Parallel Processing for Fault Tolerant Aircraft Control

by

Jamel Makky Tahir B.Sc., M.Sc. (Eng.)

Thesis submitted as a requirement for the degree of

Doctor of Philosophy

in the Department of

Automatic Control & Systems Engineering

University of Sheffield

July 1991

To my lovely family

Acknowledgement

I would like to sincerely thank my supervisor, Dr. G. S. Virk for his continued support, advice, and guidance throughout the duration of this work. I would like to express my appreciation to him for his encouragement and motivation.

My thanks are also extended to all the programming staff in the Department of Automatic Control & System Engineering and in particular, Mr. I. Durkacz, for their help in dealing with the computational problems encountered during this research. My further appreciation is due to the secretarial and technical staff of the Department.

I am grateful to the research staff at British Aerospace (Royal Military Aircraft) Brough for providing the aircraft engineering data used in this work and for their assistance during the research period.

Finally, I shall be forever grateful to my wife for her never ending support, encouragement, and understanding.

Parallel Processing for Fault Tolerant Aircraft Control

by

Jamel Makky Tahir

ABSTRACT

This thesis addresses the problem of real-time optimal control of aircraft systems using parallel processing techniques. It is shown that transputer hardware can be used in designing a suitable optimal controller for general nonlinear time-varying aircraft. In the first part of the thesis, nonlinearties and time varying aspects of the aircraft system, together with the current available solutions are investigated and suitable designs presented. Here the linear regulator approach for linear time-varying aircraft is investigated first but it is shown that real-time performance is difficult to achieve. The problem is then approached differently in that the aircraft is considered as a linear time-invariant system for short time intervals and it is then found possible to implement an optimal control solution in real-time, and suitable multi-transputer architectures are presented. The receding/moving horizon approach is applied to the aircraft system and is shown to be adequate for achieving satisfactory results. The problem of selection of the weights in the performance index of the optimal control problem is then studied and a design procedure is presented. The modeling of the aircraft as decoupled longitudinal and lateral dynamics is investigated and approached in such a way as to reduce the cross-coupling effects. Another important

aspect of this research involves the consideration of failure detection and diagnosis in the aircraft hardware. Problems including actuator failure are studied and some remedial methods for handling the failures by enabling system reconfiguration after the occurrence of the failure are presented.

The multi-processor based control system design is shown to offer a viable solution to solving complicated optimisation problems without the need for the simplification of the system dynamical equations and thereby loosing accuracy. Such simplification is usually a prerequisite for enabling practical designs. However with the use of parallel processing techniques such designs can be achieved for the more complicated (and more computationally demanding) cases as well.

Table of Contents

Chapter 1

Introduction

1.1 Historical Background

The field of automatic control has matured from an art to a technical discipline, and is still expanding rapidly. During the period from the late eighteenth to the early twentieth century, the progress in the field of control was empirical in nature. The Watt governor applied to steam engine control was perhaps the single most important development in this era of control (Widnall [54], Takahashi [51]). In the period from about 1900 to the start of World War II, the industrial progress was gaining momentum. Large-scale power generation, the aeronautic industry, communications, and electronic engineering also appeared. Such technological progress created an increasing demand for instruments and regulators. The last period dated from the early 1940's. In this period the frequency domain analysis, root-locus, sampled-data control, multivariable control systems, and optimal control theory were developed and heavily applied to both civil and military applications.

It was in this period when control system design was first approached as an optimization problem. Much of this work started with Wiener(1949), who suggested that the search for an efficient system should be formulated as an optimisation

problem. Following Wiener's lead, the field of optimal control theory has taken different approaches to the design problem. The design engineer must give considerable attention to formulating the design problem accurately. He must develop mathematical models for the plant to be controlled and the disturbances affecting it. He must know the nature of the measurements of the plant state including any sources of noise which corrupt the measurements. All the various control objectives must be gathered into an analytical statement of the cost of operating the plant, then the engineer must find the means to produce the control that minimises the cost. Mathematical tools has been found to solve the optimal control problem such as dynamic programming and the maximum principle. The digital computer came next and made it much easier to solve complicated numeric problems which were tedious when tackled manually.

An area of particular interest in which automatic control plays a vital role is automatic flight control where a balanced combination of art, science, and the effort of large project teams with adequate resources is usually required, see Howard [24]. To appreciate the research presented in this thesis it is useful to have a knowledge of the historical development of automatic control systems for flight control. For this reason such a discussion is presented here.

One century ago we were still in the era of lone inventors rather than project management teams and automatic control engineering like mechanical flight, was in its infancy. Automatic flight control started in 1873 when the Frenchman Charles Renard tested, from a high tower, an unmanned multi-wing glider incorporating an automatic control device aimed at improving the machine's directional stability. The early designs of aircraft left all the burden of controlling the aircraft to the pilot, which kept him too busy looking after the aircraft than to be able to attend any other matters modern aircraft used to do. In 1914 Lawrence Sperry demonstrated

the first fully automatic stabilised flying boat. In 1920 there arose again a series of simple automatic stabiliser inventions and by 1930 the final outcome of this early work on pilotless aeroplanes was the RAE Mark I control followed by a series of automatic stabilisers. The radio guidance and the first fully automatic landing was demonstrated in 1945. The development of the integrated circuit technology made it possible in the 1960's to produce very low weight auto-stabliser systems. The situation has advanced so much that modern aircraft nowadays carry thousands of microcircuit elements to solve their complex control and stability requirements; they also carry very powerful digital computers for the purpose of executing the extensive programmes for the testing of all the sub-systems, and the locating of failures when they occur in the automatic flight control system.

1.2 Contribution of this Research

This work is concerned with the complex problem of optimal control of nonlinear time-varying aircraft in real-time. An investigation is made to employ parallel and multi-processing techniques to achieve fast and efficient computations to provide the control within the time constraints of real-time applications, that is, within the sample time of 5 ms, when a sampling frequency of 200 Hz is used. This goal is difficult to achieve in problems of high dimensionality, time variations and nonlinearties.

Certain assumptions can however be made to make real-time solutions possible; these include reducing the system order, using linearized models to simplify the problem and allowing the feedback control strategy to be calculated off-line, and then applied on-line to provide the control in real-time. The drawback in using such an approach is usually loosing accuracy and having to accept some below par performances. Therefore, compromises have to be made between the real-time computational demands and performance demands. However with the aid of parallel

processing, it is shown here that it is possible to solve such optimal flight control problems while keeping satisfactory performance specifications.

Another problem which has been investigated in this research is the reliability and risk analysis of the aircraft system under consideration; techniques have been applied to achieve fault tolerance under actuator failures by allowing control reconfiguration.

1.3 Organisation of the Work

This work is divided into two main areas: The optimal control law design, and the fault tolerance system design. Chapter 2 presents the basic concepts of optimal control theory and the various techniques associated with it. It also presents the concepts of parallel processing by giving a brief description of some of the architectural classifications of parallel transputer systems, and the programming language used with them. Chapter 3 reviews the mathematical background related to aerodynamics and aircraft systems modelling. Chapter 4 considers the problem of designing a longitudinal optimal controller for an aircraft system. The linear regulator problem is first applied to the linearized time-varying aircraft, and then applied to the case where the aircraft is assumed to be linear and time-invariant for short time intervals. Different multi-transputer architectures are presented to provide real-time performance. Chapter 5 considers the receding horizon control problem formulation; an algorithm using this technique is developed and applied to both the longitudinal and lateral motions of the aircraft. Chapter 6 considers the problem of selecting the weighting matrices used in the optimal control problems. A practical method for the weights selection is presented to ease the design problem, and to allow adjustment of the time varying gains on-line to enable the controller to follow the aircraft time-varying aspects. Chapter 7 presents a decoupled design approach for

the longitudinal and lateral motions, where the cross-coupling effects are included to achieve an accurate representation of the aircraft dynamics. Chapter 8 considers the problem of aircraft actuator failure detection, diagnosis and suitable techniques have been used to increase the systems reliability. In Chapter 9, conclusions and suggestions for future work form the final section of this thesis.

Chapter 2

Fundamental Concepts

In this chapter the fundamental theoretical concepts which are used in the remainder of the thesis are presented. These cover quite a large area, but it are necessary to be able to appreciate the wide ranging aspects of this research. The aircraft regulator designs presented here are based on optimal control problems and so an introduction to the relevant theory is presented in section 2.1. In our approach to the real-time design of optimal controllers for the nonlinear time-varying aircraft, it was felt that the designs could only be realizable in practice by using parallel processing techniques. Therefore we introduce in section 2.2, some of the fundamental concepts of parallel processing and in particular transputer systems are discussed. Fault tolerance has became a very important property demanded in modern control systems and is the third main feature of the work described here. In light of this the fundamental aspects of fault tolerance systems are presented in section 2.3.

2.1 Optimal Control Theory

The technology of information transmission and processing has grown during recent years at an exceptionally rapid rate. The development of its most important direc-

tion namely automation, is characterized by the rapid spread of automatic systems and by an expansion of its range of application. New principles of automatic control are emerging which solve the more complex problems of control and replace man in the more complicated shares of his activity.

One of the important directions of engineering is the theory of optimal processes (see for example Widnall [54], Fel'dbaum [18]). Optimal control problems can arise in all fields of engineering, because, any control engineering problem can be solved in a "best method", that is optimal in some sense.

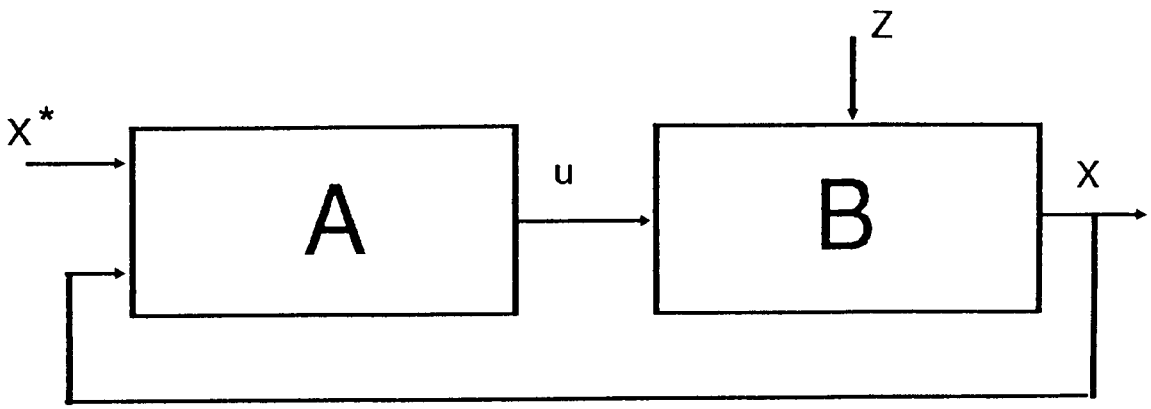

Figure 2.1: Automatic control system

The block diagram of Figure 2.1 represents an automatic system. The controller is denoted by A, and the controlled object by B. The controlled variable X is the parameter characterizing the state of the controlled object, the control action u of the controller A acts at the input of B, the driving action X^* is supplied to the input of A representing of what must be the variable X, and the perturbing action Z acting on B and having an influence on its output X. It is convenient to regard these variables as vectors.

Automatic control systems can be divided into two classes: open-loop, and closed-loop systems. The latter are also called feedback systems. In open-loop

systems the controller A does not receive information about the actual state X. The principle of feedback creates the possibility of satisfying the demands presented to the variable X.

The control purpose can be considered as the attainment of an extremum of some quantity J which is called the optimality criterion or the performance index. In the general case an optimality criterion depends both on the driving action X^* and on the state variable X; it can also depends on u as well as on time t. For definiteness let it be required that the quantity J is to be minimised:

$$J(X^*,\ X,\ u,\ t,) = Min \tag{2.1}$$

This condition is the analytical formulation of the control purpose or objective. In special cases J has the quadratic form

$$J = \int_0^T [X(t) - X^*(t)]^2 dt \tag{2.2}$$

Various engineering and economic indices can be selected as the criterion J; for example, the output of a system or the quality of production.

2.1.1 Variational Problems

In the eighteenth century general methods of solving variational problems used in determination of extreme points were given by Euler and Lagrange (see for example Widnall [54]). In the twentieth century the so-called direct methods of solving problems began to be applied in physics and engineering. The new problems have caused the appearance of new methods of solving variational problems: The method of dynamic programming developed by Bellman, and also the Maximum principle proposed by Pontryagin have all contributed to the determination of optimal solutions. Dynamic programming which we discuss next offers a convenient method for solving general optimisation problem.

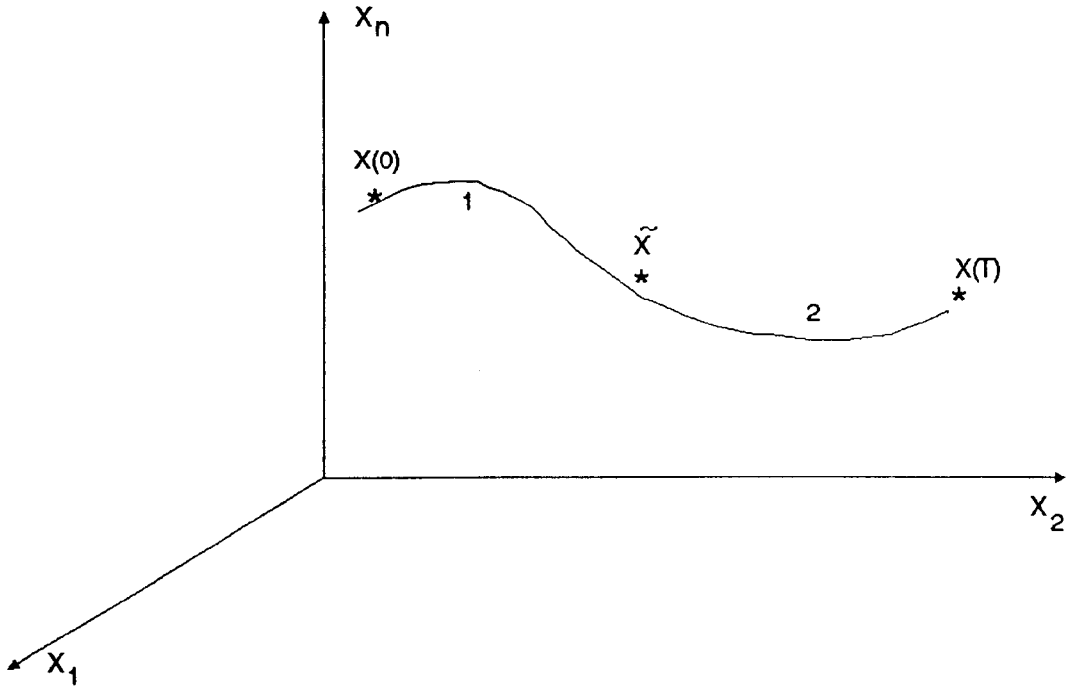

Figure 2.2: Optimal trajectory

2.1.2 Dynamic Programming

Let us consider the problem of the control of an object with the equation

$$\dot{X} = f_1(X,\ u) \tag{2.3}$$

where X is n-dimensional and u is m-dimensional, let

$$u \in \Omega(u) \tag{2.4}$$

and let it be required to minimise the integral

$$J = \int_0^T L_1(X(t),\ u(t),\ t)dt\ +\ \varphi_1(X(t)) \tag{2.5}$$

where T will be considered fixed. To illustrate the principle of optimality let us consider the optimal trajectory of Figure 2.2 with initial value of X(0) at $t = t_0$, final value X(T) at $t = T > T_0$, and some intermediate point $\tilde{X}$ corresponding to $t = \tilde{t}$, where $t_0 < \tilde{t} < T$.

The principle of optimality can be stated as: "The second section of an optimal trajectory ($\tilde{X}$ to $X(T)$) is in turn an optimal trajectory"; or "The optimal strategy does not depends on the previous history of the system but is determined only by the value of the state at the time instant under consideration".

Let us break up the interval $(0,\ T)$ into N equal parts of length Δt, and consider discrete values of $X = X(k)$ and $u = u(k)$, $(k = 1, 2, ..., N)$. Then the differential equation 2.3 can be approximately replaced by the finite difference equation

$$X(k+1) = X(k) + f(X(k),\ u(k)) \tag{2.6}$$

where

$$f(X(k),\ u(k)) = \Delta t \times f_1(X(k),\ u(k)) \tag{2.7}$$

$$X(0) = [X]_{t=0} \tag{2.8}$$

The integral 2.5 also be approximated by the sum

$$J = \sum_{i=0}^{N-1} L(X(k),\ u(k)) \ +\ \varphi(X(N)) \tag{2.9}$$

where

$$L(X(k),\ u(k)) = \Delta t \times L_1(X(k),\ u(k)) \tag{2.10}$$

$$\varphi(X(N)) = \varphi_1(X(T)) \tag{2.11}$$

To solve this discrete problem we use the concept of the "retrograde" motion from the end of the process, that is, from the terminal instant $t = T$, to the start time. Let us denote the minimal value of J_{N-1} by S_{N-1}. Evidently, this quantity depends on the state of the system at $t = (N-1)\Delta t$. Thus

$$\begin{aligned} S_{N-1} &= S_{N-1}(X(N-1)) \\ &= \min\{L(X(N-1),\ u(N-1)) + \varphi[X(N-1) + \\ &\qquad f(X(N-1),\ u(N-1))]\} \end{aligned} \tag{2.12}$$

The minimisation need only be carried out with respect to the single variable $u(N-1)$. Let us now pass to the next-to-last time segment. It can be noted that the choice of $u(N-2)$ and $u(N-1)$ appears only in the terms of the sum 2.9 that enter into the composition of the expression

$$\begin{aligned} J_{N-2} &= L(X(N-2),\ u(N-2)) + \{L(X(N-1),\ u(N-1)) \\ &\quad +\varphi(X(N))\} \end{aligned} \tag{2.13}$$

From the optimality principle it follows that only the value of $X(N-2)$ and the control objective (the minimisation of J_{N-2}) determine the optimal control for the time segment under consideration. Then we can write

$$\begin{aligned} S_{N-2} &= \min\{L(X(N-2),\ u(N-2)) + S_{N-1}\} \\ &= \min\{L(X(N-2),\ u(N-2)) + S_{N-1}(X(N-2) \\ &\quad +f(X(N-2), u(N-2)))\} \end{aligned} \tag{2.14}$$

Here the minimisation is carried only with respect to the variable $u(N-2)$, then we find $u^*(N-2)$ which is the optimal value of $u(N-2)$ and hence the quantity S_{N-2} can be obtained.

Passing in a completely analogous fashion to $S_{N-3}, ..., S_{N-k}$, we obtain the recurrence formula for determining $S_{N-k}[X(N-k)]$

$$\begin{aligned} S_{N-k} &= \min\{L(X(N-k),\ u(N-k)) + S_{N-k+1}(X(N-k) \\ &\quad +f(X(N-k),\ u(N-k)))\} \end{aligned} \tag{2.15}$$

Parallel to the process of minimising the right side of this formula, the optimal value u^* depending on $X(N-k)$ can be determined thus

$$u^*(N-k) = u^*(X(N-k)) \tag{2.16}$$

By computing the S_{N-k} successively for $k = 1, 2, ..., N$ from equation 2.15, we finally arrive at the determination of the optimal value $u^*(0)$, that is, the value of the optimal control action required at the initial time instant.

Dynamic programming permits the minimisation of a complicated function of many variables by replacing it by a sequence of minimisations. In each of these minimisation processes, the minimum of a simpler function need to be determined. It should be noted, however, that in general the solution of problems in dynamic programming can nevertheless turn out to be exceedingly tedious, but for some simple cases, it can give the control as some function of the state variables and the time to go to the final state. Complicated problems, can be only solved in practice with the aid of digital computers.

2.1.3 The Maximum Principle

One of the more outstanding recent contributions to optimal control theory is the maximum principle of Pontryagin (see for example Banks [4]). The essence of the maximum principle is that classical calculus of variation no longer holds.

Consider the performance criterion

$$J = \int_{t_0}^{t_f} L(X,\ u,\ t)dt + \varphi(X(t_f)) \tag{2.17}$$

subject to the n^{th}-order constraints

$$\begin{aligned} \dot{X} &= f(X,\ u,\ t) \\ X_0 &= X(t_0) \qquad \text{given} \\ u(t) &= \Omega \quad \text{(some subset in the } u \text{ space)} \end{aligned} \tag{2.18}$$

The problem is to find $u^*(t)$ such that J is minimised. to formulate the theorem for this problem we introduce an n-dimensional costate vector $P(t)$. The above equations and the costate vector are combined into a scalar function by the Hamiltonian H such that

$$H = L(X,\ u,\ t) + P(t)f(X,\ u,\ t) \tag{2.19}$$

The Hamiltonian is regarded as a function of $u \in \Omega$. The statement of the maximum principle is: "Let $u(t)$, $t_0 \leq t \leq t_f$, be an admissible control that moves the state from a given initial state $X(t_0)$ to a prescribed final state $X(t_f)$ at some time $(t_f - t_0)$. In order that $u^*(t)$ and the resulting trajectory $X^*(t)$ be optimal, it is necessary that there exist a nonzero continuous vector $P^*(t)$ that correspond to u^* and X^* such that, for every t, $t_0 \leq t \leq t_f$, the function H of the variable $u \in \Omega$ attain its minimum at the point $u = u^*$". Therefore, the necessary conditions for minimum J are:

$$\dot{X} = \frac{\delta H}{\delta P} = f(X,\ u,\ t) \tag{2.20}$$

$$\dot{P} = -\frac{\delta H}{\delta X} \tag{2.21}$$

$$P(t_f) = \frac{\delta \varphi(X(t_f))}{\delta X} \tag{2.22}$$

$$\frac{\delta H}{\delta u} = 0 \tag{2.23}$$

If t_f is free and that $X(t_f)$ is constrained to the terminal manifold defined by the equation

$$N(X(t_f,\ t_f) = 0 \tag{2.24}$$

Then we have the following conditions:

$$H(X^*,\ u^*,\ P,\ t) \leq H(X,\ u,\ P,\ t) \quad \text{at all } t \in [t_0, t_f] \tag{2.25}$$

$$\dot{X} = \frac{\delta H}{\delta P} \tag{2.26}$$

$$\dot{P} = -\frac{\delta H}{\delta X} \tag{2.27}$$

$$P = \frac{\delta \varphi}{\delta X} + (\frac{\delta N^T}{\delta X})v \tag{2.28}$$

$$-H = \frac{\delta \varphi}{\delta t} + (\frac{\delta N^T}{\delta t})v \ \text{ at } t = t_f \tag{2.29}$$

where v is a lagrange multiplier. The equations ($\dot{X}$, $\dot{P}$) are known as two point boundary value problem (TPBVP). Some of the initial conditions ($P(t_0)$) are unspecified, once it is known, the solution will be straight forward. The maximum principle does not circumvent the computational difficulties of the TPBVP problems, but in some simple cases it does provide considerable insight into the form of the optimal control function so that we may be able to solve the problem indirectly (Lee [30]).

2.1.4 Numerical Solutions

The development of the two theoretical approaches we have seen in the previous section have led to the solution of many optimal control problems that are analytically tractable. Unfortunately, in general cases such analytic solutions are difficult to establish, and recourse must be made to numerical techniques to obtain the solutions.

There are many numerical methods to solve the optimal problem (see for example Polak [42], Dyer [16], and Hasdorff [22]). In general, for nonlinear optimal control problems, the following algorithmic procedure is usually undertaken

Step 1 : Making an initial guess in the control $U^0(t)$ (some algorithms start with an initial guess for the missing initial conditions), $i = 0$.

Step 2 : Compute the system state $X^i(t)$ that corresponds to u^i.

Step 3 : Linearize the TPBVP over the calculated trajectory (X^i, u^i).

Step 4 : Obtain a search direction by solving the linearized TPBVP.

Step 5 : Calculate a step size in the search direction that leads to the minimum cost. If the optimal solution has been found, stop; else, continue.

Step 6 : Compute a new better control u^{i+1}. Set $i = i + 1$, and go to Step 2.

In this section we will study one of the numerical methods, namely the gradient method, to show the computational difficulties which limit their application in real-time control systems.

The gradient algorithm is a direct method in which a sequence of nominal solutions converging to the optimum are generated. The new nominal solution is generated by incrementing the old nominal in the direction of the gradient of the return function $(\frac{\delta H}{\delta u})$.

Consider the problem of minimising the performance criterion

$$J(u) = \varphi(X(t_f),\ t_f) + \int_{t_0}^{t_f} L(X,\ u,\ t)dt \tag{2.30}$$

subject to

$$\dot{X} = f(X,\ u,\ t), \quad X(t_0) \text{ specified} \tag{2.31}$$

Using the maximum principle, we have the following costate equation

$$\dot{P}(t) = -P^T(t)(\frac{\delta f}{\delta X}) + (\frac{\delta L}{\delta X}) \tag{2.32}$$

with

$$P(t_f)^T = -\frac{\delta \varphi}{\delta X} \qquad \text{at } t = t_f \tag{2.33}$$

and the gradient of the cost function $J(u)$ is given by

$$\text{grad } J(u,t) = -(\frac{\delta f}{\delta u})^T P(t) + (\frac{\delta L}{\delta u})^T \tag{2.34}$$

The following algorithm can be used to solve this problem

Step 0 : Select a nominal control sequence $u^0(t)$, $t \in [t_0,\ t_f]$.

Step 1 : Set $i = 0$.

Step 2 : Compute $X^i(t)$, $t \in [t_0,\ t_f]$, by solving 2.31 with $u(t) = u^i(t)$.

Step 3 : Compute $P^i(t)$, $t \in [t_0,\ t_f]$, by solving the costate equation backwards with $u(t) = u^i(t)$, $X(t) = X^i(t)$, and $t \in [t_0,\ t_f]$.

Step 4 : Compute the gradient from equation 2.34, for $t \in [t_0,\ t_f]$, and with $u^i,\ P^i,\ X^i$.

Step 5 : For $t \in [t_0,\ t_f]$ set

$$h(u^i, t) = -\text{grad } J(u^i, t), \tag{2.35}$$

if $h(u^i, t) = 0$, stop; else, go to Step 6.

Step 6 : Compute a $\lambda_i > 0$ such that

$$-\lambda_i(1-\alpha) \parallel \text{grad } J(u^i) \parallel_2^2 \leq \theta(\lambda_i,\ u^i) \leq -\lambda_i\alpha \parallel \text{grad } J(u^i) \parallel_2^2 \tag{2.36}$$

for some $\alpha \in [0,\ 1]$ and

$$\theta(\lambda_i,\ u^i) = J(u^i + \lambda_i h(u^i)) - J(u^i) \tag{2.37}$$

Step 7 : For $t \in [t_0,\ t_f]$, set

$$u^{i+1}(t) = u^i(t) + \lambda_i h(u^i, t) \tag{2.38}$$

set $i = i + 1$, and go to Step 2.

After a number of iterations, the algorithm converges to the solution $u^*(t)$. The time required for the algorithm to converge to the solution depends on how close the initial guess $u^0(.)$ is to the optimal solution $u^*(t)$. Numerical techniques, such as the above algorithm, must be used to provide control for optimal system in two ways:

1. Open-loop control: In which the algorithm is run off-line, and then, the resulting control sequence $u^*(t)$ is fed to the system to take it from its initial state to the required state , provided that, the system is stationary, that is, it remains in its initial state during the computation time. Furthermore, all the possible sources of disturbances remain in the same predicted state during the time interval used to take the system to its final state. An example of this systems, is launching a missile to an orbit or rendezvous point.

2. Closed-loop control: In this case, the control to be applied must depends on the system state and the time to go to the final state. To achieve this, the above algorithm should run and produce the control at every instant in time, and only the control corresponds to that instant is applied. The current state will be the initial value for the next run until the specified final state or time is reached. This requires that all the computations must be completed within some specified period of time (the sample period).

It is obvious, that the numerical solution of complex problems (nonlinear and high dimensional) is computationally demanding. The situation when considering on-line computer control of systems is further complicated due to the need for high sample rates. Because of these difficulties, designers usually try to avoid the use of numerical techniques to solve on-line computer control problems which need on-line real-time computations. They simplify their problems by considering it to be linear, piecewise linear, and/or reducing its order to make it analytically solvable or computationally feasible. Most of the designs use off-line computations of numerical algorithms to provide good insight into the form of optimal control which is much easier to implement. For example, the references Calise [11], Ellert [17], and Grimm [21] show how some of the optimal control problems can be tackled for real-time implementation.

After the recent development in the area of parallel processing, many investigators have been encouraged to use the new technology to reduce the execution time of the numerical algorithms. As a result, some existing algorithms have been modified, and new parallel algorithms developed and used in various application areas.

There are many parallel algorithms which address the optimisation problems in general and optimal control problems in particular (see for example Travassos [53], Raczynski [44], Keller [28], Chang [12], and Menon [33]). Some of these algorithms execute the most costly operations in parallel to reduce the time complexity of a sequential algorithm (Raczynski [44]). Other algorithms achieve parallelism by dividing the optimisation interval $[t_0,\ t_f]$ into a number of subintervals, thus, the optimisation problem is converted into a number of subproblems which are solved in parallel togather with introducing some criterion to achieve continuity between the subproblems. Examples of this kind of algorithm are the parallel shooting and parallel quasilinearization algorithms.

Parallelism, as far as numerical optimisation algorithms are concerned, does not simply mean using several processors to accelerate some vector or matrix operations in the algorithm because the whole algorithm will be classified as a sequential one, despite the fact that some elemental operations are executed concurrently. Thus, parallel processing must be used at a higher level. For the gradient algorithm, there are two main tasks: 1) Evaluation of the gradient, which needs one forward integration of the system equations, linearization over the whole trajectory, and backward integration of the costate equation. 2) Evaluation of the cost function to find the step size, which needs several forward integrations of the system equations and evaluations of the cost function. These two tasks must be executed in parallel, and in the cost function calculation, we can lunch a number of evaluations with

different step sizes (λ). In this case, parallelism is achieved in our algorithm and it will be of the type GY/Z (see Travassos [53]), where Y is the number of tasks, both of gradient and cost function type, which might be executed in parallel and Z is the number of cost function tasks. Thus, the number of gradient tasks being executed concurrently with the cost tasks equal $Y - Z$. The sequential algorithm is of type $G1/1$ where no parallelism exists.

Parallel implementation of optimisation algorithms can clearly speed up the execution time, and it may be widely applied to control systems of the open-loop type. But, their time complexity is still high which makes it unsuitable for systems requiring on-line computations unless they are simple. Furthermore, for closed-loop systems, the execution time can not be fixed, which is an important property in computer control, because the time to convergence depends on the initial guess. And, if a large number of processors are used, the time required for inter-processor communications may become significant and effect the efficiency of the implementation. These difficulties force us to divert our attention from such methods and instead to consider more practical solutions as we will see in the next chapters.

2.2 Parallel Processing

In recent years, the computer industry has seen a remarkable growth in computing speed. However, better devices with higher speed are not the sole factor contributing to high performance. It was evident that the Von Neuman or the sequential computer trend must soon come to an end because the laws of physics limit further significant advances in this direction. Therefore, the main way around this difficulty is to use parallel processing techniques, that is machines which possess several computing devices and are able to process several bits of information simultaneously.

Parallel processing encompasses several diverse computer architectures which

can be characterized into three distinct classes: pipelined computers, array processors, and multi-processor systems (see Bertsekas [6], Hockney [23], Patton [40], and Hwang [25]). Parallelism may be exploited at four different levels:

1. Job level
2. Task level
3. Inter-instruction level
4. Intra-instruction level

The last two levels require the user to have an intimate knowledge of the system architecture on which the program is to be executed, with the intra-instruction parallelism being directly exploited by hardware measures. Therefore, we will address only job and task parallelism.

In pipeline computers successive instructions are executed in an overlapped manner. Each pipeline cycle is set equal to the delay of the slowest stage. The flow of data from a stage is synchronized by a common clock control. Theoretically, a k-stage pipeline processor could be at most k times faster, therefore the longer the pipeline, the greater the efficiency. The pipeline architecture is attractive for vector processing, where the instruction operations are executed repeatedly.

In array processors, the architecture include a central control unit, several processing elements with individual memories, and an inter-connection network. The multiple streams of data are fed into the processing elements which then execute the instructions from the control unit simultaneously.

In multi-processor systems, several comparable processors with common memory, input/output capabilities, and peripheral devices. Each processing element contains its own control unit and so they all operate independently of each other, up to the limit imposed by communication requirements. Because of this relative

independence, this type has the greatest potential for a wide range of applications. However, the application of such systems is currently hampered by several architectural problems, such as efficient task scheduling to minimise the overall idle time of the processors, data inconsistencies, and the speed of data transfers between the processors.

The solution to the execution time constraint promised by distributed parallel algorithms can be exploited on multi-processor machines which can possess vast computational power. Such computers generally have the shared-bus architecture (SBA) shown in Figure 2.3, where all the microprocessors share the system bus and hence compete to use it. This in reality can delay the computations. An alternative

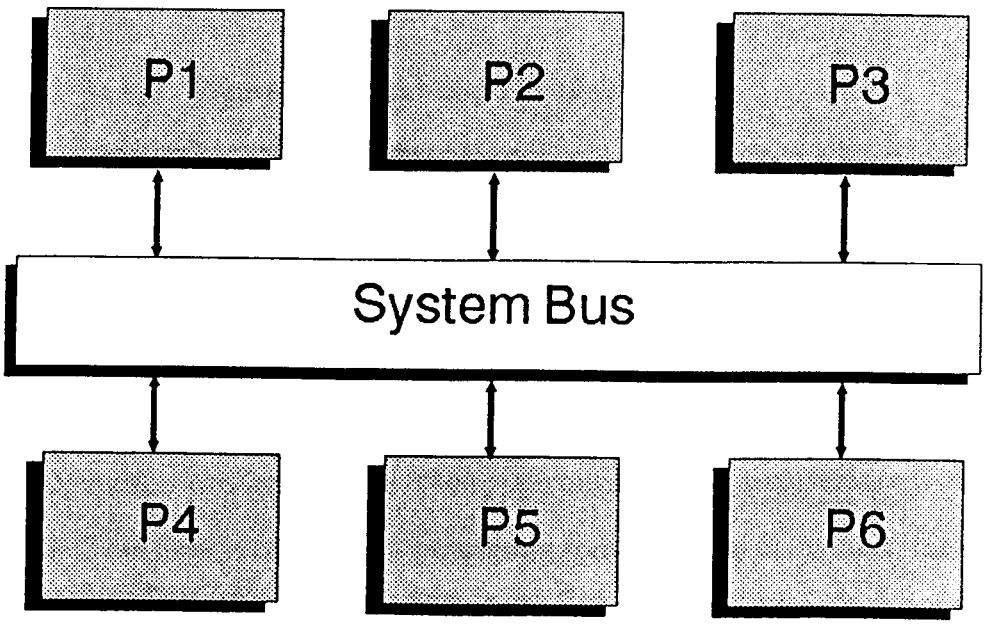

Figure 2.3: Shared-bus architecture

approach is the point-to-point (PTP) architecture shown in Figure 2.4 which allows each processor to communicate with several others through separate links, thus relieving the system from the bus bottle-neck in communications. One commercially available design of the later (PTP) architecture is the INMOS transputer which we

now concentrate on since this hardware is used in the work described here.

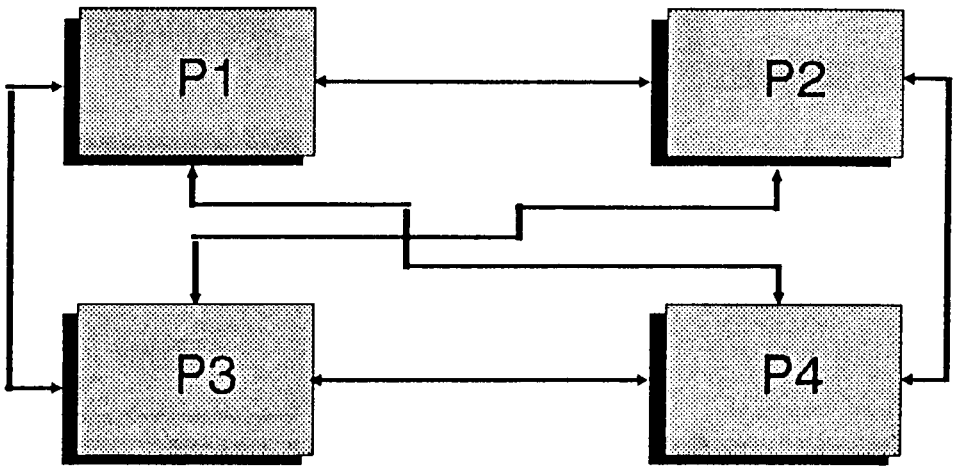

Figure 2.4: Point-to-point architecture

2.2.1 The Transputer System

The INMOS transputer is a general-purpose single-chip microprocessor designed by INMOS Ltd (see Mitchell [35], and INMOS [27]). The latest, and the most sophisticated, version available is the T800 transputer whose architecture is shown in Figure 2.5. This design combines a processing unit, floating-point unit, memory, and communication links on a single silicon subtrate.

