QUALITATIVE SIMULATION:
A Tool for Global Decision Making

by

Pentti Juhani Vesanterä

A Thesis Submitted to the Faculty of the

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

In Partial Fulfillment of the Requirements

For the Degree of

MASTER OF SCIENCE

WITH A MAJOR IN ELECTRICAL ENGINEERING

In the Graduate College

THE UNIVERSITY OF ARIZONA

1 9 8 8

STATEMENT BY AUTHOR

This thesis has been submitted in partial fulfillment of requirements for an advanced degree at The University of Arizona and is deposited in the University Library to be made available to borrowers under rules of the Library.

Brief quotations from this thesis are allowable without special permission, provided that accurate ackowledgement of source is made. Requests for permission for extended quotation from or reproduction of this manuscript in whole or in part may be granted by the head of the major department or the Dean of the Graduate College when in his or her judgement the proposed use of the material is in the interests of scholarship. In all other instances, however, permission must be obtained from the author.

SIGNED: [illegible]

APPROVAL BY THESIS DIRECTOR

This thesis has been approved on the date shown below:

Dr. François E. Cellier — December 8, 1988

F. E. Cellier
Associate Professor of
Electrical and Computer Engineering

Date

ACKNOWLEDGEMENTS

I wish to express my gratitude to my parents, Mrs. Armi Vesanterä and Mr. Pekka Vesanterä for their continual encouragement and support of my educational advances.

I also wish to gratefully acknowledge the most valuable guidance of Dr. François E. Cellier, my major advisor and thesis director, whose patience and unlimited support enabled the completion of this thesis. Acknowledged should also be the interest and presence of the committee members: Dr. Hal S. Tharp and Dr. Jerzy W. Rozenblit, and the kind help of Dr. Edwin K. Parks, professor emeritus at the Aeronautical and Mechanical Engineering with the parameters for the mathematical model.

Finally I would like to thank my dearest friend Hessam Sarjoughian for the most helpful discussions, comments, and friendship throughout these back-to-school years.

TABLE OF CONTENTS

Table of Contents–(Continued_

LIST OF ILLUSTRATIONS

ABSTRACT

As a decision making aid for the human operator of a highly automated, complex system, qualitative modeling is presented as a tool to mimic the human global assessment process by learning from the system behavior. Such a qualitative model is applied to reason about the behavior of a quantitatively simulated aircraft model, to determine on-line when a malfunction occurs in the quantitative model, to hypothesize about the nature of this malfunction, and to suggest a global strategy that would allow to operate (control) the quantitative aircraft model under the modified flying conditions. Such an algorithm could be utilized as an addition to a conventional autopilot which would allow it to remain operational after a malfunction has taken place.

CHAPTER 1
INTRODUCTION

With the continuous advances of technology in the area of automatic control, automation has become increasingly popular allowing us today to build highly complex control systems e.g. for modern jet cruisers, nuclear power plants, space stations, etc. Modern control technology allows systems analysis and control through a variety of techniques in the time- or frequency-domain. Lead-lag compensators, and state- and output-feedback designed by means of techniques such as pole-placement or parameter optimization by solving a matrix Riccati equation represent some of the methodologies presently available. These are all well understood and widely in use. Decision making *in clearly predefined situations* has also been successfully implemented with decentralized control systems, expert systems and rule-based control systems.

Even though all these techniques work perfectly well under normal, well defined conditions, they will drastically fail when facing a new, unforeseen (possibly emergency) situation such as a sudden, unexpected structural change in the system. Handling such a situation is a task that still has to be performed by a human operator with his/her inventiveness and capability of reasoning. The human mind has the skill of learning from the system behavior in the emergency, and the inventiveness of finding a new control strategy for the new situation. No automatic control technique presently available is able to adapt to unpredicted structural changes in the system.

Man-in-the-loop systems have been the answer to this problem until now. However when the degree of system complexity increases even further, human operators are being overloaded by the amount of information they are provided with. The human mind is incapable of taking many decisions instantaneously and simultaneously when these decisions are to be based on information arriving in huge bundles all at once. The amount of data to be processed is much too large for a human operator to act reliably and efficiently. Among other reasons, this is due to the fact that he loses his notion of temporal precedence of the arriving data items in the fast flowing sea of information that is formed. This disables the operator from distinguishing between causes and effects as the emergency progresses, which is critical for his decision making process, and an erroneous decision becomes likely.

Concerned with the problem of overloading the human operator in a fast moving scenario and highly dynamic environment, different approaches have been taken to try to build decision making aids for the human operator.

For example, to allow adaptability to new situations, an expert system for control system design as described in (Trankle and Markosian, 1984; Trankle et al., 1986) would be a very useful tool to handle sudden structural changes in the system. "The expert planner incorporates the knowledge of a control system designer in selecting and applying mathematical algorithms from several branches of control theory such as Kalman filter design, proportional/integral/derivative control, and optimal state feedback." The functionality of the method has been demonstrated at hand of a command tracking control system for an aircraft.

A fault monitoring and diagnosis expert system is described in (Ali and Sharnhorst, 1985; Ali et al., 1986) to assist pilots in handling in-flight faults. The Flight Expert System (FLES) distinguishes between two different types of faults:

maladjustments and malfunctions, and handles them separately. The knowledge representation is frame-based.

In (Cross, 1984), the author emphasizes the need of developing expert systems performing qualitative reasoning in the flight domain. An intelligent pilot aid would have the tasks of reducing excessive workload, reducing errors, improving performance, and adding new capabilities.

Structural models of human pilots performing tracking tasks have been discussed, for example in (Hess, 1984) and (Mooij, 1985). The models consider the interaction between the human central nervous system, the neuromuscular system, and the airplane, allowing in this way the modeling of the human reaction to, e.g., a deviation from a desired flight path.

A research study carried out at Perceptronics, Inc. is described in (Chu et al., 1980) concerning modeling goal-directed, recurrent decisions in an overly information-rich environment. This study defines an information value function based on an earlier study performed at Perceptronics as well, which developed a methodology for adaptive estimation of information value parameters, modeling techniques, and information selection behavior aiding techniques. The described information value function is a weighted synthesis of all decision related factors and can be directly used for management of information.

Another very interesting approach to solving the problem of overloading the human operator is discussed in (Riley, 1985). It consists of the implementation of a monitoring system for the human operator's performance to inform his electronic *teammate* about the states of the human operator so that it would be able to decide when and how to aid the operator and what information he or she needs at that moment. The main goals in "monitoring the monitor" (the human operator), are "to optimize operator workload and dynamically allocate tasks appropriately; keep the operator aware of the state of the world and the system; and facilitate fast appropriate or optimal response to system and environment changes".

Human decision making models have been developed for specific tasks as e.g. described in (Kleinman et al., 1983) for use in emergency state power distribution systems. An analytic model of a human decision making process is presented in (Morgan, 1983; 1985) where the author examines also the means of interaction of the model with other decision makers.

We propose to address the stated problem by building an on-line monitoring system that mimics the human global assessment process, that identifies the emergency, learns the system behavior, and comes up inductively with a new control strategy for the structurally modified system. This on-line monitoring system will be an automatic device that merges the best of both worlds: the human inventiveness and the automatic controller's speed and systematism.

General Systems Problem Solver (GSPS) (Klir, 1985) is a methodological framework arising from General Systems Theory that allows the user to define and analyze types of systems problems. In this methodology, systems are defined through a hierarchically arranged set of epistemological subsystems. Forecasting and reconstruction analysis capabilities are two examples of the capabilities of the GSPS methodological tools. An on-line monitoring system can be implemented in the GSPS framework by using its inductive inference capability to imitate the human learning process. SAPS–II (Cellier and Yandell, 1987) is a software coded at the University of Arizona that performs the basic concepts of the GSPS framework. SAPS–II has been implemented as an application function library to the control systems design software Ctrl–C (Systems Control Technology, 1986). In common A.I. terminology, it can be said that SAPS–II employs Ctrl–C as an A.I. shell.

Our stratagem was successfully implemented in SAPS–II for a longitudinal model of an aircraft in cruise flight. An *intelligent* autopilot was implemented using the GSPS framework. It is able to successfully identify structural changes in the monitored system, and to learn enough about the new (unknown, modified, broken) system to come up with a forecasting model that satisfactorily predicts future behavior of this system.

The approach taken is depicted in Figure 1.1. below:

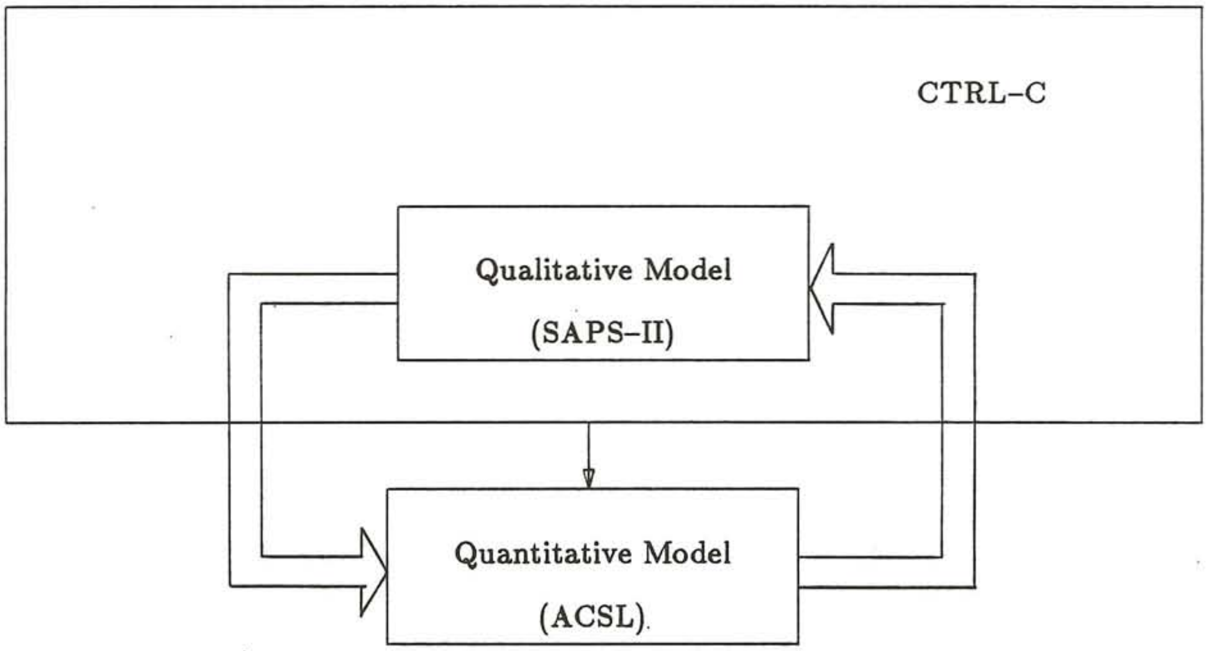

Figure 1.1 Block Diagram of the experimental setup

A *quantitative* model of the longitudinal behavior of the airplane was coded in the continuous system simulation language ACSL (Mitchell and Gauthier, 1986). The model can be exerted from within Ctrl–C through an interface between the two softwares (Systems Control Technology, 1986) which allows full use of an ACSL simulation. "Accidents" which alter the behavior of the flight are built into the aircraft model and the data extracted from the quantitative ACSL simulations are used as "measurement data" for the qualitative analysis of the system behavior using SAPS–II. SAPS–II functions are used to qualitatively and inductively reason about the measurement generated by the ACSL simulation run, determine that an accident has happened, find out when it occurred, hypothesize about the nature of the accident, and decide upon an appropriate corrective action to be taken.

