Robust and Nonlinear Control*, 9:1051–1070, 1999.
- [Mig02] Domenico Mignone. *Control and Estimation of Hybrid Systems with Mathematical Optimization*. PhD thesis, Swiss Federal Institute of Technology (ETH), January 2002.
- [NB97] Kumpati S. Narendra and Jeyendran Balakrishnan. Adaptive control using multiple models. *IEEE Transactions on Automatic Control*, 42(2), February 1997.
- [Nis92] Norman S. Nise. *Control Systems Engineering*. The Benjamin/Cummings Publishing Company, INC., 1992.
- [Pat97] R.J. Patton. Fault-tolerant control systems: The 1997 situation. *IFAC Symposium on Fault Detection Supervision and Safety for Technical Processes*, 3:1033–1054, August 1997. Available at <http://www.eng.hull.ac.uk/research/control/safepr.ps>.
- [Rau01] Marc Rauw. *FDC 1.2 - A Simulink Toolbox for Flight Dynamics and Control Analysis*, 2nd edition, May 2001. Available at <http://www.dutchroll.com/>.
- [SB99] Yuri B. Shtessel and J Buffington. Multiple time scale flight control using reconfigurable sliding modes. *AIAA Journal on Guidance, Control and Dynamics*, 22(6):873–883, 1999.
- [SHS01] Ali Saberi, Jian Han, and Anton A. Stoorvogel. Constrained stabilization problems for linear plants. 2001.
- [SHS02] Ali Saberi, Jian Han, and Anton A. Stoorvogel. Constrained stabilization problems for linear plants. *Automatica*, 38:639–654, April 2002.
- [Sht01] Yuri B. Shtessel. Sliding mode control: Overview and applications to aerospace control. Talk notes, 2001.
- [SL91] Jean-Jacques E. Slotine and Weiping Li. *Applied Nonlinear Control*. Prentice-Hall International, Inc, 1991.

- [SLT94] Ali Saberi, Zongli Lin, and Andrew R. Teel. Control of linear systems with saturating actuators. June 1994.
- [Sma97] M.H. Smaili. Flight data reconstruction and simulation of el al flight 1862. Master's thesis, Technical University Delft, November 1997.
- [SS01] Ilya A. Shkolnikov and Yuri B. Shtessel. Aircraft nonminimum phase control in dynamic sliding manifolds. *Journal of Guidance, Control and Dynamics*, 24(3), June 2001.
- [WBC⁺99] Kevin A. Wise, Joseph S. Brinker, Anthony J. Calise, Dale F. Enns, Michael R. Elgersma, and Petros Voulgaris. Direct adaptive reconfigurable flight control for a tailless advanced fighter aircraft. *International Journal of Robust and Nonlinear Control*, 9(14):999–1012, December 1999.
- [WCNF02] Sheng Wan, Giampiero Campa, Marcello Napolitano, and Mario Luca Fravolini. Sensor validation schemes for a flight control system with dual physical redundancy. In *American Control Conference*, September 2002.
- [ZHZ99] Yang Zhenyu, Shao Huazhang, and Chen Zongji. The frequency-domain heterogeneous control mixer module for control reconfiguration. In *Proceedings of the 1999 IEEE International Conference on Control Applications*. IEEE, August 1999.
- [ZJ99] Youmin Zhang and Jin Jiang. An interacting multiple-model based fault detection, diagnosis and fault-tolerant control approach. In *Proceedings of the 38th Conference on Decision & Control*, December 1999.
- [ZJ00] Youmin Zhang and Jin Jiang. Integrated design of reconfigurable fault-tolerant control systems. *Journal of Guidance*, 24(1):133–136, July 2000.

Appendix C

Author Index

- Andrisani, Dominick 28, 29
Andry, A. N. 28
Angeli, David 38
Antsaklis, Panos J. 28, 29, 30
Aström, Karl J. 8, 17, 30, 31
Balakrishnan, Jeyendran 17
Belkharraz, Abderrazak I. 29
Bemporad, Alberto 38
Blanchini, Franco 39
Blanke, M. 1
Bodson, Marc 32
Bordignon, Kenneth A. 21
Boskovic, Jovan D 15, 17, 18, 33
Brinker, Joseph S. 20, 21, 23
Buffington, J 24, 27
Bull, John 19
Bundick, W. Thomas 21
Burcham, Frank W. 19
Burken, John J. 19
Calise, Anthony J. 20, 21, 22, 23, 24
Campa, Giampiero 9
Casavola, Alessandro 38
Chung, J. C. 28
Cook, M. V. 3
Davidson, John B. 21, 28, 29
Demetriou, Michael A. 15
Denker, John S. 40, 73, 76
Durham, Wayne C. 21
Elgersma, Michael R. 20, 21
Enns, D. F. 21
Enns, Dale F. 20, 21
Etkin, Bernard 8, 42, 44
Fravolini, Mario Luca 9
Fullerton, C. Gordon 19
Gobbo, Diego Del 9
Gopinathan, Murali 15
Groszkiewicz, Joseph E. 32
Gururajan, Srikanth 9
Han, Jian 39
Hovakimyan, Naira 21, 22, 23
Huazhang, Shao 21
Huzmezan, Mihai 30, 33
Idan, Moshe 21, 22, 23
Isidori, Alberto 22
Jackson, E. Bruce 78
Jiang, Jin 19, 29
Johnson, Eric N. 21, 23
Johnson, Matthew 21, 23
Kaneshige, John 21, 23
Kanev, S. 15, 19, 33
Kerrigan, Eric 33

-
- Konstantopoulos, Ioannis K. 28, 29, 30
- Lallman, Frederick J. 21
- Lambregts, A.A. 77
- Lee, Seungjae 20, 21, 24
- Li, Sai-Ming 15, 17
- Li, Weiping 24, 25, 26
- Lin, Zongli 39
- Maciejowski, J. M. 10, 11, 91
- Maciejowski, Jan M. 33
- Maine, Trindel A. 19
- Maybeck, Peter S. 9, 19
- Mehra, Raman K. 15, 17, 18, 33
- Miani, Stefano 39
- Mignone, Domenico 33
- Mosca, Edoardo 38
- Napolitano, Marcello 9
- Narendra, Kumpati S. 17
- Nijssse, G. 19
- Nise, Norman S. 28
- Patton, R.J. 30
- Rago, Constantino 15
- Rauw, Marc 3
- Reid, Lloyd Duff 8, 42, 44
- Saberi, Ali 39
- Seanor, Brad 9
- Shapiro, E. Y. 28
- Sharma, Manu 20, 21, 24
- Shkolnikov, Ilya A. 24
- Shtessel, Yuri B. 24, 27
- Slotine, Jean-Jacques E. 24, 25, 26
- Smaili, M.H. 39
- Sobel, Kenneth 29
- Stoorvogel, Anton A. 39
- Teel, Andrew R. 39
- Verhaegen, M. 15, 19, 33
- Voulgaris, Petros 20, 21
- Wan, Sheng 9
- Wise, Kevin A. 20, 21, 23
- Wittenmark, Björn 8, 17, 30, 31
- Zhang, Youmin 19, 29
- Zhenyu, Yang 21
- Zongji, Chen 21