In any application requiring a powerful multi-processor system, a single transputer can be connected to other (any number) of similar devices using its four high-speed serial duplex links. The presence of the on-board floating-point unit significantly enhances the performance of the transputer by allowing the execution of 1.5 M flops (million of floating-point operations) per second, with a processor speed of 20 MHz. The design supports the concurrent high-level Occam programming lan-

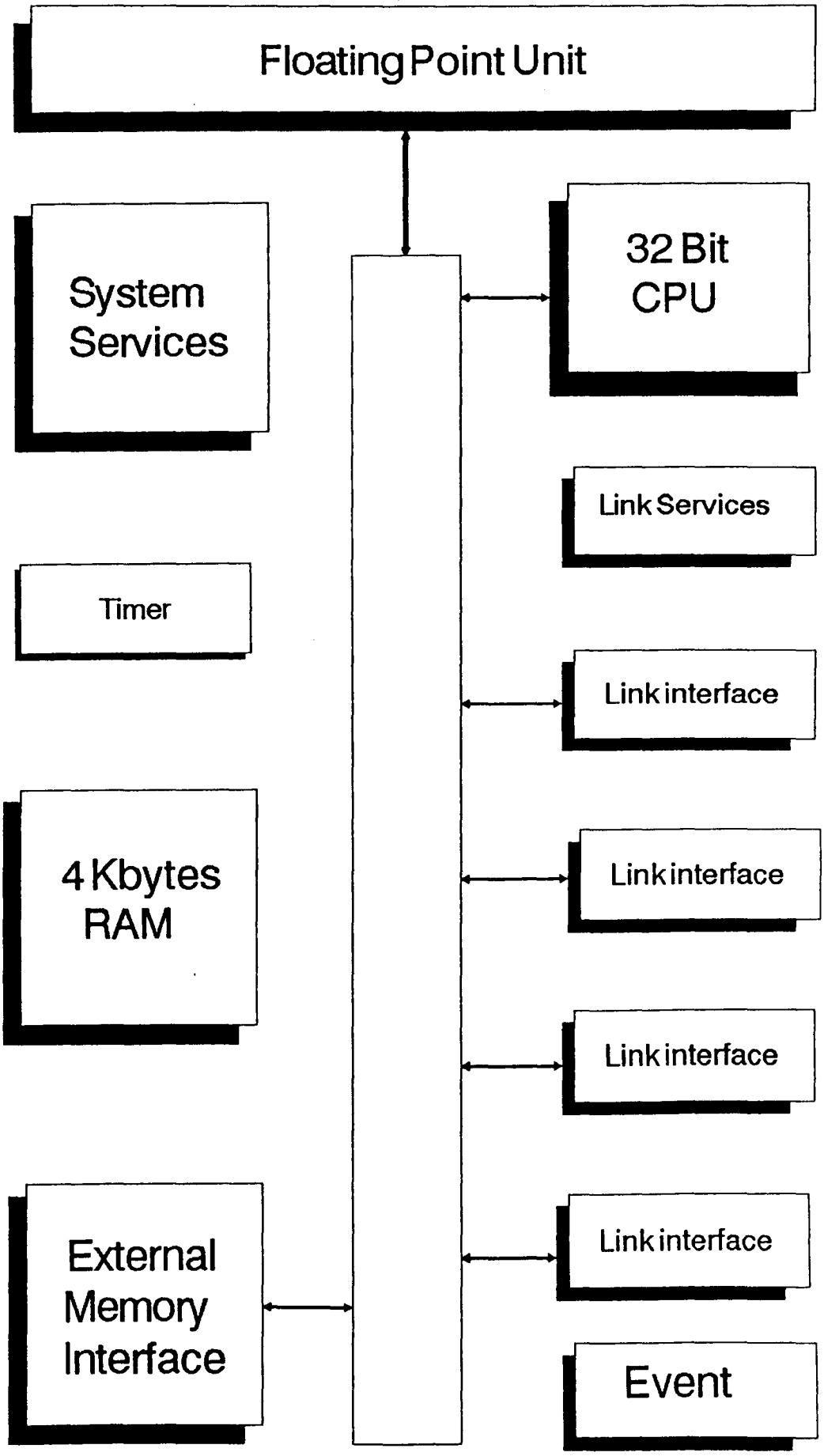

Figure 2.5: T800 transputer architecture

guage (see Mitchell [35]), and other high-level language compilers, to support the use of parallel versions of programming languages such as C, Fortran and Pascal.

In implementing a parallel application, the software source codes must be written in an appropriate programming language. Two high-level languages were available for this research namely Occam and parallel C. Parallel C with the 3L C-compiler has been used in this work. This is largely compatible with the standard C (see Purdum [43]) where the main features have been retained and parallelism is achieved in two ways:

1. Task level: A task is code for a single transputer, it contains all the necessary functions along with its "main()" function.

2. Thread level: Processes can be created at run-time, with two priority levels (urgent and nonurgent), and then terminated once finished. This is done by the dynamic allocation property associated with C.

The construction of each task has the following form:

```
#include < chan.h >
.
.
int                external variables declaration
float
.
.
main(argc, argv, envp, inp, ins, outp, outs)
int  ins, outs, argc;
char  * argv[ ], *envp[ ];
CHAN  * inp[ ], *outp[ ];
```

{

main body

}

void func1( )

{

func1 body

}

.

.

The only means of communications between different tasks is through channels (inp, outp). If variable A of 4 bytes is to be sent, and B of 4 bytes is to be received then the C structure would be

$chan - out - message(4,\ \&A,\ outp[1])$;

$chan - in - message(4,\ \&B,\ inp[1])$;

where the number 1 appearing with the channels ($outp,\ inp$) represents a software link number. A separate configuration file is set where the tasks and links arrangement with the channel number allocated to each task are defined.

From the practical experiments conducted on the transputer equipment available in the Department of Automatic Control & Systems Engineering, the execution times for arithmetic operations and point-to-point transmission of one floating point value (32 bits) are shown in Table 2.1. These values are used in the theoretical performance assessment of the algorithms presented in the forthcoming chapters.

2.3 Fault Tolerance Principles

Fault tolerance techniques have been used in computing systems, since the 1950's, to provide highly reliable hardware operations. As more systems have been employed

Table 2.1: T800 transputer Execution times

Operation	Execution time μs
Addition or subtraction	1.75
Multiply	1.95
Divide	2.25
Sequare root	8.2
sin	18.3
cos	18.5
$\tan^{-1}$	20.5
Data transmission(f.p)	11

on critical tasks, the use of fault tolerance has increased and spread in both military and industrial applications (see for example Anderson [1] and Pau [38]).

There are widely different requirements for the reliability and availability of computers and systems. Wrong outputs from a computer may simply be inconvenient, but in some applications, where human lives and/or vast sums of money may be at stake, wrong outputs cannot be tolerated. For example, the space shuttle is totally dependent on the proper operation of its computers, a mission can not even be aborted if the computers fail. Other example is the fuel-efficient aircraft, where computer control is essential to provide the fine degree of control surface actuation required to maintain its stability which the crew could not achieve. System reliability is approached by fault prevention, the goal of which is to prevent the failure of systems by ensuring that all conceivable causes of unreliability have been removed from the system before reliance is placed on its operation. There are two traditional ways to achieve fault prevention, namely fault avoidance and fault removal.

Fault avoidance is concerned with design methodologies and the selection of techniques and technologies which aim to avoid the introduction of faults during the design and construction of a system by the use of reliable components, and methodologies for coping with the complexities of hardware and software designs.

Despite the adoption of fault avoidance techniques, faults are usually present in a constructed system because of the unavailability (or cost) of fault-free hardware components and the complexity inherent in most systems.

Fault removal is concerned with checking the implementation of a system and removing any fault which are thereby exposed. A system can be extensively tested to reveal faulty hardware components and design faults. Faults are then removed and the system put into operation. But if faults develop or remain then system failure is likely to occur. Fault avoidance and removal were found to be insufficient for reliable operation of the hardware components because they age and deteriorate and can therefore become faulty. It was recognized that fault tolerance (that is, providing reliability despite the presence of faults) was often required, at least to protect the system against such hardware component faults.

There are four phases of fault tolerance which, when taken together, provide the general means by which faults can be prevented from leading to system failures. These phases are:

1. **Error detection:** The starting point for all fault tolerance strategies is the detection of an erroneous state. In principle, the more effort used in error detection the better the resulting system will be for reliable operation, since if all errors are detected and appropriate techniques applied to recover from them, then no fault can lead to system failure. In practice there will be limitations to the amount of error detection that can be provided, such as the cost of the redundancy needed and the overheads incurred at run-time by extensive

checking.

The measures for error detection that can be incorporated in systems (computer systems in particular) can take many forms; the majority of the measures adopt checks and fall into the following classifications:

(a) Replication checks;

(b) Timing checks;

(c) Reversal checks;

(d) Reasonableness checks;

(e) Structural checks;

(f) Diagnostic checks;

(g) Coding checks.

Replication checks involve some duplication, triplication, etc, of the activity of the system, then the results checked for consistency. Timing checks can be provided if the specification of a component includes timing constraints on the provision of its service. If the constraints are not met, then the timing check can raise a "time out" exception to indicate the failure of the component. Reversal checks takes the outputs from a system and calculate what the inputs should have been in order to produce that output, the calculated inputs can then be compared with the actual inputs to check whether there is an error. Reasonableness checks make checks for acceptability in which the value of an object is in an acceptable range. Structural checks are applied to complex data structures which consist of a set of elements linked together by pointers. Checks will be concerned with the consistency of information contained in a data structure. Diagnostic checks are concerned with checking the behaviour

of the components from which the system is constructed. Coding checks are based on the redundancy in the representation of an object, an example is parity checks.

2. **Damage confinement and assessment:** Once an error is identified, there can be no guarantee that all of the unwanted consequences of a fault (the damage) will be identified. Damage can spread as a result of any subsequent flow of information. The designer must be able to identify the possible system activity that could have followed an erroneous transition, and the possible flow of information. When the possible damage is identified, measures can be taken to remove it.

3. **Error recovery:** The previous two phases are passive in the sense that they are not intended to effect any changes to the system. The remaining phases are active since they do change the system and thereby enable faults and their consequences to be tolerated. The aim of error recovery is to eliminate errors from the system state that could lead to system failure. After an error has been detected and the damage assessment phase has produced an estimate of the extent to which the system state is erroneous, it will be necessary to eliminate those errors from the system state.

   State restoration is one of the measures used to recover from an error, namely to replace the entire state of the system. This is usually referred to as a "reset" of the system. the most basic reset strategy is to place a system in some predefined state, for example an initial state or a prior state of a system (backward error recovery). Forward error recovery techniques manipulate some portion of the current state to produce a new state in the hope that the new state will be error free.

4. **Fault treatment and continued service:** If the above techniques succeed in placing the system in an error free state, the system can return to normal operation since the immediate danger of failure has been averted. However, this may not be enough to ensure reliability, because those techniques leave the fault which produce the error untreated. Repeated manifestation of a fault can force a system to fail despite the efforts of the fault tolerance techniques described so far. Faults treatment techniques attempt to eradicate faults from a system so that service can be maintained. These techniques provide treatment for the fault itself and can be divided into two stages, namely "fault allocation" and "system repair".

   Techniques for system repair will necessarily be based on "reconfigurating" the system in such a way that characteristics of use of suspect components are modified to some extent. The standard approach is to make no further use of the suspect components and replace them by spare components if available.

The fundamental concepts which will be used in the research presented have been introduced and so we can now proceed to the next chapter where the modelling of the aircraft systems is discussed.

Chapter 3

Dynamic Model of the Aircraft

3.1 Introduction

Before analysing any problem, it is necessary to build the foundations and derive the general equations which describe the behaviour of the system under consideration. In this connection, the derivation of the aircraft equations of motion is reviewed here.

It is well known from classical mechanics that the translational motion of a body is described by the equation of linear momentum while the rotational motion is governed by the equation of angular momentum. The forces that influence the motion of an aircraft are of six types (see for example Babister [2], and Blakelock [7]):

1. Inertial forces arising from the mass distribution, linear, and angular acceleration of the aircraft.

2. Aerodynamic forces and moments depending on the angular velocities of the aircraft.

3. Aerodynamic forces and moments depending on the linear velocities of the

aircraft.

4. Aerodynamic forces and moments due to the application of controls.

5. Gravitational forces.

6. Propulsive forces.

The motion of an aircraft can be determined by considering the above forces acting on a rigid body, which is free to move in any direction.

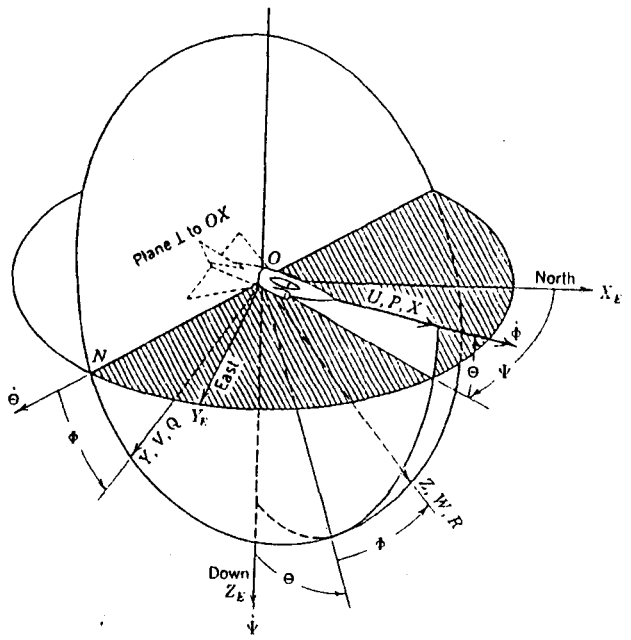

Figure 3.1: Aircraft and earth axes

3.2 The Aircraft Attitude with Earth

In order to describe the motion of the aircraft with respect to the earth or inertial space, it is necessary to specify the orientation of one axis system with respect to another. This can be done through the use of a set of angles called "Euler angles".

Consider an earth axis system with its origin at the centre of gravity of the aircraft and non-rotating with respect to the earth (see Figure 3.1).

Let OX_E and OY_E be in the horizontal plane and OZ_E vertical and down. OX_E may be taken as North or any other fixed direction. Let the following angles indicate the rotation of the XYZ axis from the earth axis.

Ψ the angle between OX_E and the projection of the OX axis on the horizontal plane.

$\dot{\Psi}$ is the a vector along OZ_E.

Θ the angle between the horizontal and the OX axis measured in the vertical plane.

$\dot{\Theta}$ is a vector along ON.

Φ the angle between ON and the OY axis measured in the OYZ plane.

$\dot{\Phi}$ is a vector along OX.

Thus, the angles Ψ, Θ, and Φ specify the orientation of the aircraft axis system with respect to the earth. The positive direction of these angles is indicated in Figure 3.1.

It is necessary to be able to transform the components of angular velocity of the aircraft from the earth axis to the aircraft axis system; This is done as follows. We take the components $\dot{\Psi}$, $\dot{\Theta}$, $\dot{\Phi}$ and project them along OX, OY, OZ axes to give

$$P = \dot{\Phi} - \dot{\Psi} \sin\Theta \tag{3.1}$$

$$Q = \dot{\Theta} \cos\Phi + \dot{\Psi} \cos\Theta \sin\Phi \tag{3.2}$$

$$R = -\dot{\Theta} \sin\Phi + \dot{\Psi} \cos\Theta \cos\Phi \tag{3.3}$$

These equations can be solved for $\dot{\Psi}$, $\dot{\Theta}$, and $\dot{\Phi}$ to yield

$$\dot{\Theta} = Q \cos\Phi - R \sin\Phi \tag{3.4}$$

$$\dot{\Phi} = P + Q \sin\Phi \tan\Theta + R \cos\Phi \tan\Theta \tag{3.5}$$

$$\dot{\Psi} = \frac{Q \sin\Phi + R \cos\Phi}{\cos\Theta} \tag{3.6}$$

where P, Q, R are the angular velocities (in rad/sec) of the aircraft about the OX, OY, OZ axes respectively.

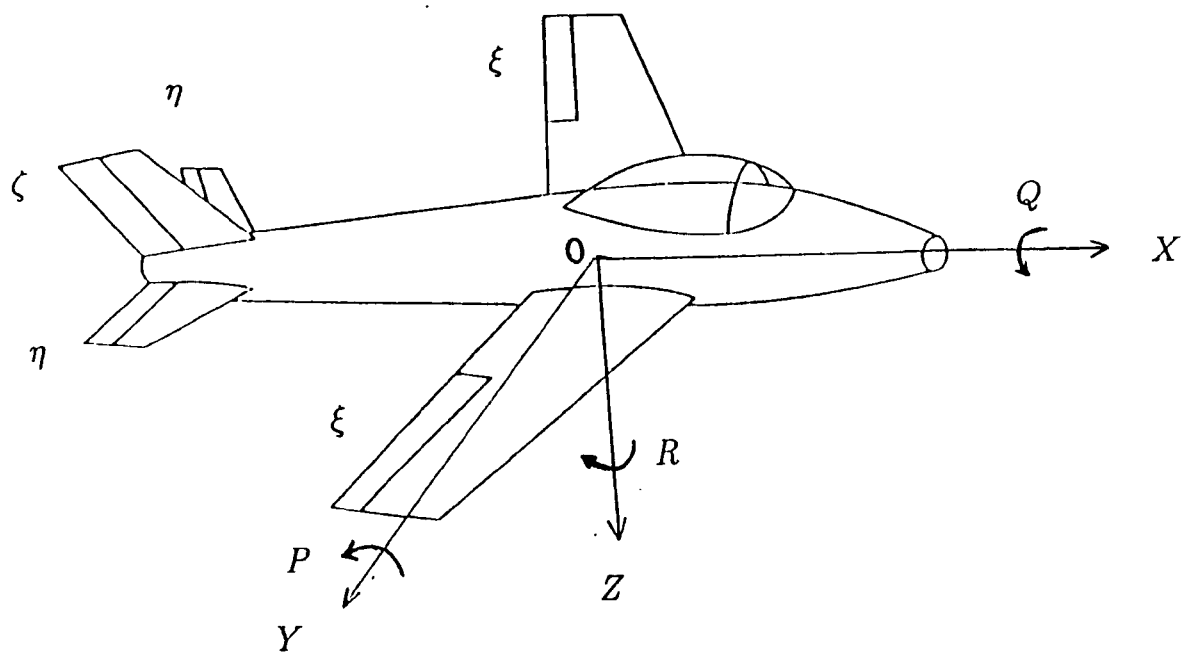

Figure 3.2: Aircraft axes and velocity components

The components of the gravity force along the aircraft axes are along

$$OX: \quad -mg \sin\Theta \tag{3.7}$$

$$OY: \quad mg \cos\Theta \sin\Phi \tag{3.8}$$

$$OZ: \quad mg \cos\Theta \cos\Phi \tag{3.9}$$

where $mg = m \times g$ is the aircraft weight.

3.3 The Aircraft Dynamic Equations

Let us take a set of rectangular axes $OXYZ$, as shown in Figure 3.2. The axes (body axes) are fixed in the aircraft and move with it. OX and OZ are in the plane of symmetry of the aircraft, with OZ downwards and OY to starboard. Let U, W, V be the velocity components of the centre of gravity along OX, OY, OZ respectively; let P, Q, R be the components of angular velocity of the axis frame $OXYZ$ about OX, OY, OZ respectively. The positive senses of these angular velocities are the clockwise directions about the respective axes. Let m be the mass of the aircraft, I_x, I_y, I_z the moments of inertia of the aircraft about OX, OY, OZ respectively, and let I_{yz}, I_{xz}, I_{xy} denote the products of inertia with respect to OYZ, OZX, OXY respectively. The aircraft is assumed to be acted upon by external forces which have components F_x, F_y, F_z along OX, OY, OZ respectively. The moments of the external forces about OX, OY, OZ are R_m, P_m, Y_m respectively.

Considered as a rigid body, the motion of the aircraft is completely defined by the following six equations:

Translation motions:

1. Parallel to OX:

$$m(\dot{U} - RV + QW) = F_x \tag{3.10}$$

2. Parallel to OY:

$$m(\dot{V} - PW + RU) = F_y \tag{3.11}$$

3. Parallel to OZ:

$$m(\dot{W} - QU + PV) = F_z, \tag{3.12}$$

Angular motions:

4. About OX:

$$\begin{aligned} R_m &= I_x\dot{P} - (I_y - I_z)QR - I_{yz}(Q^2 - R^2) - I_{xz}(\dot{R} + PQ) \\ &\quad -I_{xy}(\dot{Q} - RP) \end{aligned} \tag{3.13}$$

5. About OY:

$$\begin{aligned} P_m &= I_y\dot{Q} - (I_z - I_x)RP - I_{xz}(R^2 - P^2) - I_{xy}(\dot{P} + QR) \\ &\quad -I_{yz}(\dot{R} - PQ) \end{aligned} \tag{3.14}$$

6. About OZ:

$$\begin{aligned} Y_m &= I_z\dot{R} - (I_x - I_y)PQ - I_{xy}(P^2 - Q^2) - I_{yz}(\dot{Q} + RP) \\ &\quad -I_{xz}(\dot{P} - QR) \end{aligned} \tag{3.15}$$

The external forces and moments will be of two types: aerodynamic (and propulsive) forces and moments, and gravitational forces. There are no moments due to gravity, since we have taken the axes to pass through the centre of gravity.

The mass distribution of the aircraft is taken to be symmetrical with respect to the plane OXZ. Thus

$$I_{xy} = \sum xy\ \delta m = 0 \tag{3.16}$$

$$I_{yz} = \sum yz\ \delta m = 0 \tag{3.17}$$

while I_{xz} will not be zero unless OX and OZ are the principal axes of inertia. Therefore, from the equations 3.4 to 3.17 we have the following equations which define the motion of the aircraft about the body axes:

$$\dot{U} = \frac{F_x}{m} + RV - QW \tag{3.18}$$

$$\dot{W} = \frac{F_z}{m} + QU - PV \tag{3.19}$$

$$\dot{V} = \frac{F_y}{m} + PW - RU \tag{3.20}$$

$$\dot{Q} = \frac{P_m + (I_z - I_x)RP + I_{xz}(R^2 - P^2)}{I_y} \tag{3.21}$$

$$\dot{P} = \frac{R_m + (I_y - I_z)QR + I_{xz}(\dot{R} + PQ)}{I_x} \tag{3.22}$$

$$\dot{R} = \frac{Y_m + (I_x - I_y)PQ + I_{xz}(\dot{P} - QR)}{I_z} \tag{3.23}$$

$$\dot{\Theta} = Q \cos\Phi - R \sin\Phi \tag{3.24}$$

$$\dot{\Phi} = P + (Q \sin\Phi + R \cos\Phi)\tan\Theta \tag{3.25}$$

$$\dot{\Psi} = \frac{Q \sin\Phi + R \cos\Phi}{\cos\Theta} \tag{3.26}$$

$$\dot{H} = U \sin\Theta - W \cos\Theta \cos\Phi - V \sin\Phi \cos\Theta \tag{3.27}$$

For trajectory studies, the above equations are divided into two sets of equations, namely, longitudinal (the equations 3.18, 3.19, 3.21, 3.24, and 3.27) and lateral (the equations 3.20, 3.22, 3.23, 3.25, and 3.26). The two motions are then normally decoupled by means of engineering hypotheses (see chapter 7) and then handled separately.

3.4 The Aerodynamic Forces and Moments

The forces and moments which appear in the equations 3.18 to 3.23 are defined as follows:

$$F_x = Th - (D + X_d)\cos\alpha_b + L \sin\alpha_b - mg \sin\Theta \tag{3.28}$$

$$F_z = -(D + X_d)\sin\alpha_b - L\ \cos\alpha_b + mg\ \cos\Theta\cos\Phi \tag{3.29}$$

$$F_y = Y + mg\ \sin\Phi\ \cos\Theta \tag{3.30}$$

$$\begin{aligned} P_m &= P_{mt} + L(C_g - C_1)\bar{C}\ \cos\alpha_b + D(C_g - C_1)\bar{C}\ \sin\alpha_b \\ &\quad + X_d(C_2 + (C_g - C_1)\bar{C})\sin\alpha_b + Gp \end{aligned} \tag{3.31}$$

$$R_m = R_{mt}\ \cos\alpha_b - Y_{mt}\ \sin\alpha_b \tag{3.32}$$

$$Y_m = Y_{mt}\ \cos\alpha_b + R_{mt}\ \sin\alpha_b + Y(C_g - C_1)\bar{C} + Gy \tag{3.33}$$

where Th is the engine thrust; D, L are drag and lift forces respectively; α_b is the angle of attack relative to fuselage; X_d is the engine intake drag force; Y is the aerodynamic side force; P_{mt} is the aerodynamic pitching moment about the wing quarter chord; R_{mt}, Y_{mt} are the aerodynamic rolling and yawing moments about the stability axis; C_g is the position of the centre of gravity behind the leading edge of the standard mean chord $\bar{C}$ as a fraction of $\bar{C}$; C_1, C_2 are constants; Gp and Gy are the engine gyroscopic effects.

The lift, drag, and pitching moment are given by:

$$L = \frac{1}{2}\rho V_r^2 S_w C_l \tag{3.34}$$

$$D = \frac{1}{2}\rho V_r^2 S_w C_d \tag{3.35}$$

$$P_{mt} = \frac{1}{2}\rho V_r^2 S_w C_m \tag{3.36}$$

The aerodynamic derivatives (see Babister [2]) C_l, C_d, and C_m depend upon the angle of attack; the flap setting; the setting angle of the tail (the stabilizer); the elevator control angle η; the wing area S_w; the tail area S_t; the wing span b; the height of the tail above fuselage and the distance between the wing and the tail quarter chord. It also depends upon other quantities which are functions of the Mach number M, and are given in the form of graphs in the aircraft engineering data [8]. ρ is the air density and $V_r = \sqrt{U^2 + W^2 + V^2}$ is the aircraft relative velocity.

The rolling, yawing moments and side force are given by:

$$Y = \frac{1}{2}\rho V_r S_w Y_v V + \frac{1}{2}\rho V_r^2 S_w Y_\zeta \zeta \tag{3.37}$$

$$\begin{aligned} R_{mt} = {} & \frac{1}{2}\rho V_r S_w b[L_v V + V_r L_\xi \xi + V_r L_\zeta \zeta + bL_p \\ & (P\ \cos\alpha_b + R\ \sin\alpha_b) + bL_r(R\ \cos\alpha_b - P\ \sin\alpha_b)] \end{aligned} \tag{3.38}$$

$$\begin{aligned} Y_{mt} = {} & \frac{1}{2}\rho V_r S_w b[N_v V + V_r N_\xi \xi + V_r N_\zeta \zeta + bN_p \\ & (P\ \cos\alpha_b + R\ \sin\alpha_b) + bN_r(R\ \cos\alpha_b - P\ \sin\alpha_b)] \end{aligned} \tag{3.39}$$

where ξ, ζ are the aileron and rudder angles (lateral control variables); the aerodynamic coefficients Y_v, $Y_\zeta, \ldots,$ N_r are functions of the angle of attack which were given in the form of graphs in the aircraft engineering data. The C_g position as well as the inertial moments (I_x, I_y, I_z, I_{xz}) are functions of the aircraft mass (m) and were also given as graphical data.

3.5 The Engine Model

The aircraft under consideration has one turbojet engine whose dynamics are as follows [8]:

Let T_s, P_s be the static temperature (K) and static pressure (N/m^2); let M be the Mach number (the ratio of aircraft velocity to the speed of sound); and let N be the engine rotor speed (number of revolutions per minute). Then the total temperature and pressure is given by

$$T_t = T_s(1 + 0.2M^2) \tag{3.40}$$

$$P_t = P_s(1 + 0.2M^2)^{3.5} \tag{3.41}$$

let

$$E_1 = \frac{N}{\sqrt{T_t}} \tag{3.42}$$

$$E_2 = C_3 + C_4 E_1 + C_5 E_1^2 \tag{3.43}$$

Then the thrust will be

$$Th = [E_2(1 + C_6 M^2)^{3.5} - 1] C_7 P_s \tag{3.44}$$

and the intake momentum drag is

$$X_d = C_8 V_r \frac{P_t}{\sqrt{T_t}} (C_9 + C_{10} E_1 + C_{11} E_1^2) \tag{3.45}$$

The engine gyroscopic moments are

$$Gp = C_{12} NR \tag{3.46}$$

$$Gy = C_{13} NQ \tag{3.47}$$

Let the demanded throttle position be denoted by γ, then the actual throttle position is given by

$$\gamma_a = \frac{1}{1 + 2S} \gamma \qquad \gamma \in [0, 1] \tag{3.48}$$

and the corresponding engine rotor speed is

$$N = N_{idle} + (N_{max} - N_{idle}) \gamma_a^{1/2} \tag{3.49}$$

where

$$N_{idle} = C_{14} + C_{15} H + C_{16} V_a \tag{3.50}$$

and where V_a is the equivalent airspeed (knots), and N_{max} is the maximum permissible rotor speed. From the equations 3.48 to 3.50 we have

$$\dot{N} = \dot{N}_{idle} + \frac{\gamma (N_{max} - N_{idle})^2 - (N - N_{idle})^2}{4(N - N_{idle})} \tag{3.51}$$

Equation 3.51 is then combined with the aircraft dynamic equations (3.18 to 3.27) to form the aircraft nonlinear equations of motion. The constants (C_1 to C_{16}) and N_{max} are obtained from the aircraft engineering data.

3.6 The Atmosphere Model

To assess the aircraft modelling, it is important that a standard atmosphere be defined. The vertical distribution of the physical properties of the atmosphere depends on the height and time. For the altitude region between sea level and height $H = 6.5 \times 10^4$ ft, the atmospheric properties can be obtained by taking into account the variation of both the acceleration of gravity (g) and the air molecular weight with the altitude (see Miele [37]).

There are two basic equations which must be satisfied by the gas composing any atmosphere, these are

$$dP = -\rho g dH \tag{3.52}$$

$$P = \rho R T \tag{3.53}$$

where P is the pressure; ρ is the density; g is the acceleration of gravity; H is the altitude; R is the gas characteristic constant; and T is the temperature. If we denote the quantities evaluated at sea level by the subscript o, then the geopotential altitude ($\bar{H}$) is defined as

$$\bar{H} = \frac{1}{g_o} \int_0^H g dH \tag{3.54}$$

and the molecular temperature (static temperature) is defined as

$$T_s = T \frac{w_o}{w} = T \frac{R}{R_o} \tag{3.55}$$

where w is the molecular weight; From equations 3.53 to 3.55 we can get the set of differential equations

$$\frac{dP}{P} = -\frac{g_o}{R_o T_s} d\bar{H} \tag{3.56}$$

$$\frac{d\rho}{\rho} = -\left(\frac{g_o}{R_o} + \vartheta\right) \frac{d\bar{H}}{T_s} \tag{3.57}$$

where $\vartheta = \frac{dT_s}{dH}$ is the gradient of the molecular temperature.

The atmosphere data available for this research is known as the ARDC (see Miele [37]) model atmosphere and is based on the following assumptions

1. The space immediately surrounding the Earth is divided into eleven concentric layers, in each of which the gradient of the molecular temperature is constant.

2. For the six layers belonging to the lower atmosphere (from sea level to 3×10^5 ft), the composition of the air is constant.

3. For the five layers belonging to the upper atmosphere (3×10^5 ft to 2.3×10^6 ft), the composition of the air is variable, and the molecular weight is represented by inverse trigonometric functions in terms of the geopotential altitude.

4. The acceleration of gravity varies with altitude according to

$$g = g_o(\frac{r_o}{r_o + H})^2 \tag{3.58}$$

where $r_o = 20.9 \times 10^6$ ft is the radius of the Earth. This atmosphere model may be replaced by a simplified model using the International Standard Atmosphere model (ISA) [9] which consists of height bands which are identified by their temperature profiles:

Troposphere which has a constant temperature gradient from 288 K at sea level to 217 K at 36089 ft.

Lower Stratosphere has a constant temperature of 217 K from 36089 ft to 65616 ft.

Upper Stratosphere and Lower Chemosphere have a constant temperature gradient from 217 K at 65616 ft to 229 K at 104986 ft.

For a constant temperature gradient the local static pressure P_s is

$$P_s = P_{so}(\frac{T}{T_o})^k \tag{3.59}$$

For a constant temperature

$$P_s = P_{so}e^{n(H-H_o)} \tag{3.60}$$

where H_o is the altitude at the base of the height band, T is ISA standard temperature (K),

$P_{so} = 2116.2166\ lbf/ft^2$, $k = 5.2558797$ (Troposphere),

$P_{so} = 472.68000\ lbf/ft^2$, $n = -0.00004863461\ 1/ft$ (Lower Stratosphere),

$P_{so} = 114.34520\ lbf/ft^2$, $k = -34.163218$ (Upper Stratosphere and Lower Chemosphere).

We may now add an offset to the temperature profile to produce the desired sea level temperature T_{s1} (deg C), and then compute the speed of sound A, and air density ρ:

$$T_s = T + (T_{s1} - 15.0) \tag{3.61}$$

$$A = \sqrt{\mu R T_s} \tag{3.62}$$

$$\rho = \frac{P_s}{RT_s} \tag{3.63}$$

where $\mu = 1.4$, and $R = 3089.8114\ ft^2/sec^2/K$.

3.7 Conclusion

This chapter has presented the modeling assumptions used in this research. Much of these are standard but are given here for completeness so that the thesis form a complete and coherent piece of work. It is now possible to present the technical findings of this research, and is the next chapter, the first autopilot results are presented.

Chapter 4

Aircraft Longitudinal Optimal Control

4.1 Introduction

In the preceding chapters we have discused some of the fundamental aspects of computer control and aircraft systems. Now we start presenting the main contributions in this thesis, and in this chapter an optimal autopilot using the British Aerospace aircraft data for the longitudinal motion is described. This is a simplification of the overall control problem but it allows an easy and convenient stepping stone to the complete aircraft consideration presented in chapter 7. As we have seen in chapter 2, that for optimal solutions several steps have to be performed to reach the desired objectives, namely a linearization done about an operating point, the adjoint equations solved backwards in time over the optimisation horizon, the problem solved to give a descent direction, a new operating point deduced, the equations relinearized at the new conditions and so on. Assuming all the stages are well constructed, this iteractive scheme results in a solution to the original nonlinear problem. It is clear that the procedure is quite tortuous and in computer control applications the pro-

cessing needs to be performed in real-time. The situation, when considering optimal control of aircraft systems is further complicated because large dimensional models are necessary for adequate representation and fast sample rates are also required giving rise to a heavy computational burden that is extremely difficult to achieve in real-time implementation. A sampling rate of 200 Hz is normally expected to achieve good real-time control performance and so the control computations needs to be accomplished within a sampling period of 5 milliseconds. Due to the complexity of the computing task this is difficult to achieve using sequential computer systems and in fact are difficult even for the parallel computers which have recently became available.

Because of the above difficulties, optimal control problems are usually approached using simplified system dynamic equations. Here, we also simplify the problem to make it tractable by assuming that the nonlinear aircraft is pointwise linearized at particular instants. The concepts of parallel processing is then used to achieve real-time optimal control of the aircraft longitudinal motion.

4.2 The Aircraft Longitudinal Equations

As mentioned earlier, the aircraft equations presented in chapter 3 can be split into two sets where one set describes the longitudinal motion of the aircraft and the other describes the lateral motion. This can be achieved by the use of proper assumptions which uncouples the two sets. Thus, by assuming

$$P = R = V = 0 \tag{4.1}$$

the longitudinal equations are

$$\dot{U} = \frac{F_x}{m} - QW \tag{4.2}$$

$$\dot{W} = \frac{F_z}{m} + QU \tag{4.3}$$

$$\dot{Q} = \frac{P_m}{I_y} \tag{4.4}$$

$$\dot{\Theta} = Q \tag{4.5}$$

$$\dot{H} = U \sin\Theta - W \cos\Theta \tag{4.6}$$

$$\dot{N} = \dot{N}_{idle} + \frac{\gamma(N_{max} - N_{idle})^2 - (N - N_{idle})^2}{4(N - N_{idle})} \tag{4.7}$$

where the notation is as defined in chapter 3. These nonlinear equations can be represented as:

$$\dot{X} = f(X,\ u) \tag{4.8}$$

where $X = [U,\ W,\ Q,\ \Theta,\ H,\ N]^T$ is an 6-vector of state variables, and $u = [\eta,\ \gamma]^T$ is an 2-vector of control inputs.

Trajectory optimisation for the system (4.8) continue to be computationally challenging because the application of Pontryagin's maximum principle produces the general nonlinear two-point boundary value problems which are computationally demanding as mentioned earlier. In the approach here to solve this problem, successive linearizations are made to produce successive linear optimal problems which are solved to obtain the optimal control. As time progresses the linearizing point and optimising interval moves forwards thus enabling real-time performance.

4.3 linearization of the Aircraft Model

Linear differential equations are easier to handle than the general nonlinear ones; it is therefore natural that most applicable control theory is based on linear models which approximate the system under consideration.

In many practical applications the ideal operating condition for the system is where all system inputs, states and outputs are constant in time. In such applications, the role of the control system is primarily that of a regulator to return

the system variables to their ideal values following disturbances. Suppose that the system is described by the nonlinear model

$$\begin{aligned} \dot{X}(t) &= f(X(t),\ u(t)), \quad \text{for all } t \\ Y(t) &= g(X(t),\ u(t)) \end{aligned} \tag{4.9}$$

where the vector functions f and g do not depend explicitly on time t. Let X_o, u_o, Y_o be an operating point; the process of linearization of (4.9) in the vicinity of these operating conditions consists of approximating of the vector functions f, g by linear functions

$$f(X,\ u) \simeq f(X_o,\ u_o) + A(X - X_o) + B(u - u_o) \tag{4.10}$$

$$g(X,\ u) \simeq g(X_o,\ u_o) + C(X - X_o) + D(u - u_o) \tag{4.11}$$

where A, B, C, D are constant matrices. Defining the perturbation variables

$$\tilde{X} = X(t) - X_o \tag{4.12}$$

$$\tilde{u} = u(t) - u_o \tag{4.13}$$

$$\tilde{Y} = Y(t) - Y_o \tag{4.14}$$

then we obtain

$$\frac{d\tilde{X}(t)}{dt} = \frac{d}{dt}(X(t) - X_o) = \frac{dX(t)}{dt} \tag{4.15}$$

$$Y(t) = Y_o + C\tilde{X}(t) + D\tilde{u}(t) \tag{4.16}$$

$$f(X(t),\ u(t)) \simeq A\tilde{X}(t) + B\tilde{u}(t) \tag{4.17}$$

and the relations

$$\begin{aligned} \frac{d\tilde{X}(t)}{dt} &\simeq A\tilde{X}(t) + B\tilde{u}(t) \\ \tilde{Y}(t) &\simeq C\tilde{X}(t) + D\tilde{u}(t) \end{aligned} \tag{4.18}$$

These expressions suggest that the dynamic behaviour of the perturbations $\tilde{X}$ and $\tilde{Y}$ in response to the perturbation input $\tilde{u}$ can be approximated by the solution of the linear time-invariant model obtained from (4.18) by replacing the approximation signs by equalities. The matrices A, B, C, D have elements calculated by

$$\begin{aligned} a_{ij} &= \frac{\delta f_i}{\delta X_j} \qquad b_{ij} = \frac{\delta f_i}{\delta u_j} \\ c_{ij} &= \frac{\delta g_i}{\delta X_j} \qquad d_{ij} = \frac{\delta g_i}{\delta u_j} \end{aligned} \tag{4.19}$$

at $X = X_o$ and $u = u_o$. Using the above results, the aircraft longitudinal equations can be linearized about the operating point $(X_o,\ u_o)$ to be

$$\dot{e}(t) = Ae(t) - B\Delta u(t) \tag{4.20}$$

where

$$e(t) = X(t) - X_o \tag{4.21}$$

$$\Delta u(t) = u(t) - u_o \tag{4.22}$$

and the matrices A, B will be of the form

$$A = \begin{bmatrix} F_{xu} & F_{xw} - Q & F_{xq} - W & F_{x\theta} & 0 & F_{xn} \\ F_{zu} & F_{zw} & F_{zq} & F_{z\theta} & 0 & 0 \\ P_{mu} & P_{mw} & P_{mq} & P_{m\theta} & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ \sin\Theta & -\cos\Theta & 0 & H_\theta & 0 & 0 \\ N_u & 0 & 0 & 0 & N_h & N_n \end{bmatrix} \tag{4.23}$$

$$B = \begin{bmatrix} F_{x\eta} & 0 \\ F_{z\eta} & 0 \\ P_{m\eta} & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & N_{\gamma} \end{bmatrix} \tag{4.24}$$

where $F_{xu} = \frac{\delta F_x / \delta u}{m}$, ... etc, are known as the aerodynamic derivatives which can be calculated using the corresponding equations presented in chapter 3.