CHAPTER 2
MODEL DESCRIPTION

2.1 Basic Physical Laws

Among all vehicles available for transportion today, aircrafts belong to a class that requires an essentially more complex form of control. Controlling motion in a three-dimensional space is certainly more complicated than the control of surface-bound or line-bound vehicles. Furthermore, modern aircrafts require increasingly sophisticated controls to fulfil the requirements of today's air transportation trend.

Flight stability can basically be studied through two independent models: longitudinal and lateral. Longitudinal motions can be modeled independently from the lateral ones if the following simplifying assumptions are valid:

1. The airplane is perfectly symmetrical with respect to its median longitudinal plane.
2. There are no gyroscopic effects of spinning masses (engine rotors, airscrews, etc) acting on the aircraft.

This text will adopt the assumptions above and will consider the longitudinal model of an airplane in cruise flight at high altitude. A longitudinal flight is characterized by the absence of forces and moments that would cause its lateral motion. Furthermore, the aeroelastic nature of the airplane's structure will be neglected as well, so that the *rigid body* equations of motion would apply to the model.

The mathematical model described in the following sections models an essentially longitudinal flight restricted to longitudinal deviations from a trimmed reference flight condition. This reference flight is characterized by the requirement that the resultant force and moment acting on the aircraft's center of mass are zero.

It is not the scope of this text to provide full details about the stability study of flight. To follow the reasoning process here presented, one may consult specialized literature from that area (Etkin, 1972; 1982; Hacker, 1970; Irving, 1966)

Motion of a rigid body

The forces and moments acting on the center of mass of a moving rigid body can be equated through the Newton's second law of motion, along with it's rotational analog presented by Euler. The equations below describe translational and rotational motions expressed in a space-fixed coordinate system.

$$\mathbf{F} = \frac{d}{dt}(m\mathbf{v}) \tag{2.1a}$$

$$\mathbf{M} = \frac{d}{dt}(\mathbf{I\Omega}) \tag{2.1b}$$

$\mathbf{F}$ is the resultant of the forces acting on the center of gravity (CG) of the body. The mass m of the body is assumed to be independent of time and concentrated in the center of gravity, and $\mathbf{v}$ is the linear velocity vector of the CG. $\mathbf{M}$ is the resultant torque moment acting on the CG, $\mathbf{I}$ is the inertial tensor of the body, and Ω is its angular velocity vector.

When transforming these equations into a body-fixed coordinate system, we have to consider the contribution of the rotation of this new reference frame with respect to the previous one.

$$\mathbf{F} = \frac{d}{dt}(m\mathbf{v}) + \Omega \times (m\mathbf{v}) \tag{2.2a}$$

$$\mathbf{M} = \frac{d}{dt}(\mathbf{I}\Omega) + \Omega \times (\mathbf{I}\Omega) \tag{2.2b}$$

When considering an essentially longitudinal flight, the resultant force **F** can be decomposed into its tangential component F_t and its normal component F_n. The only acting moment will be the one about the axis that is perpendicular to the longitudinal symmetry plane (call it y-axis), and the generalized equations of motion (2.2) can be written in the simpler form:

$$\mathbf{F} = F_t + F_n \tag{2.3a}$$

$$M_y = I_y \frac{d\Omega}{dt} \tag{2.3b}$$

The reference flight condition

We will define a reference flight condition as being characterized by a steady longitudinal and horizontal flight where the resultant force and moment acting on the plane are zero. The headwind is assumed to be constant and perfectly horizontal.

The stability axes

The theory presented in this text will be developed with respect to a set of body-fixed axes named stability axes. The origin of this coordinate system is the center of gravity of the airplane: the x-axis points in the direction of the motion of the airplane in the reference flight condition, the z-axis points 'downward,' and y-axis runs spanwise and points to the right.

The reference angles

Three angles are defined to describe the relative position of the velocity vector of the center of gravity of the airplane with respect to an earth-fixed reference frame and a fuselage-fixed reference frame.

α is the **angle of attack** (or incidence) of the airplane which describes the inclination of the resultant velocity vector **v** to the x axis of the body-fixed coordinate system. The usual notation of the velocity components in the stability axes is u for the x-axis component and w for the z-axis component. Hence

$$\alpha = tan^{-1}\left(\frac{w}{u}\right) \tag{2.4}$$

γ is the **flight path angle** of the aircraft, representing the inclination of the velocity vector to the horizontal, i.e., to the x-component of the earth-fixed reference frame.

θ is the **pitch angle**, being the one that is best sensed by a human pilot, for it represents the relative position between the two reference frames. It is defined in terms of the previous angles as follows:

$$\theta = \gamma + \alpha \tag{2.5}$$

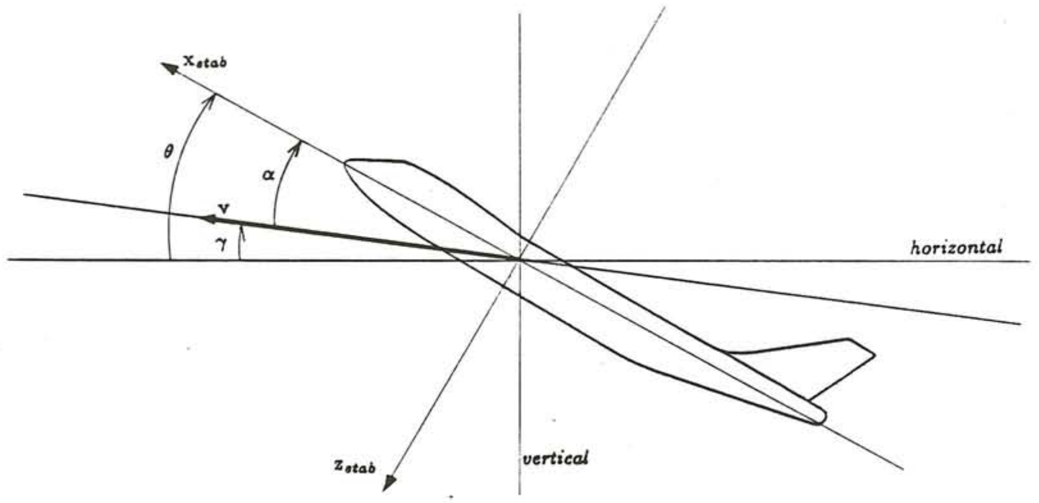

Figure 2.1 The reference angles

Forces and Moments

Finally, the tangential and normal components of the resultant force and the moment about the center of gravity of the airplane considered as a rigid body whose mass is constant over time, can be written in terms of the reference angles γ and θ as:

$$F_t = m\frac{d\mathrm{v}}{dt} \tag{2.6a}$$

$$F_n = m\mathrm{v}\frac{d\gamma}{dt} \tag{2.6b}$$

$$M_y = I_y\frac{d^2\theta}{dt^2} \tag{2.6c}$$

The quantities affecting the airplane in flight are its weight W, the thrust T developed by the engines, the aerodynamic forces, Lift L and Drag D, and the aerodynamic pitching moment M.

The weight of the aircraft will be considered constant (thus the weight of the fuel consumed during the flight is neglected).

The thrust developed by the propulsive system will be considered as being a function of the flight velocity and of its own control variable δ_T, the throttle opening. For reasons of simplicity, the thrust line will be assumed to coincide with the x-axis of the stability axes. The center of gravity, by definition, is in this axis and therefore the thrust does not affect the moment directly.

The aerodynamic forces L and D compose the force response of the aircraft to the motion. They act in the mean aerodynamic center of the wing, causing the aerodynamic moment M about the center of gravity, which is defined to be positive for a nose up effect. The Lift is defined as being the normal component of the aerodynamic force with respect to the flight path, and the Drag is its tangential component.

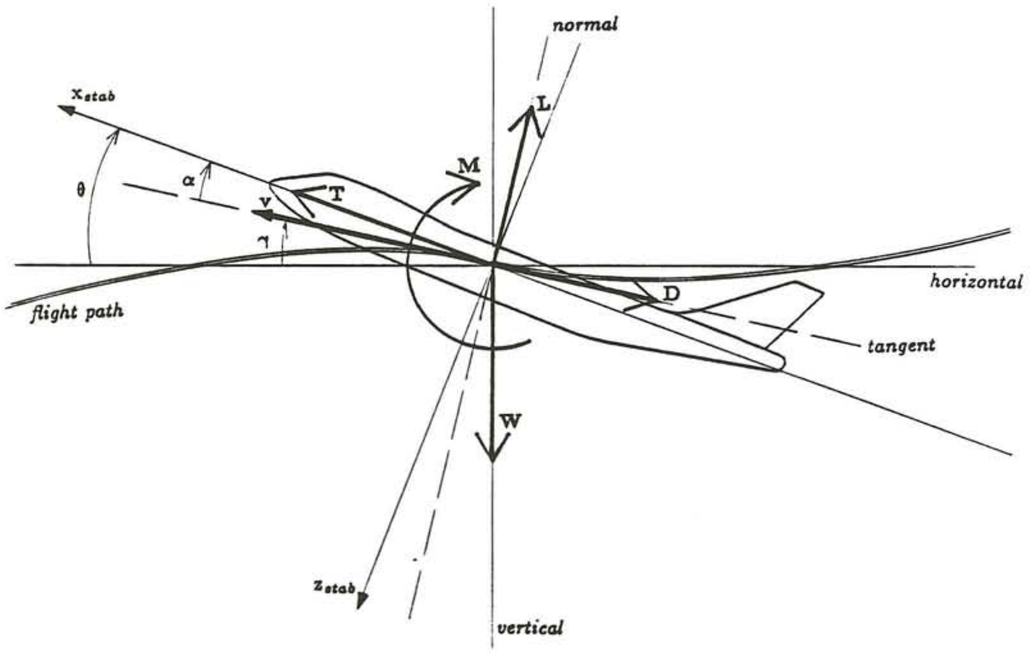

Figure 2.2 Forces and Moment acting on the airplane

The aerodynamic reactions L, D and M

The standard way of expressing the aerodynamic forces L and D and the longitudinal aerodynamic moment M is through their nondimensional aerodynamic coefficients C_L, C_D and C_M:

$$L = \frac{1}{2}\rho v^2 S C_L \tag{2.7a}$$

$$D = \frac{1}{2}\rho v^2 S C_D \tag{2.7b}$$

$$M = \frac{1}{2}\rho v^2 S \frac{\bar{c}}{2} C_M \tag{2.7c}$$

which shows their direct dependence on the local air density ρ, the square of the cruising speed v and the size of the aerodynamic surface S of the airplane.

Parameter $\frac{\bar{c}}{2}$ in the expression for the moment stands for the characteristic length (for the nondimensional coefficients), taken as half of the mean aerodynamic chord $\bar{c}$ of the wing.

The nondimensional coefficients C_L, C_D, and C_M

These three nondimensional coefficients express the aerodynamic response of the airplane to variations in the following aerodynamic variables:

1. α , the angle of attack
2. δ_e , the elevator deflexion
3. $\dot{\alpha}$, the angle of attack rate
4. q , the pitch rate

Equations (2.8*a*), (2.8*b*) and (2.8*c*) below describe the nondimensional aerodynamic coefficients expressed by a Taylor series expansion around an initial value (subscript 0) for which α, δ_e, $\dot{\alpha}$ and q are zero:

$$C_L = C_{L_0} + \frac{\partial C_L}{\partial \alpha}\alpha + \frac{\partial C_L}{\partial \delta_e}\delta_e + \frac{\partial C_L}{\partial \dot{\alpha}}\dot{\alpha} + \frac{\partial C_L}{\partial q}q \tag{2.8a}$$

$$C_D = C_{D_0} + \frac{\partial C_D}{\partial \alpha}\alpha \tag{2.8b}$$

$$C_M = C_{M_0} + \frac{\partial C_M}{\partial \alpha}\alpha + \frac{\partial C_M}{\partial \delta_e}\delta_e + \frac{\partial C_M}{\partial \dot{\alpha}}\dot{\alpha} + \frac{\partial C_M}{\partial q}q \tag{2.8c}$$

The stability derivatives

The aerodynamic reactions of the airplane can be represented approximately by means of stability derivatives, i.e., the coefficients of the Taylor series expansion above.