Because the linearizing process is repeated successively when time progresses, the linear time-invariant model given by equation (4.20) can be considered as time-varying system of the form

$$\dot{e}(t) = A(t)e(t) + B(t)\Delta u(t) \tag{4.25}$$

4.4 The Linear Quadratic Regulator Problem

The linear quadratic regulator is an automatic feedback-control oriented approach to optimal control problems, in which, feedback can be realized in the optimal control of linear systems based upon a quadratic performance index. We consider the minimisation of the quadratic cost function

$$J = e^T(t_f)Fe(t_f) + \int_{t_0}^{t_f} [e^T(t)Qe(t) + \Delta u^T(t)R\Delta u(t)]dt \tag{4.26}$$

subject to the linear dynamics (4.25), where F and Q are (6×6) positive semidefinite symmetric matrices, and R is an (2×2) positive definite symmetric matrix. F, Q, R are the weighting factors which indicate the relative importance of the various terms in the control performance, and so they must be stated such that the objective is satisfied.

It is well known that problem (4.25) and (4.26) produces a two-point boundary value problem (see for example Banks [4]), which can be solved by applying the maximum principle.

We have seen in chapter 2 that the necessary conditions for an extremum are

$$\dot{\lambda} = -\frac{\delta H}{\delta e} = -Qe - A^T\lambda \tag{4.27}$$

$$\lambda(t_f) = \frac{d(e^T(t_f)Fe(t_f))}{de} \tag{4.28}$$

$$\frac{\delta H}{\delta \Delta u} = R\Delta u + B^T\lambda \tag{4.29}$$

$$\dot{e} = \frac{\delta H}{\delta \lambda} = A(t)e(t) + B(t)\Delta u(t) \tag{4.30}$$

where we define the Hamiltonian H by

$$\begin{aligned} H(e,\ \Delta u,\ \lambda, t) &= e^T(t)Qe(t) + \Delta u^T(t)R\Delta u(t) \\ &+ \lambda^T(t)\ (A(t)e(t) + B(t)\Delta u(t)) \end{aligned} \tag{4.31}$$

From equation (4.30) we get

$$\Delta u(t) = -R^{-1}B^T(t)\lambda(t) \tag{4.32}$$

and from equations (4.27), (4.30), and substituting from equation (4.32) we have

$$\begin{bmatrix} \dot{e} \\ \dot{\lambda} \end{bmatrix} = \begin{bmatrix} A & -BR^{-1}B^T \\ -Q & -A^T \end{bmatrix} \begin{bmatrix} e \\ \lambda \end{bmatrix} \tag{4.33}$$

with the initial and final conditions

$$e(t_0) = e_0 \tag{4.34}$$

$$\lambda(t_f) = Fe(t_f) \tag{4.35}$$

respectively.

Equation (4.33) is the two-point boundary value problem which can be solved by assuming

$$\lambda(t) = P(t)e(t) \tag{4.36}$$

where P is an (6×6) positive definite symmetric matrix. By differentiating the last equation with respect to time t, we get

$$\dot{\lambda} = \dot{P}e + P\dot{e} \tag{4.37}$$

Substituting from equation (4.33) we have

$$(\dot{P} + PA + A^T P + Q - PBR^{-1}B^T P)e = 0 \tag{4.38}$$

The last equation must hold for non-zero e, thus the term inside the round brackets must be zero, and so

$$\begin{aligned} \dot{P} &= -P(t)A - A^T P(t) - Q + P(t)BR^{-1}B^T P(t) \\ P(t_f) &= F \end{aligned} \tag{4.39}$$

which is known as the "Riccati equation", and it defines a system of (6×6) first order differential equations. However, since P is symmetric, only $6(6+1)/2$ first order equations need to be solved. The optimal control is then given by

$$\Delta u(t) = -R^{-1}B^T P(t)e(t) \tag{4.40}$$

For time-varying systems, the above problem can be solved to provide optimal control for the system such that the states will be driven to and maintained at desired values. This can be achieved using the following algorithmic procedure:

Algorithm 1:

Step 0 : Initialise $u(t_0)$, set $i = 0$, t_0, t_f, h, and the desired state X_d.

Step 1 : Linearize the system equations about the current state and control $(X(t_i),\ u(t_i))$.

Step 2 : Integrate the Riccati equation (4.39) with the set $(A,\ B)$, provided from Step 1, from t_f to t_i and store the gain matrix$P(t_i)$.

Step 3 : Calculate the optimal control $u(t_{i+1})$ for the next step as follows:

$$e(t_i) = X(t_i) - X_d \tag{4.41}$$

$$\Delta u = -R^{-1}B^T P(t_i)e(t_i) \tag{4.42}$$

$$u(t_{i+1}) = u(t_i) + \Delta u \tag{4.43}$$

$$t_{i+1} = t_i + h \tag{4.44}$$

Step 4 : Calculate $X(t_{i+1})$ by integrating the nonlinear system equations from the current state and using the new control $u(t_{i+1})$.

Step 5 : Set $i = i + 1$; if $t_i = t_f$, set $x(t_0) = X(t_i)$, $u(t_0) = u(t_i)$, $i = 0$, and go to Step 1 (that is, to start new interval $(t_0,\ t_f)$); else go to Step 1.

Equation (4.43) reflects the fact that the state and control variables used in the linearized system equations are perturbations of the original state and control variables. The above algorithm suggests that at all moments in time (each sampling interval), the Riccati equation must be integrated from (t_f) to the current time (t_i) using the current linearized system matrices $(A,\ B)$ to give the current gain matrix $P(t_i)$ which is to be used to provide the control at that moment. The system may need several intervals $(t_f - t_0)$ to drive the state error to zero; this will depend upon the size of the error and the length of the optimisation interval. In practice, it is difficult to perform the above calculations within a sampling period of 5 milliseconds as we will illustrate in the next section.

4.5 The Computational Complexities

We consider now equation (4.39), for the aircraft longitudinal motion, which gives rise to a Riccati equation of dimension

$$\bar{n} = \frac{n(n+1)}{2} = 21 \tag{4.45}$$

since $n = 6$.

The following equations need to be solved in order to evaluate the derivative function and to calculate the control deviation Δu.

$$\begin{aligned} s_{ij} &= \sum_{k=0}^{1} \frac{b_{ik} b_{jk}}{r_{kk}}, && i \leq j \\ s_{ij} &= s_{ji}, && i,j = 0,1,\ldots,5 \end{aligned} \tag{4.46}$$

$$\begin{aligned} \dot{p}_{ij} &= \sum_{l=0}^{5} p_{il} \sum_{k=0}^{5} s_{lk} p_{kj} - \sum_{k=0}^{5} a_{ki} p_{kj} - \sum_{k=0}^{5} a_{kj} p_{ik}, && i < j \\ \dot{p}_{ij} &= \sum_{l=0}^{5} p_{il} \sum_{k=0}^{5} s_{lk} p_{kj} - 2 \sum_{k=0}^{5} a_{kj} p_{kj} - q_{ii}, && i = j = 0,1,\ldots,5 \end{aligned} \tag{4.47}$$

$$RB_{ji} = \frac{b_{ij}}{r_{jj}}, \qquad i = 0,1,\ldots,5; \quad j = 0,1 \tag{4.48}$$

$$RBP_{ji} = \sum_{k=0}^{5} RB_{jk} p_{kj}, \qquad i = 0,1,\ldots,5; \quad j = 0,1 \tag{4.49}$$

$$\Delta u_j = -\sum_{k=0}^{5} RBP_{jk} e_k, \qquad i = 0,1,\ldots,5; \quad j = 0,1 \tag{4.50}$$

Equations (4.46) and (4.48) need to be solved once during the linearization process, while equation (4.47) needs to be solved for each function evaluation (derivation), and the equations (4.49) and (4.50) need to be solved once at the end of each integration process (at each sample).

Let n_a, n_s, n_d, n_m be the number of additions, subtractions, divisions, and multiplications respectively, then for one function evaluation (equation (4.47)) we have

$$\begin{aligned} n_a &= 45\bar{n} - 5n = 915 \\ n_s &= 2\bar{n} = 42 \\ n_m &= 54\bar{n} - 5n = 1104 \\ n_d &= 0 \end{aligned} \tag{4.51}$$

and for the calculation of Δu we have

$$\begin{aligned} n_a &= 60 \\ n_s &= 12 \\ n_m &= 84 \\ n_d &= 0 \end{aligned} \tag{4.52}$$

When we remove all the terms in (4.47) and (4.49) which are multiplied by zero elements of A and B, then for one function evaluation we have

$$\begin{aligned} n_a &= 205 \\ n_s &= 27 \\ n_m &= 269 \end{aligned} \tag{4.53}$$

and for the calculation of Δu

$$\begin{aligned} n_a &= 12 \\ n_s &= 12 \\ n_m &= 36 \end{aligned} \tag{4.54}$$

The total time required to integrate (4.39) from the final to the initial time and calculate Δu can be estimated by using the equation

$$T_t = T_l + N_s(T_f + T_{up}) + T_u + T_{com} \tag{4.55}$$

where T_l, T_f, T_u, T_{up}, T_{com} are the execution times of the linearization routine, function evaluation, Δu calculation, updating the P gains, and interprocessor communications respectively; and where N_s is the number of integration steps $((t_f - t_0)/h)$. Let t_a, t_s, t_m, t_d, t_{com} be the execution times of one floating point addition, subtraction, multiplication, division, and interprocessor transmission respectively; then by using the values given in Table 2.1 we can estimate the total execution time required to calculate the control using Algorithm 1

4.6 Optimal Control Problem Parallelisation

Algorithm 1 is implemented sequentially using the transputer based system of Figure 4.1(a), and in parallel using the transputer array system shown in Figure 4.1(b) and (c). Where the ellipses represent processes mapped onto the different processors (represented by the rectangles). Processor P_0 performs the aircraft simulation, and is connected to a personal computer (PC) which serves as a link between the user and the transputer network. The processors P_1 to P_4 are used to perform the numerical integration of (4.39) as well as supplying P_0 with the control input signals. Parallelism is achieved by using parallel numerical integration methods to reduce the computation time (see for example Miranker [34], and Franklin [19]).

Two methods of numerical integration are demonstrated here to realise real-time implementation of Algorithm 1.

Method 1:

The Euler integration method can be used; this method needs one function evaluation in each integration step. Parallelism is achieved by partitioning the equations (4.47), (4.49) and (4.50) between the processors.

For the one processor case (see Figure 4.1(a)), an optimisation interval $(t_f - t_0)$ of 0.5 second and a sampling period (the integration step, h) of 5 milliseconds were used and gave the following results

$$N_s = \frac{t_f - t_0}{h} = \frac{0.5}{5 \times 10^{-3}} = 100$$

$$T_f = 205t_a + 269t_m + 27t_s = 930.55\ \mu s$$

$$T_u = 12t_a + 36t_m + 12t_s = 112.2\ \mu s$$

$$T_{up} = \bar{n}(t_s + t_m) = 77.7\ \mu s$$

$$T_l = 3000\ \mu s$$

$$T_{com} = 0$$

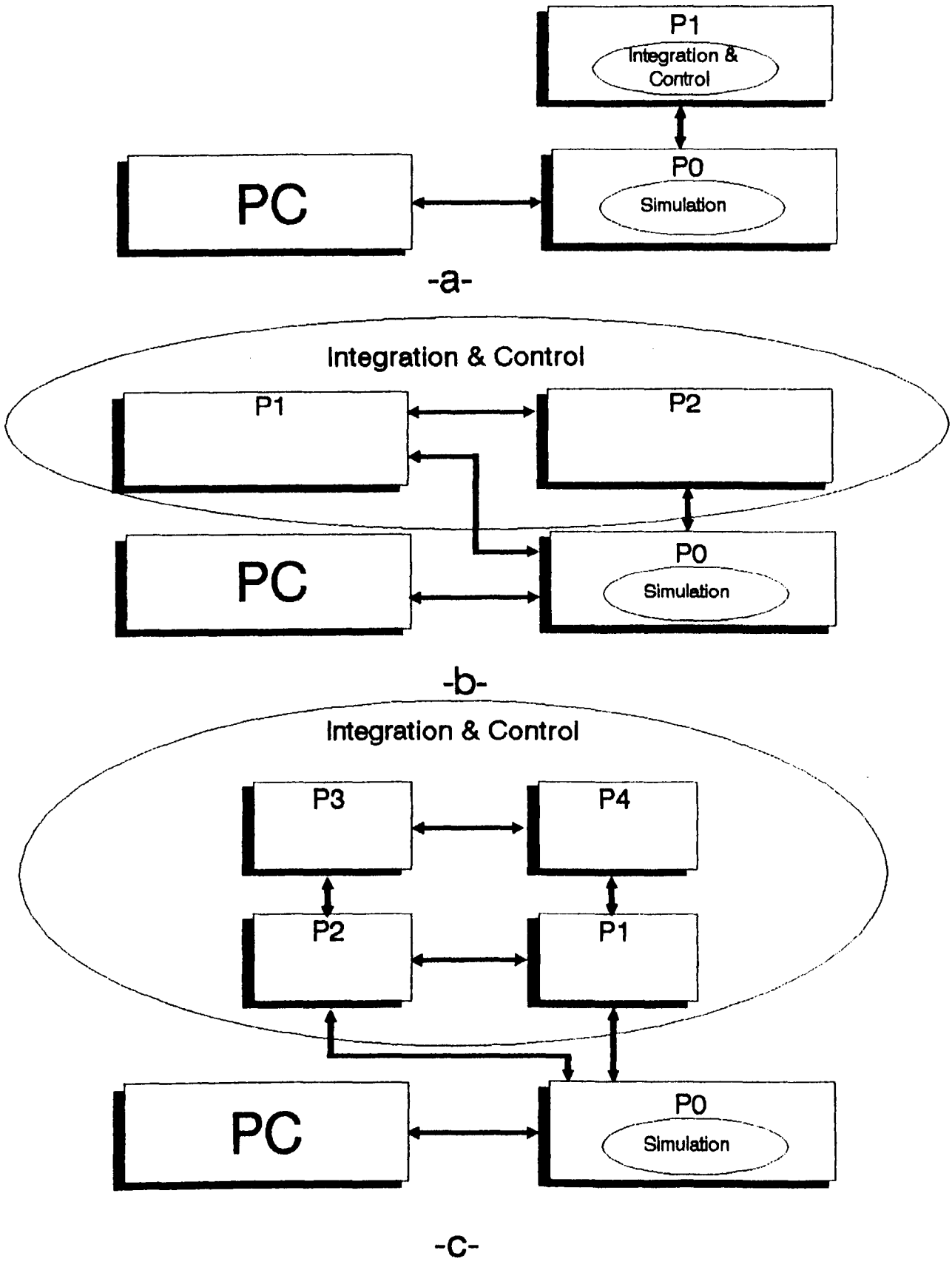

Figure 4.1: Transputer network for Algorithm 1

Hence using equation (4.55), the total time is $T_t = 0.103$ seconds.

Using the transputer network of Figure 4.1(b); and by partitioning the equations such that:

1. processor P_1 is used to update the gains

   $[p_{00}, p_{04}, p_{05}, p_{14}, p_{15}, p_{24}, p_{25}, p_{34}, p_{35}, p_{44}, p_{45}, p_{55}]$

   and calculate Δu_0;

2. processor P_2 is used to update the gains

   $[p_{01}, p_{02}, p_{03}, p_{11}, p_{12}, p_{13}, p_{22}, p_{23}, p_{33}]$

   and calculate Δu_1.

The maximum time required is as follows

$$T_f = 118t_a + 141t_m + 14t_s = 512.95\ \mu s$$

$$T_u = 12t_a + 6t_s + 24t_m = 46.8\ \mu s$$

$$T_{up} = 12(t_s + t_m) = 44.4\ \mu s$$

Each processor has a copy of the linearization routine and the two processors need 21 transmissions to exchange the gains P_{ij} which are needed to evaluate the derivative function. Therefore

$$T_{com} = N_s(\bar{n}t_{com}) = 23100\ \mu s \tag{4.56}$$

Hence the total time is $T_t = 0.082$ seconds. By further partitioning and using the transputer network of Figure 4.1(c), such that

1. P_1 is used to update the gains $[p_{01}, p_{02}, p_{11}, p_{12}, p_{22}]$ and calculates Δu_0,

2. P_2 is used to update the gains $[p_{03}, p_{13}, p_{23}, p_{33}]$ and calculates Δu_1,

3. P_3 is used to update the gains $[p_{00}, p_{04}, p_{14}, p_{24}, p_{34}, p_{44}]$,

4. P_4 is used to update the gains [p_{05}, p_{15}, p_{35}, p_{45}, p_{55}].

Each processor has a copy of the linearization routine and the four processors need 32 transmissions to exchange the gains in each integration step. Therefore the maximum time required to perform the different operations is

$$\begin{aligned} T_f &= 56t_a + 73t_m + 7t_s = 252.5\ \mu s \\ T_u &= 12t_a + 6t_s + 24t_m = 46.8\ \mu s \\ T_{up} &= 6(t_s + t_m) = 22.2\ \mu s \\ T_{com} &= N_s(32t_{com}) = 35200\ \mu s \end{aligned}$$

And therefore the total time is $T_t = 0.067$ seconds.

Method 2:

The parallel predictor-corrector integration method is used here which uses the following formulas

$$Y^p_{i+1} = Y^c_{i-1} + \frac{h}{3}(8F^p_i - 5F^c_{i-1} + 4F^c_{i-2} - F^c_{i-3}) \tag{4.57}$$

$$Y^c_i = Y^c_{i-1} + \frac{h}{24}(9F^p_i + 19F^c_{i-1} - 5F^c_{i-2} + F^c_{i-3}) \tag{4.58}$$

where i indicates the step number; p, c indicate the predicted and corrected values respectively, and when used in conjunction with a function evaluation (F) indicate that the function is evaluated using the corresponding predicted or corrected values. Two function evaluations are required in each integration step. This method has better accuracy than the previous one and may be used with larger integration steps for a given error tolerance. The execution time can be estimated by using the equation

$$T_t = T_l + 2N_s(T_f + T_{up}) + T_u + T_{com} \tag{4.59}$$

The values of T_f, T_l, T_u are as given in Method 1, while the updating time becomes

$$T_{up} = \bar{n}(2t_a + 2t_s + 4t_m + t_d) \tag{4.60}$$

Therefore in one processor we have

$$\begin{aligned} T_{up} &= 358.05\ \mu s \\ T_{com} &= 0 \\ T_t &= 0.260\ s \end{aligned}$$

In the two processor case, the time for function evaluations and updating will be halved since

1. the predictor equation and its function evaluation is solved by the predictor processor P_2, and
2. the corrector equation and its function evaluation is solved by the corrector processor P_1.

The two processors need to communicate $3\bar{n}$ times in each step as all the Y^c and F^c values must be passed from the corrector to the predictor processor and F^p values from the predictor to the corrector processor. Therefore

$$\begin{aligned} T_{com} &= N_s(3\bar{n}t_{com}) = 69300\ \mu s \\ T_t &= T_l + N_s(T_f + T_{up}) + T_u + T_{com} = 0.201\ s \end{aligned}$$

If we use four processors such that P_1, P_2 work as the corrector group, P_3, P_4 as the predictor group; and the equations are partitioned in each group as shown in the two processor case of Method 1, then we will have

$$\begin{aligned} \text{Inter} - \text{group communications} &= 3 \times 12 = 36 \\ \text{Inter} - \text{processor communications} &= \bar{n} = 21 \end{aligned}$$

Therefore

$$
\begin{aligned}
T_{com} &= N_s(56t_{com}) = 61600\ \mu s \\
T_f &= 512.95\ \mu s \\
T_{up} &= 12(2t_a + 2t_s + 4t_m + t_d) = 204.6\ \mu s \\
T_u &= 46.8\ \mu s \\
T_t &= T_l + N_s(T_f + T_{up}) + T_u + T_{com} = 0.136\ s
\end{aligned}
$$

Table 4.1: Execution times for algorithm 1

Number of Transputers	Execution time (sec.)			
	Method 1		Method 2	
	Estimated	Actual	Estimated	Actual
1	0.103	0.124	0.260	0.290
2	0.082	0.097	0.201	0.242
4	0.067	0.079	0.136	0.182

The estimated and actual execution time for the different cases considered are as shown in Table 4.1. It can be seen that the time required for inter-processor communications is likely to dominate the total time when the number of processors is increased. Although the optimisation interval used is short (0.5 second), the results obtained show that the execution time is far beyond the goal of 5 milliseconds and no significant improvement can be obtained by further partitioning of the system equations. Therefore Algorithm 1 is not suitable for real-time control application. Real-time solution is possible if the algorithm can be modified to suit the time constraints. In the next section we illustrate how this is possible.

4.7 A Real-time Algorithm

Algorithm 1 is a good approximation for handling the nonlinear time-varying aircraft problem, but as we have seen, it is not suitable for real-time implementation in practical systems since the execution time can not be reduced as required (5 milliseconds). The modification proposed here is that, instead of linearizing the system equations and calculating the time varying gains for each sampling interval, the aircraft equations are linearized and the gain matrix ($G = -R^{-1}B^TP(t)$, $t \in [t_1,\ t_2]$) calculated and held constant for several samples (between t_1 and t_2). During this time the aircraft model is assumed to be time-invariant, when time t_2 is reached the aircraft is relinearized at the new state, etc. The precise details of the algorithm are given next.

Algorithm 2:

Step 0 : Initialise $u(t_0)$, t_0, t_f, h, and the desired state X_d; set $i = 0$.

Step 1 : linearize the system equations about the current state and control ($X(t_i)$, $u(t_i)$); set $u_0 = u(t_i)$.

Step 2 : Integrate the Riccati equation from t_f to the beginning of the next subinterval $\bar{t}$ (within which the system is assumed to be time-invariant); store the gains ($G(t) = -R^{-1}B^TP(t)$, $t \in [\bar{t}, t_f]$).

Step 3 : Calculate the optimal control $u(t_{i+1})$ for the next sampling step as follows

$$\begin{aligned} e(t_i) &= X(t_i) - X_d \\ \Delta u &= -R^{-1}B^TP(t_i)e(t_i) \\ u(t_{i+1}) &= u_0 + \Delta u \\ t_{i+1} &= t_i + h \end{aligned}$$

Step 4 : Calculate $X(t_{i+1})$ by integrating the nonlinear system equations from $x(t_i)$ and using $u(t_{i+1})$.

Step 5 : Set $i = i + 1$; if $t_i = t_f$, set $X(t_0) = X(t_i)$, $u(t_0) = u(t_i)$, $i = 0$, and go to Step 6; else go to Step 6.

Step 6 : Is it time for the next linearization? If yes, go to Step 1; If not go to Step 3.

Algorithm 2 suggests that the optimisation horizon interval should be divided into subintervals of length equal to the expected execution time required to prepare the gains for the next subinterval. Within these intervals the system is assumed to be time-invariant. It is necessary to calculate and store the whole sequence of gains $G(t)$ for $t \in [\bar{t},\ t_f]$, and because of this we are unable to perform the gain calculations in a processor and send it to a controller processor which works in real-time to provide the control signals. This is because the communication time required to send the time sequence of the (2×6) gain matrix $G(t)$ is likely to cause undesirable delay in the controller and the integrator processor. To overcome this problem, we let the controller processor and the integrator processor switch from the control to integration job and vice versa. That is, the processor which calculates the gains for the next subinterval is switched to the control task in that interval as shown in the time table of Figure 4.2. The algorithm is implemented using the transputer network of Figure 4.3(a), where P_0 is the aircraft simulator and also controls the switching procedure. P_1 and P_2 perform one of the following sub-tasks at a given subinterval

1. To calculate Δu using the stored gains, transmit the gains to P_0, and read the current state at each sample. This will be referred to as the control task.

Processor	Time (seconds)					
	0.0	0.108	0.192	0.27	0.33	0.37 0.5
P1	Control Task	Integration Task (0.5 - 0.192)	Control Task	Integration Task (0.5 - 0.33)	Control Task	Integration Task (0.5 - 0.0)
P2	Integration Task (0.5 - 0.108)	Control Task	Integration Task (0.5 - 0.27)	Control Task	Integration Task (0.5 - 0.37)	Control Task

Figure 4.2: Time table for Algorithm 2

2. Linearize the aircraft equations about the operating point at the switching instant, integrate the Riccati equation backwards from t_f to the next switching time, and store the gains. This will be referred to as the integration task.

The interval $(t_f - t_0)$ is divided according to the actual execution time required for the integration task as given in Table 4.1 for Method 1. The time required for updating the gains can be reduced by increasing the number of subintervals, that is, by reducing the execution time of the integration task. This can be achieved by adding extra processors as shown in Figure 4.3(b). The extra processor P_3 will combine with the processor performing the integration task to give an array of processors as we have seen in the two processors case in the previous section. The time diagram for the later case is shown in Figure 4.4.

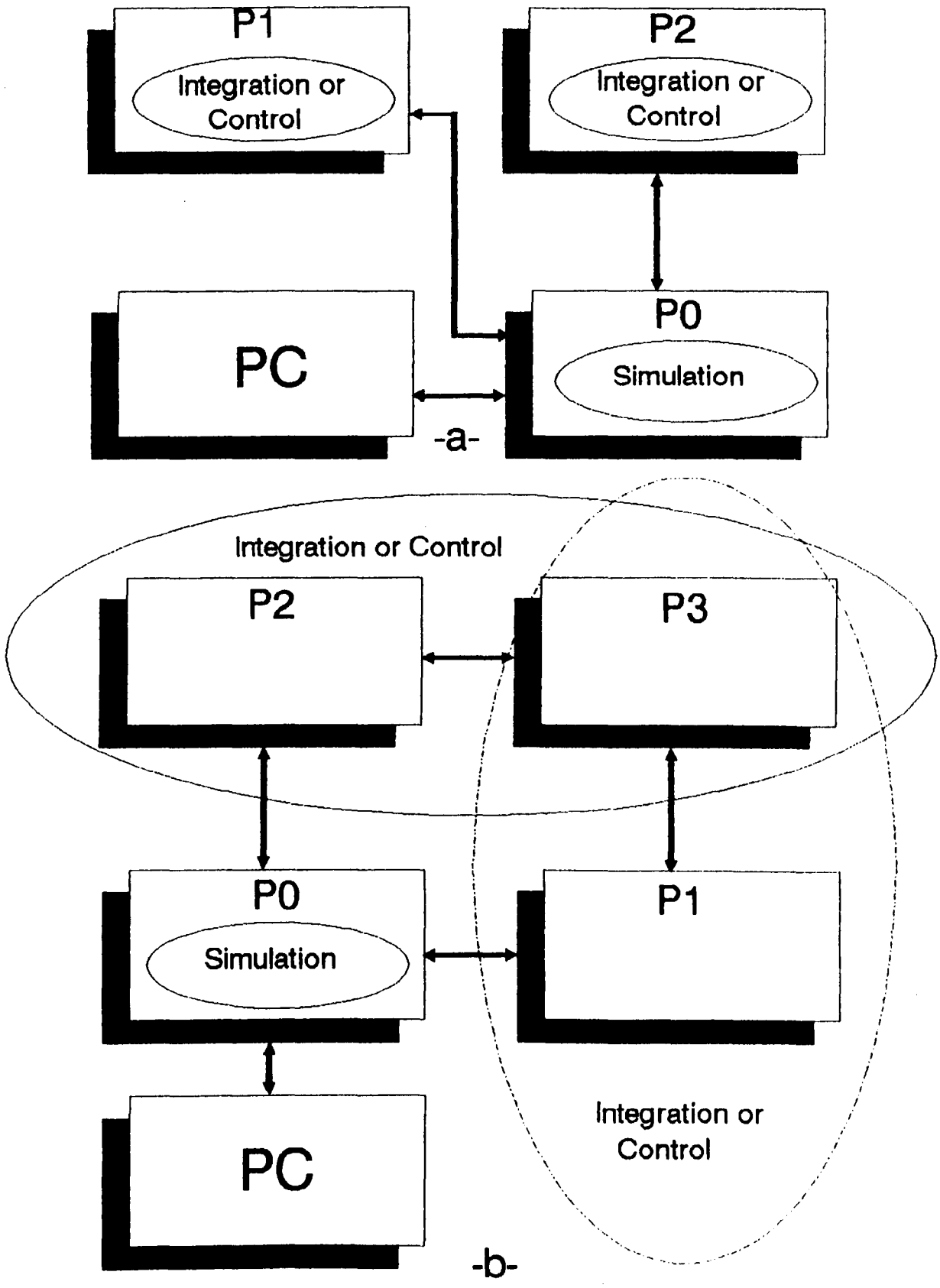

Figure 4.3: Transputer network for Algorithm 2

Processor	Time(seconds)							
	0.0	0.086	0.16	0.225	0.285	0.33	0.37	0.4 0.5
P1	Control Task	Integration Task with P3 (0.5 - 0.16)	Control Task	Integration Task with P3 (0.5 - 0.285)	Control Task	Integration Task with P3 (0.5 - 0.37)	Control Task	Integration Task with P3 (0.5 - 0.0)
P2	Integration Task with P3 (0.5 - 0.086)	Control Task	Integration Task with P3 (0.5 - 0.225)	Control Task	Integration Task with P3 (0.5 - 0.33)	Control Task	Integration Task with P3 (0.5 - 0.4)	Control Task

Figure 4.4: Time table for Algorithm 2 with extra processors

4.8 Simulation Results

Algorithm 2 can be implemented in real-time using the following weighting matrices which are chosen to satisfy the states and control constraints

$$Q = \text{diag}[0.15,\ 0,\ 100,\ 1000,\ 2,\ 5 \times 10^{-6}]$$

$$F = 0$$

$$R = \text{diag}[10000,\ 1000]$$

The method for the selection of these weights is discussed in chapter 6. The initial aircraft state is assumed to be [145, 0, 0 , 0, 4900, 6600], the desired state is [150, 5, 0, 0.033, 5000, 6615]. The optimisation interval is 0.5 second, and the sampling period is 5 milliseconds. The optimal control results are shown in Figure 4.5 which show good performance. The algorithm presented therefore offers a viable solution for the real-time optimal control problem, and gives a good approximation to the nonlinear aircraft problem.

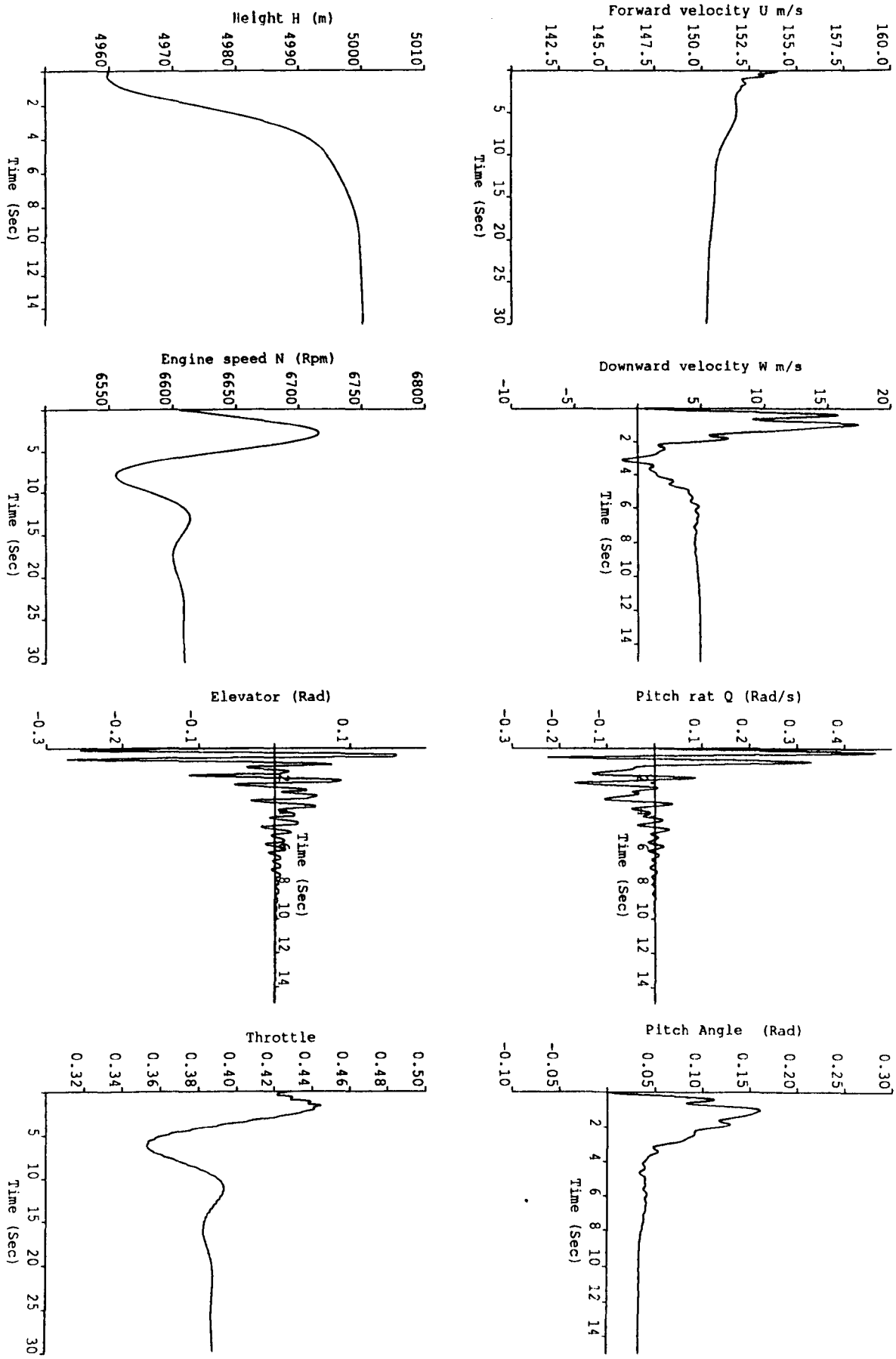

Figure 4.5: Longitudinal optimal responses

Chapter 5

Real-time Algorithm Via the Receding Horizon Method

5.1 Introduction

In this chapter we will introduce an alternative method of obtaining a suitable optimising control strategy for our time-varying system, namely, the receding/moving horizon method. The reasons for introducing it here are due to its lower requirements in computing resources as discused below. For these reasons it is the technique adapted for the majority of the control designs presented in this thesis. In fact for linear time-invariant systems, there exist several approaches to enable the design of optimal control laws. One of these which guarantied robust stabilising feedback control is the steady-state feedback law where the optimal stabilising state-feedback gains are found by solving (only once) an algebraic Riccati equation (see Richardson [47], Illinois [26]). The corresponding solution for the linear time-varying system problem requires the backward integration of a matrix Riccati equation over an infinite horizon or the solution of a difficult and time consuming algebraic Riccati equation at all moments in time. Clearly, this is not a very practical way of ob-

taining the control law. A natural way of attempting to reduce the computational difficulties is to assume that at all moments t, we need to calculate the optimal control for a fixed, finite horizon of length T (see Shaw [50], Kwon [29]).

The standard regulator problem addresses the problem of determining the optimal control $u^0(.)$ to be applied to a linear system in order to minimise a cost function over a given interval $(t_i,\ t_f)$. In the case of a quadratic performance index, the resulting optimal control is a straight forward state-feedback law; the gain computation involves the solution of the Riccati equation backwards in time. The control applied at time t, using a moving horizon of length T, would therefore be the initial step in minimising a quadratic performance index over $(t,\ t+T)$. The procedure leads to a state-feedback control law. The deference between this approach and the one presented in the last chapter is that "only the initial gain at time t is applied".

Although the receding horizon is reasonable in concept, it does not have any obvious interpretation in terms of optimising some standard form of performance index over some predetermined interval $(t_i,\ t_f)$; its value is in the fact the approach gives a practical and computationally realizable technique for calculating optimal control for general time-varying linear systems.

5.2 Receding Horizon Optimal Control

Consider the time-varying linear system described by

$$\dot{e}(t) = A(t)e(t) + B(t)\Delta u(t) \tag{5.1}$$

where $e(.) \in \Re^n$ and $\Delta u(.) \in \Re^m$. Let J be a standard quadratic cost function over a fixed interval $(t_i,\ t_f)$ defined as

$$J \;=\; e^T(t_f)Fe(t_f) + \int_{t_i}^{t_f} [e^T(\tau)Qe(\tau) + \Delta u^T(\tau)R\Delta u(\tau)]d\tau \tag{5.2}$$

Here Q, R, F are design parameters or weighting matrices. As we have seen in chapter 4, the optimal control input that minimises J is provided by the following state-feedback law

$$\Delta u^*(t) = -R^{-1}B^T(t)P(t,\ t_f; F)e(t) \tag{5.3}$$

and the gain $P(t,\ t_f;\ F)$ is computed using the Riccati equation

$$\begin{aligned}\frac{d}{d\tau}P(\tau,\ t_f;\ F) &= -P(\tau,\ t_f;\ F)A(\tau) - A(\tau)P(\tau,\ t_f;\ F) + \\ &\quad P(\tau,\ t_f;\ F)B(\tau)R^{-1}B^T(\tau)P(\tau,\ t_f;\ F) - Q\end{aligned} \tag{5.4}$$

backwards in time using the final condition $P(t_f,\ t_f;\ F) = F$. This can be modified to give rise to the receding horizon optimal control problem; the receding horizon optimal control $\Delta u^m(t)$ is defined as the input at time t that would be needed to minimise over $(t,\ t+T)$ the following criterion

$$J_m = e^T(t_f)Fe(t_f) + \int_t^{t+T}[e^T(\tau)Qe(\tau) + \Delta u^T(\tau)R\Delta u(\tau)]d\tau \tag{5.5}$$

This also results in a control law which is a state-feedback law, given by

$$u^m(t) = -R^{-1}B^T(t)P(t,\ t+T;\ F)e(t) \tag{5.6}$$

The interpretation of this control is that, at each moment t, the input applied is chosen as if optimisation of the criterion J_m over $(t,\ t+T)$ was the overall objective. In order to compute $P(t,\ t+T;\ F)$ one must solve at all moments in time the Riccati equation backwards from final conditions at $t+T$ given by F. As we have seen from the complexity of Algorithm 1 presented in chapter 4, this is not computationally feasible. However it is straight forward to realize that in the case of time-invariant systems, the receding horizon method yields a "constant" feedback gain given by

$$u^m(t) = -R^{-1}B^T P_T e(t) \tag{5.7}$$

where P_T can be obtained from

$$\begin{aligned} \frac{d}{dt}P &= -PA - A^TP + PBR^{-1}B^TP - Q; \\ P(t+T,\ t+T;\ F) &= F \end{aligned} \tag{5.8}$$

The last equations makes use of the same assumption used in Algorithm 2, that is, the system is assumed to be time-invariant for short intervals $T_{lin} < T$ to overcome the excessive computational requirements.

The feedback control law must provide asymptotic stability for the closed-loop system. Therefore, there must be a finite horizon T which will ensure stability for the system. For the particular case where $F = 0,\ Q > 0,\ R > 0$, it is well known that: (see Kwon [29])

"If the pair $(A,\ B)$ is controllable, then there exists a finite horizon T such that the moving (receding) horizon control law stabilises the system".

For large enough T, the value $P(t,\ t+T;\ F)$ approaches the steady state law $P(t,\ \infty;\ F)$. Therefore, the actual control used in practise differs only slightly from the steady-state feedback law $(P(t,\ \infty;\ F))$, which is known to stabilise the time-varying systems. Kwon [29] and Shaw [50] have presented a method for choosing the horizon length T for the case of $F = \infty$, and Thomas [52] suggests that T should be made roughly equal to the rise-time of the system.