Note that as C_D is strongly influenced by the angle of attack, all other influences can be neglected.

The α derivatives C_{L_α}, C_{D_α}, and C_{M_α} describe how changes in the angle of attack α affect the aerodynamic forces and moments. An increase in the angle of attack generally induces an increase in the Lift, an increase in the Drag and a negative pitching moment.

The δ_e derivatives $C_{L_{\delta_e}}$ and $C_{M_{\delta_e}}$ describe the effect that a deflexion of the elevator has on the Lift and on the Pitching Moment. A positive elevator deflexion is defined as being *elevator down*, which causes an increase in the Lift and a negative pitching moment increment.

The $\dot{\alpha}$ derivatives $C_{L_{\dot{\alpha}}}$ and $C_{M_{\dot{\alpha}}}$ basically represent the adjustment of the pressure distribution on the aerodynamic surfaces to sudden changes in the angle of attack, as, for example, when sudden changes in the incidence of the headwind occur.

The q derivatives C_{L_q} and C_{M_q} represent the aerodynamic effects induced by a rotation of the airplane about its spanwise axis when the angle of attack is kept constant, e.g., keeping the fuselage tangential to an arbitrarily varying flight path.

These two rotational effects can be visualized considering a flight along an arbitrary flight path: first with the fuselage of the plane always tangent to the flight path (angle of attack kept zero) and second, with the fuselage always horizontal (pitch angle kept zero).

The nondimensional aerodynamic coefficients are expressed in terms of the stability derivatives in the set of equations (2.9):

$$C_L = C_{L_0} + C_{L_\alpha}\alpha + C_{L_{\delta_e}}\delta_e + \frac{\bar{c}/2}{v}\left[C_{L_{\dot\alpha}}\dot\alpha + C_{L_q}q\right] \qquad (2.9a)$$

$$C_D = C_{D_0} + C_{D_\alpha}\alpha \qquad (2.9b)$$

$$C_M = C_{M_0} + C_{M_\alpha}\alpha + C_{M_{\delta_e}}\delta_e + \frac{\bar{c}/2}{v}\left[C_{M_{\dot\alpha}}\dot\alpha + C_{M_q}q\right] \qquad (2.9c)$$

Note that the rotational derivatives $C_{L_{\dot\alpha}}$, C_{L_q}, $C_{M_{\dot\alpha}}$ and C_{M_q} are multiplied by $\frac{\bar{c}/2}{v}$. This is due to the fact that these derivatives are, in fact, taken with respect to the quantity $\frac{\dot\alpha\bar{c}/2}{v}$ (or $\frac{q\bar{c}/2}{v}$) where $\bar{c}$ is the mean aerodynamic chord of the wing and v is the cruising velocity.

For example, consider $C_{L_{\dot\alpha}}$:

$$\begin{aligned} C_{L_{\dot\alpha}} &= \frac{\partial C_L}{\partial \frac{\dot\alpha\bar{c}/2}{v}} \\ &= \frac{\partial C_L}{\frac{\bar{c}/2}{v}\partial\dot\alpha} \end{aligned}$$

therefore, $$\frac{\partial C_L}{\partial\dot\alpha} = \frac{\bar{c}/2}{v}C_{L_{\dot\alpha}}$$

Longitudinal Flight Control

Longitudinal flight control basically means control of the velocity vector **v** acting on the center of gravity of the airplane. The two available control elements are δ_e for the elevator deflexion and δ_T for the throttle control of the Thrust.

The immediate response of the aircraft to a $\Delta\delta_e$ at constant throttle is a brusque rotation in pitch and a consequent change in both angle of attack and Lift, followed by a curvature $\dot{\gamma}$ of the flight path. After this first fast transient, the new steady state flight is characterized by the new values of γ_{ss} and u_{ss}. The steady-state speed u_{ss} is fixed by the value of $C_{L_{ss}}$, which, in turn, is determined by δ_e (see Etkin, 1982, sections 2.5 and 9.1).

The immediate effect of a positive $\Delta\delta_T$ with fixed δ_e is essentially a change in the velocity, followed by a change in the flight path angle γ. But as a given δ_e fixes a constant steady state velocity, the final effect of opening the throttle will be a change in the flight path angle without changing the speed.

2.2 The Mathematical Model

Equations of motion

Referring to figure 2.2, we can write the final equations of motion from the equations 2.6. This will not be done in the stability axes, but in the tangent and normal axes with respect to the flight path, because this simplifies somewhat the equations:

$$m\dot{v} = T cos\alpha - D - W sin\gamma \qquad (2.10a)$$

$$mv\dot{\gamma} = T sin\alpha + L - W cos\gamma \qquad (2.10b)$$

$$I_y\dot{q} = M \qquad (2.10c)$$

$$\dot{\theta} = q \tag{2.10d}$$

Relation (2.5) gives the relationship between the reference angles:

$$\theta = \gamma + \alpha. \tag{2.5}$$

and the position of the airplane with respect to the ground is given by the equations (2.11):

$$\dot{h} = v\ sin\gamma \tag{2.11a}$$

$$\dot{x} = v\ cos\gamma \tag{2.11b}$$

Aerodynamic equations

Equations (2.7) give the aerodynamic quantities L, D, and M:

$$L = \frac{1}{2}\rho v^2 S C_L \tag{2.7a}$$

$$D = \frac{1}{2}\rho v^2 S C_D \tag{2.7b}$$

$$M = \frac{1}{2}\rho v^2 S \frac{\bar{c}}{2} C_M \tag{2.7c}$$

and the nondimensional coefficients C_L, C_D and C_M are given by the equations (2.9):

$$C_L = C_{L_0} + C_{L_\alpha}\alpha + C_{L_{\delta_e}}\delta_e + \frac{\bar{c}/2}{v}\left[C_{L_{\dot{\alpha}}}\dot{\alpha} + C_{L_q}q\right] \tag{2.9a}$$

$$C_D = C_{D_0} + C_{D_\alpha}\alpha \tag{2.9b}$$

$$C_M = C_{M_0} + C_{M_\alpha}\alpha + C_{M_{\delta_e}}\delta_e + \frac{\bar{c}/2}{v}\left[C_{M_{\dot{\alpha}}}\dot{\alpha} + C_{M_q}q\right] \tag{2.9c}$$

Closed loop equations

The two control laws implemented in the model are standard procedure in stability and control of flight. Feedback of pitch angle deviation from its trimmed value (for which the airplane is in steady horizontal reference flight) into the elevator deflexion suppresses efficiently the phugoid mode of the airplane which is slow and very lightly damped. The second control loop was similarly implemented feeding back velocity into Thrust.

$$\delta_e = \delta_{e_{trim}} + K_\theta(\theta - \theta_{trim}) \tag{2.12a}$$

$$T = T_{trim} + K_u(u - u_{trim}) \tag{2.12b}$$

The subscript $_{trim}$ refers to the trimmed value of the variable, and u is the x-component of the velocity in the stability axes, or

$$u = v\ cos\alpha \tag{2.13}$$

Model Parameters

Listed below are all the values used for the flight related constants. The airplane related physical data was taken for a large commercial/cargo jet plane as a Boeing 747 in cruise flight, at 20000 ft and Mach Number .5 ($\approx$ 500 ft/s). The aerodynamic coefficients were adapted for a trimmed reference flight with a given set of initial conditions which is characterized by a horizontal steady flight at 500 ft/s, altitude 20,000 ft, zero angle of attack, elevator deflexion of 1.6 degrees (.0279 rad) and constant thrust of 33,000 pounds. These initial conditions were then trimmed such that the flight would start perfectly trimmed, since approximation errors in the aerodynamic constants had still to be corrected. The model is theoretical but effort was made in building it in the most realistic fashion.

Airplane Constants

I_y	= 27,000,000.0	$[slug\ ft^2]$
W	= 500,000.0	$[lb]$
$\bar{c}$	= 27.3	$[ft]$
S	= 6,000.0	$[ft^2]$

Physical Constants

g	= 32.2	$[lb/ft^2]$
ρ	= 0.0012	$[slug/ft^3]$, at 20,000 ft

Aerodynamic Constants

C_{L_0}	= 0.5455	[ ]
C_{L_α}	= 5.2	$[1/rad]$
$C_{L_{\delta_e}}$	= 0.36	$[1/rad]$
$C_{L_{\dot{\alpha}}}$	= 2.0	$[1/\frac{rad}{s}]$
C_{L_q}	= 5.5	$[1/\frac{rad}{s}]$
C_{D_0}	= 0.036667	[ ]
C_{D_α}	= 0.26	$[1/rad]$
C_{M_0}	= 0.039	[ ]
C_{M_α}	= -0.74	$[1/rad]$
$C_{M_{\delta_e}}$	= -1.4	$[1/rad]$
$C_{M_{\dot{\alpha}}}$	= -8.0	$[1/\frac{rad}{s}]$
C_{M_q}	= -22.0	$[1/\frac{rad}{s}]$

Feedback Gains

K_θ	$= 0.25$	[ ]
K_u	$= 40.0$	[ ]

Initial Conditions

v_0	$= 500.1375$	$[ft/s]$
h_0	$= 20{,}000.0$	$[ft]$
x_0	$= 0.0$	$[ft]$
q_0	$= 0.0$	$[rad/s]$
α_0	$= -0.000055$	$[rad]$
θ_0	$= -0.000055$	$[rad]$
γ_0	$= 0.0$	$[rad]$
δ_{e_0}	$= 0.027886$	$[rad]$
T_0	$= 33{,}005.5$	$[lb]$

2.3 Model Implementation

The mathematical model of the longitudinal flight was implemented in ACSL (Advanced Continuous Simulation Language) which has been designed for modeling and evaluating the performance of continuous systems described by time dependent, non-linear differential equations (Mitchell and Gauthier, 1986; Systems Control Technology, 1986).

The model used is similar to the sample program listed in the ACSL User Guide/ Reference Guide (Mitchell and Gauthier, 1986 page A.51) entitled "Longitudinal Study" which was used in the Ctrl–C/ACSL Interface Description (Systems Control Technology, 1986, page 8) as well, however with some modifications.

In the present text, the model is still basically the same, but many aerodynamic parameters have been changed to make the model more realistically represent the longitudinal flight of a large commercial/cargo jet plane cruising at high altitude. The initial conditions were also set to values such that the flight would start perfectly trimmed, so that the initial trimming phase of the original models is not really needed here, even though it is still included in the program for situations when the initial conditions are changed.

The initial trimming phase adjusts non-trimmed guesses of initial conditions such that the flight would start as a smooth cruise flight since that is what the model represents. After being started, the model can be perturbed, and the control loops will stabilize it normally, but it is desired that all simulation runs start trimmed. The initial values of the flight path angle, pitch angle, and elevator deflexion are iteratively adjusted such that a weighted mean square error of the derivatives of the linear velocity, pitch rate, and flight path angle would be minimized (acceptable if less than 0.1) before the model is started.

Two control loops were implemented into the model, one in each of the control elements. The first of them was used in the Ctrl–C/ACSL Interface Description as well, and is common usage for the phugoid (long period) mode supression in aerodynamic stability design: feedback of deviations of the pitch angle from its trimmed value into the elevator deflexion.

A second control loop was implemented in the driving force of the aircraft, allowing the user to change the intensity of thrust developed by the engines. This was done simply by changing the value of T_{trim}, considering, in this way, the throttle opening directly proportional to the thrust. A feedback loop was also implemented into this control element to automatically compensate for changes in the velocity.

Figures 2.3 and 2.4 show the response of the system to step changes in the reference values of the two control elements of the model.