5.3 Real-time Receding Horizon Algorithm

The receding horizon method can be implemented using a procedure similar to Algorithm 1 where to reduce the computational requirements, the aircraft equations are assumed to be linear and time-invariant for short intervals $T_{lin} < T$; the algorithm presented below may be used

Algorithm 3

Step 0 : Initialise U_c, t, T, T_{lin}, h, and X_d.

Step 1 : linearize the system equations about the current state $X(t)$ and the control U_c (the centre position for the control); set $\bar{t} = t$.

Step 2 : Integrate the Riccati equation from $t + T$ to t and store the gain matrix $G = -R^{-1}B^T P_T$.

Step 3 : Calculate the control for the next step as follows:

$$e(t) = X(t) - X_d \tag{5.9}$$

$$\Delta u = Ge(t) \tag{5.10}$$

$$u(t+h) = U_c + \Delta u \tag{5.11}$$

$$t = t + h \tag{5.12}$$

Step 4 : Apply the control $u(t)$ to the system and obtain the new state $X(t)$.

Step 5 : If $t \geq \bar{t} + T_{lin}$, go to Step 1; else, go to Step 3.

It should be noted that the nonlinear system equations are linearized about the control U_c instead of the current control $u(t)$ to prevent the integral action in the calculated control ($u(t+h) = u(t) + \Delta u$) which causes undesired response such as control saturation and excessive switching. That is, we calculate at each step the actual amount of control and not the change required to the current control as in Algorithm 2. Algorithm 3 can be implemented in real-time using the transputer network shown in Figure 5.1, in which

1. Processor P_0 is the system simulator (it performs step 4 of the algorithm).

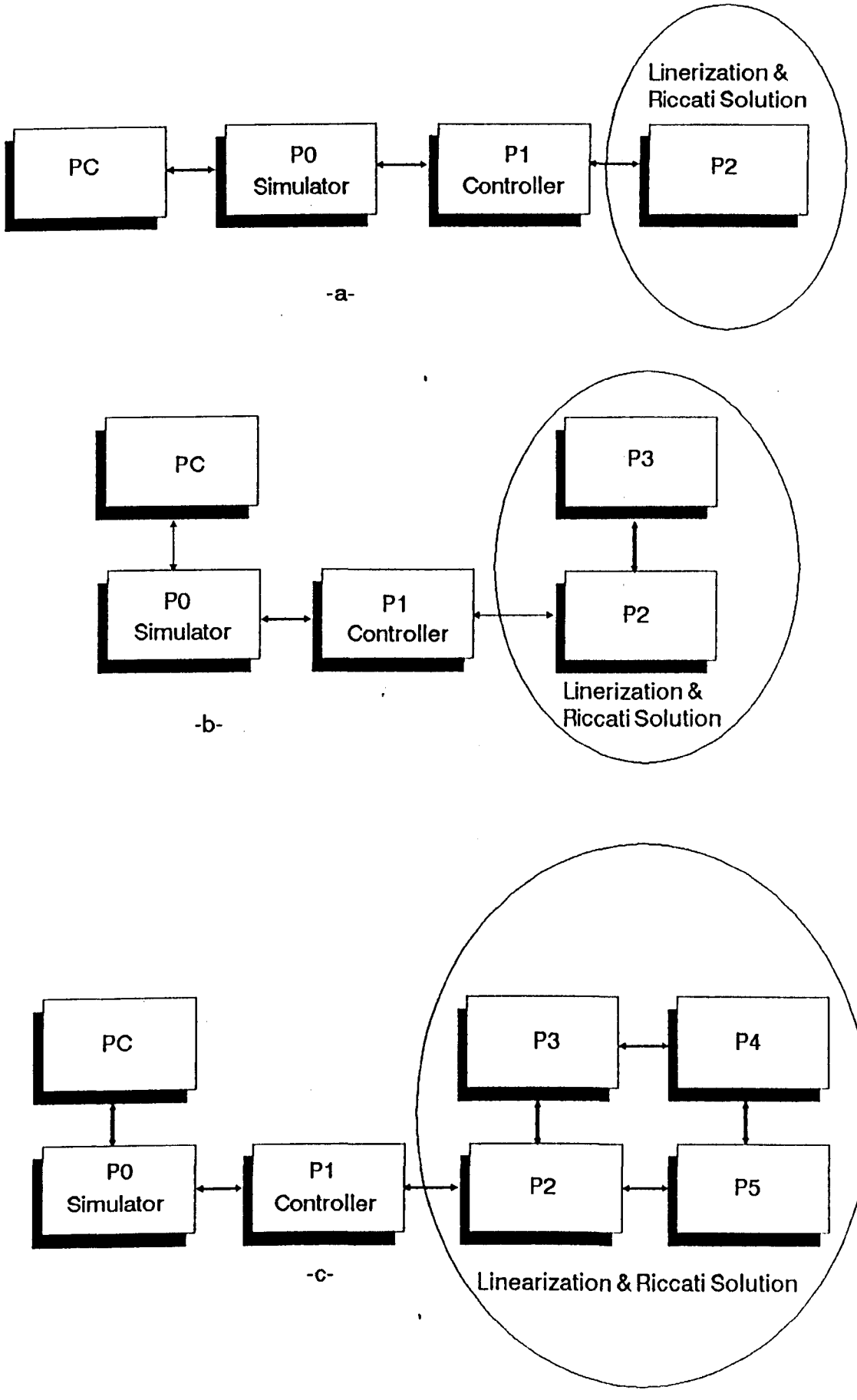

Figure 5.1: Transputer network for Algorithm 3

2. Processor P_1 acts as the controller providing the control updates every 5 milliseconds using the current value of the gain matrix G (it performs Step 3 of the algorithm).

3. The network of processors P_2 to P_5 are used to update the gain matrix G (to perform the steps 1 and 2 of the algorithm).

The next two sections discuss the computational aspects of Algorithm 3 when it is used to provide the control inputs to the aircraft. The sample period used is $h = 5$ ms, the horizon depth is $T = 5$ s, and the Riccati equation is integrated in steps of $h_s = 20$ ms.

5.4 The Longitudinal Motion

The aircraft longitudinal motion is modelled using the equations (4.2) to (4.7). Following the same procedure as in chapter 4, the computational timings can be estimated. The number of integration steps N_s will be

$$N_s = \frac{T}{h_s} = \frac{5}{20 \times 10^{-3}} = 250 \tag{5.13}$$

Using equation (4.55) we can estimate the execution time required to perform Steps 1 and 2 of Algorithm 3 (which is made equal to the length of the linearization interval T_{lin}); This is calculated to be the following:

One processor (Figure 5.1 a):

T_{lin} = 255.13 ms

Two processors (Figure 5.1 b):

T_{lin} = 200.12 ms

Four processors (Figure 5.1 c):

T_{lin} = 159.71 ms

The actual times found in real implementations are as given in Table 5.1. From these results, it is clear that the feedback gains can be updated every 295, 212, and 169 milliseconds with respect to the above three cases respectively.

Processor P_1 needs to communicate with P_0, that is, it transmits the control and receives the system states. It also calculates the control and monitors the gain updating processor(s). The maximum execution time in P_1 is then

Communications with $P_0 = 8 \times t_{com} = 88\ \mu s$

Communications with $P_2 = 18 \times t_{com} = 198\ \mu s$

Control updating $= 12 \times t_m + 14 \times t_a = 47.9\ \mu s$

Total time $= 333.9\ \mu s$

The total time required by the controller processor (P_1) is very small when compared by the time constraints of 5 milliseconds which is the sample period. This leaves some extra spare capacity in $P1$ to perform other tasks as we will see in the following chapters.

5.5 The Lateral Motion

Now we consider the complete set of equations of motion for the whole aircraft given in chapter 3. If we use the following assumptions

$$W = \dot{W} = 0 \tag{5.14}$$

$$Q = \dot{Q} = 0 \tag{5.15}$$

$$U = U_0 \tag{5.16}$$

Then, the aircraft lateral motion can be described by the following set of nonlinear equations

$$\dot{V} = \frac{F_y}{m} - RU_0 \tag{5.17}$$

$$\dot{P} = \frac{R_m + I_{xz}\dot{R}}{I_x} \tag{5.18}$$

$$\dot{R} = \frac{Y_m + I_{xz}\dot{P}}{I_z} \tag{5.19}$$

$$\dot{\Phi} = P \tag{5.20}$$

$$\dot{\Psi} = R\cos\Phi \tag{5.21}$$

where the notations, forces and moments are again as that defined in chapter 3. The above nonlinear equations can be represented as

$$\dot{X} = f(X,\ u) \tag{5.22}$$

where $X = [V,\ P,\ R,\ \Phi,\ \Psi]^T$ is the state vector, and $u = [\xi,\ \zeta]^T$ is the control vector. The above equations can be linearized about an operating point $(X_o,\ U_c)$ to give

$$\dot{e} = Ae(t) + B\Delta u(t) \tag{5.23}$$

where

$$e(t) = X(t) - X_o \tag{5.24}$$

$$\Delta u(t) = u(t) - U_c \tag{5.25}$$

and the matrices A, B will be in the form

$$A(t) = \begin{bmatrix} F_{y_v} & 0 & -U_0 & g\cos\Phi & 0 \\ R_{mv} & R_{mp} & R_{mr} & 0 & 0 \\ Y_{mv} & Y_{mp} & Y_{mr} & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & \cos\Phi & -R\sin\Phi & 0 \end{bmatrix} \tag{5.26}$$

$$B(t) = \begin{bmatrix} 0 & F_{y_\zeta} \\ R_{m\xi} & R_{m\zeta} \\ Y_{m\xi} & Y_{m\zeta} \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \tag{5.27}$$

where $F_{y_v} = \frac{\delta F_y/\delta v}{m}$, ... etc, are obtained by differentiating the corresponding equations presented in chapter 3. Following the same procedure as in chapter 4, the computational timings of Algorithm 3 when used to provide the lateral control, are as follows

$$\bar{n} = \frac{n(n+1)}{2} = 15$$

One processor (Figure 5.1 a):

$$\begin{aligned} T_f &= 164t_a + 121t_m = 522.95\ \mu s \\ T_{up} &= \bar{n}(t_m + t_a) = 55.5\ \mu s \\ T_g &= 15t_a + 25t_m + 15t_d = 108.75\ \mu s \\ T_{com} &= 0 \\ T_l &= 2700\ \mu s \\ T_{lin} &= T_l + N_s(T_f + T_{up}) + T_g + T_{com} = 147.42\ \text{ms} \end{aligned}$$

where T_f, T_{up}, T_{com}, T_l are as defined in chapter 4, T_g is the estimated execution time to perform the gain equation $G = -R^{-1}B^T P_T$ at the end of each Riccati integration phase, and T_{lin} is the time required to update the lateral gains.

Two processors (Figure 5.1 b):

$$\begin{aligned} T_f &= 61t_a + 83t_m = 268.6\ \mu s \\ T_{up} &= 9(t_m + t_a) = 33.3\ \mu s \\ T_g &= 15t_a + 25t_m + 15t_d = 108.75\ \mu s \end{aligned}$$

$$T_{com} = N_s(\bar{n}t_{com}) = 41250\ \mu s$$

$$T_l = 2700\ \mu s$$

$$T_{lin} = T_l + N_s(T_f + T_{up}) + T_g + T_{com} = 119.53\ \text{ms}$$

Each processor has a copy of the linearization routine and the gain matrix G is calculated in processor P_2 because no significant improvement can be achieved in partitioning these operations due to communication cost. The Riccati equation is partitioned (as equally as possible) such that processor P_2 updates the gains

$[p_{03},\ p_{04},\ p_{13},\ p_{14},\ p_{23},\ p_{24},\ p_{33},\ p_{34},\ p_{44}]$

as well as calculating the gain matrix G; processor P_3 updates the gains

$[p_{00},\ p_{01},\ p_{02},\ p_{11},\ p_{12}]$.

Four processors (Figure 5.1 c):

The equations are partitioned such that, processor P_2 updates the gains

$[p_{03},\ p_{13},\ p_{23},\ p_{33}]$ and calculates the matrix G; P_3 updates

$[p_{04},\ p_{14},\ p_{24},\ p_{34},\ p_{44}]$; P_4 updates

$[p_{02},\ p_{12},\ p_{22}]$; and P_5 updates $[p_{00},\ p_{01},\ p_{11}]$.

The total execution time is obtained as follows

$$T_f = 34t_a + 45t_m = 147.25\ \mu s$$

$$T_{up} = 4(t_m + t_a) = 14.8\ \mu s$$

$$T_g = 15t_a + 25t_m + 15t_d = 108.75\ \mu s$$

$$T_{com} = N_s(23t_{com}) = 63250\ \mu s$$

$$T_l = 2700\ \mu s$$

$$T_{lin} = T_l + N_s(T_f + T_{up}) + T_g + T_{com} = 106.57\ \text{ms}$$

The actual execution times are shown in Table 5.1.

Table 5.1: Execution times for Algorithm 3

Network	Execution time T_{lin} (ms)			
	Longitudinal		Lateral	
	Estimated	Actual	Estimated	Actual
1 processor	255.13	295	147.42	190
2 processors	200.12	212	119.53	130
4 processors	159.71	169	106.57	113

5.6 Simulation Results

Algorithm 3 is implemented in real-time to process the longitudinal and lateral control laws using the transputer network shown in Figure 5.1. The following design parameters are used:

Horizon $T = 5$ s,

integration step $h_s = 20$ ms,

and sample period $h = 5$ ms.

The weighting matrices for the longitudinal controller are:

$Q_{long} = diag[1.3,\ 0.6,\ 200,\ 40,\ 0.004,\ 0.00002]$,

$R_{long} = diag[8,\ 2]$,

$F_{long} = 0$.

For the lateral controller the corresponding matrices are:

$Q_{lat} = diag[0.008,\ 1,\ 20,\ 30,\ 200]$,

$R_{lat} = diag[20,\ 20]$,

$F_{lat} = 0$.

For the longitudinal motion, the initial and the desired states are:

$X_0 = [150,\ 0,\ 0,\ 0,\ 4900,\ 6605]$

$X_d = [150,\ 5,\ 0,\ 0.033,\ 5000,\ 6605]$

and for the lateral:

$X_0 = [0,\ 0,\ 0,\ 0,\ 0.2]$

$X_d = [0,\ 0,\ 0,\ 0,\ 0]$

The "sub-optimal" controlled trajectories are shown in Figure 5.2 for the longitudinal and Figure 5.3 for the lateral motion. We can see from these trajectories, that the real-time control performances are adequate. The setting of the weighting factors ($Q,\ R$) determines this performance, and so the selection of the ($Q,\ R$) matrices is quite critical. We will discuss this in the next chapter.

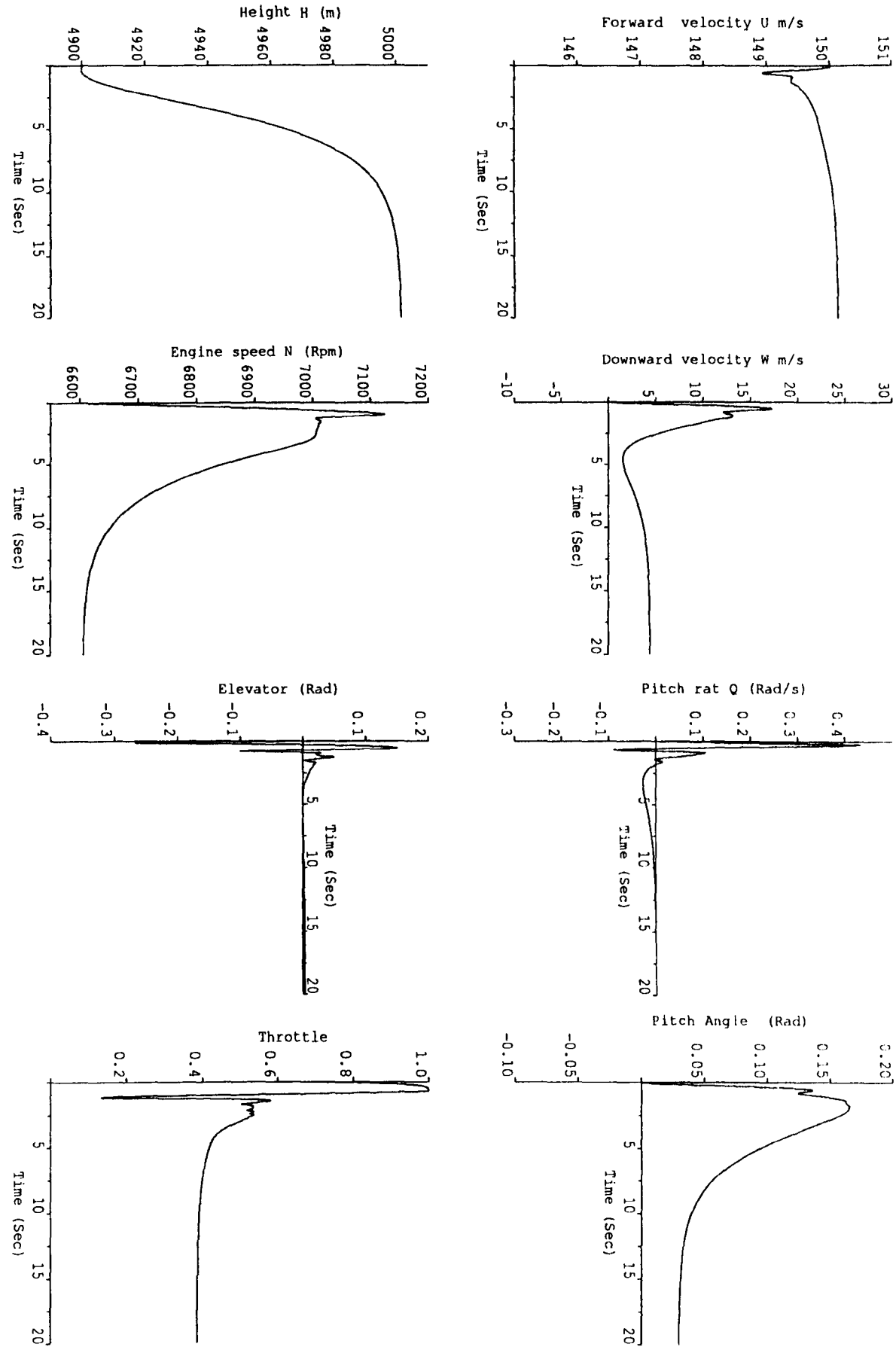

Figure 5.2: Longitudinal responses

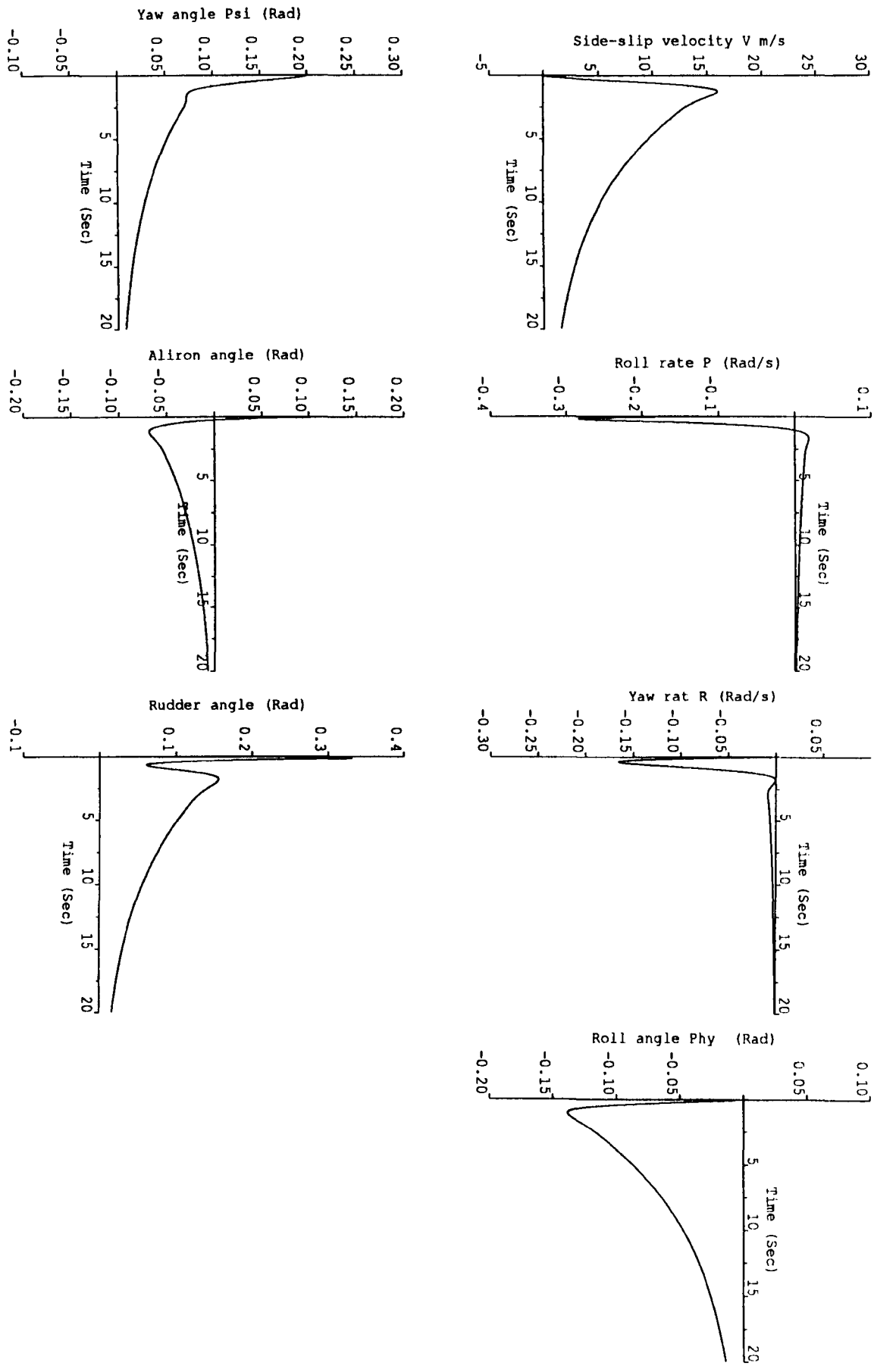

Figure 5.3: Lateral responses

Chapter 6

Selection of Weights in Time-varying Optimal Flight Control

6.1 Introduction

In the previous chapters, we have discused the design of optimal controllers for nonlinear time-varying aircraft. The critical factor in such designs is the choice of the performance index, and in particular, the choice of the weighting elements to achieve the desired responses. In this chapter, we present a design procedure for the selection of these weights. This is quite a complicated task since the performance assessment is usually very subjective with possibly many points to take into consideration such as percentage overshoot, steady-state errors, settling times and rise-times.

The commonest cost function in current use is a quadratic function taking the form

$$\min_{u} \int_0^T \left[x^T(t)\,Qx(t) + u^T(t)\,Ru(t)\right] dt \tag{6.1}$$

where Q is an $n \times n$ symmetric positive semi-definite matrix, R is an $m \times m$ symmetric positive definite matrix for a system of state $x(t)$, with dimension n, and m inputs $u = [u_1, \ldots, u_m]^T$.

In problems of this kind it is well known that the $x^T Q x$ term relates to system error and $u^T R u$ relates to control effort. By varying Q and R, the emphasis given to the individual terms can be varied for different objectives. For example as Q is increased, tighter regulatory performance is sought, which usually requires larger controlling signals. Similarly as R is increased, control effort is penalised more, and so large control signals are avoided, but this generally leads to worsening of the errors. It is therefore clear that a proper balance between Q and R needs to be reached to ensure that good error regulation is obtained but without using excessive control.

Now although the theory of optimal control is well documented (see for example Banks [4]; Bryson and Ho [10]), and leads to the determination of optimal solutions for general problems, the proper formulation of such problems is not. The Q and R matrices in the objective function (6.1) are usually found by following essentially an educated trial and error approach until an adequate response is obtained. Such a procedure for selecting the weights can get very complicated when multivariable systems are being considered, and where there is significant cross-coupling between the various subsystems. Moreover, for time-varying systems, a particular set of Q and R matrices which give a satisfactory response at a given operating condition may not yield good behaviour at another operating point; hence the Q and/or R weights may need to be made time-varying and dependant upon operating conditions.

These difficulties, together with the significant computational demands for on-line control, and the level of mathematical analysis required, have restricted the widespread adoption of optimal control techniques in practical applications, and it

is felt that better design methodologies are required. In this chapter a practical systematic approach is suggested which enables the Q and R matrices to be determined. We will concentrate on the optimal autopilot design problem to illustrate the approach but the method is equally applicable to any multivariable problem. As we have seen in chapter 5, a receding/moving horizon control problem has been shown to be suitable for handling the time-varying aircraft equations in real-time. Here we have the system described by

$$\begin{aligned} \dot{e}(t) &= A(t)e(t) + B(t)\Delta u(t) \qquad \text{for } t \in [t_0, t_f] \\ e(t_0) &= e_0 \end{aligned} \tag{6.2}$$

The quadratic cost function J is to be minimised over a fixed interval $[t_0, t_0 + T]$ for some horizon length of T seconds, and

$$J = \frac{1}{2}\int_{t_0}^{t_0+T} \left[e^T(t)Qe(t) + \Delta u^T(t)R\Delta u(t)\right] dt + \frac{1}{2}e^T(t_0+T)Fe(t_0+T) \tag{6.3}$$

Here Q, R, and F are symmetric matrices which contain the designed weighting parameters. We will assume that Q and F are $n \times n$ positive semi-definite matrices and R is an $m \times m$ positive definite matrix. This is the standard regulator problem where the Q, R, and F matrices need to be selected so that the minimising control will drive the initial error e_0 towards the origin in an optimal manner. As already discussed the optimal control input that minimises equation (6.3) is provided by the following state-feedback law

$$\Delta u^o(t) = -R^{-1}B^T P(t_0)e(t) \qquad \text{for } t \in [t_0, t_0 + T] \tag{6.4}$$

where $P(t_0)$ is an $n \times n$ matrix-valued function calculated from solving a Riccati equation, and $T_{lin} < T$ is the time duration over which time-invariance is assumed for the aircraft.

6.2 Selection of Q and R

It is clear that to achieve the desired control performance, the " correct" weighting matrices in the objective function have to be determined. There are no rigid guidelines which enable such a design, and usually some ad hoc approach is used. In this section we present a practical procedure which enables the selection of the Q and R matrices so that the overall control performance is as required. From equation (6.4) we have

$$\Delta u^{o}(t) = G(t_0)\, e(t) \tag{6.5}$$

where $G(t_0)$ is an $m \times n$ matrix given by

$$G(t_0) = -R^{-1}B^{T}P(t_0) \qquad \text{for } t \in [t_0, t_0 + T_{lin}] \tag{6.6}$$

at the start of each linearizing interval. The components of the control input are therefore calculated by

$$\Delta u_j^{o}(t) = g_{j1}e_1(t) + g_{j2}e_2(t) + \ldots + g_{jn}e_n(t) \qquad \text{for } j = 1, 2, \ldots, m \tag{6.7}$$

and so each control input Δu_j^{o} has n gain elements of G associated with it. These gains are multiplied by the corresponding state errors to give individual contributions to the overall control input signals which need to be applied. As can be seen from the above equations, the gain elements of G are effected by the weighting matrices Q, R, and F. For convenience we will assume that $F = 0$ and concentrate on the selection of Q and R. The important points to bear in mind when Q and R are being designed include the following:

1. The Q and R matrices for the given horizon, must be found to achieve a proper balance between the terms in the Riccati equation so that an individual component term does not dominate the solution. Of course, in certain applications

this may be precisely what is required to simplify the design procedure or emphasise a particular design objective.

2. Each gain g_{ji} should therefore be of "reasonable" magnitude so that its authority (that is the term $g_{ji}e_i$) will not dominate the resulting control. Also the g_{ji} magnitudes must not lead to excessive control (actuator saturation). Hence, in the design procedure, all the weights must initially be roughly the same order, and then altered according to the required performance. The authority assignment to each gain is clearly system dependant and on the performance requirements sought.

3. When the above aspects have been considered and the elements of Q and R have been selected for a given operating point, it is advisable to study the system responses to assess the controller's performance. It may be possible to make further changes to fine-tune the weights to obtain the desired results.

4. In some applications the system under study is time-varying, and so the time-varying characteristics of the gains g_{ji} have to be established. Do these follow the time-varying characteristics of the system? If not the controller needs to be made time-varying so that it stays tuned to the system. To determine how this can be achieved it is necessary to establish which of the system states have the major time-varying influence on the system. The dominant state(s) can be found by analysing the time-varying properties of elements in the A and B matrices. For example in the aircraft dynamic equations considered in this work, the velocity has the major influence on the system dynamics; higher velocities means that the elements of B and some elements of A will increase. This effects the balance designed for in the optimal flight control. The designed values of Q and R (for the operating velocity) must therefore

be tested along the full working range of velocities. From our study of the aircraft system, we found it difficult to calculate fixed weights which covered the entire range of velocities adequately, and hence some elements need to be made time-varying. This time-varying nature has to be established so that Q and R can be adjusted, on-line, as the operating velocity varies with time.

In the next section we shall show how the above points can be used to yield a suitable procedure for designing Q and R. Although the procedure presented is with particular reference to fixed wing aircraft, it can be used as a guide-line for other general dynamical systems if some prior knowledge concerning the application is available.

6.3 Longitudinal Motion Controller

Here we design an optimal control law for the longitudinal motion of the aircraft. From equation (6.5) – (6.7) we have

$$\Delta\eta = g_{11}e_1 + g_{12}e_2 + \cdots + g_{16}e_6 \tag{6.8}$$

$$\Delta\gamma = g_{21}e_1 + g_{22}e_2 + \cdots + g_{26}e_6 \tag{6.9}$$

where the g_{ji}'s gains are of the form

$$g_{1i} = \frac{-1}{r_{11}}(F_{x\eta}p_{1i} + F_{z\eta}p_{2i} + P_{m\eta}p_{3i}) \qquad \text{for } i = 1,2,\ldots,6 \tag{6.10}$$

$$g_{2i} = \frac{-1}{r_{22}}N_{\gamma}p_{6i} \qquad \text{for } i = 1,2,\ldots,6 \tag{6.11}$$

Also $e_1, e_2, \ldots, e_6$ are the state errors, p_{ij} are the elements of the Riccati matrix $P(t_0)$, and r_{jj} are the elements of the diagonal matrix R.

Bearing in mind the aspects stated above, the following procedure can be undertaken to select the weighting matrices for our aircraft application:

1. Choose an operating condition, such as for example, the current state

   $X_0 = [150, 0, 0, 0, 4900, 6605]^T$, (say)

   and let the desired state be

   $X_d = [150, 0, 0, 0.033, 5000, 6605]^T$, (say).

2. Determine suitable working ranges for the states U, W, Q, Θ, H, N, and the controls η, and γ. Then select values within these ranges of possible errors for each state (say 5% of these values), and determine the average values for the control inputs. To give equal emphasis to all the error and control effort terms, the diagonal elements of the Q and R matrices can be selected as follows:

   The q_{ii} and r_{jj} elements are chosen inversely proportional to the corresponding errors and average inputs respectively. Considering the longitudinal motion, we have the data shown in the Tables 6.1 and 6.2. Hence the q_{ii} and r_{jj} elements can be chosen initially so that

   $$e_{i_{5\%}} q_{ii} = 1, \qquad \text{for } i = 1, 2, \ldots, 6$$

   and

   $$u_{j_{av}} r_{jj} = 1, \qquad \text{for } j = 1, 2$$

   Therefore the initial QR weighting matrices are:

   Q_0 =diag [0.13, 0.6, 40, 40, 0.004, 0.003],

   R_0 =diag [3.85, 2].

3. Using these initial Q and R in the optimal controller algorithm, it is possible to calculate the gain matrix G at the initial state values X_0. This G is found

Table 6.1: Initial q_{ii} weighting

i	State e_i	Typical value	5% value $e_{i_{5\%}}$	q_{ii}
1	U	150 m/s	7.5	0.13
2	W	30 m/s	1.5	0.6
3	Q	0.5 rad/s	0.025	40
4	Θ	0.5 rad	0.025	40
5	H	5000 m	250	0.004
6	N	6600 rpm	330	0.003

Table 6.2: Initial r_{jj} weighting

j	Control Δu_j	Range of Control input	Average $u_{j_{av}}$	r_{jj}
1	η	-0.36 to 0.16 rad	0.26 rad	3.85
2	γ	0 to 1	0.5	2

to equal

$$G = \begin{bmatrix} -0.24 & 0.27 & 4.06 & 14.7 & 0.025 & -0.000006 \\ -0.05 & -0.004 & 0.001 & 0.12 & -0.0002 & -0.014 \end{bmatrix} \tag{6.12}$$

The first row corresponds to the elevator control $\Delta\eta$ and the second to the engine throttle control $\Delta\gamma$.

4. These initial Q and R result in the responses shown in Figure 6.1 from which it is obvious that the responses are unsatisfactory since large oscillations are present in many of the responses. The performance index needs to be modified and better values for the QR matrices need to be selected. To ease the design

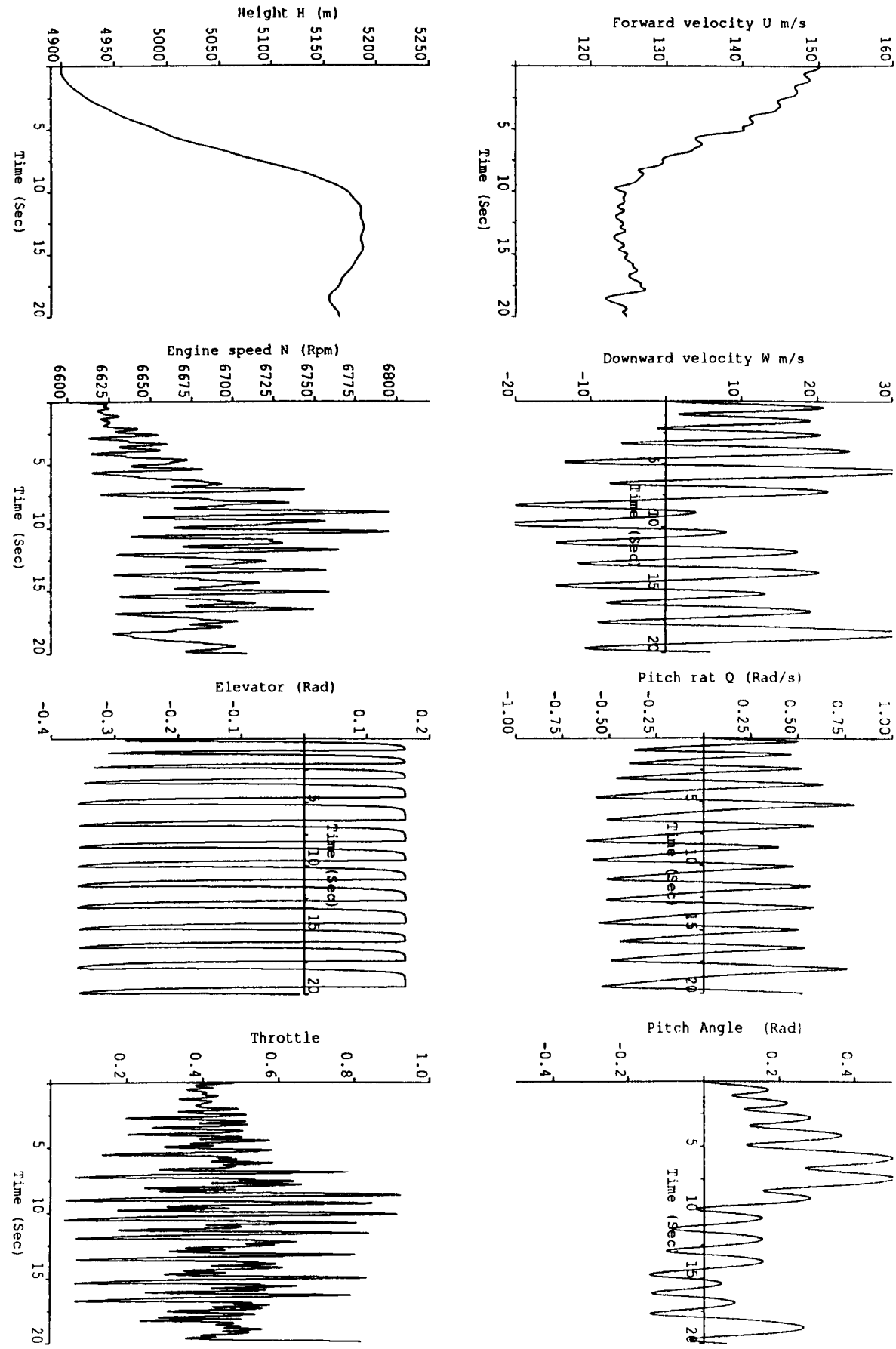

Figure 6.1: Longitudinal responses

problem and because of the interaction between the two control loops, it is possible to consider one loop at a time:

(a) The aircraft engine rotor speed (N) is seen to be responding poorly to the velocity error. The reason for this may be found by examining the gains g_{21} (corresponding to the forward velocity U) and g_{26} (corresponding to rotor speed N); if these are multiplied by the respective 5% state errors in U and engine speed from Table 6.1 it can be seen that the magnitudes of their contributions $\mid g_{ji}e_i \mid$ to the total control signal are

$$g_{21} \ : \ 0.05 \times 7.5 = 0.375$$

$$g_{26} \ : \ 0.014 \times 330 = 4.62$$

This means that g_{26} dominates the engine throttle control; therefore it is necessary to increase the contribution from g_{21} and/or decrease that from g_{26}. In practice this corresponds to increasing q_{11} and/or decreasing q_{66} in the Q matrix until a reasonable compromise is achieved. Suitable values for these q_{ii}'s were found to be $q_{11} = 1.3$ and $q_{66} = 0.00002$.

(b) In the elevator control loop, the calculated control $\Delta\eta$ is excessive and the elevator is likely to saturate for small errors in the pitch angle Θ as seen by the contribution from $g_{14} = 14.7 \times 0.025 = 0.3675$; since η has a maximum limit of 0.36 rad, it is seen that a 5% error in the pitch angle causes the elevator to saturate. The situation can be remedied by increasing r_{11} so that the sensitivity is reduced. Therefore after one tuning phase, the following Q and R are arrived at:

Q_1 =diag[1.3, 0.6, 40, 40, 0.004, 0.00002],

R_1 =diag[8, 2].

These weighting matrices do indeed lead to a better response as shown in

Figure 6.2. The resulting gains at the initial state for these QR weights are

$$G = \begin{bmatrix} -0.088 & 0.18 & 3.09 & 10.52 & 0.017 & -0.000055 \\ -0.76 & -0.033 & 0.07 & 6.2 & -0.006 & -0.003 \end{bmatrix} \quad (6.13)$$

(c) It can be seen from Figure 6.2 that the forward velocity and engine speed responses are now satisfactory, but the responses in aircraft altitude and attitude are still unsatisfactory. The balance between the gains g_{12} to g_{15} needs to be adjusted to give a better control performance; how this is achieved can be found by analysing the aircraft responses and the $g_{ij}e_j$ contributions which form the elevator control. These contributions for the respective elements are:

g_{12} : $0.18 \times 1.5 = 0.27$

g_{13} : $3.09 \times 0.025 = 0.077$

g_{14} : $10.52 \times 0.025 = 0.263$

g_{15} : $0.017 \times 250 = 4.25$

It is clearly seen that the g_{13} contribution is significantly smaller than the others. This corresponds to the q_{33} element in Q, and so the effectiveness of q_{33} has to be increased by increasing the magnitude of q_{33}; such reasoning leads to the determination of the following Q and R matrices:

Q_2 =diag[1.3, 0.6, 200, 40, 0.004, 0.00002] ,

R_2 =diag[8, 2]

These yields the results shown in Figure 6.3, where it can be seen that the performance is reasonably satisfactory, and so suitable diagonal weighting matrices have been obtained. Of course further modifications can be made to improve the situation if necessary, but we stop here since the procedure has been demonstrated adequately.

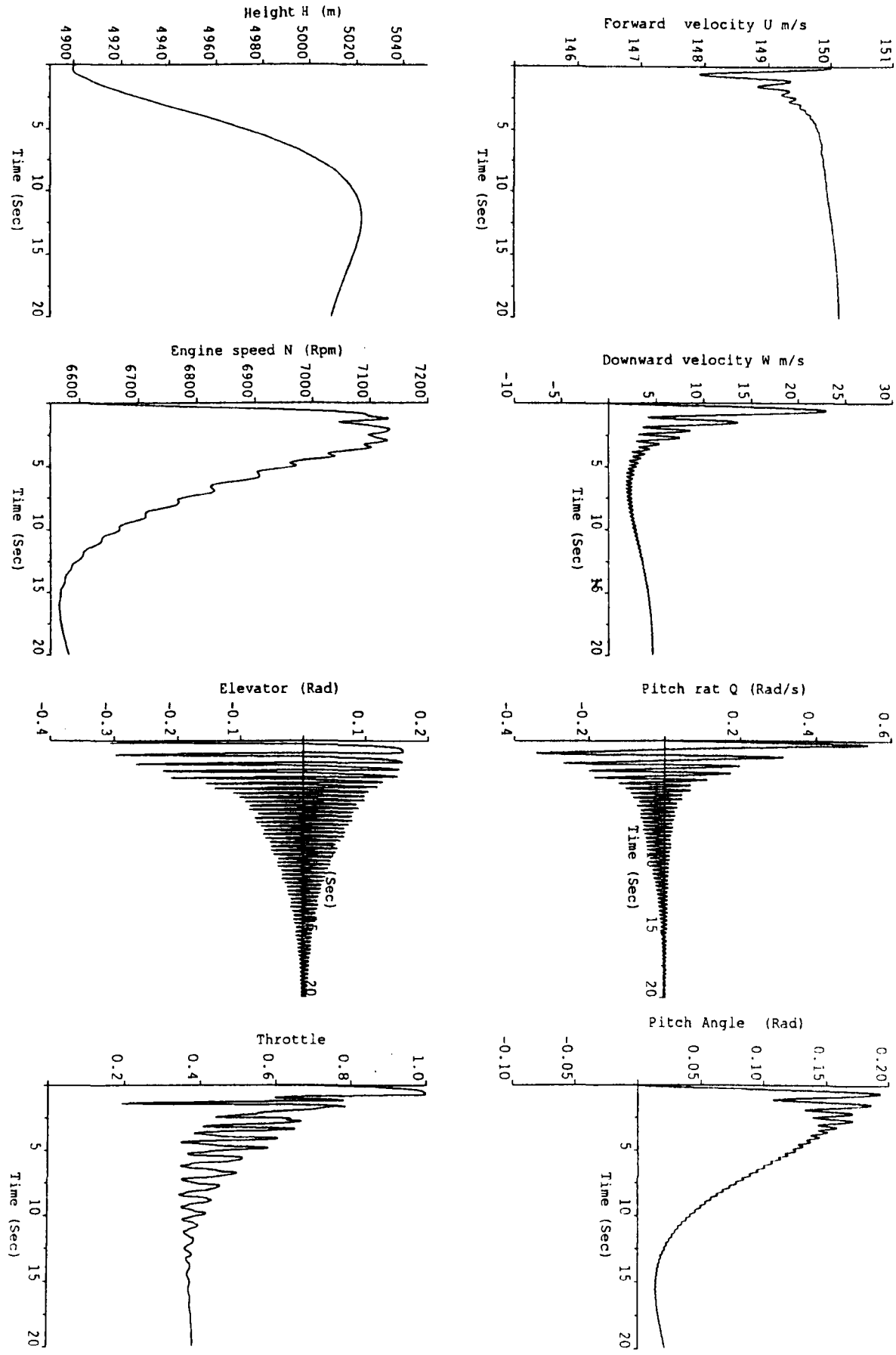

Figure 6.2: Longitudinal responses

6.4 Time Variational Effects

In the previous section, a procedure for selecting the QR weights for optimally controlling the longitudinal motion of an aircraft operating at some point has been presented. Since the system is time-varying, the time dependency has to be accounted for in the optimisation procedure. One practical method of obtaining a suitable strategy is to analyse the system equations and determine which states have the major influence on the time variational behaviour. From a study of the system considered, it was found that aircraft velocity caused the largest changes in the system description matrices. Therefore, the optimal controller design needs to be tested for different velocities in the working range of the aircraft. The result of this assessment showed that the above designed values for Q and R are satisfactory only in a small neighbourhood of the chosen operating velocity; as the aircraft velocity changes, the elements of the matrix B (mainly the ones corresponding to the elevator) increase significantly thereby disrupting the balance that has been designed for in the Riccati equation. It is found that the $PBR^{-1}B^TP$ term will have more effect than the other terms (for higher velocities), and so reasonable results can be achieved if this major effect can be cancelled. Since the change in the $PBR^{-1}B^TP$ term results mainly from the increase in the elements of B, a suitable correction can be obtained by adjusting the weighting matrix R. The time variational behaviour is most apparent in the elements of B which are related to the elevator control, hence increasing the corresponding R element, that is, increasing r_{11} with velocity will give the necessary adjustment. The practical procedure suggested is to simply increase the velocity in steps within the operating range and vary r_{11} until a satisfactory response is achieved at each range of operational velocity. The experimental results of this variation of the optimising r_{11} with aircraft velocity showed that the following relation can be derived:

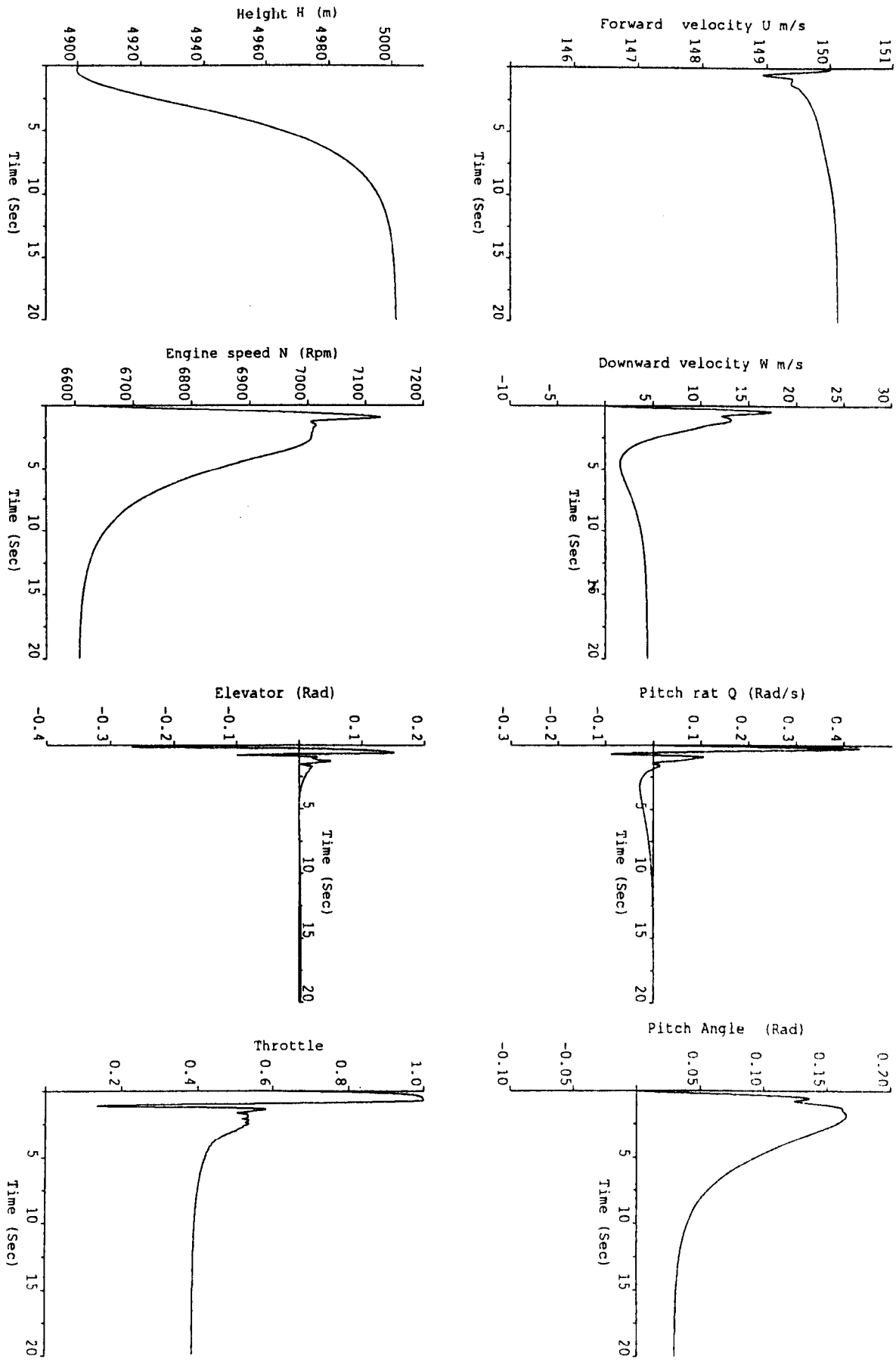

Figure 6.3: Longitudinal responses

$$r_{11} = -0.0022V_r^2 + 1.802V_r - 212.7 \tag{6.14}$$

Equation (6.14) can then be used with each Riccati equation integration to adjust the weighting element r_{11} on-line such that the resulting optimising feedback gains are adjusted according to the operating velocity; Figure 6.4 shows the system responses, for the same state error, at two different operating velocities (150 m/s and 250 m/s) where it can be seen that both performances are similar; the procedure can therefore lead to good aircraft control over a large operating range. Some of the Q elements may also be altered in the same way if it is required to compensate for any other undesired behaviour caused by the time-varying effects.

6.5 Lateral Motion Controller

The lateral motion dynamical equations are given in chapter 5. In the same way as for the longitudinal motion, we can design an optimal control law and its corresponding weighting matrices. Here we have $X = [V, P, R, \Phi, \Psi]^T$ is the state vector, $u = [\xi, \zeta]^T$ is the control input vector, and e is the state error vector.

From equation (6.5) – (6.7) we have

$$\Delta\xi = g_{11}e_1 + g_{12}e_2 + \cdots + g_{15}e_5 \tag{6.15}$$

$$\Delta\zeta = g_{21}e_1 + g_{22}e_2 + \cdots + g_{25}e_5 \tag{6.16}$$

where the gains g_{ji}'s are of the form

$$g_{1i} = \frac{-1}{r_{11}}(R_{m_\xi}L_{2i} + Y_{m_\xi}L_{3i}) \quad \text{for } i = 1, 2, \ldots, 5 \tag{6.17}$$

$$g_{2i} = \frac{-1}{r_{22}}(F_{y_\zeta}L_{1i} + R_{m_\zeta}L_{2i} + Y_{m_\zeta}L_{3i}) \quad \text{for } i = 1, 2, \ldots, 5 \tag{6.18}$$

and where, p_{ij}'s are the elements of the Riccati matrix $P(t_0)$ and r_{jj} are the diagonal elements of the lateral weighting matrix R.

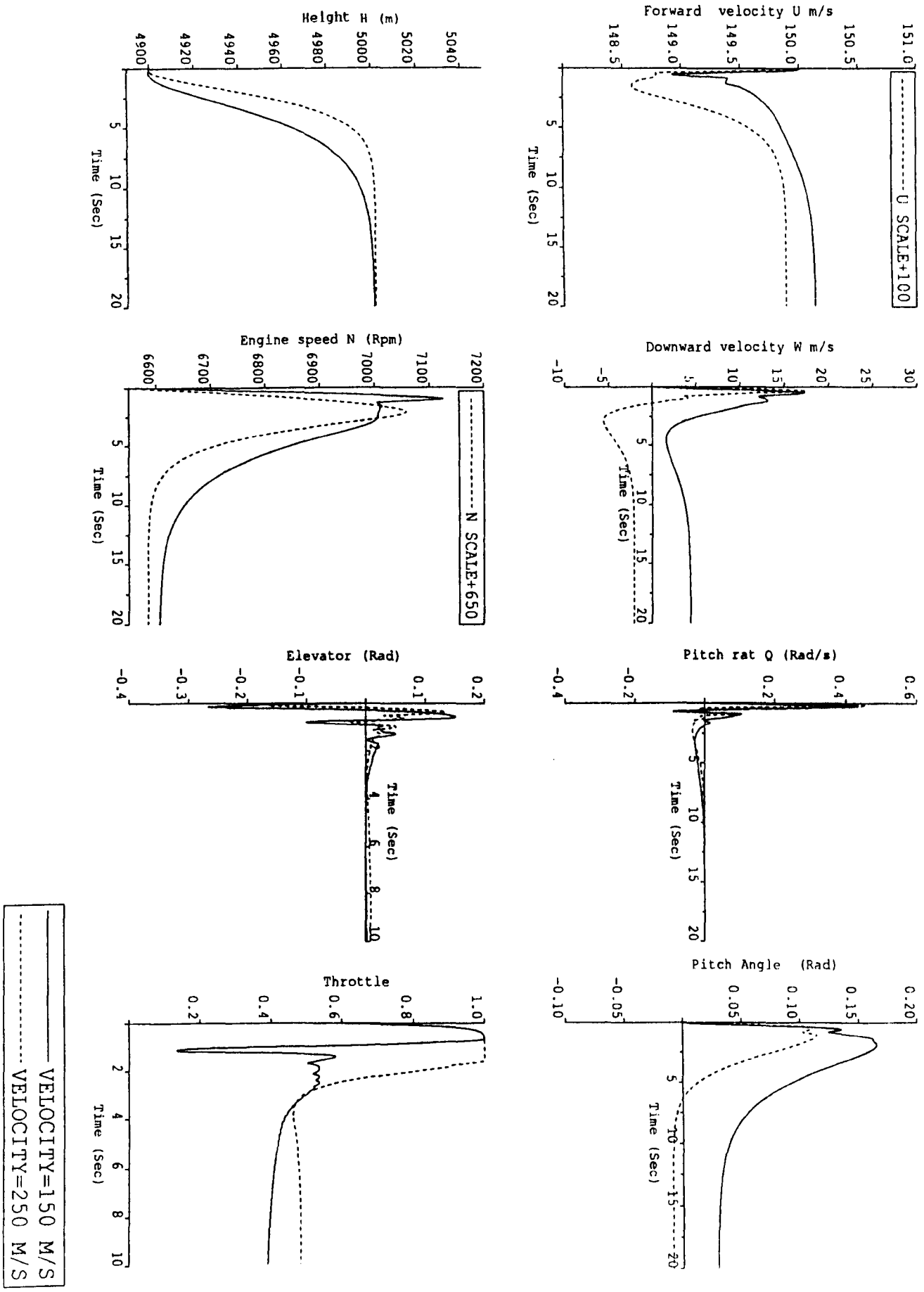

Figure 6.4: Longitudinal responses for two operating velocities

Using an initial state $X_0 = [0,\ 0,\ 0,\ 0,\ 0.2]^T$, a desired state $X_d = [0,\ 0,\ 0,\ 0,\ 0]^T$, and following the procedure presented in section 6.4 for the longitudinal motion, suitable Q and R weighting matrices for the lateral controller can also be designed. These weightings are

$Q = \text{diag}\,[0.008,\ 1,\ 20,\ 30,\ 200]$,

$R = \text{diag}\,[20,\ 20]$.

and give the control performance shown in Figure 6.5, and a lateral feedback gain matrix at the initial state as

$$G = \begin{bmatrix} 0.003 & 0.3 & -0.037 & 1.24 & 0.85 \\ -0.0042 & -0.017 & 1.52 & 0.33 & 2.6 \end{bmatrix} \tag{6.19}$$

Again, as for the longitudinal motion case, the effect of time variations on the performance of the lateral motion controller need to be taken into consideration. In this respect it was found that, as the aircraft velocity increases, the performance can be maintained by allowing for extra error in the velocity of side-slip (V); that is, keeping the same allowance for the angle of side-slip ($\beta = \sin^{-1}\ V/V_r$). This corresponds to the weighting element q_{11} which must be reduced as the aircraft velocity increases. As a result of off-line investigations, it was found that the following linear relationship for calculating the optimal q_{11} element can be used at different velocities:

$$q_{11} = \frac{(-0.003V_r + 1.25)}{100} \tag{6.20}$$

When this adjustment is made, the lateral control performance, shown in Figure 6.6, gives similar optimised performances for identical errors at different operating velocities. The responses shown are for an error in Ψ of 0.2 rad at operating velocities of 150 m/s and 250 m/s.

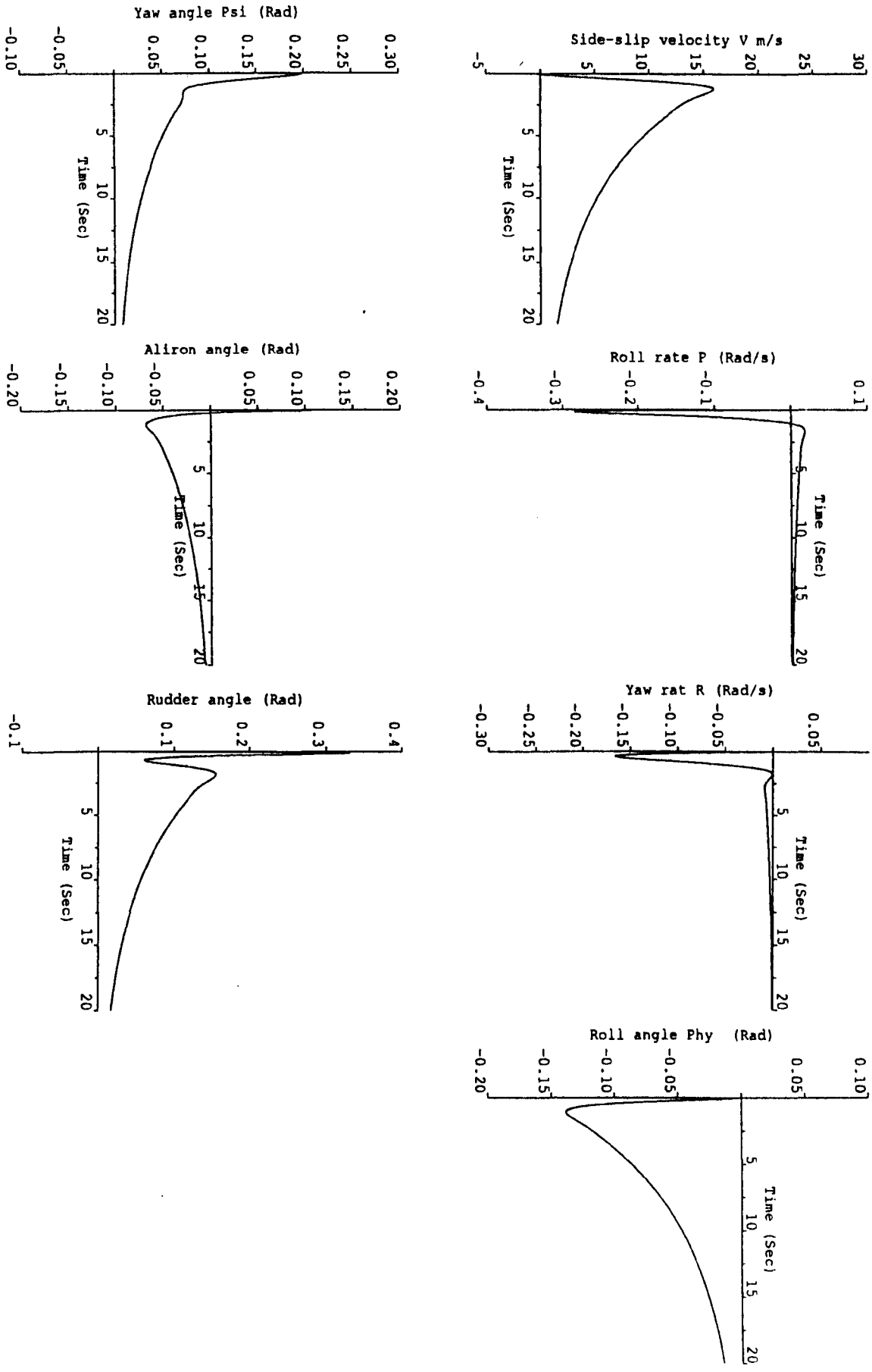

Figure 6.5: Lateral responses

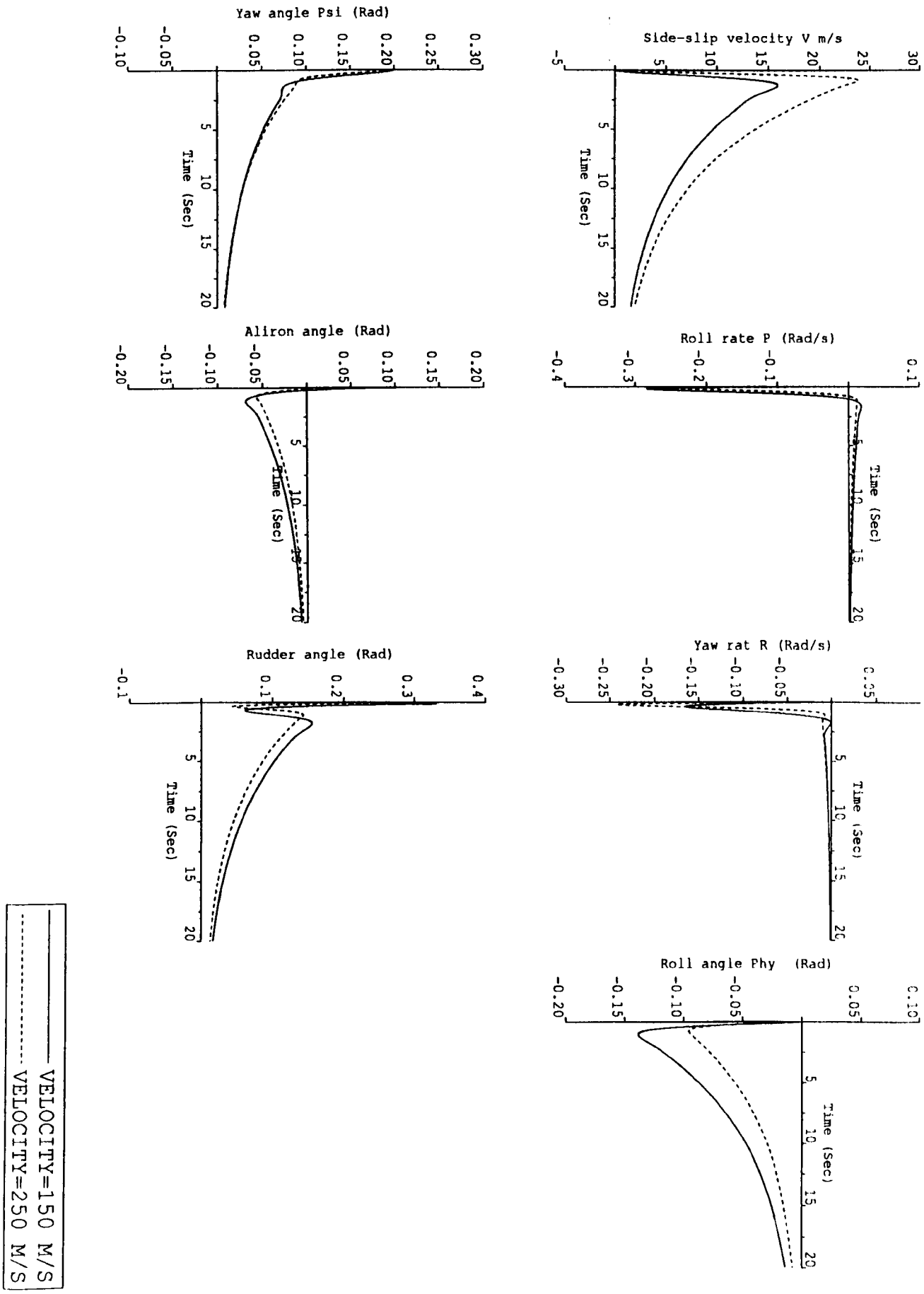

Figure 6.6: Lateral responses for two operating velocities

Chapter 7

Parallel Flight Controller Implementations

7.1 Introduction

In this chapter the optimal flight controller for the whole aircraft will be considered and how the control law computations can be parallelised using various transputer networks to achieve real-time performance.

As we have seen earlier, it is not possible to linearize the aircraft (and update the time-varying gains) every 5 ms, but after much longer time intervals dictated by the computing hardware. Since the aircraft is a time-varying system, the approximations may therefore yield poor descriptions – this clearly has consequences which are detrimental to the performance of the flight controller. To improve the performance, the linearization rate can be increased by using parallel processing techniques which allow the overall computing task to be partitioned into several sub-tasks that are mapped onto a multi-processor system. A regulator optimal control problem is considered where the controller is designed to drive initial errors towards the origin. The problem can be reformulated in a straight forward way to

yield the tracking problem case where the objective is to calculate the optimal flight control laws in response to pilot demands. Similar reasoning as for the longitudinal autopilot discussed in chapter 5 is used to arrive at a suitable control strategy for the whole aircraft. The cross-coupling effects are assumed to be constant over the linearization intervals. It is shown that when the processing is partitioned over several processors the linearization time intervals can be reduced to well within 400 ms second enabling accurate representations of the nonlinear time-varying aircraft, and thereby achieving good control performances as compared to the results when a single computing device is used. The efficiency of the parallel implementations is highlighted by comparing different multi-processor solutions with a uni-processor solution. Timings of the multi-processor designs are analysed to study the effects of communication overheads, idle times, etc, and how the parallelisation efficiency can be improved.

7.2 The Aircraft Model

The nonlinear equations which define the motion of the aircraft are given in chapter 3 and may be put in the form

$$\dot{X}(t) = f(X(t), u(t)) \tag{7.1}$$

where $X = [U, W, Q, \Theta, H, N, V, P, R, \Phi, \Psi]^T$ is the state vector and $u = [\eta, \gamma, \xi, \zeta]^T$ is the control vector. These equations can be written in a linearized form to highlight

the cross-coupling terms between the longitudinal and lateral motions as

$$
\left[\begin{array}{c} \dot{u} \\ \dot{w} \\ \dot{q} \\ \dot{\theta} \\ \dot{h} \\ \dot{n} \\ \hline \dot{v} \\ \dot{p} \\ \dot{r} \\ \dot{\phi} \\ \dot{\psi} \end{array}\right] = \left[\begin{array}{cccccc|ccccc} f_{1u} & f_{1w} & f_{1q} & f_{1\theta} & 0 & f_{1n} & R & 0 & V & 0 & 0 \\ f_{2u} & f_{2w} & f_{2q} & f_{2\theta} & 0 & f_{2n} & -P & -V & 0 & f_{2\phi} & 0 \\ f_{3u} & f_{3w} & f_{3q} & f_{3\theta} & 0 & f_{3n} & 0 & f_{3p} & f_{3r} & f_{3\phi} & 0 \\ 0 & 0 & f_{4q} & 0 & 0 & 0 & 0 & 0 & f_{4r} & f_{4\phi} & 0 \\ f_{5u} & f_{5w} & 0 & f_{5\theta} & 0 & 0 & f_{5v} & 0 & 0 & f_{5\phi} & 0 \\ f_{6u} & 0 & 0 & 0 & f_{6h} & f_{6n} & 0 & 0 & 0 & 0 & 0 \\ \hline f_{7u} & f_{7w} & 0 & f_{7\theta} & 0 & 0 & f_{7v} & f_{7p} & f_{7r} & f_{7\phi} & 0 \\ f_{8u} & f_{8w} & f_{8q} & 0 & 0 & 0 & f_{8v} & f_{8p} & f_{8r} & 0 & 0 \\ f_{9u} & f_{9w} & f_{9q} & 0 & 0 & f_{9n} & f_{9v} & f_{9p} & f_{9r} & 0 & 0 \\ 0 & 0 & f_{10q} & f_{10\theta} & 0 & 0 & 0 & f_{10p} & f_{10r} & f_{10\phi} & 0 \\ 0 & 0 & f_{11q} & f_{11\theta} & 0 & 0 & 0 & 0 & f_{11r} & f_{11\phi} & 0 \end{array}\right] \left[\begin{array}{c} u \\ w \\ q \\ \theta \\ h \\ n \\ \hline v \\ p \\ r \\ \phi \\ \psi \end{array}\right]
$$

$$
+ \left[\begin{array}{cc|cc} f_{1\eta} & 0 & 0 & 0 \\ f_{2\eta} & 0 & 0 & 0 \\ f_{3\eta} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & f_{6\gamma} & 0 & 0 \\ \hline 0 & 0 & 0 & f_{7\zeta} \\ 0 & 0 & f_{8\xi} & f_{8\zeta} \\ 0 & 0 & f_{9\xi} & f_{9\zeta} \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{array}\right] \left[\begin{array}{c} \eta \\ \gamma \\ \hline \xi \\ \zeta \end{array}\right] \tag{7.2}
$$

where $f_{1a} = \partial f_1 / \partial a$ is the partial derivative of the first element of function f, that is f_1, with respect to state a, etc, and $u, w, q, \ldots, \psi$ are the state perturbations about the linearized point. In compact form this reduces to:

$$
\left[\begin{array}{c} \dot{x}_1(t) \\ \dot{x}_2(t) \end{array}\right] = \left[\begin{array}{cc} A_{11} & A_{12} \\ A_{21} & A_{22} \end{array}\right] \left[\begin{array}{c} x_1(t) \\ x_2(t) \end{array}\right] + \left[\begin{array}{cc} B_1 & 0 \\ 0 & B_2 \end{array}\right] \Delta u(t) \tag{7.3}
$$

where A_{11}, A_{12}, A_{21} and A_{22}, are the partial derivations of the nonlinear function f in equation (7.1) with respect to the states, B_1 and B_2 are the derivatives with respect to the controls, $x_1 = [u, w, q, \theta, h, n]^T$ is made up of variables related to the longitudinal motion and $x_2 = [v, p, r, \phi, \psi]^T$ has variables connected to the lateral motion. Hence A_{12} and A_{21} represents the cross-coupling terms, and if these are absent the longitudinal and lateral motions are decoupled and can be handled independently of each other. It is well known, see for example Blakelock [7], that the cross-coupling terms can be removed by assuming:

1. the aircraft is in straight and unaccelerated flight and then disturbed by deflections of the control surfaces.

2. the elevator deflection causes only a pitching moment about the OY axis and causes no rolling or yawing moments.

3. the aileron and rudder deflections causes rotations only about the OX and OZ axes respectively.

These assumptions are not strictly valid in many modern aircraft. Furthermore the designs are evolving towards aircraft having more weight concentrated in the fuselage and their wings are becoming thinner, shorter and hence lighter. This weight shift is causing the cross-coupling effects to increase because the magnitudes of the moments of inertia in equations (3.21)–(3.23) are changing significantly. As more weight is concentrated along the longitudinal axis, the moment of inertia about the OX axis, I_x decreases while the moments of inertia about the OY and OZ axes increase. This phenomena increases the interaction between the longitudinal and lateral motions, and can best be seen by examining these moment equations. As I_x becomes much smaller than I_y and I_z the moment of inertia difference terms $(I_z - I_x)$ and $(I_x - I_y)$ become large. Hence if a rolling moment is introduced, it

results in some yawing moment and the $P(t)R(t)(I_z - I_x)$ term may become large enough to cause considerable pitching. Other factors which must be considered are the aerodynamic cross-coupling effects. For example, the lateral aerodynamic derivatives are proportional to the angle of attack (α), which is dependent upon the longitudinal states (i.e, $\alpha = \tan^{-1} W/U$). Also, in the design of standard flight controllers, pointwise linearizations of the aircraft are used – this means the aircraft may not be in straight and level unaccelerated flight at the linearization instants as normally assumed above leading to errors in the models used in the controller design.

It is clear that the standard assumptions are not always valid and so a more general approach is necessary for obtaining improved system performance. In this respect we propose to decouple the longitudinal and lateral motions but to assume that the cross-coupling effects are constant over the linearization intervals. Equation (7.2) shows that the longitudinal motion can be separated if we assume that the lateral state perturbations, that is $[v, p, r, \phi, \psi]^T$ have small values and can be neglected. In the same way we can separate the lateral motion equations by neglecting the longitudinal perturbations. Therefore if X is a vector of the state variable then

$$X(t) = X_0 + x(t) \tag{7.4}$$

where X_0 is the value of the state vector at the linearization instant, $x(t)$ is the state vector perturbation. This forms the basis for the decoupling of the aircraft motions; the equations for the two motions are presented next.

7.2.1 Decoupled Aircraft Motions

Longitudinal Motion

When considering the longitudinal motion, the lateral state perturbations are assumed to be small, and so their rates of change can be neglected during an interval

T_{lin} between two successive linearizations. Hence $\dot{V} = \dot{P} = \dot{R} = \dot{\Phi} = \dot{\Psi} = 0$, and this leads to the following longitudinal equations (not showing the time dependence for convenience):

$$\dot{U} = \frac{X_f}{M} - QW + (R_0 V_0)^* \tag{7.5}$$

$$\dot{W} = \frac{Z_f}{M} + QU - (P_0 V_0)^* \tag{7.6}$$

$$\dot{Q} = \left\{P_m + (I_z - I_x) P_0 R_0 + I_{xz}\left(R_0^2 - P_0^2\right)\right\} / I_y \tag{7.7}$$

$$\dot{\Theta} = Q\cos\Phi_0 - (R_0 \sin\Phi_0)^\dagger \tag{7.8}$$

$$\dot{H} = U\sin\Theta - W\cos\Theta\cos\Phi_0 - V_0\cos\Theta\sin\Phi_0 \tag{7.9}$$

$$\dot{N} = \dot{N}_{idle} + \frac{\gamma(N_{max} - N_{idle})^2 - (N - N_{idle})^2}{4(N - N_{idle})} \tag{7.10}$$

Lateral Motion

In the same way the effects of the longitudinal perturbations can be ignored by assuming $\dot{U} = \dot{W} = \dot{Q} = \dot{\Theta} = \dot{H} = \dot{N} = 0$, which gives rise to the following lateral equations:

$$\dot{V} = \frac{Y_f}{M} + PW_0 - RU_0 \tag{7.11}$$

$$\dot{P} = \left\{R_m + (I_y - I_z) RQ_0 + I_{xz}\left(\dot{R} + PQ_0\right)\right\} / I_x \tag{7.12}$$

$$\dot{R} = \left\{Y_m + (I_x - I_y) PQ_0 + I_{xz}\left(\dot{P} - RQ_0\right)\right\} / I_z \tag{7.13}$$

$$\dot{\Phi} = P + (Q_0 \sin\Phi + R\cos\Phi)\tan\Theta_0 \tag{7.14}$$

$$\dot{\Psi} = \{Q_0 \sin\Phi + R\cos\Phi\} / \cos\Theta_0 \tag{7.15}$$

These sets of equations will be used for short time intervals T_{lin} over which the aircraft is linearized and suitable optimal control laws designed. It is clear that in this way the cross-coupling effects between the two motions are allowed for, but are assumed to be constant over the linearizing intervals. For example when considering the longitudinal motion, the lateral variables (V, P, R, Φ, Ψ) are assumed to be

constant at their values $(V_0, P_0, R_0, \Phi_0, \Psi_0)$ when the linearization is performed, and vice versa. In the longitudinal equations some of the cross-coupling terms (assumed constant) appear as constants, and hence will vanish in the linearization process. Other terms however, eg. $Q\cos\Phi_0$ in equation (7.8) are combined with the longitudinal terms and will not vanish. In the lateral motion equations all the (assumed constant) longitudinal terms are combined with the (assumed varying) lateral terms. Therefore $(U_0, W_0, Q_0, H_0, N_0)$ will not vanish in general from the linearized lateral equations, and hence they will effect the optimal feedback gains calculated by the control algorithm.

A procedure for handling the cross-coupling terms which vanish in the linearization (and hence are not handled in the control algorithm) in the longitudinal equations is possible along the lines indicated below.

1. The terms marked * in equations (7.5) and (7.6) normally have a small effect on the longitudinal motion and can therefore be removed from the equations without affecting the performance significantly.

2. The inertial cross-coupling terms in equation (7.7) are normally the most important terms. They can be neutralised by applying an equal and opposite amount of pitching moment using the elevator. Such a result can be achieved by setting

$$\Delta\eta = \left\{PR(I_x - I_z) + I_{xz}\left(P^2 - R^2\right)\right\} / \frac{\partial P_m}{\partial \eta} \tag{7.16}$$

3. The term marked † in equation (7.8) can be eliminated by changing the pitch rate Q by an amount ΔQ where

$$\Delta Q = R\tan\Phi \tag{7.17}$$

This ΔQ can be added to the demanded pitch rate Q_d causing an error which generates (when multiplied by the corresponding feedback gain) an elevator control action dependent upon yaw rate and roll angle to keep the pitch angle and aircraft height at desired values.

In this way it is not necessary to wait until significant errors accumulate in the longitudinal states before remedial action is taken since such an action can be computed and applied as soon as the errors arise in the lateral attitude angles and rates using equation (7.16) and (7.17). In this way these cross-coupling effects can be kept to a minimum.

7.3 Real-time Implementation

The approach taken here is to formulate an optimal control problem for the complete linearized aircraft, separate the motions into the longitudinal and lateral dynamics, taking into account some of the cross-coupling effects explicitly and others implicitly in the optimal control algorithm, and solve the two "decoupled" subproblems using a multi-transputer network in real-time. Algorithm 3, presented in chapter 5, is used to provide the control for each motion provided that when is used to control the longitudinal motion, the extra terms needed to account for the inertial cross-coupling $\Delta\eta$ from equation (7.16) should be added to the calculated elevator control input and the additional pitch rate ΔQ from equation (7.18) should be added to the desired pitch rate as discussed earlier. Clearly for real-time performance all the computations have to be performed iteratively within the time scales of the aircraft and as already mentioned, a sampling rate of approximately 200 Hz is normally needed to achieve satisfactory control; hence Algorithm 3 should be processed to provide control updates for the two motions every 5 ms.

7.3.1 Single Transputer Implementation

The aircraft data supplied by British Aerospace was coded in Parallel C to run on a single T800 transputer, and as a benchmark to assess the parallel implementations, the optimal control algorithm was executed on another single T800 transputer interacting with the aircraft simulator processor as shown in Figure 7.1.