A switch INPT was implemented in the code to change the driving functions that will perturb the model from a trimmed reference flight. TMX specifies the length of the simulation in seconds, CINT the communication interval, and SEED the seed to be used in the generation of random numbers. The listing of the ACSL model code can be found in the Appendix at the end of this thesis and should be consulted for further details about the model.

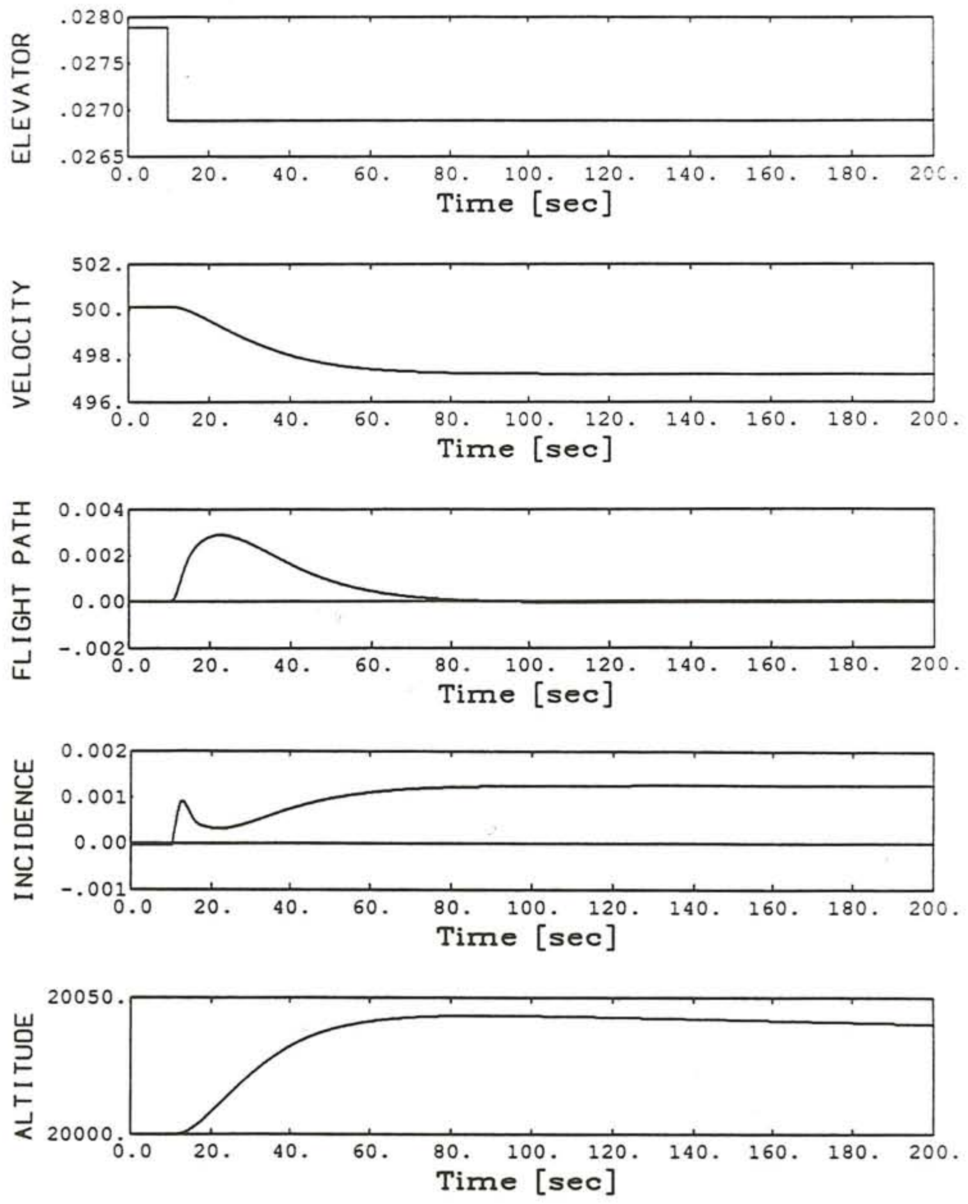

Figure 2.3 Closed loop response to a step change of -.001 rad in $\delta_{e_{trim}}$.

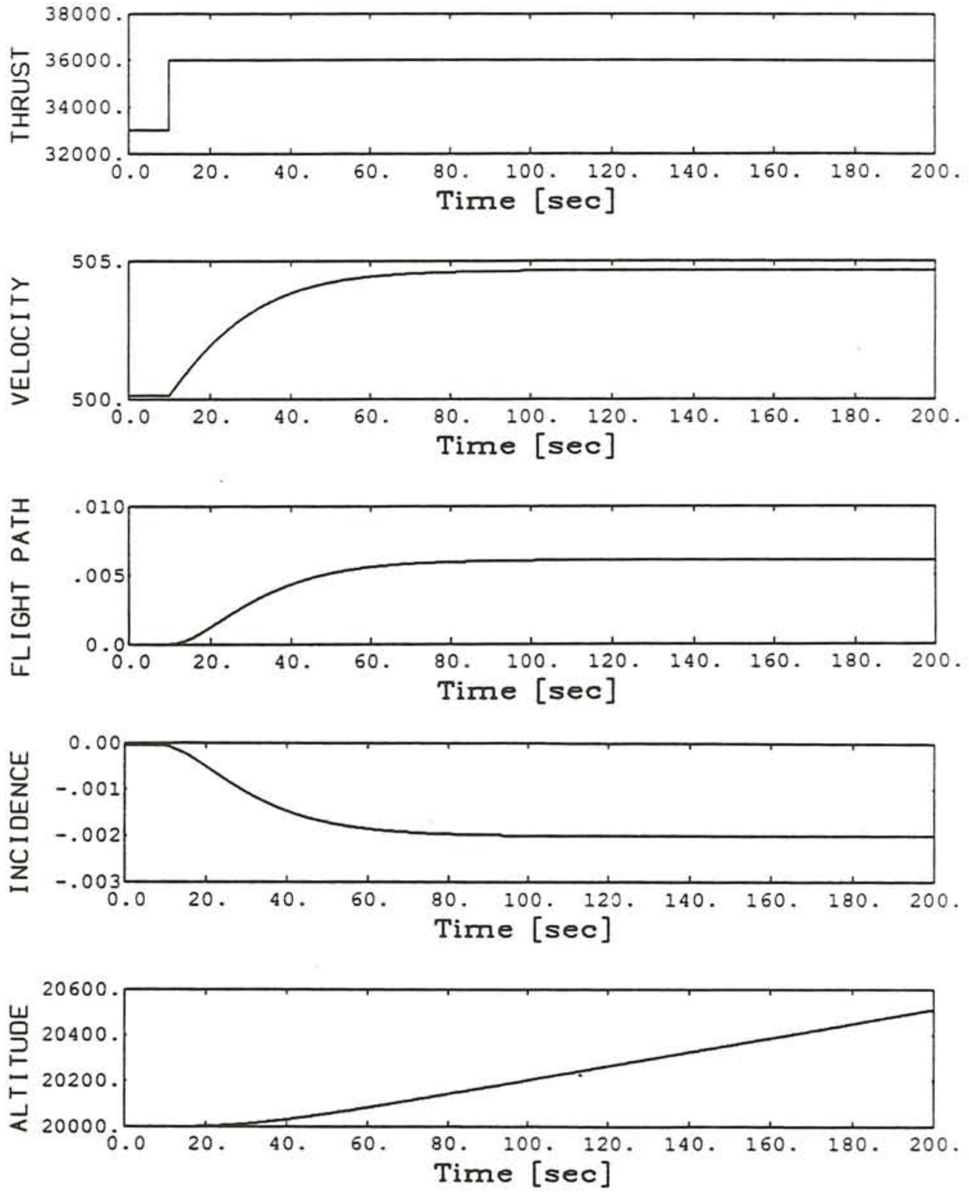

Figure 2.4 Closed loop response to a step change of 3000 lb in T_{trim}.

CHAPTER 3
GENERAL SYSTEMS THEORY

3.1 Conceptual Framework

Systems problem solving or analysis through General Systems Theory starts by defining a region in the universe where the system and the observer coexist and interact.

A system in this context can be interpreted as a set of relations between some objects that belong to that region of the universe and in which the observer is interested.

Therefore, the first step to the problem solving, or analysis, is the definition of the system: what is it, that is of interest to us concerning the problem under study? A set of variables to represent the system has to be chosen, and this set is to be classified into *input variables* and *output variables*, which is a natural classification of the variables: input variables depend on the environment and control the output variables.

"Let a conceptual framework through which types of systems problems are defined together with methodological tools for solving problems of these types be called a *general systems problem solver* (or GSPS, in abbreviation)." (Klir, 1985 page 10)

3.2 Epistemological Hierarchy

The GSPS framework is a hierarchically arranged set of epistemological subsystems. Starting at level one, the amount of knowledge in the systems increase as we climb the epistemological ladder.

The four first epistemological subsystems are:

level one : Source System

level two : Data System

level three : Behavior System

level four : Structure System

The lower level subsystems are contained in the ones that are at higher epistemological levels.

Source System

At the lowest epistemological level, we find the Source System which represents the system as it is recognized by the observer. The amount of information present at this level represents the basic description of the problem in which the observer is interested: which are the variables that are relevant to the problem, what causal relationships are present among them (which are inputs and which are outputs to the system), and which are the states these variables can possibly assume along their time-history (this study will only consider systems where the only support variable is time).

To illustrate the definition, let us consider the first flying lesson of a future pilot in a B747 flight simulator. Let us assume that the simulation starts off from a stabilized (trimmed) longitudinal flight at high altitude. So, he starts by playing around with all the controls that he has available, very cautiously in the beginning. At one time, he detects a control that makes the nose of the aircraft go up and down with respect to the horizon: he "senses" the pitch angle. Then he gets curious about the velocity and checks out the speedometer: the speed is around "500". The number does not mean much to him, since he does not even know the units it is coded in. But it is a reference value because the plane is flying all right; so that must be a good value for the air speed. He tries the controls again and observes the variation of the speed until he is able to code the reading of the instrument to something like: "somewhere near 500", "above 500" and "below 500". Then he moves to the other variables that he can identify, and within his capabilities of observation, analyses them and codes them in the way he understands and feels them. When he gets enough confidence in his understanding of the way things work in the cockpit, he starts to experiment more aggressively. Things like "What happens if" start crossing his mind, and he starts restricting his attention to certain aspects of the flight, defining in this way an area of interest in that sea of instruments and different sensations he is experiencing for the first time. The knowledge that our aspiring pilot has now acquired — a set of variables of interest and a set of states these variables can potentially assume — is defined as a Source System in GSPS.

The reason(s) for the observer to have chosen the system as such is not arbitrary and can be interpreted as the link between this system and its environment.

The number of states, or levels that each variable can potentially assume is essentially problem dependent. It should be kept as low as possible without unacceptable loss of information. Consider again the speedometer with its needle at 500. Let us assume that that is the standard cruise velocity for the airplane and therefore, in that region, the scale is larger and with more subdivisions. In his first experiments, the "pilot" was doing fine interpreting the velocity as "about 500", "high" and "low", but now he may want to be a little more accurate because he wants to try flying faster and slower, and he may decide that five levels are more appropriate for this new task.

Data System

The next epistemological level in the hierarchy is represented by the Data System. It includes the Source System and, additionally, the recoded time history of all its variables. Data may be supplied by a quantitative simulation of a system model or by observation in system analysis problems, or may be imposed as desired states in system design problems.

A Data Model in the GSPS framework is an $n_{rec} \times n_{var}$ matrix where n_{rec} is the number of recordings (data points) collected in the time span covered by the Data Model and n_{var} is the number of variables present in the model. This is a matrix representation of the time-history of the system, where the input variables are conventionally located in the leftmost columns and time increases from top to bottom of the matrix.