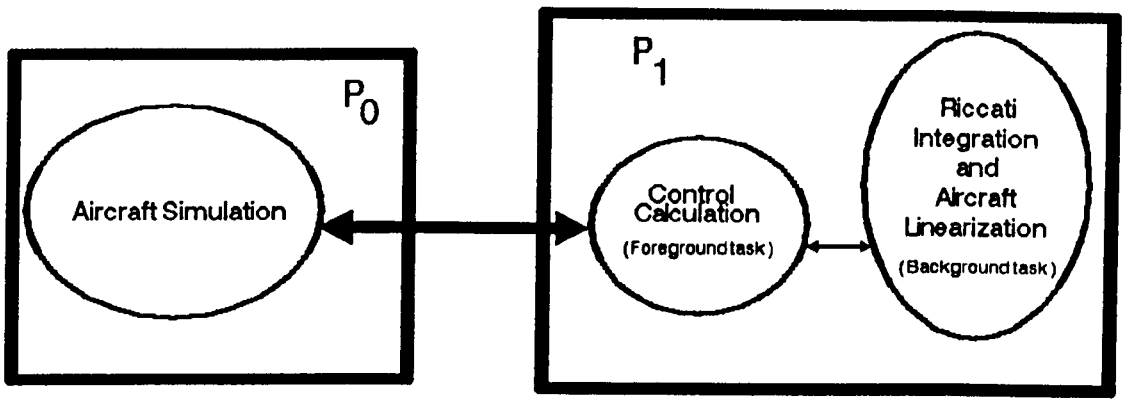

Figure 7.1: Single transputer autopilot implementation

Since the transputers currently available have only two levels of priority, the linearization and Riccati integrations for the lateral and longitudinal motions were executed as background tasks (non-urgent tasks) and the control updates as foreground tasks (urgent tasks) so that the control signals can be provided every 5 ms (see for example Bennett [5]). It was found that real-time performance was achieved using an optimising horizon T =5 seconds and integration step size of 20 ms and a linearizing updates every T_{lin} =600 ms. This time is quite satisfactory for the aircraft system considered. However for some advanced military aircraft with much faster dynamics, and which is required to perform complex manoeuvres, much faster linearizing rates are required to maintain good system representations at all times. To illustrate this aspects for the aircraft under consideration the linearization up-

date rate was dilebrately slowed to show the performance compared with the faster update rate. Figure 7.2 shows the responses for T_{lin} =600 ms and T_{lin} =4 seconds when an initial state X_{ic} = [180, 0, 0, 0, 4900, 6615, 0, 0, 0, 0, 0]^T and a desired state X_d = [150, 5, 0, 0.033, 5000, 6615, 0, 0, 0, 0, 0]^T are assumed. The lateral states are not shown because there is no significant change in them. The figure clearly shows that the control action is slower for the T_{lin} =4 seconds case because the aircraft linearization is carried out at high velocity, but as time passes the aircraft velocity is reduced thus changing the aircraft dynamics, but these are not followed adequately with the slow linearizing rate and leads to the poorer response. The faster update implementation is able to follow the aircraft changes more closely. Such poor modelling can lead to situations where instability results.

In addition fast linearization updates are very useful in instances where aspects such as fault tolerance are considered (see chapter 8) and which use analytical redundancy to detect and isolate failures. Such applications require fast updates to the reference models so that quick and reliable fault detection can occur.

To improve the situation the linearization update time can be reduced by using multiple processor systems, as discussed next.

7.3.2 Multi-Transputer Implementations

It has been shown in the previous chapters, that the longitudinal and lateral motions of the aircraft can be optimally controlled using a parallel processing approach where the linearizing intervals T_{lin} can be made equal to the execution time for Steps 1 and 2 of the control algorithm (Algorithm 3). Therefore if extra processors are used to execute the algorithm, the time interval T_{lin} can be reduced as required. Here we extend the results to cover the whole aircraft. A number of different parallel implementations are presented and each solution is analysed in terms of its timings

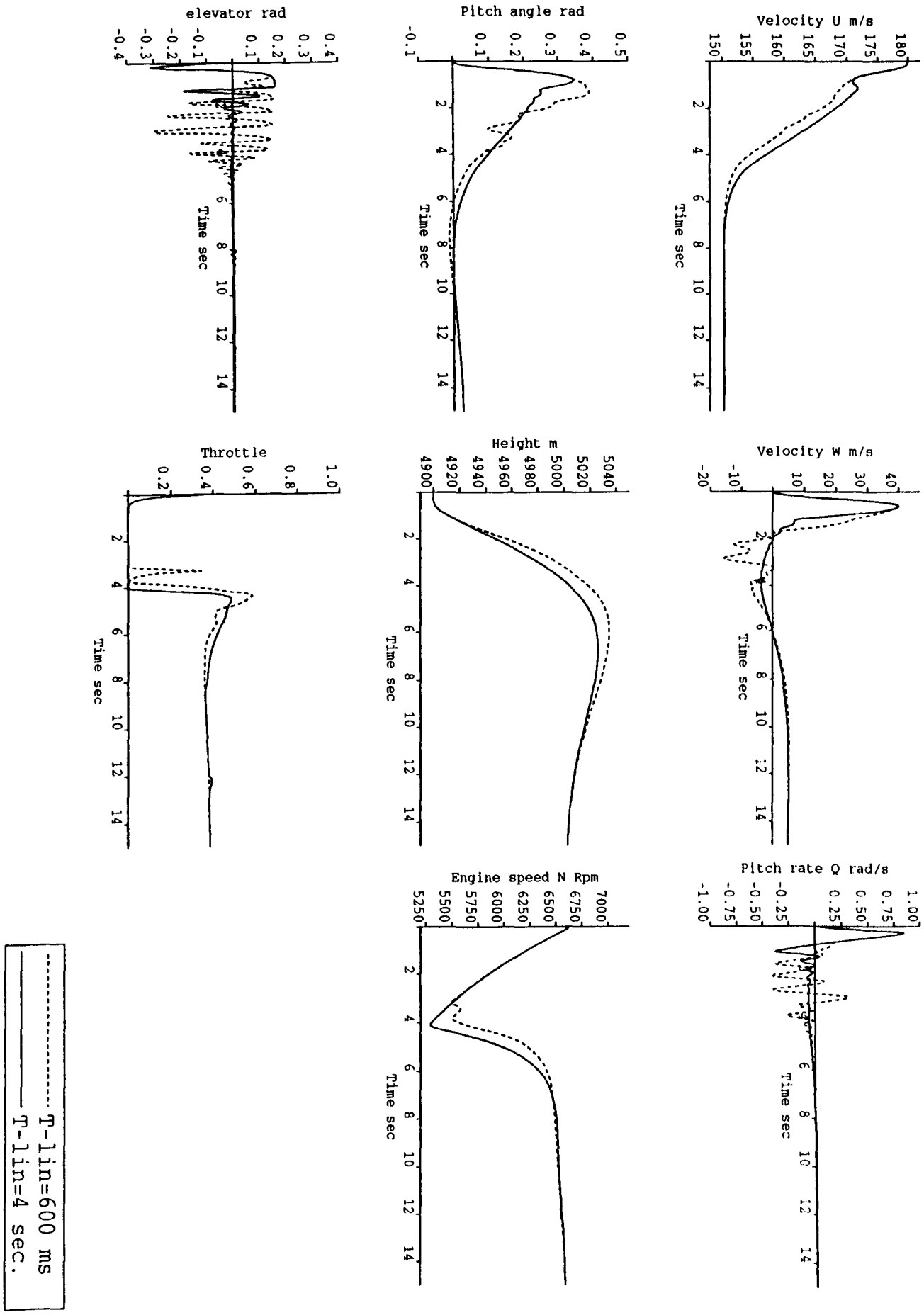

Figure 7.2: Control performances of fast and slow updates

and efficiency.

Two Transputer implementation

The first multi-processor solution to be presented is obtained by partitioning the longitudinal and lateral motions into two sub-tasks which are mapped onto two separate processors. The transputer network system shown in Figure 7.3 can be used to provide the real-time optimal control solution for this approach. The tasks

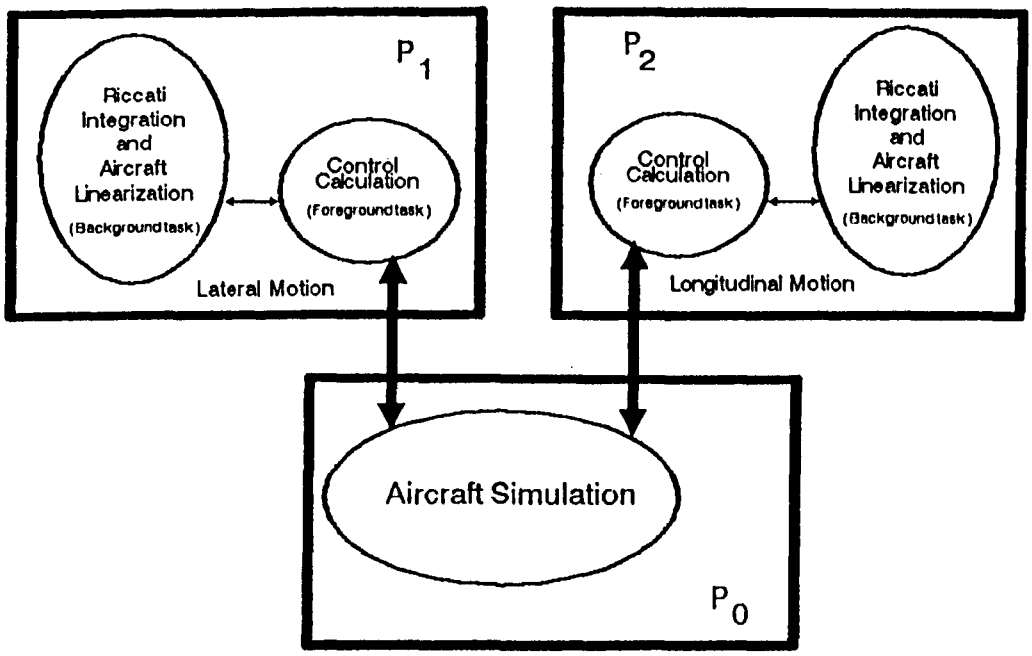

Figure 7.3: Two transputer autopilot implementation

of the separate processors in this configuration are as follows:

Transputer P_0 is the aircraft simulator.

Transputer P_1 handles the lateral motion, that is provide aileron (ξ) and rudder (ζ) control actions in its foreground task. Transputer P_1 performs these control updates by using the gains P_T calculated by a background task which performs the linearizations and solves the Riccati integration, that is Steps 1 and 2 of the control algorithm for the lateral motion.

Transputer P_2 controls the longitudinal motion of the aircraft, that is provide the elevator (η) and engine throttle (γ) controls. Essentially P_2 performs the same operations as P_1 but for the longitudinal motion.

Both the lateral and longitudinal transputers (P_1 and P_2) read the complete state from the simulator transputer P_0 so that the dominant cross-coupling effects not included in the optimal controller can be allowed for as discussed earlier. For a receding horizon interval T of 5 s, and if the integration step used in Step 2 of the algorithm is 20 ms then it is found that the lateral Riccati equations can be up dated every 215 ms ($= T_{l,lat}$) and the longitudinal ones every 345 ms ($= T_{l,long}$). These figures can be further reduced if the linearization and Riccati solution tasks are separated and distributed onto extra processors as discussed next.

Tree Network Autopilot Implementation

Since the Riccati integration sub-task is performed after the aircraft linearization sub-task has been completed, it is not worthwhile to map these two tasks onto separate processors as they are essentially sequential in nature. It may however be useful to distribute the individual sub-tasks onto more processors so that the cycle time is reduced. This is not done here but we speed up the cycle rate of the linearization by separating the control calculation sub-tasks and mapping them onto a separate transputer as shown in Figure 7.4. The control calculation sub-tasks are relatively lightweight and so this transputer (P_1) has significant spare processing capacity which can be used in a variety of ways to speed up the overall cycle time. For example P_1 can be used to share some of the processing for the aircraft linearization and/or assisting in the Riccati integration calculations. Another possibility, as discussed in chapter 8 is to use the spare capacity to monitor and maintain other aircraft functions such as fault tolerance by being able to reconfigure the control

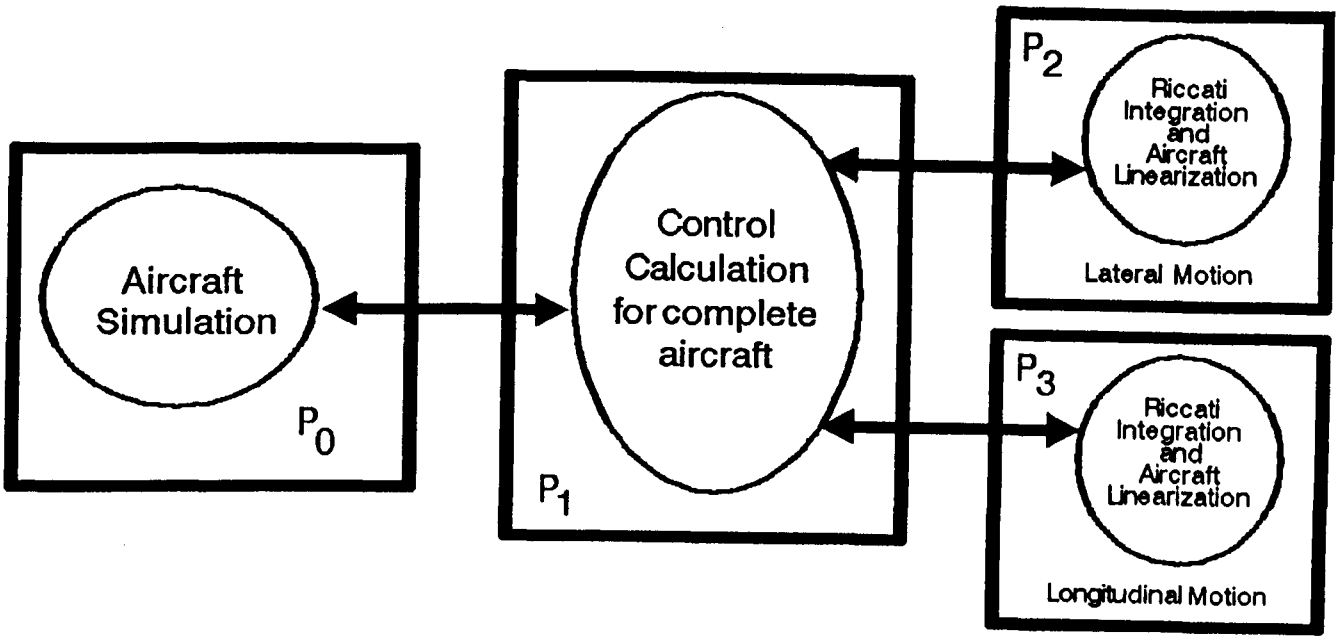

Figure 7.4: Tree network implementation

forces under failure conditions.

Using the network shown in Figure 7.4 it was found that the lateral motion transputer P_2 can update the gains every 190 ms, the longitudinal transputer P_3 can cycle every 295 ms. The control update transputer P_1 requires 993 μs to perform the controlling and communication tasks leaving the remainder of the 5 ms sample time to perform "other operations".

It should be noted that if the transputer P_2 and P_3 are used together to perform linearization and Riccati calculations for the overall aircraft as shown in Figure 7.5, that is, not splitting the processing into the longitudinal and lateral parts, then the results will not be as good as the tree network implementation of Figure 7.4. Here transputers P_2 and P_3 first solve the linearization task of the two motions and then perform the Riccati equation integrations by splitting the job of each motion between them. Extra communications are necessary in this implementation and so the efficiency is reduced. In this case it was found that P_2 and P_3 used collectively gives a linearization and feedback gains update every 342 ms.

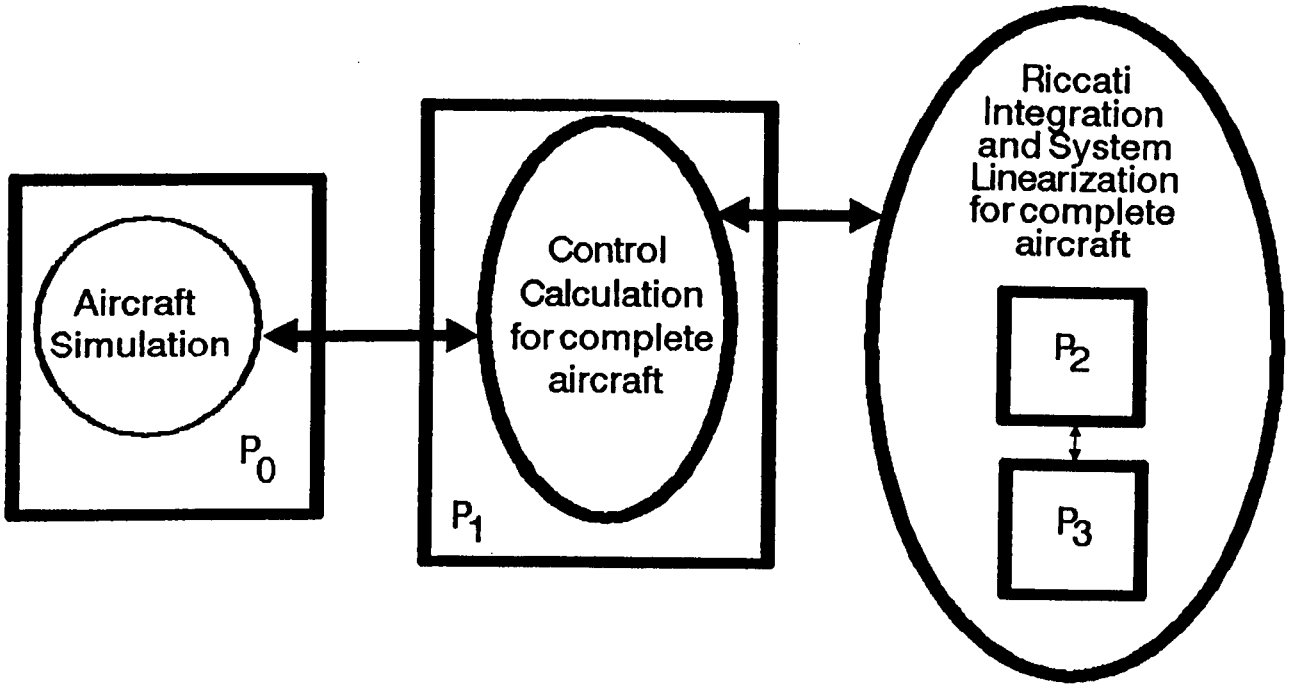

Figure 7.5: "Collective Mode" network implementation

The processing and communication times are significantly increased if the whole aircraft is considered as one optimisation problem due to the larger dimensionality of the problem. The timings for the various parallel implementations discussed can be broken down into actual computation time, and communication times. These are shown in Table 7.1, and Table 7.2 shows the performance of the various transputer autopilot implementations. The efficiency of the multi-transputer implementations are calculated by using

$$\text{Efficiency} = \frac{\text{single transputer cycle time}}{\text{multi} - \text{transputer cycle time} \times \text{no. of transputers}} \tag{7.18}$$

7.4 Simulation Results

The real-time algorithm was coded to run on the various T800 transputer networks considered in section 7.3. For the state vector $X = [U, W, Q, \theta, H, N, V, P, R, \Phi, \Psi]^T$, an initial value of $X_{ic} = [\ 150,\ 5,\ 0,\ 0.033,\ 5000,\ 6615,\ 0,\ 0,\ 0,\ 0,\ 0.2]^T$ and a desired value of $X_d = [\ 150,\ 5,\ 0,\ 0.033,\ 5000,\ 6615,\ 0,\ 0,\ 0,\ 0,\ 0]^T$ is assumed, and

Table 7.1: Timings for different transputer implementations

Network		Cycle time	Processing time	Communication time
Two transputers	T_1	215 ms	208.9 ms	6.2 ms
	T_2	345 ms	335.1 ms	9.9 ms
Tree network	T_1	5 ms	728 μs	275 μs
	T_2	190 ms	189.89 ms	110 μ s
	T_3	295 ms	294.86 ms	132 μs
"Collective Mode" network	T_1	5 ms	728 μs	385 μs
	T_2	342 ms	242.8 ms	99.2 ms
	T_3	342 ms	242.8 ms	99.2 ms

so the aircraft is required to change its direction by 11.5°.

The weighting matrices shown in Table 7.3 were designed using the procedure presented in chapter 6.

The optimal trajectories and controls for the multi-transputer implementations are shown in Figure 7.6 (the longitudinal motion) and Figure 7.7 (the lateral motion). From these we can observe the following:

1. The elevator positive deflection in the first few seconds is due to the additional cross-coupling control $\Delta\eta$, while later on the elevator responds optimally to

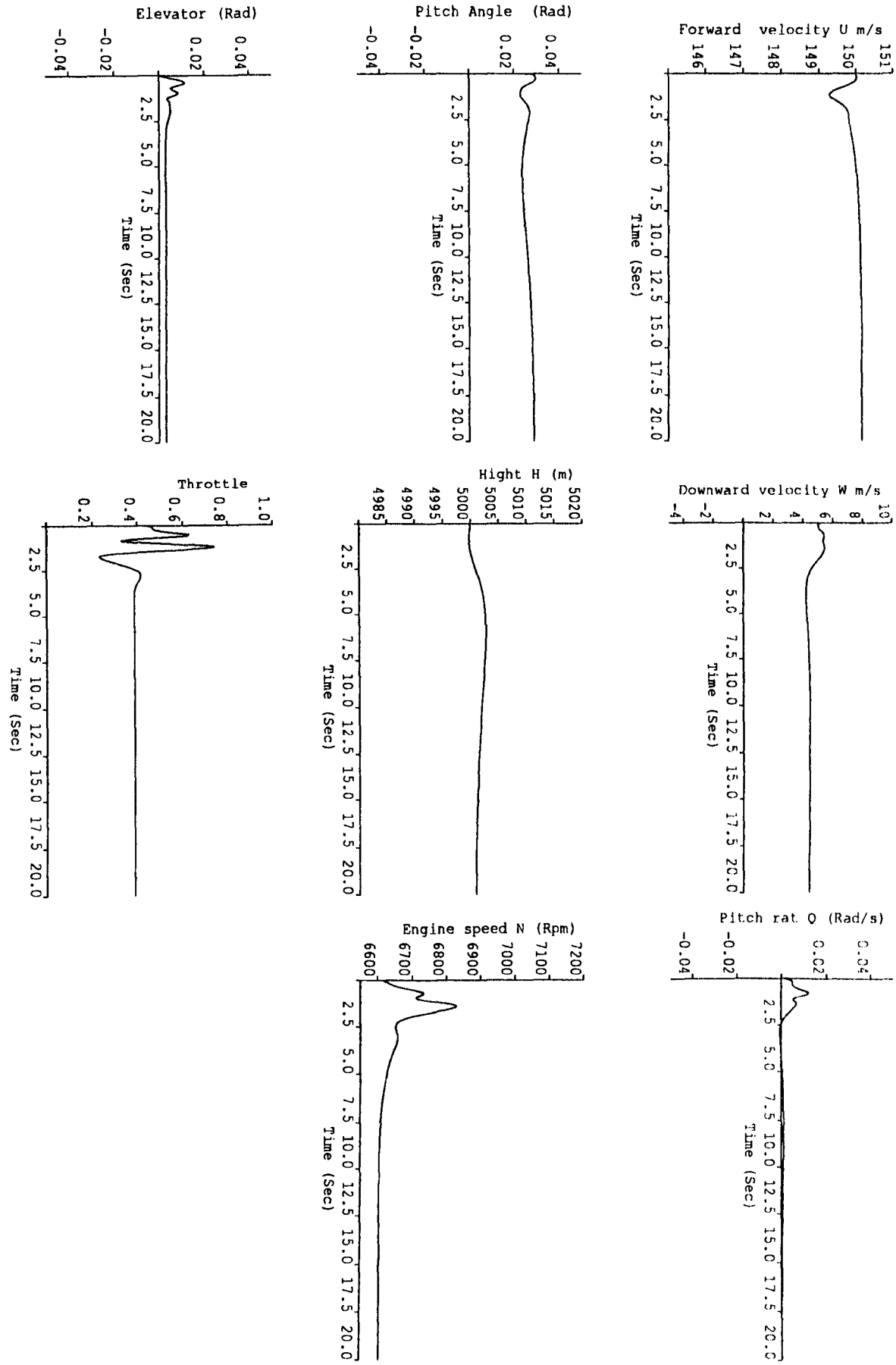

Figure 7.6: Longitudinal aircraft motion

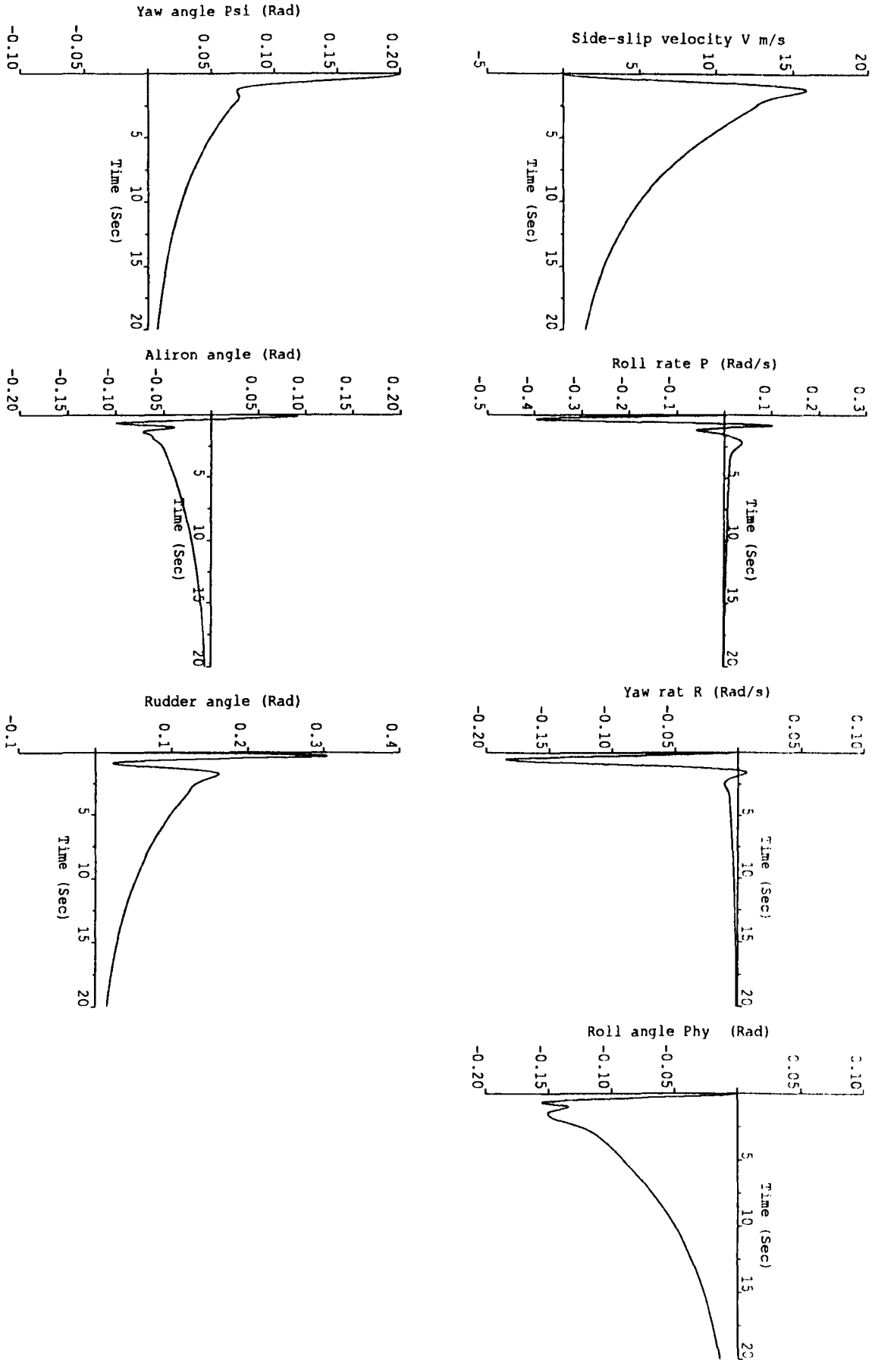

Figure 7.7: Lateral aircraft motion

Table 7.2: Performances for different transputer outopilot implementations

Network	Cycle times /ms	Efficiency	Spare capacity
Uni-processor	$T_l = 600$	1.0	No
Two processors	$T_{l,lat}$ =215, $T_{l,long}$ =345	0.87	No
Tree network	$T_{l,lat}$ =190, $T_{l,long}$ =295	0.68	Yes
"Collective Mode" network	T_l =342	0.59	Yes

Table 7.3: Weights used in the autopilot

Longitudinal motion	Lateral motion
Q_{long} = diag [1.3, 0.6, 200, 40, 0.004, 2×10^{-5}]	Q_{lat} = diag[$q_{11}(V_r)$, 1, 20, 30, 200]
R_{long} = diag [$r_{11}(V_r)$, 2]	R_{lat} = diag [20, 20]
$F_{long} = 0$	$F_{lat} = 0$

reduce the errors in the pitch angle and aircraft height.

2. The reduction in Θ even though Q is positive is due to the cross-coupling effect from yaw rate and roll angle but it has been kept to a minimum due to the additional ΔQ term of equation (7.17).

3. The reduction in forward speed U is caused mainly by the positive pitch rate and slightly by the cross-coupling term (RV). This reduction in U is remedied optimally by a slight increase in throttle control.

4. The response to an error in the yaw direction (Ψ) is acceptable and the negative error in the roll angle (Φ) reduces the angle of side-slip. The amount of roll can

be reduced by increasing the weighting elements q_{11}, q_{33} and r_{11} in the lateral motion, but at the expense of increasing the settling time for the yaw angle heading.

Overall, the results shown in Figures 7.6 and 7.7 are adequate. Hence the linear time invariant aircraft model, and the cross-coupling assumptions are valid and useful when considering problems of this kind.

It has been shown that good performance levels are possible by splitting the computational task functionally into smaller sub-tasks which are processed by different processors. The individual processing devices can be configured into suitable architectures for optimising the computations for the application being considered. It is imperative that such flexibility be available for the effective employment of parallel processing techniques to a wide variety of applications.

Chapter 8

A Fault Tolerant Optimal Flight Control System

8.1 Introduction

The various real-time distributed optimal autopilot designs, presented in chapter 7, allow a viable design methodology. The methods ensure that each element in the control loop performs to required specifications using suitably selected weighting matrices for the "decoupled" longitudinal and lateral problems. The control algorithm, although allowing good performance, is not designed to efficiently exploit the resulting control power if the primary control surfaces become inoperative due to failures in the aircraft. Such issues are important in military aircraft systems and hence lead to very stringent reliability standards that dictate extensive redundancy in the hardware. This extra hardware results in additional cost but reduces the mean time between failures.

Most aircraft have excess control power which can be used when required; this excess power can allow an alternative to the duplication or triplication of the control linkages, and such an approach is taken in this paper to give rise to a fault-tolerant

autopilot design. Although a level of redundancy is inevitable in a realistic design, we show that it is possible to reconfigure and distribute the forces and moments of a failed control surface to the remaining functional surfaces. In this way good control performance can be maintained without resorting to redundancy in the actuator and servo hardware. The available aircraft mathematical model is used to design a method for effective fault-detection and identification of failures in control surfaces. This is performed using "digital logic" that compares state predictions with actual sensor measurements taken from the aircraft so that failures in control surface operation can be detected and hence acted upon. When a fault is detected it is necessary to distribute the failed control surface forces to the remaining, still functional, surfaces. Such a result is achieved by using control reconfiguration that allows redistribution of the control inputs taking the failure into account. The aircraft model provided by British Aerospace did not include split controlling surfaces – only differential ailerons ξ, collective elevator η, single rudder ζ, and one engine throttle control γ, were available. In order to be able to apply the control reconfiguration to the aircraft, it is necessary to modify the model so that split elevator and ailerons can be used when required. For example, if the aileron fails it is then possible to use the left and right elevators differentially to produce a rolling moment.

Under normal conditions, we have the situation shown in Figure 8.1 which is the ordinary optimal autopilot presented in chapter 7. Here the aircraft is linearized about some point to give:

$$\begin{bmatrix} \dot{x}_1(t) \\ \dot{x}_2(t) \end{bmatrix} = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} + \begin{bmatrix} B_{11} & 0 \\ 0 & B_{22} \end{bmatrix} \Delta u(t) \tag{8.1}$$

or

$$\dot{x} = Ax(t) + B\Delta u(t) \tag{8.2}$$

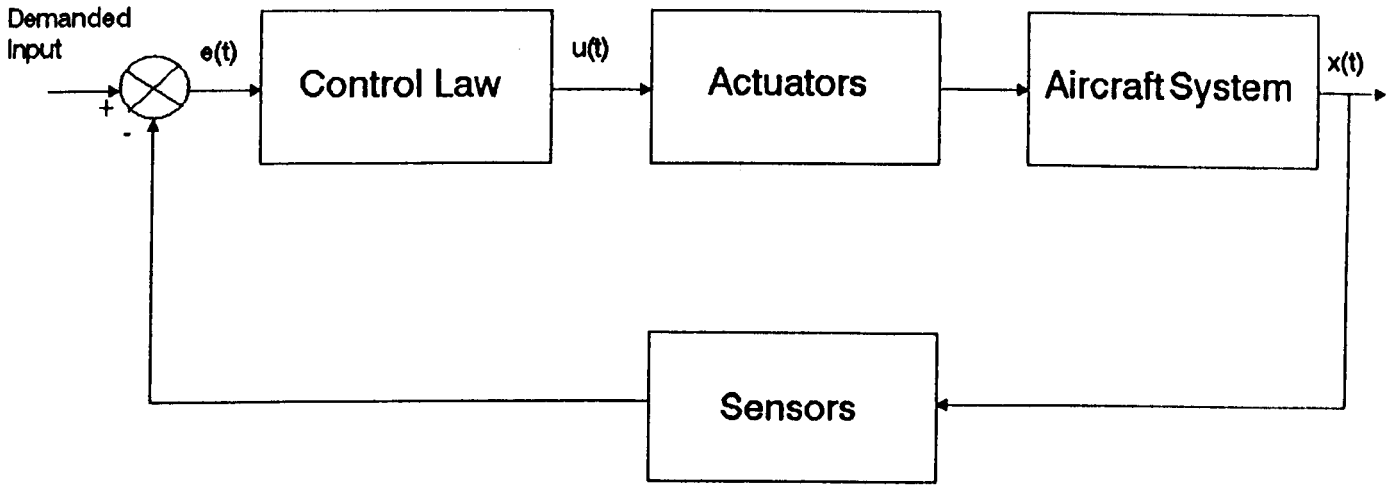

Figure 8.1: Original optimal flight control system

where as before $x_1 = [u, w, q, \theta, h, n]^T$ is the longitudinal motion state, $x_2 = [v, p, r, \phi, \psi]^T$ is the lateral motion state, and $\Delta u = [\eta, \gamma, \xi, \zeta]^T$ are the control inputs. We have seen that a receding-horizon optimal control law can be computed for an aircraft system, in real-time, using a multi-transputer network. Here we extend this autopilot design to enable a degree of fault-tolerance.

Under a failure condition, assuming the failed surface is locked to the centre position, that is, the input force to the aircraft from this surface is zero, the control signals are reconfigured using reconfiguration matrices so that the new system is as close to the original non-failed system as possible. The block diagram of the proposed fault-tolerant solution is shown in Figure 8.2, and which we will discuss next.

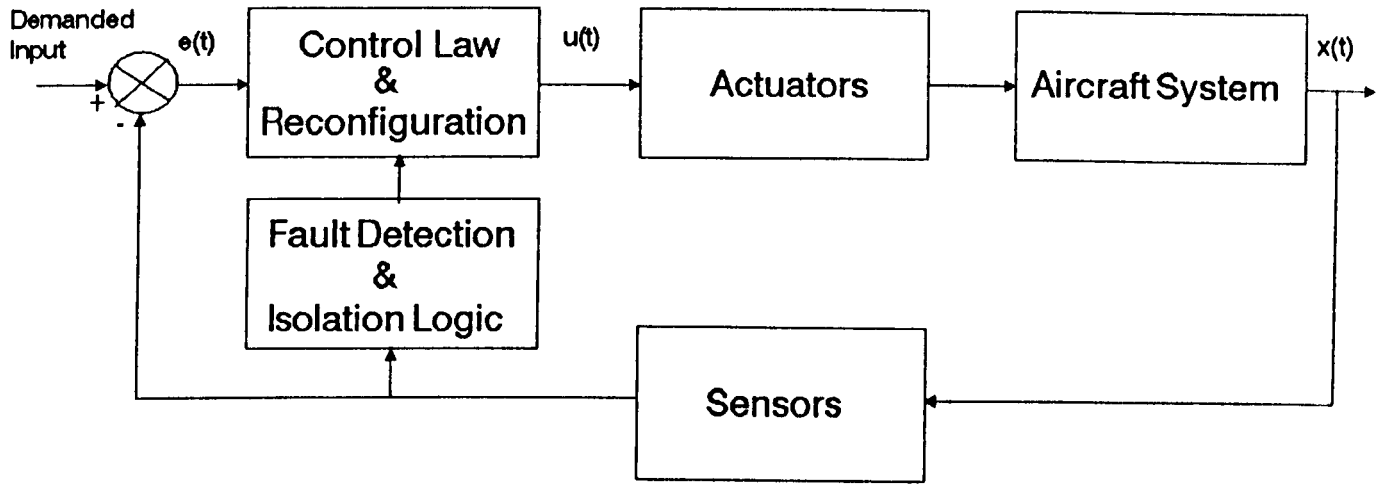

Figure 8.2: Reconfigurable flight control system

8.2 The Aircraft Optimal Control Law

Equation (8.1) shows the linearized form of the coupled aircraft equations. As we have seen in chapter 7, the aircraft two motions can be "decoupled" and we can design an optimal control law for each motion which will be

$$\Delta u^*_{long} = -R^{-1}_{long} B^T_{11} P_{long}(t_0) e_{long}(t) \tag{8.3}$$

$$\Delta u^*_{lat} = -R^{-1}_{lat} B^T_{22} P_{lat}(t_0) e_{lat}(t) \tag{8.4}$$

where the subscripts "long" and "lat" denote the longitudinal and lateral motions control law which are discused in the previous chapters and so $\Delta u^*_{long} = [\eta, \gamma]^T$ and $\Delta u^*_{lat} = [\xi, \zeta]^T$. The modified control input vector is as follows:

$$\Delta u = [\eta_r, \eta_l, \gamma, \xi_r, \xi_l, \zeta]^T$$

Therefore only the matrix B in equation 8.1 will be changed to become

$$B_0 = \begin{bmatrix} \bar{B}_{11} & \bar{B}_{12} \\ \bar{B}_{21} & \bar{B}_{22} \end{bmatrix} = \left[\begin{array}{ccc|ccc} \frac{f_{1\eta}}{2} & \frac{f_{1\eta}}{2} & 0 & f_{1\xi_r} & f_{1\xi_i} & 0 \\ \frac{f_{2\eta}}{2} & \frac{f_{2\eta}}{2} & 0 & f_{2\xi_r} & f_{2\xi_i} & 0 \\ \frac{f_{3\eta}}{2} & \frac{f_{3\eta}}{2} & 0 & f_{3\xi_r} & f_{3\xi_i} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & f_{8\gamma} & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & f_{7\zeta} \\ f_{8\eta_r} & f_{8\eta_i} & 0 & \frac{f_{8\xi}}{2} & -\frac{f_{8\xi}}{2} & f_{8\zeta} \\ f_{9\eta_r} & f_{9\eta_i} & 0 & \frac{f_{9\xi}}{2} & -\frac{f_{9\xi}}{2} & f_{9\zeta} \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{array}\right] \tag{8.5}$$

For the unimpaired aircraft there will be no effect from the submatrices $\bar{B}_{12}$ and $\bar{B}_{21}$ due to symmetry. Therefore if we examine the term $PBR^{-1}B^TP$ in the Riccati equation (5.8) we will find the new optimal control problem (with the modified control) is equivalent to the previous one provided that the (2×2) weighting matrices R_{long} and R_{lat} become (3×3) defined as follows:

$$\bar{R}_{long} = \text{diag}[\frac{r_{11}}{2}, \frac{r_{11}}{2}, r_{22}]^T$$

$$\bar{R}_{lat} = \text{diag}[\frac{r_{11}}{2}, \frac{r_{11}}{2}, r_{22}]^T$$

where r_{ii}'s are the corresponding diagonal elements of the matrices R_{long} and R_{lat}. The Riccati solution of the new problem will give the same gain matrices P_{long} and P_{lat} and only the feedback gain matrices will change to be

$$G_{long} = -\bar{R}_{long}^{-1}\bar{B}_{11}^T P_{long} \qquad (3 \times 6) \tag{8.6}$$

$$G_{lat} = -\bar{R}_{lat}^{-1}\bar{B}_{11}^T P_{lat} \qquad (3 \times 5) \tag{8.7}$$

Hence, the results obtained in the previous chapters can be used to calculate the new feedback gains.