Note that data, in the Data System, refers to *recoded* data, i.e., they must be represented in terms of the levels that meaningfully represent the states each variable asssumes along its time history, and which were chosen in the Source System.

Behavior System

One epistemological level higher, we find the Behavior System which holds, in addition to the knowledge inherent to both, Source and Data Systems, a set of time-invariant relationships existent among these variables, for a given set of initial or boundary conditions. Behavior Systems can be considered basic cells for higher epistemological level systems, so called Structure Systems. The time-invariant relationships among the variables are *translation rules* mapping these variables into their common spaces. They can be used to generate new states of the variables within the time span defined in the Data Model, allowing in this way an *inductive system modeling* feature in the methodology. It is based on this feature that a monitoring device for the system can be built to detect structural changes in it. Due to this characteristic, Behavior Systems are also called Generative Systems.

3.3 The Concept of Mask

A mask is the matrix representation of a time-invariant translation rule relative to a given Data Model, hence, it is the matrix representation of the Behavior Model of the system. The dimensions of a mask are $(d+1) \times n_{var}$, where d is the depth of the mask and represents the number of sampling intervals it covers.

Elements of a mask are negative, zero, or positive. The non-zero elements represent the "sampling variables" of the mask and the zero elements are neutral entries.

Sampling variables are the variables that form the translation rule which the mask represents. They are the state variables themselves, with a temporal tag attached to them. For example, consider the identities

$$s_1 = x(t - \Delta t)$$

$$s_2 = x(t)$$

where the sampling variable s_1 is defined as being the state variable x considered one time interval Δt back in time, and the sampling variable s_2 is defined as being the same state variable x, but considered at the present time t defining therefore a distinct sampling variable (s_2).

Sampling variables, as previously stated, can be negative or positive entries in the mask, which is how input (or generating) sampling variables (negative entries) are distinguished from the output (or generated) sampling variables (positive entries). These variables are numbered separately, input sampling variables from -1 down and output sampling variables from $+1$ up. Among the sampling variables of the same type (inputs *vs* outputs), the numbering sequence is arbitrary. In this text, we adopted the following convention: sampling variables are numbered consecutively from the left to the right and from the top to the bottom (like one does when writing in English).

For example, consider the following mask which has been designed for a Data Model composed of two input state variables (v_1 and v_2) and three output state variables (v_3, v_4, and v_5):

$$\begin{array}{c}
 \\ t-\Delta t \\ t \\ t+\Delta t
\end{array}
\begin{array}{c}
\begin{array}{ccccc} v_1 & v_2 & v_3 & v_4 & v_5 \end{array} \\
\begin{pmatrix} 0 & 0 & -1 & 0 & 0 \\ -2 & 0 & 0 & -3 & 0 \\ 0 & -4 & 0 & 0 & +1 \end{pmatrix}
\end{array}$$

The shown mask corresponds to the translation rule :

$$v_5(t+\Delta t) = f\Big(v_3(t-\Delta t), v_1(t), v_4(t), v_2(t+\Delta t)\Big)$$

where $v_i(\tau)$ represents the state assumed by the variable v_i at time $t = \tau$.

Note that the mask's only output sampling variable (entry "+1") is located at its bottom row which has the most advanced time tag associated with it (conveniently specified as $t + \Delta t$), which gives the mask generative qualities, i.e., the state of the state variable v_5 at a future time $t + \Delta t$ can be *generated* based on the states of the input sampling variables, once a behavioral pattern for the output in question (v_5) has been extracted from the Data Model.

Sampling interval

Note that the translation rule (and the respective mask) in the previous example uses samples of the Data Model taken at every Δt seconds to predict the state of v_5. Hence, Δt is the sampling interval t_s of the collected data set. There is not a precise way of determining the most efficient sampling interval to be used, but a good rule of thumb to be used is that the mask should cover the dynamics of the slowest mode in the model (Cellier, 1987). In the case of the given example, the mask has depth 2 and the sampling interval Δt should then be about half of the slowest time constant of the model.

3.4 Optimal Mask Analysis

Given a Data Model, any mask associated with it is "valid" since it is the representation of a relationship among the sampling variables it contains. The question now is "How *good* is the mask?", "How valid is the translation rule it represents?". There are innumerous possible masks that can be written for one set of variables, and our discussion now focuses on the determination of the mask that will have the least uncertainty in its generating capability. To describe the methodology used to compare different potential mask candidates, let us first describe some of the tools available in our GSPS software package, SAPS–II.

Behavior Model Analysis

A behavior analysis can be performed on a given Data Model by *sliding* the mask associated with it over the Data Model in the positive time direction, and writing down the states assumed by the sampling variables at every sampling interval. Note that the resulting Behavior Model will have d (mask depth) less data sets than the Data Model, and the size of the data sets read at every sampling interval, is the number of sampling variables present in the mask used.

To visualize the process of building a behavior model out of a Data Model using a certain mask, picture the mask as being, physically, a piece of cardboard with the same physical dimensions as the Data Model in the horizontal direction and both matrices having the same spacing between rows. Now picture the non-zero entries of the mask as holes in the cardboard such that when the mask is slid over the Data Model, one can read the (recoded) states of the state variables in the positions determined by the sampling variables (the *holes* in the mask) and write each row of the behavior model from these readings at every sampling interval. It is the convention to write first, from left to right, the inputs (negative entries in the mask: -1, -2, ...) and then the outputs (positive entries: $+1$, $+2$, ...).

Listed below are a generative mask, the ten first lines of a Data Model for which the mask has been created and the associated Behavior Model which is obtained by applying the mask on the Data Model as described above:

$$mask = \begin{pmatrix} 0 & 0 & -1 & 0 & 0 \\ -2 & 0 & 0 & -3 & 0 \\ 0 & -4 & 0 & 0 & +1 \end{pmatrix}.$$

Time	Data Model					Behavior Model				
$t = 0$	1	3	3	2	2	–	–	–	–	–
Δt	1	2	2	2	2	–	–	–	–	–
$2\Delta t$	2	3	3	1	1	3	1	2	3	1
$3\Delta t$	1	1	1	2	3	2	2	1	1	3
$\vdots$	3	2	2	3	3	3	1	2	2	3
	2	2	3	1	1	1	3	3	2	1
	1	1	2	3	2	2	2	1	1	2
	3	2	3	2	1	3	1	3	2	1
	1	1	3	2	2	2	3	2	1	2
	2	3	2	2	3	3	1	2	3	3

A simplified representation of the Behavior Model is obtained by performing a basic behavior analysis on it, which consists of a sorted listing of all possible combinations of the states of the Behavior Model associated with their frequency of occurrence in the Behavior Model. This is where SAPS–II starts extracting information from the Data Model in a more comprehensive way.

State Transition Analysis

State transition analysis is a more complete behavior analysis, which outputs conditional probabilities as well, relating the probability with which certain inputs occur to the occurrence of certain outputs. It is assumed that the Data Model under study is large enough to allow our interpreting of the frequencies of occurrence as **probabilities** of occurrence.

The State Transition Model generated by SAPS–II gives the frequencies (or probabilities) of occurrence of sets of inputs, sets of outputs, and mixed input/output occurrence frequencies (conditional probabilities of a certain output to occur, given the occurrence of a set of inputs).

The Concept of an Optimal Mask

An OPTIMAL MASK is a mask that represents the translation rule that has the least uncertainty associated with its capability of generating future states of a certain set of outputs. In this thesis, we will only consider single output masks and generation of different output variables will be done with separate masks.

SAPS–II allows the user to search for an optimal mask relative to a certain set of inputs, where the "goodness-of-fit" of different masks is based on a quality factor q which is evaluated for each mask using Shannon Entropy (e.g., Levine and Tribus, 1978) as a measure of uncertainty associated with the proposed generative properties of the mask.

MASK CANDIDATES are masks with the same dimensions as the desired optimal mask and whose non-zero elements are either "-1" or "$+1$". Such masks allow the user of SAPS–II to specify all possible input sampling variables (-1 entries) that may possibly affect the output sampling variables that he also specifies in the same mask candidate ($+1$ entries). Zero elements are neutral.

For example, the mask:

$$\begin{array}{c} \\ t-\Delta t \\ t \\ t+\Delta t \end{array} \begin{array}{c} \begin{array}{ccccc} i_1 & i_2 & o_1 & o_2 & o_3 \end{array} \\ \begin{pmatrix} -1 & -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & -1 & -1 \\ -1 & -1 & 0 & 0 & +1 \end{pmatrix} \end{array}$$

is a mask candidate built for a Data Model composed of two input state variables (i_1 and i_2) and three output state variables (o_1, o_2, and o_3). In building such a mask candidate, the user has specified almost every element of the mask as a possible influencing element (input sampling variable) in the generation of a future state of the state variable o_3 one sampling interval (Δt) ahead in time. The sampling variables in a mask candidate do not need numbering since their function is solely to point out which are the possible input sampling variables that may (possibly) affect the behavior of the output variables that are also specified in the mask candidate.

Similar mask candidates will be used for the other two output variables. Note that the only two zero entries in the mask candidates are relative to the other two output state variables at time $t + \Delta t$. By not permitting a direct interaction among the three outputs ate same time, we prevent the forecating (behavior generation) algorithm from entering algebraic loops, since otherwise it could, e.g., happen that

$$o_3(t + \Delta t) = f_3(o_1(t + \Delta t))$$

$$o_1(t + \Delta t) = f_1(o_3(t + \Delta t))$$

which is perfectly logical since future outputs should not influence the behavior of the generated output.

Based on a mask candidate and a Data Model, SAPS–II is able to perform an exhaustive search through all mask complexity levels (number of non-zero entries in the mask) to find the mask that has the least uncertainty associated with its capability of generating future states of a certain set of output sampling variables.

For example, with the previous sample mask candidate, the search starts at mask complexity 2. At this complexity level each input sampling variable is considered individually as being the only element affecting the state assumed by the one output sampling variable present in the mask: $v_5(t + \Delta t)$. A quality factor relative to each considered mask is computed based on the behavior pattern observed throughout the Behavior Model that was obtained applying the mask on the Data Model under consideration. Once the best mask at this level has been found, SAPS–II goes through all possible combinations of pairs of inputs (at mask complexity 3 there are three non-zero elements in a mask, i.e., as we are considering only one output sampling variable, there must be two input sampling variables) and computes the quality factors associated with each of these masks, finding in this way the best mask at complexity level 3. In this same fashion, all possible masks at all levels (up to a maximum complexity level specified by the user) will have their quality factors evaluated and the mask with the highest quality factor found in the search is the optimal mask that we are looking for. In the mask history, SAPS–II preserves the knowledge of the best masks found for each complexity level.

The Quality Factor of a Mask

Shannon Entropy measure is used to determine the uncertainty associated with the assumption that the proposed translation rule is true, i.e., that a certain output occurs when a given set of states occur in the input sampling variables. The entropy relative to one input is calculated from the equation:

$$H_i = -\sum_{\forall o} p(o \mid i)\ log_2 p(o \mid i) \tag{3.1}$$

where $p(o \mid i)$ is the conditional probability of a certain output occurring, given that the input occurred.

The overall entropy of a mask is calculated as the sum:

$$H_m = \sum_{\forall i} p_i H_i \tag{3.2}$$

where p_i is the probability of that input occurring. Maximum entropy is measured when all the probabilities are equal, and zero entropy occurs when the relationship is deterministic.

A normalized overall entropy reduction is defined for the masks as H_r by:

$$H_r = 1 - \frac{H_m}{H_{max}} \tag{3.3}$$

where H_{max} is the maximum possible entropy of the mask. H_r will be such that

$$0 \leq H_r \leq 1$$

In order to find the optimal mask that best represents the relationship existent among the set of sampling variables under consideration, SAPS–II goes through an extensive search that is carried out in levels. It first computes the quality of the simplest masks (with the smallest number of non-zero elements in them), chooses the best at that level and steps to the next level. In order to compare masks at different levels, a mask complexity weight C_m is defined as:

$$C_m = \frac{n_{var}\, d_{act}\, n_{compl}}{d_{max}} \tag{3.4}$$

where:

n_{var} is the number of variables in the source model,

d_{act} is the actual depth of the mask plus one,

n_{compl} is the number of non-zero entries in the mask, and

d_{max} is the maximum possible depth the mask could have (the depth of the chosen mask candidate) plus one.

For example, the complexity weight of the following mask

$$\begin{pmatrix} 0 & 0 & 0 & 0 \\ -1 & 0 & -2 & 0 \\ -3 & 0 & 0 & 1 \end{pmatrix}$$

can be evaluated to

$$C_m = \frac{4 \times 2 \times 4}{3} = 10.667$$

Finally, the quality measure Q is defined as

$$Q = \frac{H_r}{C_m} \tag{3.5}$$

Once an optimal mask has been found, SAPS–II allows the user to forecast future behavior of the system, based on the knowledge inherent to the generative model represented by the mask, and in the behavioral knowledge held within the Data Model. The probabilities associated with the forecast states are outputted as well, and the forecasting can be aborted at a minimum probability, if desired.

CHAPTER 4
A QUALITATIVE AIRCRAFT MODEL

4.1 General Description

This chapter focuses on the ability of generative systems to perform inductive reasoning based on past behavior of the system under analysis. We will use our longitudinal flight model to demonstrate how a system failure detector can be implemented in the GSPS framework forecasting the future behavior of the system and comparing this forecast with the actual measured data. As the forecast is based purely on past behavior of the system, it is adaptive to slow changes in it (such as aging of its components), but a sudden structural change is immediately detected since the behavior of the system cannot be forecast any more with the optimal masks that have been evaluated for the system under observation.

A set of five variables will compose the source system of our qualitative model, and data will be generated first to extract information from the system and load it into the GSPS framework to generate the optimal masks relative to the outputs. Then, data relative to a cruise flight will be analysed and monitored with the aid of the forecasting tool of SAPS–II using the optimal masks. An unforeseen system failure in the system is simulated by a structural change in the model, and the change is immediately noticed by the online monitoring system.

4.2 The Source Model

The Source Model chosen to run the experiments has two input variables and three output variables. The input variables are step perturbations affecting the reference values of the two control variables of our model, $\delta_{e_{trim}}$ and T_{trim}. The *trimmed* values of these variables are preset, such that the model starts out in a perfectly stable horizontal flight. A change in any of these variables will perturb the model forcing it to a new steady state, with a new cruise velocity and/or flight path angle. The set of outputs that we will analyse are the two components of the aerodynamic force, lift L and drag D, and the flight path angle γ.

4.3 The Data Model

The Data Model to be used for the optimal mask analysis must be very rich in information about the dynamics of the system. To concentrate as much information as possible in this Data Model, a theoretical "shaken" flight phase where the controls are exerted frequently and in different ways was implemented into the model as describes the ACSL-code below:

```
program
initial
            ⋮
      constant tpulse = 0.0, dde = 0.001, dtr = 3000.0, ...
            sint = 6.0, delta1 = 0.9, delta2 = 1.1
            ⋮
end $"of initial"
dynamic
            ⋮
      if(inpt .eq. 2) go to c2 $"shaken flight phase"
            ⋮
      c2..continue
      if(t .ge. tpulse) go to p1
```

```
        go to e1
              ⋮
        p1..continue
        tpulse = tpulse + sint*unif(delta1, delta2)
        pde = int(unif(1, 3.9999))
        ptr = int(unif(1, 3.9999))
        detrim = dez + (pde - 2)*dde
        trtrim = trz + (ptr - 2)*dtr
        go to e1
              ⋮
        e1..continue
        termt(t .ge. tmx) $"stopping criterion"
end $"of dynamic"
end $"of program"
```

In the shaken flight phase, the trim values of the control variables are changed in steps at time intervals randomly determined between 0.9 and $1.1 \times sint$, the sampling interval of the model under study. The steps are randomly positive, negative, or null and their magnitude has a default magnitude of $\Delta\delta_{e_{trim}} = .001$ rad and $\Delta T_{trim} = 3000$ lb. Giving so small steps in the elevator, and so large ones in the thrust (10%) may not be realistic, but these values were chosen since the model is very sensitive to changes in the elevator deflexion. The use of more realistic values like .087 rad ($\approx$ 5 degrees) for $\Delta\delta_{e_{trim}}$ and 1500 lb for ΔT_{trim} outputs a behavior that is almost not dependent at all of the changes in thrust. The effects of both inputs are desired to be of comparable magnitude since we know that both variables do affect the system, and therefore they are both of interest. A solution to this problem would be the use of more recoding levels for the output variables, which was not possible because of the data amount this would require. The reason for this will be explained in due course.

The communication interval *cint* is set to match the sampling interval which, in turn was taken as half of the slowest time constant of the output variables, a reasonable procedure for masks of depth 2, as discussed in the previous chapter. The following Ctrl–C code shows how the Ctrl–C/ACSL interface allows us to compute the eigenvalues and the time constants of the linearized model.

```
[> acsl('set tmx = 0')
[> freeze('x, h')
[> start
[> a = jacobian
```

$$a = \begin{pmatrix} -0.6034 & 0.0236 & 0.6137 & 0.0003 \\ 0.5538 & -0.7408 & -0.8705 & 0.0001 \\ 0.0000 & 1.0000 & 0.0000 & 0.0000 \\ -17.1213 & 0.0000 & -15.0787 & -0.0059 \end{pmatrix}$$

```
[> lambda = eig(a)
```

$$\text{lambda} = \begin{pmatrix} -0.0850 + 0.0056i \\ -0.0850 - 0.0056i \\ -0.5901 + 0.8722i \\ -0.5901 - 0.8722i \end{pmatrix}$$

```
[> tau = [-1/real(lambda(1)); -1/real(lambda(3))]
```

$$\text{tau} = \begin{pmatrix} 11.7648 \\ 1.6948 \end{pmatrix}$$

```
[> tsample = round(tau/2)
```

$$\text{tsample} = \begin{pmatrix} 6. \\ 1. \end{pmatrix}$$

As shows the code above, the Ctrl–C/ACSL interface offers the user the capability to evaluate the state transition matrix of a nonlinear system's linearized model about the current point in the state space. This is done by numerical perturbation through the "a = **jacobian**" command. For the output of the command to be valid, the model has to be trimmed at the point of evaluation (all derivatives equal zero) and outputs of open loop integrators have to be eliminated from the state vector. Both displacement variables (horizontal displacement x and vertical displacement h) are outputs of open-loop integrators (see Appendix for the ACSL program listing) and can be eliminated from the state vector with the command **freeze**('x, h'). Trimming the model is not needed since we compute the Jacobian at time zero when our model is perfectly trimmed due to the carefully chosen initial conditions. The **start** command starts the simulation run, but as tmx (maximum simulation time, used as stopping criterion in the model) is set to zero, all that happens is the computation of the start values for all variables based on the given initial conditions. Therefore, the state transition matrix, the eigenvalues and the time constants calculated refer to a linear model that matches the trimmed reference flight of our aircraft. The only use of the "A-matrix" is to give an idea of the sampling interval to be used, but all data needed for the qualitative analysis of the model will be collected from simulation runs of the nonlinear ACSL-model describing the longitudinal flight.

The slowest time constant of the linearized model is 12 seconds, a time span which should be fully covered by the mask that we want to use on the data model. We will assume that, for global decision making purposes, only the slowest mode of the observed system is of interest. Therefore, when using masks of depth 2 on the data model (assuming that two samples within every interval of one time constant will yield enough information about the dynamics of the system), we are setting the sampling interval of the data model to 6 seconds.

Had we wanted to include the faster mode ($\tau_2 = 2$ sec) in our analysis, the dimensions of the Behavior Model would have grown beyond the current capabilities of SAPS–II if no information about the relationships among the sampling variables is available. The following mask is an example of a very general mask candidate that would cover both time constants, sampling each of them twice:

$$\begin{pmatrix} -1 & -1 & -1 & -1 & -1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ -1 & -1 & -1 & -1 & -1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ -1 & -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & -1 & -1 \\ -1 & -1 & 1 & 0 & 0 \end{pmatrix}$$

Another solution to this problem whould be to perform two separate optimal mask analyses, one for each mode, and use the information about the causal relationships gotten from these analyses to build less complex mask candidates and so, iteratively find the optimal masks for each output.

The following two sets of Ctrl–C/ACSL code were used to retrieve the step input response of the system shown in figures 4.1 . First the system is started trimmed at time zero, perturbed with a negative step of magnitude $-.001$ rad in the reference value of the elevator deflexion scheduled at time $t = 10$ seconds:

```
[>  acsl('set tmx = 200, cint = 0.1')
[>  acsl('set inpt = 1, dtr1 = 0')
[>  [t, de, detrim, l, d, ga] = start;
[>  save >temp
```

The time history of the perturbation step $\delta_{e_{trim}}$, and its effect on the elevator deflexion, lift, drag and flight path angle, and the independent variable itself are saved temporarily. The next code describes the simulation of the model with a perturbation step of magnitude $\delta_T = 3000$ lb in the thrust also scheduled at time $t = 10$ seconds.

```
[>  acsl('set tmx = 200, cint = 0.1')
[>  acsl('set inpt = 1, ddel = 0, ttrl = 10')
[>  [tr, trtrim, l, d, ga] = start;
[>  ltr = l; dtr = d; gatr = ga;
[>  clear l d ga
[>  load <temp
[>  term = '4100'; hard = 'tekf';
[>  window('211'), plot(t, [de, detrim])
[>  xlabel('time [sec]', ' lll  lll '), ylabel('de,detrim')
[>  window('212'), plot(t, [tr, trtrim])
[>  xlabel('time [sec]', ' lll  lll '), ylabel('tr,trtrim')
[>  replot
[>  erase, window('211'), plot(t, [l, ltr])
[>  xlabel('time [sec]', ' lll  lll '), ylabel('lift [lb]', ' lll  ll ')
[>  window('212'), plot(t, [d, dtr])
[>  xlabel('time [sec]', ' lll  lll '), ylabel('drag [lb]', ' lll  ll ')
[>  replot
[>  erase, window('211'), plot(t, [ga, gatr])
[>  xlabel('time [sec]', ' lll  lll '), ylabel('g [rad]', 'g  lll ')
[>  replot
[>  quit

$   rename ctrlc.tekf figure4_1
```

The outputs of this second ACSL simulation are the perturbation step input, and its effects on the thrust, lift, drag and flight path angle. The last three variables are renamed so that we can load back the time histories of these same variables affected by the step in the elevator.