8.3 Failure Detection

It is evident that before remedial action can be taken any fault present needs to be detected and isolated. The proposed solution of the failure detection problem is based on the reference model approach used in observer theory, see for example Patton [41]. An observer used for fault monitoring can be designed using eigenstructure assignment such that the eigenvalues of the closed-loop observer matrix are chosen not only to provide good state estimation but also to let certain faults manifest themselves in the observer/filter residuals. Therefore, an aircraft failure can be identified by using a state observer constructed from a reference model in conjunction with sensed aircraft data as shown in Figure 8.3. The observer matrix D

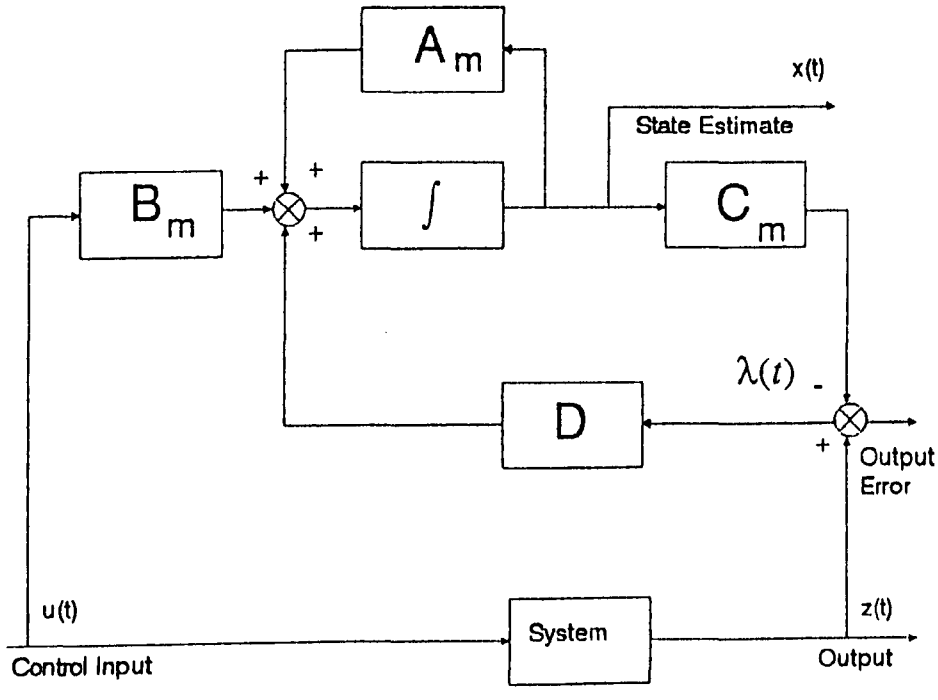

Figure 8.3: State estimator for fault detection

is normally chosen so that the eigenvalues of $(A_m - DC_m)$ are stable and that good state estimates are obtained, where the "m" subscripts denote that model matrices are being considered. However, in the present application, the primary concerns are that a stable observer be designed and certain failures be accentuated in the errors,

defined by

$$\lambda(t) = z(t) - C_m\hat{x}(t) \tag{8.8}$$

The triad (A_m, B_m, C_m) is assumed to accurately describe the aircraft model under normal unfailed conditions, and so $(A_m, B_m, C_m) \approx (A, B, C)$ and the subscripts can be omitted for convenience. Under failure conditions the aircraft can be modelled by

$$\dot{x}(t) = Ax(t) + B\Delta u(t) - b_i\Delta u_i(t) \tag{8.9}$$

$$z(t) = Cx(t) \tag{8.10}$$

for $t \geq t_0$, where b_i is the i^{th} column of B relating to the failed component of the input, and t_o is the (unknown) time of failure; hence an off-failure in the i^{th} control input is modelled such that u_i does not affect $\dot{x}(t)$. The observer/filter equation will be

$$\dot{\hat{x}}(t) = A\hat{x}(t) + B\Delta u(t) + D\lambda(t) \tag{8.11}$$

There are many techniques for designing failure sensitive filters, see for example Willsky [55]; one method is to choose the observer matrix D so that particular failure modes manifest themselves as residuals which remain in a fixed direction or plane. To illustrate this consider the case when $C = I_n$, so that the differential equation for the residuals is

$$\dot{\lambda}(t) = [A - D]\,\lambda(t) - b_i\Delta u_i(t) \tag{8.12}$$

If $D = \delta I_n + A$, where $\delta > 0$, it is straightforward to show

$$\lambda(t) = e^{-\delta(t-t_0)}\lambda(t_0) - \frac{b_i\Delta u_i}{\delta}\left(1 - e^{-\delta(t-t_0)}\right) \tag{8.13}$$

Hence as the initial conditions die out, $\lambda(t)$ maintains a fixed direction $-b_i\Delta u_i$ under failure conditions. Hence in this case a fault can be detected and alarms sounded

when $\mid \lambda(t) \mid \geq \varepsilon$, for some $\varepsilon > 0$. It is obvious that the state estimator (8.11) is asymptotically stable as the eigenvalues of $(A - D)$ have negative real parts. For good fault-tolerance the values of δ and ε have to be chosen carefully; small δ values correspond to good fault detection since they lead to large λ's, but also to poor state estimation since the observer dynamics are slow; small ε values lead to false alarms, whereas large values give reductions in the fault detections – clearly a compromise needs to be reached; both δ and ε need to be small and this is only possible if the estimation properties of the observer is good.

The problem with the above approach is its restriction to deterministic systems which are linear and time-invariant in their unfailed states. For our nonlinear time-varying aircraft, the filter reference model is updated in periods of T_{lin} with the feedback gains (see chapter 7). Therefore the linear time-varying model used in the filter can closely describe the aircraft system and give satisfactory results. However, the approximations used in the linearization process and if long T_{lin} periods are used can lead to poor performance. To improve the state estimation and hence the filter performance equation (8.11) can be modified to be

$$\dot{\hat{x}}(t) = f(\hat{x}(t), u(t)) + D\lambda(t) \tag{8.14}$$

where $f(.)$ represent the aircraft nonlinear equations and the matrix D for the two cases (for the linear time-varying and the nonlinear filters) will be

$$D = \delta I + A \tag{8.15}$$

and where A is the local linear representation to the aircraft system which is to be updated every T_{lin} seconds. In the nonlinear filter case, δ and ε may be kept small to improve the filter performance.

To reduce the computational complexity of our filter we also use a reduced-order observer/filter in which we only use the residuals of the q, p, r states since these

are directly affected by the above controls; it is therefore sufficient to detect these residuals to enable isolation of any failure. That is, we estimate three states instead of eleven. Therefore

$$\lambda = [\lambda_q, \lambda_p, \lambda_r] \tag{8.16}$$

and letting

$$s = [q, p, r]^T \tag{8.17}$$

$$x = [u, w, q, \theta, h, n, v, p, r, \phi, \psi]^T \tag{8.18}$$

$$\hat{x} = [u, w, \hat{q}, \theta, h, n, v, \hat{p}, \hat{r}, \phi, \psi]^T \tag{8.19}$$

our observer/filter equations will be

$$\dot{\hat{s}}(t) = \bar{A}\hat{x}(t) + \bar{B}u(t) + \bar{D}\lambda(t) \quad \text{linear filter} \tag{8.20}$$

$$\dot{\hat{s}}(t) = \bar{f}(\hat{x}(t) + u(t)) + \bar{D}\lambda(t) \quad \text{nonlinear filter} \tag{8.21}$$

where $\lambda = s - \hat{s}$, $\bar{A}$ is the (3×11) matrix consisting of rows 3, 8 and 9 of the A matrix, $\bar{B}$ is the (3×6) matrix consisting of rows 3, 8, 9 of the B matrix, $\bar{f}(.)$ is the corresponding three nonlinear moments equations, and $\bar{D}$ is given by

$$\bar{D} = \delta I + \begin{bmatrix} f_{3_q} & f_{3_p} & f_{3_r} \\ f_{8_q} & f_{8_p} & f_{8_r} \\ f_{9_q} & f_{9_p} & f_{9_r} \end{bmatrix} \tag{8.22}$$

where $\delta I = \text{diag}\,[\delta_{11}, \delta_{22}, \delta_{33}]$ and f_{i_a} are the corresponding elements of A. As soon as one of the elements of λ exceeds ϵ an alarm is raised. In this work we are not considering the effect of disturbances on the three states (Q, P, R); this will be considered in future work.

Having detected a fault the problem has to be isolated, that is, it is necessary to determine which actuator has failed. Such fault isolation can be achieved by using

digital logic. Here we are concerned in detecting any failures in the control actuators $\eta_r, \eta_l, \xi_r, \xi_l$, and ζ (where the subscripts relate to the left and right surfaces of the modified aircraft). It is easily seen that any fault can be isolated by analysis of the elements of λ; the element which exceeds ϵ first corresponds to the failed control while the other elements will be shifted into some direction depending on the failure. Thus the element of λ which goes larger than ϵ, together with the signs of the other elements and the applied controls gives an indication of which control has failed. For example, when $\mid \lambda_q \mid \geq \epsilon$, one of the elevators η_r or η_l must be in failure. To determine which one has failed, it is necessary to check the signs of λ_p and the elevator deflections as required by the controller. If it is required that the elevators be at negative deflection at the moment when the alarm is raised, and λ_p is positive, then the left elevator has failed and vice versa. In the same way we can detect and isolate aileron failures while rudder faults are isolated when only $\mid \lambda_r \mid \geq \epsilon$.

8.4 System Reconfiguration

Reconfiguration of the flight control law after effector failure is studied by several researchers, see for example Ratton [45], [46], McLean [32], and Russ [49]. The usual objective is to obtain a gain reconfiguration matrix so that the output of the reconfigured system is as close to the original unfailed system as possible. In our work, to simplify the analysis it will be assumed that when a failure occurs, the control surface is locked to the centre position, that is, the input to the aircraft from the failed surface is zero. To implement reconfiguration of the control signals, it is necessary to compute the gain reconfiguration matrices which distribute the forces and moments of the failed surface to the remaining, still functional, surfaces. This needs to be done for all possible faults so that in practice the reconfiguation can be achieved as a gain scheduling algorithm to apply the appropriate reconfiguration

matrix. In the fully functional aircraft case the gain matrix of the unimpaired aircraft is used.

As discussed in the previous chapters, an optimal autopilot for the linearised aircraft

$$\dot{x}(t) = Ax(t) + B_0\Delta\bar{u}(t) \tag{8.23}$$

is possible using $\Delta\bar{u} = G_0e(t)$, where G_0 is the unimpaired (or original) optimal controller gain matrix obtained by solving the Riccati equations and e is the error in the system states defined by $e = X - X_d$. The component of $\dot{x}$ due to the control inputs is given by the term $B_0G_0e(t)$. When a fault develops and the aircraft is impaired, the feedback matrix must be altered (reconfigured) and the $\dot{x}$ control component becomes $B_IG_Ie(t)$, where the subscripts "I" denote impaired matrices. It is clear that for the aircraft system to be unaffected by the impairment, we must have

$$B_IG_I = B_0G_0 \tag{8.24}$$

Therefore equation (8.24) provides a method of determining the unknown gain matrix G_I namely

$$G_I = B_I^+B_0G_0 \tag{8.25}$$

where B_I^+ is the pseudo-inverse of matrix B_I, see Noble [39]. If $n > m$, that is, the state dimension is greater than the number of control inputs then the solution of equation (8.24) which minimises the sum of the squares of the errors $J = r^Tr$ can be used, where $r = B_0G_0 - B_IG_I$. This gives

$$G_I = \left(B_I^TB_I\right)^{-1}B_I^TB_0G_0 \tag{8.26}$$

Since the various components can have different ranges and different limits, a weighted least-squares error estimate, or normalisation of the values by their physical limits,

can be used. In these cases equation (8.24) is modified to $WB_IG_I = WB_0G_0$ which yields

$$G_I = \left(WB_I^T\right)^+ WB_0G_0 \tag{8.27}$$

where W is a weighting matrix which needs to be designed. From equation (8.26), equation (8.27) can be written as

$$G_I = (B_I^T H B_I)^{-1} B_I^T H B_0 G_0 \tag{8.28}$$

where $H = W^T W$. Defining the reconfiguration matrix to be

$$M_I = (B_I^T H B_I)^{-1} B_I^T H B_0 \tag{8.29}$$

Equation (8.25) can be solved by minimising the error $J = r^T r + G_I^T K G_I$ to give the solution

$$M_I = (B_I^T H B_I + K)^{-1} B_I^T H B_0 \tag{8.30}$$

Here two weighting matrices H and K have to be designed, where H is used to optimise equation (8.24), and K is used to constrain the amplitude of the elements of the required control gain matrix G_I (see for example McLean [32]). Then the reconfigured feedback gain matrix will be

$$G_I = M_I G_0 \tag{8.31}$$

The reconfiguration matrices M_I can be computed off-line for different failure conditions and the correct one utilised as necessary on-line.

8.4.1 Calculation of the Reconfiguration Matrices

The aircraft equations were linearized about some operating point and the control matrix B_0 is found to be

$$B_0 = \begin{bmatrix} 0.07 & 0.07 & 0 & -0.17 & -0.17 & 0 \\ -6.28 & -6.28 & 0 & -9.8 & -9.8 & 0 \\ -5.8 & -5.8 & 0 & 0.7 & 0.7 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1477 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1.9 \\ -3.0 & 3.0 & 0 & -12.2 & -12.2 & 0.04 \\ 0 & 0 & 0 & -0.1 & 0.1 & -2.04 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{8.32}$$

To avoid the difficulties associated with the solution of equation (8.29) and (8.30), which result from the weighting matrices H and K, another approach of reasoning logic is used to design the reconfiguration matrices M_I as follows:

1. The gains of the failed surface are to be given to the remaining one; if the two control surfaces are failed then its gains must be distributed to the remaining control surfaces that are able to replace them.

2. The effect of loosing symmetry must be removed such that equation (8.24) holds.

3. The elements of M_I is to be calculated by comparing the effectiveness of the failed and the remaining surfaces and then adjusted using simulation tests.

Therefore the values of the reconfiguration matrices for the failures considered are as given below:

$$M_{I,\eta_r} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ m_1 & 0 & 0 & 1 & 0 & 0 \\ -m_1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}; \quad M_{I,\xi_r} = \begin{bmatrix} 1 & 0 & 0 & -m_2 & 0 & 0 \\ 0 & 1 & 0 & -m_2 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 1 & 0 \\ 0 & 0 & 0 & m_3 & 0 & 1 \end{bmatrix};$$

$$M_{I,\zeta} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & -m_4 \\ 0 & 1 & 0 & 0 & 0 & m_4 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & m_5 \\ 0 & 0 & 0 & 0 & 1 & -m_5 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}; \quad M_{I,\xi_r+\xi_l} = \begin{bmatrix} 1 & 0 & 0 & m_6 & 0 & 0 \\ 0 & 1 & 0 & 0 & m_6 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix};$$

$$M_{I,\xi_r+\eta_l} = \begin{bmatrix} 1 & 1 & 0 & -2m_2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2m_1 & 0 & -1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}; M_{I,\eta_r+\eta_l} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ -m_7 & 0 & 0 & 1 & 0 & 0 \\ 0 & -m_7 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

Where the m_i elements are calculated in the following way:

1. η_r failure:

   (a) The new gains of the remaining elevator ($\bar{g}_{\eta_l}$) must be doubled, that is

   $$\bar{g}_{\eta_l} = g_{\eta_l} + g_{\eta_r}.$$

(b) The roll effectiveness of the left elevator is to be cancelled using the ailerons, therefore

$$\begin{aligned} \bar{g}_{\xi_r} &= g_{\xi_r} + m_1 g_{\eta_r}; \\ \bar{g}_{\xi_l} &= g_{\xi_l} - m_1 g_{\eta_r}; \end{aligned}$$

where

$$m_1 = \frac{\text{Roll effectiveness of } \eta_r}{\text{Roll effectiveness of ailerons}} = \frac{3}{24.4} = 0.123.$$

However simulation tests show that a better performance can be achieved using $m_1 = 0.02$.

(c) The yaw effictiveness of η_l is neglected.

2. ξ_r failure:

(a) The gains of the remaining aileron are doubled

$$\bar{g}_{\xi_l} = g_{\xi_l} - 1 \times g_{\xi_r}.$$

(b) The pitch effectiveness of ξ_l is to be cancelled by the elevators, therefore

$$\begin{aligned} \bar{g}_{\eta_r} &= g_{\eta_r} - m_2 g_{\xi_r}; \\ \bar{g}_{\eta_l} &= g_{\eta_l} - m_2 g_{\xi_r}; \\ m_2 &= \frac{\text{Pitch effectiveness of } \xi_r}{\text{Pitch effectiveness of elevators}} = \frac{0.7}{11.6} = 0.06. \end{aligned}$$

(c) The yaw effectiveness of ξ_l is to be cancelled by the rudder, therefore

$$\begin{aligned} \bar{g}_{\zeta} &= g_{\zeta} + m_3 g_{\xi_r}; \\ m_3 &= \frac{\text{Yaw effectiveness of } \xi_r}{\text{Yaw effectiveness of rudder}} = \frac{0.1}{2.04} = 0.05. \end{aligned}$$

3. ζ failure:

There is no other rudder surface, therefore the rudder gains are to be distributed to the ailerons and the elevators to produce a bank angle such that the

starboard weight component produces the necessary yawing moment. therefore

$$\begin{aligned}
\bar{g}_{\eta_r} &= g_{\eta_r} - m_4 g_\zeta; \\
\bar{g}_{\eta_l} &= g_{\eta_l} + m_4 g_\zeta; \\
\bar{g}_{\xi_r} &= g_{\xi_r} + m_5 g_\zeta; \\
\bar{g}_{\xi_l} &= g_{\xi_l} - m_5 g_\zeta; \\
m_4 &= \frac{\text{Yaw effectiveness of } \zeta}{\text{Yaw effectiveness of elevators}} = \frac{2.04}{0.1} = 20; \\
m_5 &= \frac{\text{Yaw effectiveness of } \zeta}{\text{Yaw effectiveness of ailerons}} = \frac{2.04}{0.2} = 10.
\end{aligned}$$

Simulation tests show that a better performance can be achieved using $m_4 = m_5 = 1$.

4. $(\xi_r + \xi_l)$ failure:

   The gains of the failed ailerons are to be given to the elevators to enable them to work differentially to produce the required rolling moment. Therefore

$$\begin{aligned}
\bar{g}_{\eta_r} &= g_{\eta_r} + m_6 g_{\xi_r}; \\
\bar{g}_{\eta_l} &= g_{\eta_l} + m_6 g_{\xi_l}; \\
m_6 &= \frac{\text{Roll effectiveness of one aileron}}{\text{Roll effectiveness of one elevator}} = \frac{12.2}{3} = 4.06.
\end{aligned}$$

5. $(\xi_r + \eta_l)$ failure:

   (a) The gains of the remaining aileron and elevator are to be doubled.

   (b) The pitch effectiveness of the left aileron and the roll effectiveness of the right elevator have to be cancelled. Therefore

$$\begin{aligned}
\bar{g}_{\xi_l} &= g_{\xi_l} - 1 \times g_{\xi_r} + 2 m_1 g_{\eta_l}; \\
\bar{g}_{\eta_r} &= g_{\eta_r} + g_{\eta_l} - 2 m_2 g_{\xi_r}.
\end{aligned}$$

6. $(\eta_r + \eta_l)$ failure:

   The gains of the failed elevators are to be given to the ailerons to enable them

to work collectively to produce the required pitching moment.

$$\begin{aligned}
\bar{g}_{\xi_r} &= g_{\xi_r} - m_7 g_{\eta_r}; \\
\bar{g}_{\xi_l} &= g_{\xi_l} - m_7 g_{\eta_l}; \\
m_7 &= \frac{\text{Pitch effectiveness of one elevator}}{\text{Pitch effectiveness of one aileron}} = \frac{5.8}{0.7} = 8.3.
\end{aligned}$$

8.5 Simulation Results

To achieve real-time performance the above fault-tolerant design can be implemented on the transputer network shown in Figure 8.4. Transputer P_0 is the aircraft simula-

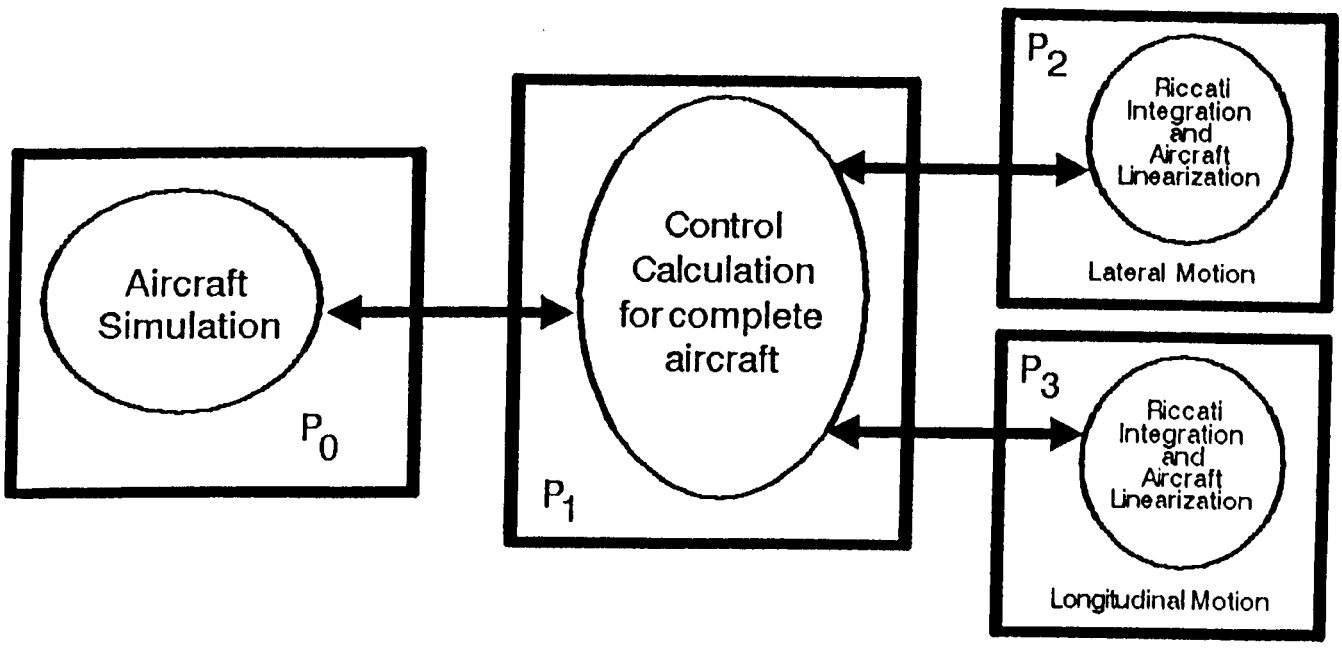

Figure 8.4: Transputer network for aircraft control

tor. Transputer P_1 handles the control tasks, that is, it provides ailerons, elevators, rudder and engine speed controls, the failure detection and isolation task, and the reconfiguration of the feedback gains according to the failure. P_1 receives the longitudinal and lateral feedback gains as well as the elements of the filter matrices ($\bar{A}$, $\bar{B}$, and $\bar{D}$) from transputers P_2 and P_3 at intervals of 295 ms and 190 ms respectively.

Transputer P_2 performs the aircraft linearization and solves the Riccati equation for the longitudinal motion and supplies P_1 with the longitudinal gains and some of the elements in $\left(\bar{A}, \bar{B}, \bar{D}\right)$. Transputer P_3 performs the same operations as P_2 but for the lateral motion. This is when a sampling rate of 200 Hz with a horizon length of 5 seconds is used. Table 8.1 shows the execution times taken for the different tasks performed by transputer P_1 for the linear time-varying filter and the nonlinear filter cases; this shows that the total time is well below the sampling period of 5 ms. The

Table 8.1: Transputer P_1 timings

Computational task	Execution time μs	
	Linear filter	Nonlinear filter
Control calculations	728	728
Fault detection + isolation	326	950
Gains reconfiguration	402	402
Communications with T_0	187	187
Communications with T_2	385	286
Communications with T_3	451	297
Minimum total time	1241	1865
Maximum total time	2094	2564

desired aircraft states are assumed to be

$$X_d = [150,\ 5,\ 0,\ 0.03,\ 5000,\ 6605,\ 0,\ 0,\ 0,\ 0,\ 0]^T$$

Six different cases were considered, namely,

1. an error of 100 m in the height and the right elevator fails (see Figure 8.5 and 8.6);

2. an error of 0.6 rad in the roll angle and the right aileron fails (see Figure 8.7 and 8.8);

3. an error of 0.2 rad in the yaw angle and the rudder fails (see Figure 8.9 and 8.10);

4. an error of 0.6 rad in the roll angle and the two ailerons failed (see Figure 8.11 and 8.12);

5. an error of 0.6 rad in the roll angle and the right aileron together with the left elevator failed (see Figure 8.13 and 8.14); and

6. an error of 100 m in the height and the two elevators failed (which resulted in an unstable aircraft).

As shown in Figures (8.5) to (8.14) good results are possible for the one aileron, one elevator, two ailerons, and an aileron + an elevator failures. The impaired and unimpaired responses are seen to be similar, indicating that the aircraft can be controlled adequately under such failures. For the rudder failure, good results are obtained for the manoeuvre shown; however, the remaining surfaces are not sufficient to regulate large errors in the yaw rate using the side-force produced by rolling the aircraft and so another rudder is ideally required to adequately control the aircraft. The same may be said about the two elevators failure, where simulation test shows that the ailerons do not have enough spare power to replace the elevators because there pitch effectiveness is very small when compared with the elevators. Also they adversely effect the vertical velocity (when used collectively) which produces large angles of attack leading to instability. This can clearly seen from equation (8.32) where the effectiveness of the aileron to the vertical velocity (9.8) is quite high comparatively.

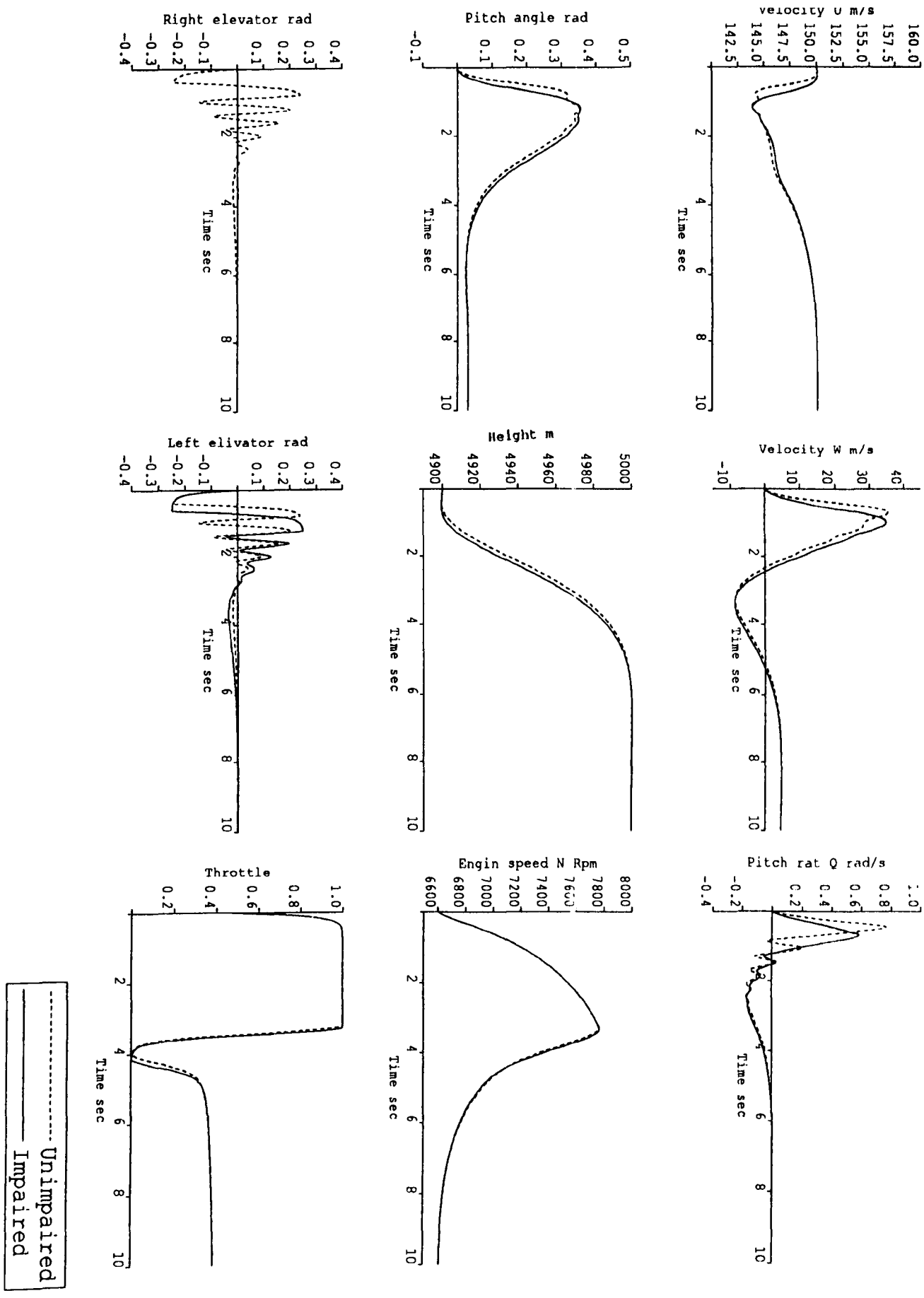

Figure 8.5: Right elevator failure (longitudinal)

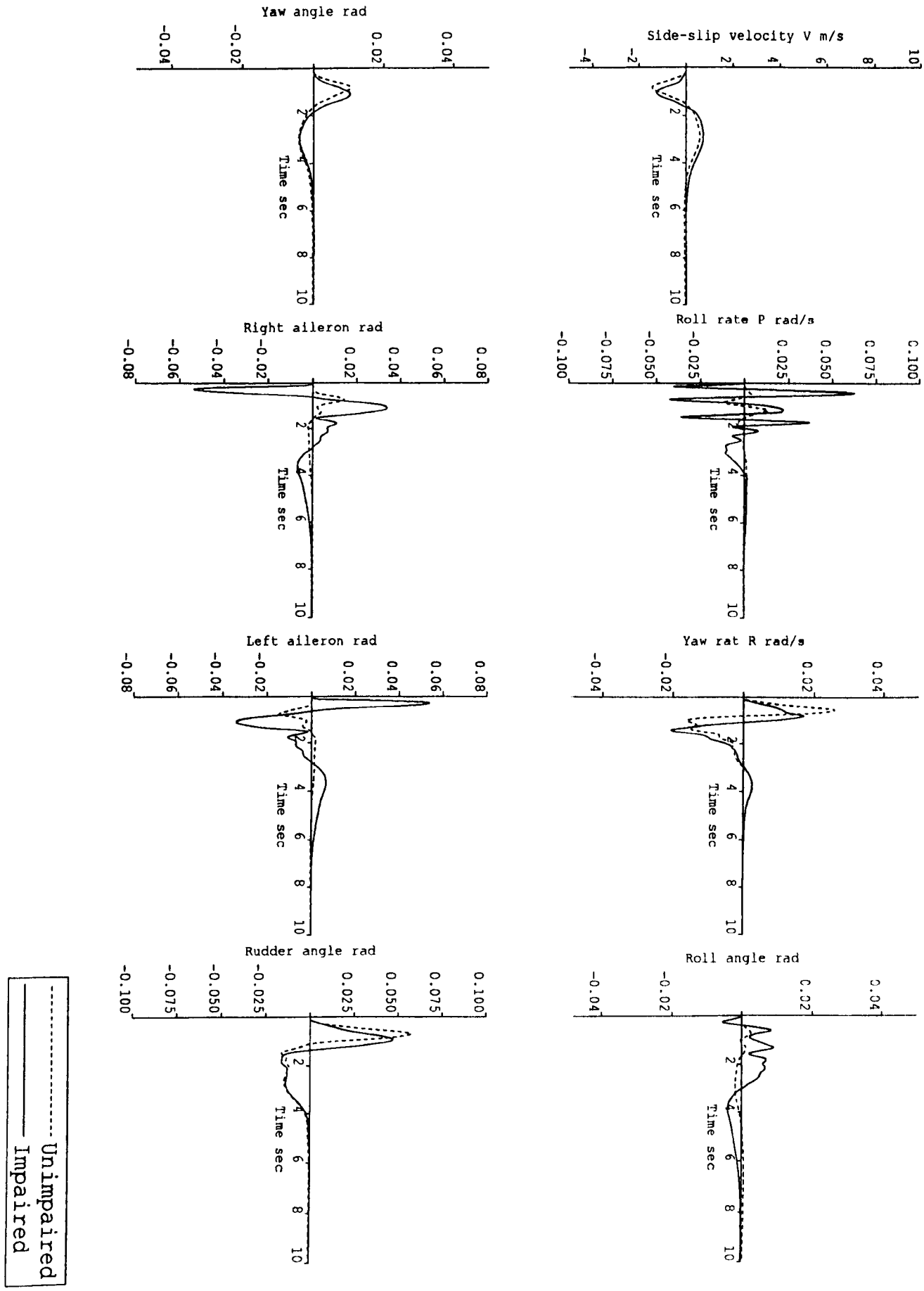

Figure 8.6: Right elevator failure (lateral)

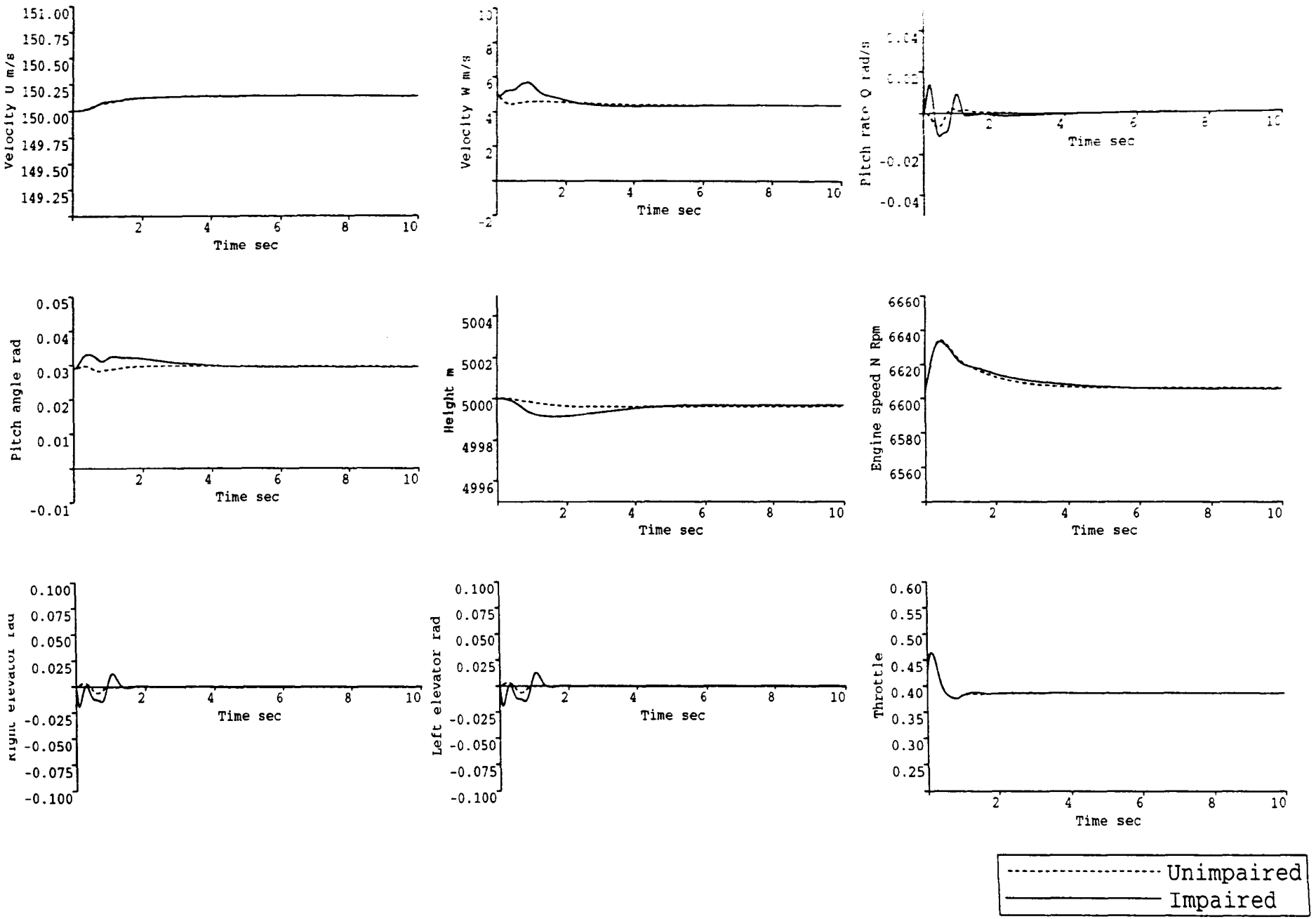

Figure 8.7: Right aileron failure (longitudinal)