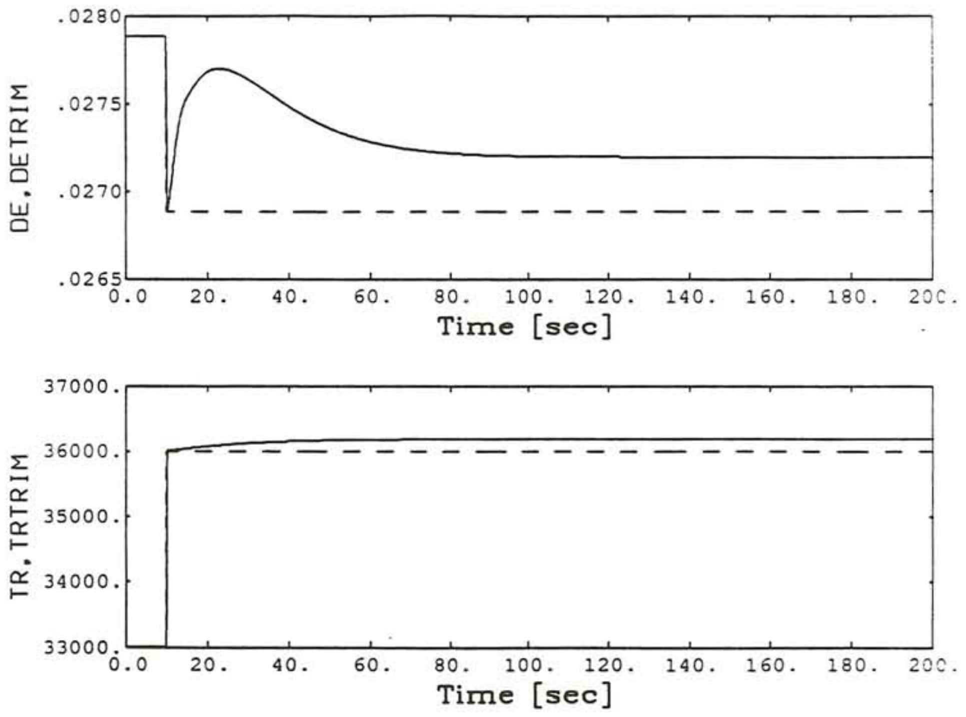

Figure 4.1a Effect of the perturbations on the controls

Figure 4.1a above shows the plots of each perturbation superposed with its effect on the respective control variable, and figure 4.1b shows the response of the three output variables of our source system to each of the perturbations. The plots in solid line represent the response of the system to a step change in the elevator deflexion, at fixed throttle opening and the plots in dashed line represent the response at fixed elevator deflexion and a step change in the thrust. Both perturbations are given from the trimmed reference flight characterized by the initial conditions.

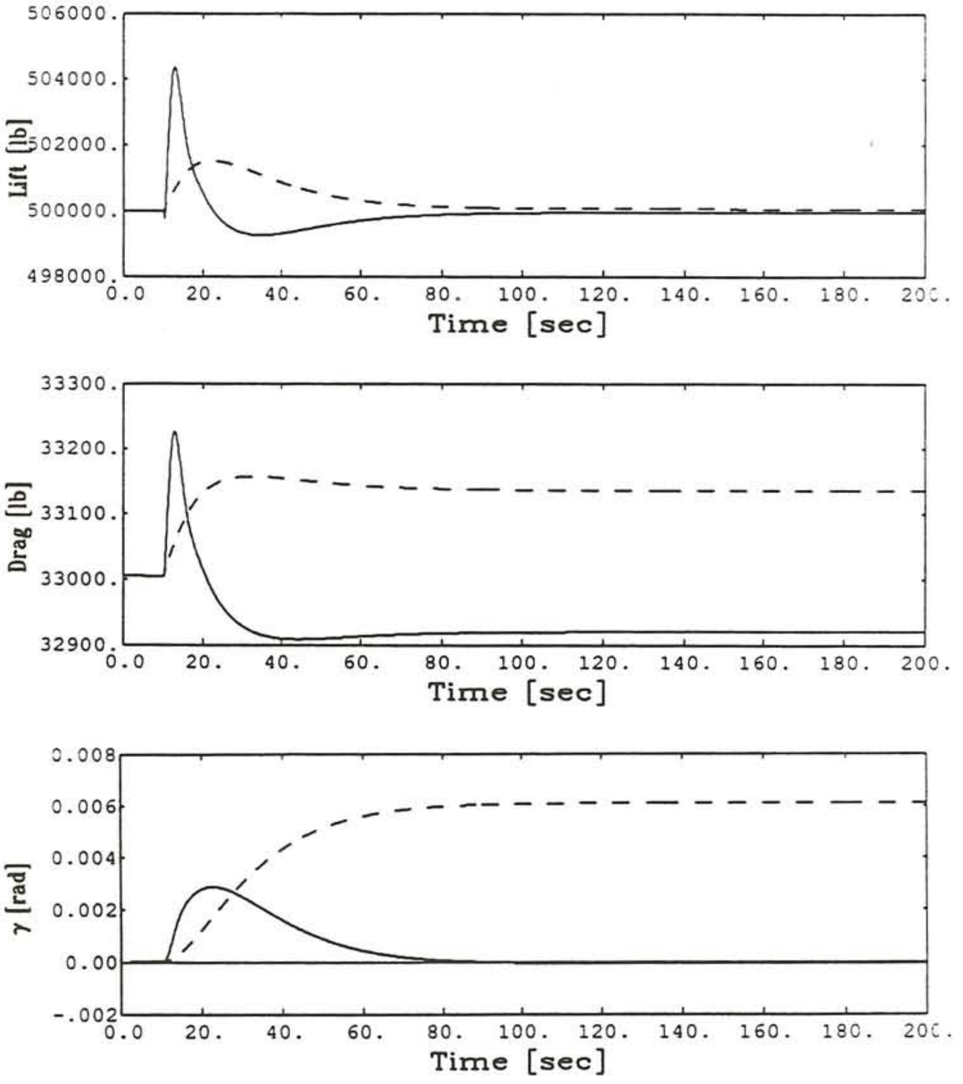

Figure 4.1b System response to the perturbations

Collecting Data

One last parameter has still to be decided upon, before we run the model in its shaken phase: data size. The necessary length of the simulation is closely related to the number of recoding levels that we want to use to recode our raw data. Even though the number of recoding levels is a parameter that theoretically should already have been decided upon in the source system, here we will have to work the other way around due to restricted data handling capabilities of the computational tools available. The Ctrl–C environment workspace currently allows the use of 200,000 elements which is sometimes restrictive, as, for example in the Ctrl–C/SAPS–II user-defined data recoding function **rekode** (which will explained in detail a few sections ahead, in this same chapter), the data size cannot be larger than 5000 data points for five variables.

The SAPS–II function **optmask**, the standard SAPS–II function for optimal mask evaluation, currently restricts the data size to 2500 data points, which starts being a problem if we need to have more than three recoding levels and five variables.

The minimum number of recordings that a Data Model should contain can be calculated as being five times the total number of possible combinations of the states assumed by the sampling variables that form the Behavior Model matrix of the system. The factor five is widely used in statistical and classification analyses (Law and Kelton, 1982).

According to this assumption, the minimum number of recordings necessary to have enough events of every possible type happening, is given by

$$n_{rec} = 5 \prod_{i=1}^{n_{compl}} n_{lev_i} \tag{4.1}$$

where n_{rec} is the suggested number of recordings for the Data Model, n_{compl} is the maximum mask complexity of all masks that will be used on the Data Model, and

n_{lev_i} is the number of recoding levels of the ith sampling variable of the Behavior Model.

The number of columns in the Behavior Model matrix of a system is dictated by the complexity of the mask used to build it. In this thesis all optimal mask analyses will be carried on up to complexity level five. This limitation in the methodology is mainly imposed by the computational tools, but is perfectly acceptable since we will only consider single output masks in this work, allowing in this way up to four input sampling variables to influence one output.

Assuming that all optimal masks evaluated are of complexity five, the minimum number of recordings for a set of five variables, all recoded into three levels, is 1215, and our tools are more than capable of handling that. The two input variables are discrete variables with three states each, so there is no doubt about their recoding: they are already naturally recoded into three levels. Now, considering the outputs, we could recode them all into four levels which would require a Data Model size of 2880. This number is acceptably close to the maximum allowable number of data points (2500), but in the case of our problem, we want to recode the outputs into an odd number of levels, for the reason that a central level is desired where one state of the variable can be considered "normal". Using five levels for all three output variables would require a Data Model with 5625 recordings which is not currently possible to handle with the available tools. Recoding two of the outputs into five levels and one into three requires 3375 samples and two of them into three levels and one into five, requires 2025 data points.

We will first use 2500 data points from the shaken flight model, recode all variables into three states each, and then use the recoded data model to calculate the optimal masks.

Shaking the Airplane

The following Ctrl–C/ACSL interface code sets the simulation time to 15,000 seconds and the input selection switch *inpt* to 2, to choose the shaken flight phase.

```
[>  acsl('set tmx = 15000, inpt = 2')
[>  [pde, ptr, l, d, ga] = start;
```

The communication interval *cint* (equivalent to the sampling interval in this model) has a default value of 6 seconds yielding in this way a total of 2501 data points for all variables. The default values for the magnitude of the steps is $\Delta\delta_{e_{trim}} = 0.001$ rad and $\Delta T_{trim} = 3000$ lb.

The **start** command starts the simulation and the variables to be outputted are defined inside square brackets, on the left hand side of the command. Figure 4.2 shows plots of the shaken behavior of the output variables obtained through the Ctrl–C/ACSL interface code below:

```
[>  t = 0: 6: 15000;
[>  term = '4100';
[>  hard = 'tekf';
[>  window('211'), plot(t, l)
[>  xlabel('time [sec]',' lll  lll '), ylabel('lift [lb]',' lll  ll ')
[>  window('212'), plot(t, d)
[>  xlabel('time [sec]',' lll  lll '), ylabel('drag [lb]',' lll  ll ')
[>  replot
[>  erase, window('211'), plot(t, ga)
[>  xlabel('time [sec]',' lll  lll '), ylabel('g [rad]','g  lll ')
[>  replot
[>  quit
$ rename ctrlc.tekf figure4_2
```

where the first line of the code creates the support variable *t* since it had not been specified as an output for the ACSL simulation and it was needed for the plots.