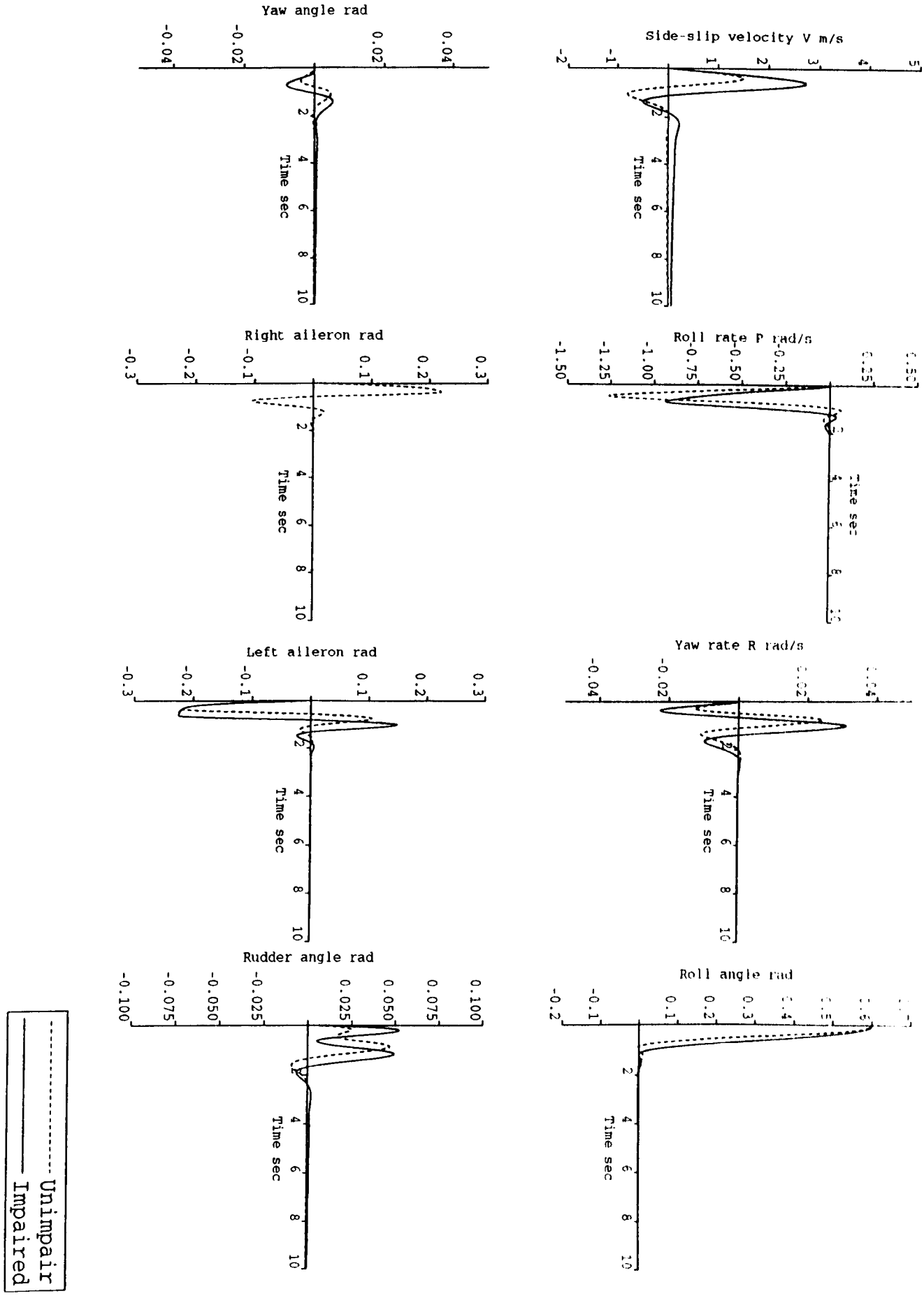

Figure 8.8: Right aileron failure (lateral)

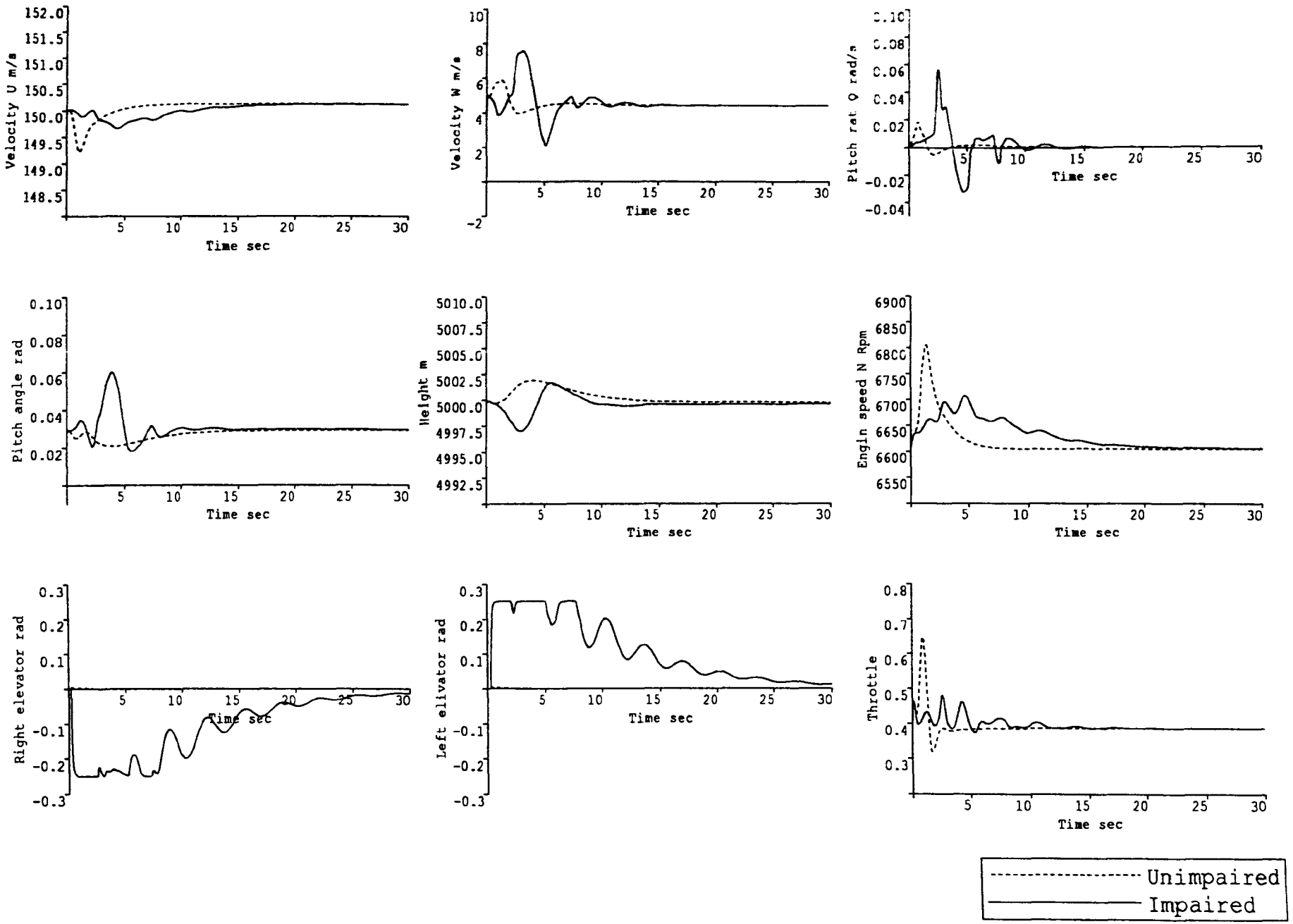

Figure 8.9: Rudder failure (longitudinal)

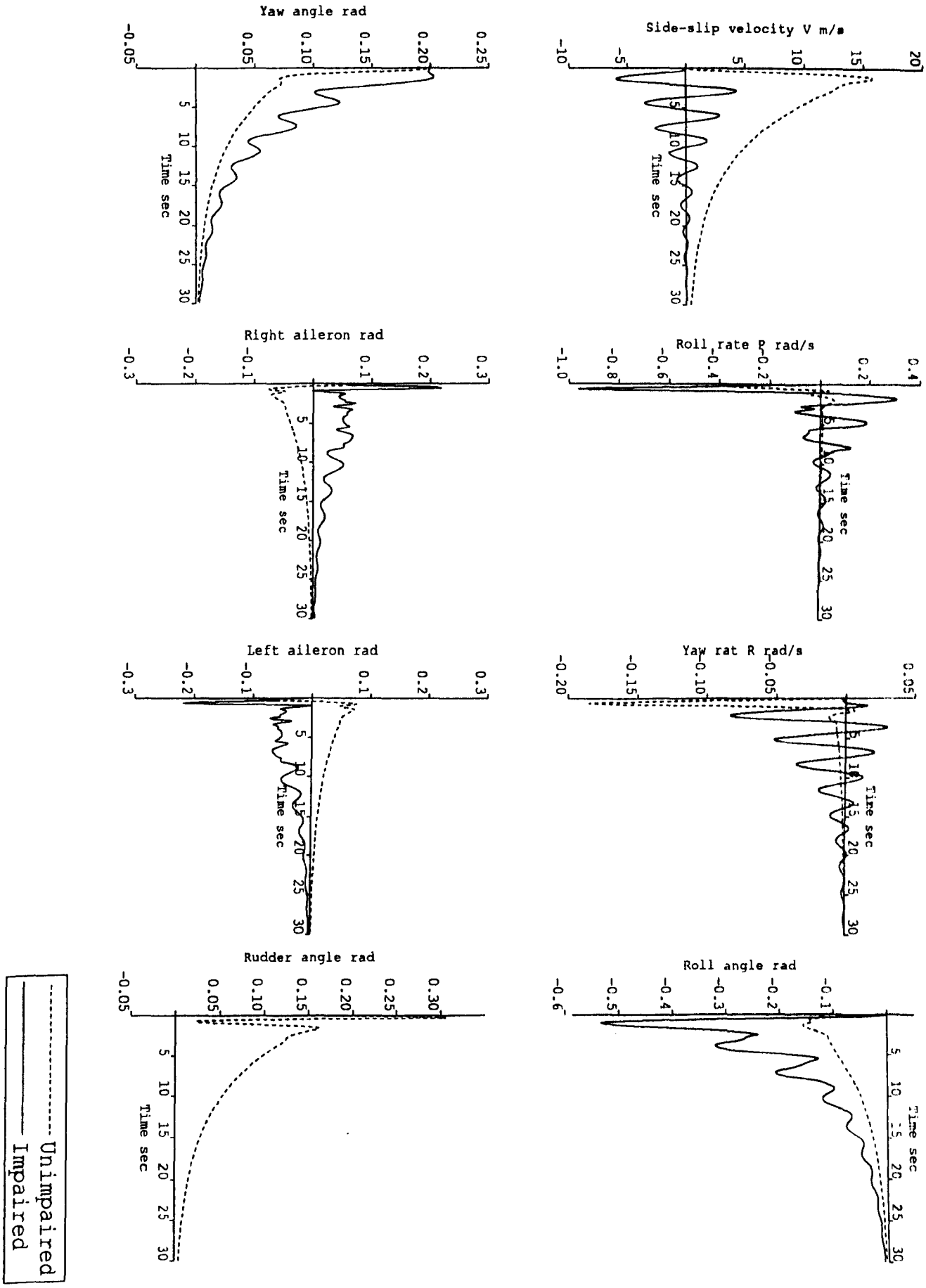

Figure 8.10: Rudder failure (lateral)

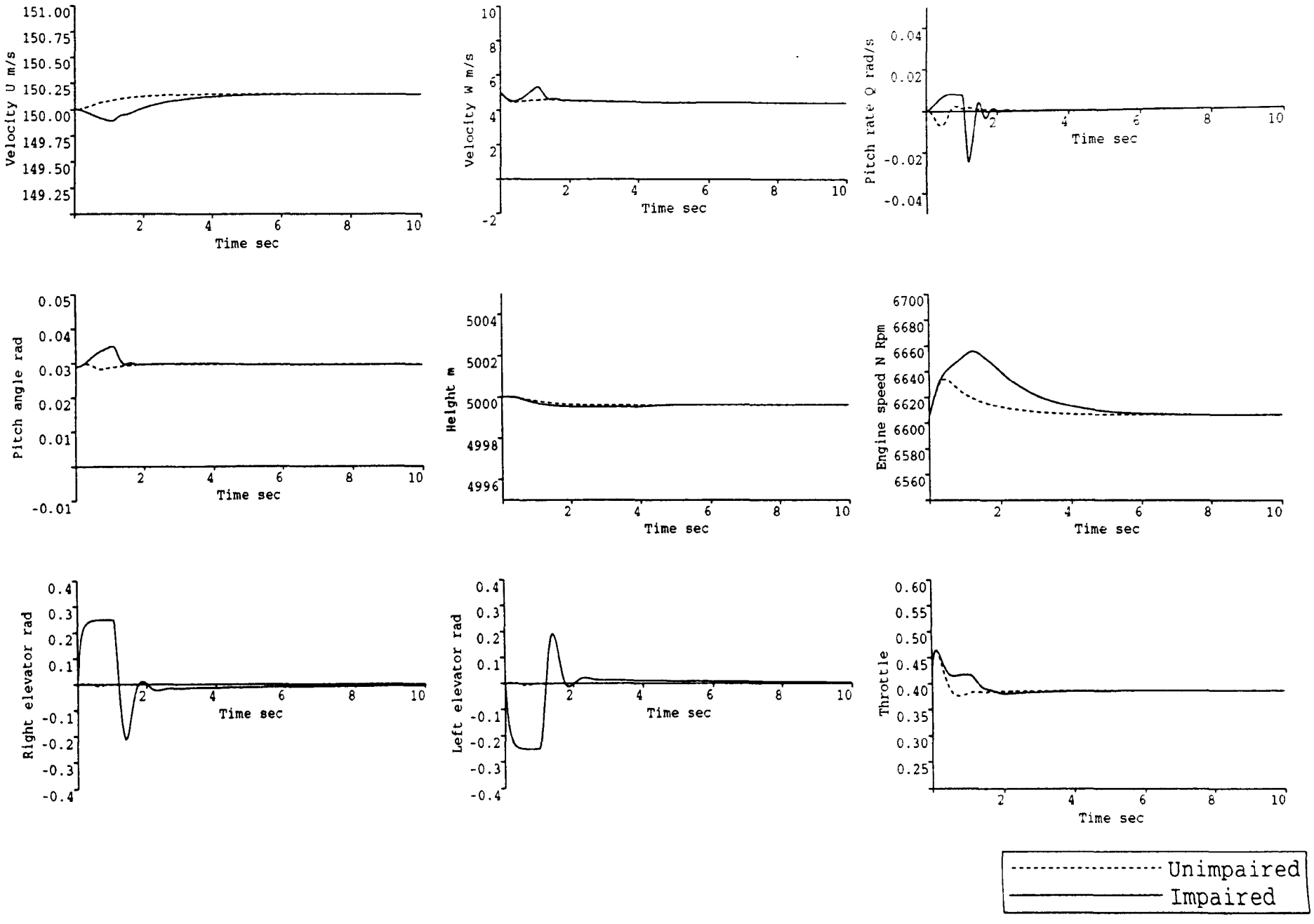

Figure 8.11: Two ailerons failure (longitudinal)

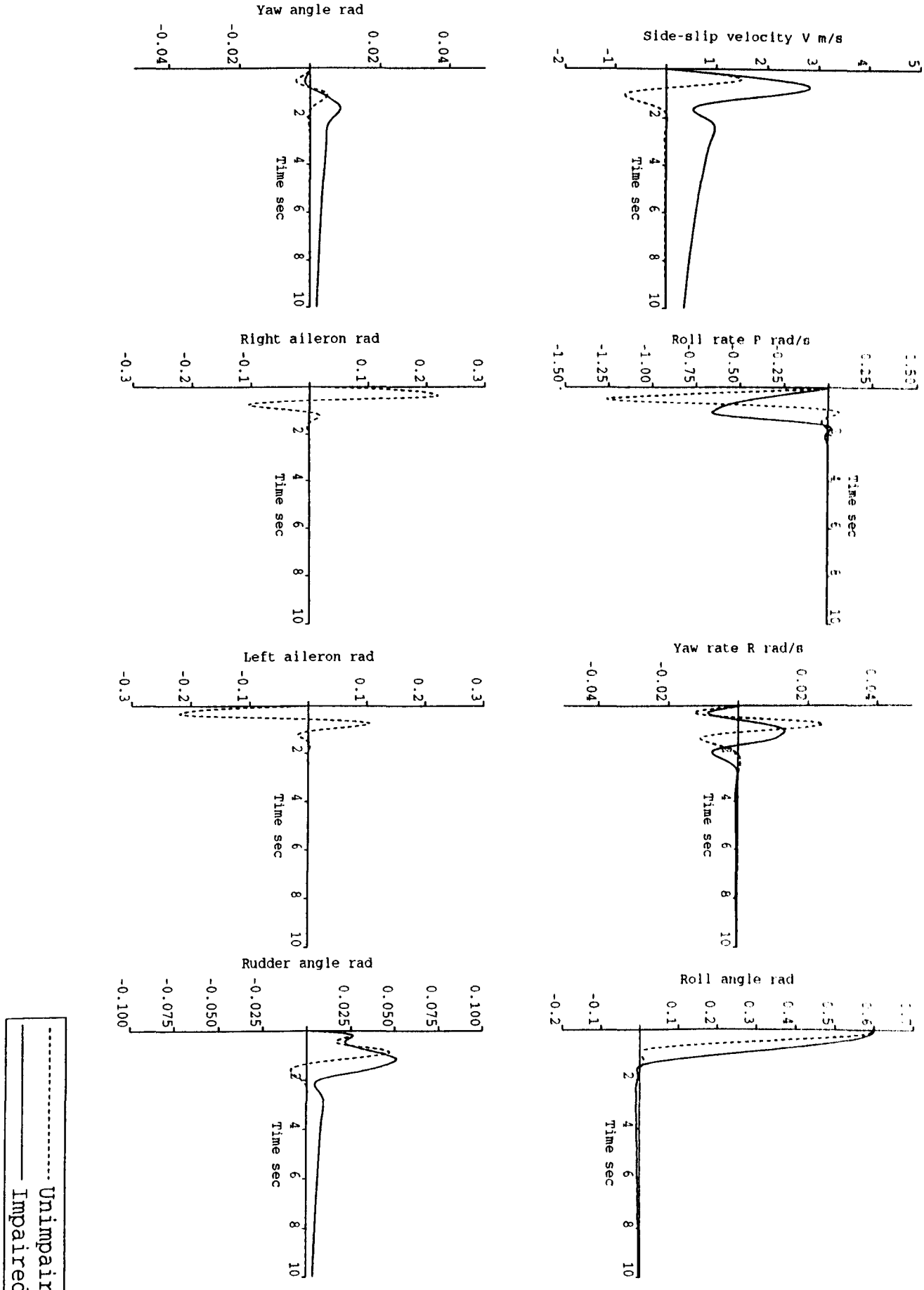

Figure 8.12: Two ailerons failure (lateral)

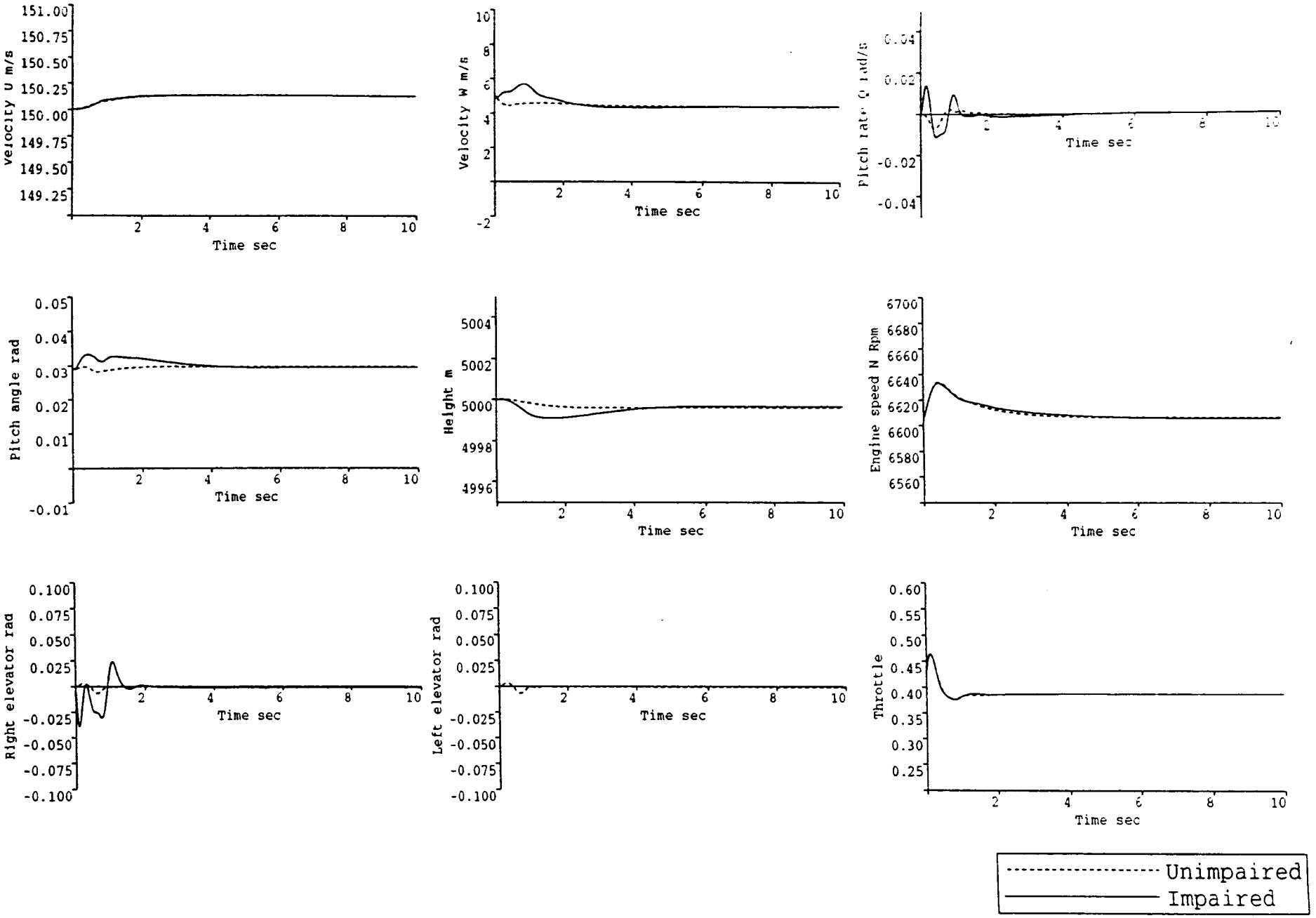

Figure 8.13: An aileron + elevator failure (longitudinal)

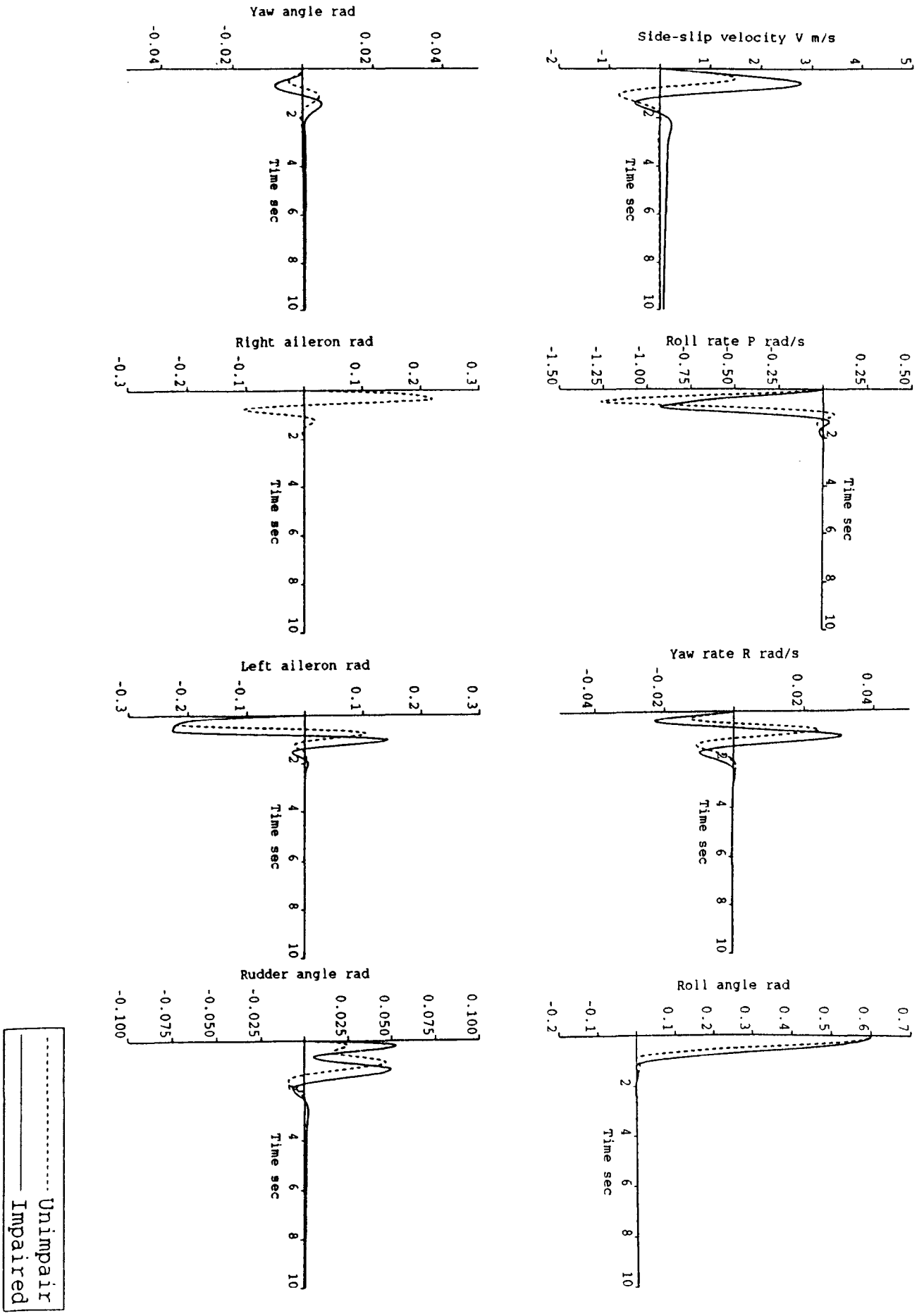

Figure 8.14: An aileron + elevator failure (lateral)

Chapter 9

Conclusion and Future Work

The main aims of the research presented in this thesis has been to investigate the design of optimal controllers for nonlinear time-varying aircraft, and to design a reconfigurable control law which enables the aircraft to retain its flight performance in the presence of control failures caused by combat damage or actuator failures. The mathematical model of a typical fixed wing aircraft is used as the basis to demonstrate the design; this was used to decouple the model into two smaller models, the longitudinal and lateral motions which can be treated as two subproblems with some cross-coupling effects.

Several approaches were used to solve the optimal control problem, for nonlinear time-varying systems but to no avail; the main reasons are that these approaches require excessive computational resources to provide real-time control. Therefore an alternative, namely the linear quadratic regulator (LQR), approach is used for a linear time-varying aircraft. This is also computationally demanding but by using parallel processing techniques it is demonstrated that suitable transputer architectures can be used to solve the problem. The solution fulfills the control objective but suffers from the following aspects:

1. The integration step sizes in the Riccati solution are restricted to equal the

sampling period; this increases the computation time for high sample rates.

2. A large memory capacity is needed to store the gain sequence for the complete horizon interval.

3. A relatively large communication time is required to transmit the updated gain elements from an "off-line" processor to the "on-line" controller processor.

In view of these, a receding horizon approach is used and found to be adequate for real-time aircraft control.

Due to the difficulties in formulating optimal control problems, a design procedure is presented which allows the weighting matrices to be selected for such problems. Some of the weighting elements in the matrices are time-varying to allow for good performances over the entire operating range of the aircraft. The stability of the Riccati solution and the time-varying problem are solved by using velocity dependent weights.

Several transputer networks are used to handle the "decoupled" aircraft motions with the cross-coupling effects of the two motions allowed for. Good real-time performances are achieved by splitting the computational task functionally into smaller sub-tasks which are processed on different processors, and the individual processing devices are configured to minimise the communication aspects of the application.

Another aspect of the work presented here is the fault tolerance nature of the approach. It is shown that reconfiguration of the flight control system is possible. Some results are presented where single and double control surfaces failures are introduced. These show that the remaining surfaces can compensate for the failures. However, such recovery can only be achieved with an aircraft which has been designed with other surfaces available and which have sufficient power. The failures can be detected and isolated using linear and nonlinear time-varying observer/filter

combined with digital logic. The nonlinear filter gives better performances but require more processing times than the linear filter.

It is hoped that the fault tolerant optimal flight control designs presented here is a useful addition to the area of control engineering. However the sign of good research is that it leads to further avenues for investigation. In this respect we have merely touched an a small aspect of the work which is necessary for the design of flight control systems. Other areas of work which have been exposed by our investigations include:

1. the design of controllers which respond to pilot demands (the "tracking problem");

2. the need to design a separate control loop which sets the angle of the stabiliser to achieve open-loop stability for the entire working range of the aircraft. In the current work this is assumed to be operated manually. This could be an important feature in the fault tolerance and reconfiguration strategy since it allows a secondary control input force for the elevators;

3. the need to consider the effect of disturbances on the failure detection filter such that robust design can be achieved;

4. the consideration of other types of failure, such as, "on" failures and computer failures; and

5. the consideration of different modelling and control strategies.

It is hoped that future work can consider such issues and that the work presented here helps in this respect.

Bibliography

[1] Anderson T, and Lee P A, Fault tolerance principles and practice, Prentec-Hall 1981.

[2] Babister A W, Aircraft dynamic stability and response, Pergamon press 1980.

[3] Balakrishnan A V, Control theory and the calculaus of variations, Academic press 1969.

[4] Banks S P, Control systems engineering, Prentic-Hall 1986.

[5] Bennett S, Real-time computer control, Prentice-Hall, London, 1988.

[6] Bertsekas D P, and Tsitsiklis J N, Parallel and distributed computations, numerical methods, Prentice-Hall 1989.

[7] Blakelock J H, Automatic control of aircraft and missiles, John Wiley & sons 1965.

[8] British aerospace aircraft engineering data, (private correspondence).

[9] British standard aerospace series, B.S.2G 199, 1984.

[10] Bryson A E, and Ho Y C, Applied optimal control, Ginn and Co, 1969.

[11] Calise A J, Corban J E, and Flandro G A, Trajectory optimisation and guidance law development for national aerospace plane applications, American control conferance, Atlanta, Georgia, June 15-17, 1988.

[12] Chang T, Jin X, and Luh P B, A parallel algorithm for large scale convex optimal control problems, proceedings of 1987 American control conferance, Minneapolis USA, 10-12 June 1986.

[13] Chakravarty A, Four-dimensional fuel-optimal guidance in the presence of winds, J. Guidance, vol. 8, no. 1, Jan.-Feb. 1985.

[14] Chazan D, and Miranker W L, A nongradient and parallel algorithm for unconstrained minimisation, SIAM J. control, vol. 8, no. 2, May 1970.

[15] Dixon L C W, Adjoint-control transformations for solving practical optimal control problems, optimal control applications & methods, vol. 2, pp 365-381, 1981.

[16] Dyer P, The computation and theory of optimal control, Acadenic press 1970.

[17] Ellert F J, and Merriam C W, Synthesis of feedback controls using optimisation theory - an example, IEEE Trans. on automatic control 1963.

[18] Fel'dbaum A A, Optimal control systems, Academic press 1965.

[19] Franklin M A, Parallel solution of ordinary differential equations, IEEE Trans. C-27(5), pp. 413-420, 1978.

[20] Fu S J, Aircraft guidance for formation flying based on optimal control, proceedings of the American control conferance, Minneapolis USA, 10-12 June 1987.

[21] Grimm W, A numerical approach for on-line guidance of aircraft, proceedings of 25^{th} IEEE conferance on decision and control, Athens, Greece, 10-12 Dec. 1986.

[22] Hasdorff L, Gradient optimisation and nonlinear control, John Wiley & sons 1969.

[23] Hockney R W and Jesshope C R, Parallel computers 2: Architecture programming and algorithms, Adams Hilger, Bristol, 1988.

[24] Howard R W, Automatic flight controls in fixed wing aircraft - the first 100 years, Aeronatical journal, Nov. 1973.

[25] Hwang K, and Briggs F A, Computer architecture and parallel processing, Mcgraw-Hill 1984.

[26] Illinois R B, and Bielefeld V M, Symmetric updating of the solution of the algebraic Riccati equation, methods operation research, vol. 54, pp. 117-25, 1986.

[27] INMOS Ltd, IMS T800 transputer engineering data, 1980.

[28] Keller H B, Numerical solution of two-point boundary value problems, society for industrial and applied mathematics, Philadelphia, Pennsylvania 19103, vol. 24, 1976.

[29] Kwon W H, Bruckstein A M, and Kailath T, Stablising state-feedback design via the moving horizon method.

[30] Lee R C K, Optimal estimation, identification, and control, M.I.T. press 1964.

[31] Lee E B, and Markus L, Foundations of optimal control theory, John Wiley & sons 1967.

[32] McLean D, and Aslam-Mir S, Reconfigurable flight control system, International conference on Control'91, vol. 1, pp. 234-242, 25-28 March 1991.

[33] Menon P K A, and Lehman L L, A parallel quasi-linearization algorithm for air vehicle trajectory optimisation, J. Guidance, vol. 9, no. 1, Jan.-Feb. 1986.

[34] Miranker W L, and Liniger W M, Parallel methods for the numerical integration of ordinary differential equations, J. Math. Comput. vol. 21, pp. 303-320, 1967.

[35] Mitchell D A P, Thompson J A, Manson G A, and Brookes G R, Inside the transputer, Blackwell scientific publications 1990.

[36] Miele A, Well K H, and Tietze J L, Multipoint approach to the two-point boundary value problem, Journal of mathematical analysis and applications, vol. 44, pp. 625-642, 1973.

[37] Miele A, Flight mechanics: Theory of flight path, Pergamon press 1962.

[38] Pau L F, Failure diagnosis and performance monitoring, Marcel Dekker inc 1981.

[39] Noble B, Applied linear algebra, Prentice-Hall, 1969.

[40] Patton P C, Multi-processors: Architectures and applications, computer, vol. 18, no. 6, pp. 29-40, 1985.

[41] Patton R, Frank P, and Robert C, Fault diagnosis in dynamic systems, Prentice-Hall 1989.

[42] Polak E, Computational methods in optimisation, Academic press 1971.

[43] Purdum J, C programming guide, Que corporation 1988.

[44] Raczynski S, On parallel algorithm for real-time optimal control problems, proceedings of the 1986 American control conference, Seattle, USA, 18-20 June 1986.

[45] Ratton K S, Evaluation of control-mixer concept for reconfiguration of flight control systems, NAECON, vol. 2, pp. 560-569, 1985.

[46] Ratton K S, Reconfiguration of flight control systems after effector failure , proc. of fourth international conferance on systems Engineering, Coventry Polytechnic, 1985.

[47] Richardson T J, and Kwong R H, On positive definite solution to the algebraic Riccati equation, systems & control letters, vol. 7, pp. 99-104, 1986.

[48] Roberts S M, and Shipman J S, Two-point boundary value problems: shooting methods, Elsevier 1972.

[49] Russ D E, Reconfigurable digital control laws for the 7D Digitac II aircraft with failed primary control surface, proc. of workshop on multivariable control systems, 1983.

[50] Shaw L, Nonlinear control of linear multivariable systems via state-dependent feedback gains, IEEE trans. control, vol. 24, no. 1, feb. 1979.

[51] Takahashi Y, Rabins M J, and Auslander D M, Control and dynamic systems, Addison-Wesley publishing company 1965.

[52] Thomas Y A, Linear quadratic optimal estimation and control with receding horizon, electronic letters, vol. 11, no. 1, 9^{th} jan. 1975.

[53] Travassos R, and Kaufman H, Parallel algorithms for solving nonlinear two-point boundary value problems which arise in optimal control, Journal of optimisation theory and applications, vol. 30, no. 1, Jan. 1980.

[54] Widnall W S, Applications of optimal control theory to computer controller design, M.I.T. press 1968.

[55] Willsky A S, A survey of design methods for failure detection in dynamic systems, Automatica, vol. 12, pp. 601-611, 1976.

Appendix A

Publications

1. Tahir J,M and Virk G S, A real-time distributed algorithm for an aircraft longitudinal optimal autopilot, Concurrancy: Practice and Experience, vol 2(2), pp 109-121, 1990.

2. Virk G S and Tahir J M, Parallel optimal control algorithms for Aircraft, IEE Colloquium on navigation, guidance and control in aerospace. Digest No.1989/142, 3/1 - 3/5, November 1989.

3. Virk G S, Tahir J M, and Kourmoulis P K, Parallel processing in aerospace control system, proceedings of the second international conference on applications of transputers,11-13 july 1990, Southampton UK.

4. Virk G S and Tahir J M, Selection of weights in time-varying optimal flight, submitted to Optimal Control Applications and Methods, 1991.

5. Virk G S and Tahir J M, A fault tolerant optimal flight control system, IEE International Conference 'Control 91 ',vol. 2, pp. 1049-1055, 25-28 March 1991, Heriot-Watt university, Edinburgh, UK.

6. Virk G S and Tahir J M, The design of optimal controller for aircraft, IEE

Colloquium on integrating issues in aerospace control, Digest No. 1991/072, 4 April 1991.

7. Virk G S and Tahir J M, Parallel processing for real-time flight control, invited chapter in Transputer Control, Research Studies Press Ltd, to be published.

Appendix B

Table of Notations

U	m/sec	forward component of aircraft velocity
W	m/sec	downward component of aircraft velocity
V	m/sec	velocity of side-slip
V_r	m/sec	aircraft relative velocity
Q	rad/sec	pitch rate
P	rad/sec	roll rate
R	rad/sec	yaw rate
Θ	rad	pitch attitude angle
Φ	rad	roll attitude angle
Ψ	rad	yaw attitude angle
H	metres	aircraft height
N	rev/min	engine speed
η	rad	elevator deflection

ξ	rad	aileron deflection
ζ	rad	rudder deflection
γ	dimensionless	engine power setting
X_f	N	total force acting along OX axis
Y_f	N	total force acting along OY axis
Z_f	N	total force acting along OZ axis
P_m	Nm	total pitching moment
Y_m	Nm	total yawing moment
R_m	Nm	total rolling moment
M	kg	total aircraft mass
I_x	kgm^2	moment of inertia about OX axis
I_y	kgm^2	moment of inertia about OY axis
I_z	kgm^2	moment of inertia about OZ axis
I_{xz}	kgm^2	the cross product of inertia about OZX axes
T_s	ms	sampling period
T_l	ms	time between successive linearizations
T	sec	horizon depth
x		vector of state variables
x_d		vector of desired states
u_{in}		vector of control variables
α	rad	angle of attack