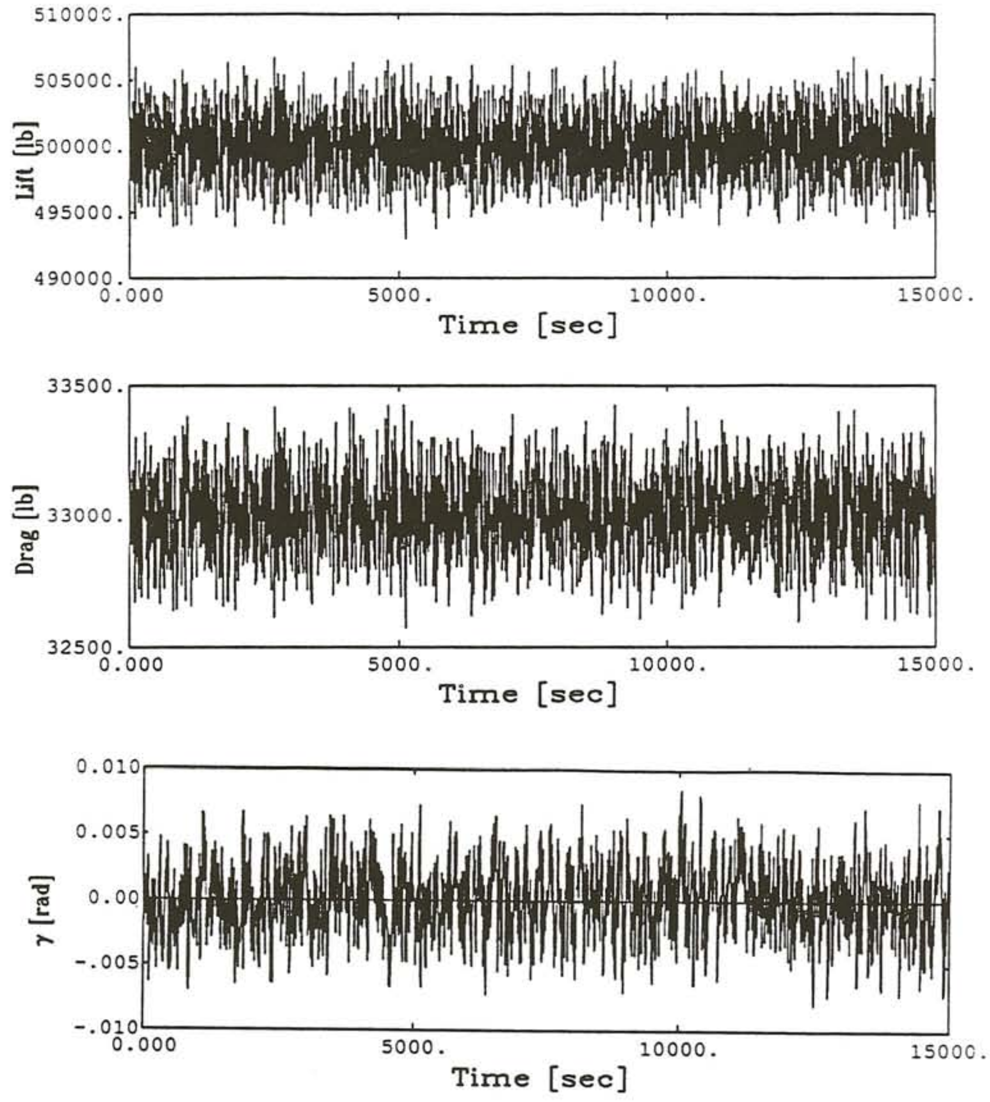

Figure 4.2 Shaken variables

Recoding

The inputs of the source model are the perturbations $\Delta\delta_{e_{trim}}$ and ΔT_{trim} affecting the model through step changes in the trimmed reference value of the elevator deflexion and of the thrust developed by the engines. The shaken flight phase has been implemented in the model such that these step changes occur at every communication interval being randomly negative, null, or positive, and, therefore, each of these variables can assume three different states. The state they will assume is dictated by *pde* and *ptr* which are the integer part of the numbers between 1.0000 and 3.9999 generated by two distinct (uniformly distributed) random number generators. Perturbations *pde* and *ptr* are already coded to { 1 2 3 } and are ready to be used, needing no recoding.

The first 20 randomly chosen states of the input variables are listed below:

```
[>  inrec = [pde, ptr];
[>  clear pde ptr
[>  in20 = inrec(1:20, :)

in20  =     1     2
            1     2
            2     3
            3     3
            3     2
            2     2
            3     3
            1     3
            1     1
            1     1
            2     2
            1     3
            3     2
            3     1
            2     3
            1     3
            1     3
            1     2
            3     1
            3     1
```

The output variables $\{ L \quad D \quad \gamma \}$ do require recoding since they are not discrete variables as the inputs. The number of recoding levels to be used for each variable has, intuitively, to be odd, if we want to have a 'normal' range of operation and variations about it. Let us try the analysis first with just 3 levels for each variable, and then decide if that gives meaningful enough results, or whether we should try a more complex data system with one or two of the output variables recoded into 5 levels (as mentioned in the previous section, we already know that recoding all three output variables into five levels would require a data model too large for the current version of SAPS–II).

User-defined recoding function rekode

The function **rekode** is based on a code written by Dr. François Cellier in September 1987, which he then called **nlrec**. His function calculates the number of levels to be used, based on the data size using expression (4.1) and applies the same number of levels to all variables. This feature is not used in **rekode**, which assumes the number of recoding levels to be three, but otherwise the code is the same. Both functions choose the limits of the recoding levels such that each of them contains the same amount of data points. This is done by sorting each variable completely, dividing the sorted vector into three equal parts, and choosing the limits as the first and last elements of each of these parts. These limits yield the from-matrix for each variable while the to-vector is the same for all: $to = [\ 1 \quad 2 \quad 3\ ]$. Then the standard SAPS–II function **recode** is used to recode each variable with the *'domain'* option and the respective from- and to- matrices. The following code lists the Ctrl–C/SAPS–II commands used by the function:

```
//[recdata, from] = rekode(rawdata)
[>  [nr, nv] = size(rawdata);
[>  from = zrow(1, 3);
[>  recdata = zrow(nr, 1);
[>  for i = 1: nv, ...
[>    [tag, d1] = sort(rawdata(:,i)); ...
[>    fr(1, 1) = d1(1); ...
[>     fr(2, 1) = 0.5*(d1(round(nr/3)) + d1(round(nr/3)+1)); ...
[>     fr(1, 2) = fr(2,1); ...
[>     fr(2, 2) = 0.5*(d1(round(2*nr/3)) + d1(round(2*nr/3)+1)); ...
[>     fr(1, 3) = fr(2,2); ...
[>     fr(2, 3) = d1(nr); ...
[>     r = recode(rawdata(:,i), 'domain', fr, 1:3); ...
[>     recdata = [recdata, r]; ...
[>     from = [from; fr]; ...
[>     end
[>   recdata = recdata(:, 2:nv+1);
[>   from = from(2:2*nv+1, :);
[>   return
```

The following Ctrl–C/SAPS–II code loads the SAPS-function library into the Ctrl–C workspace, loads the user-defined Ctrl–C/SAPS–II function **rekode** which will be used to recode the variables, and carries out the recoding of the three outputs:

```
[>   do saps:saps
[>   deff rekode
[>   xraw = [l, d, ga];
[>   clear l d ga
[>   [xrec, from] = rekode(xraw);
```

The outputs of the **rekode** recoding function are *xrec*, the recoded data model and *from*, a matrix formed by concatenating from below all three from-matrices: the first two rows compose fl (from-matrix for lift), third and fourth rows represent fd (from-matrix for drag), and the fifth and sixth rows compose the from-matrix for γ, fga.

```
[>   fl=from(1:2,:);
[>   fd=from(3:4,:);
[>   fga=from(5:6,:);
```

$$fl = \begin{pmatrix} 4.9352 & 4.9885 & 5.0110 \\ 4.9885 & 5.0110 & 5.0710 \end{pmatrix} \times 10^{+05}$$

$$fd = \begin{pmatrix} 3.2589 & 3.2936 & 3.3073 \\ 3.2936 & 3.3073 & 3.3425 \end{pmatrix} \times 10^{+04}$$

$$fga = \begin{pmatrix} -0.0078 & -0.0013 & 0.0013 \\ -0.0013 & 0.0013 & 0.0078 \end{pmatrix}$$

To illustrate the recoding of the raw data model, the first twenty elements of each output variable are plotted below in their raw and recoded forms:

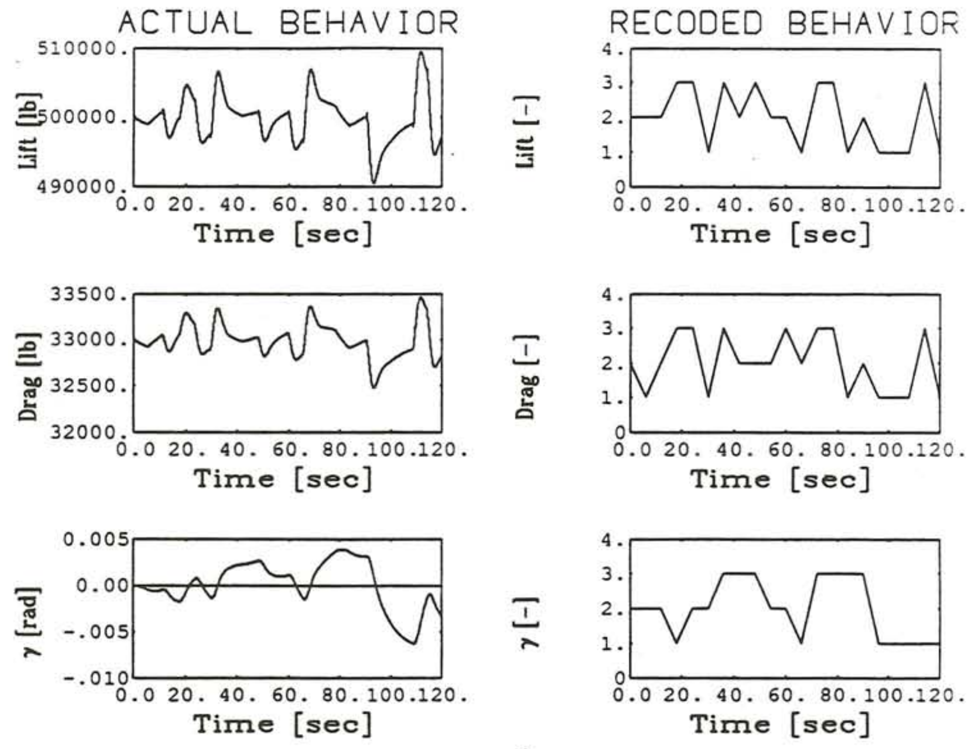

Figure 4.3 Recoding of the output variables

4.4 The Generative Model

Now, with the recoded data model at hand, we are ready to proceed with our task of building a qualitative model for the system through the evaluation of the optimal masks for it. The only information we have about the causal relationship among the variables stems from the source model where the basic variables were defined as inputs and outputs. Very little, however, is known about the 15 possible sampling variables available in a five variable mask of depth 2 that we are going to use. The difference here is that the sampling variables hold causal behavior knowledge not only about the variables, but about the variables at one, two, up to the depth of the mask sampling intervals behind in time. Therefore, there is not much help we can give to SAPS–II in terms of hinting which are the most probable causes to the effects observed in each of the outputs, or even which sampling variables definitively have no effect. The solution is to simply fill the mask candidate with -1s, meaning that any sampling variable is a possible input to the I/O Model the mask will generate. This will make the search for an optimal mask quite extensive but the optimal mask will certainly be found.

Intuitively, separate masks for each output variable give better results, since the magnitude of the effect of one influencing sampling variable may vary from one output to another, and, in this way, its influence on one of the outputs could easily be discarded, even if this were the most influencing sampling variable to that particular output.

Also, as we are interested in generating a tool with inductive capabilities, we are interested in how past behavior relates to future behavior, and therefore, our output sampling variables should be located in the mask candidates such that they are ahead in time from the possible inputs. As the convention adopted for the data model is that time increases when we move down in the data matrix, our outputs should be located in the bottom row of the masks.

We will also assume that present states of the outputs do not affect each other, but present states of the input variables may affect any of the outputs. The following set of mask candidates was used for the optimal mask evaluation:

$$mcan_l = \begin{pmatrix} -1 & -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & -1 & -1 \\ -1 & -1 & +1 & 0 & 0 \end{pmatrix}$$

$$mcan_d = \begin{pmatrix} -1 & -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & -1 & -1 \\ -1 & -1 & 0 & +1 & 0 \end{pmatrix}$$

$$mcan_\gamma = \begin{pmatrix} -1 & -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & -1 & -1 \\ -1 & -1 & 0 & 0 & +1 \end{pmatrix}$$

The following Ctrl–C/SAPS–II code loads the user-defined function **omhis** and evaluates the optimal masks using it:

```
[>  deff omhis
[>  [msk, hm, hr, q, mhis] = omhis([inrec, xrec], 5, 0);
```

Omhis is a user-defined Ctrl–C/SAPS–II function that uses the standard SAPS–II optimal mask analysis function **optmask** and was created specially for this problem. Its main purpose is to generate the mask candidates $mcan_l$, $mcan_d$ and $mcan_\gamma$ and run optimal mask analyses for each of the three mask candidates. Listed below is the code used:

```
//[msk, hm, hr, q, mhis] = omhis(rawrec, n, r)
mcan1 = [-ones(3, 2), [-ones(2, 3); zrow(1, 3)]];
mcan2 = mcan1;
```

```
mcan3 = mcan1;
mcan1(3, 3) = 1;
mcan2(3, 4) = 1;
mcan3(3, 5) = 1;
repo = r;
[msk1, hm1, hr1, q1, mhis1] = optmask(rawrec, mcan1, n);
[msk2, hm2, hr2, q2, mhis2] = optmask(rawrec, mcan2, n);
[msk3, hm3, hr3, q3, mhis3] = optmask(rawrec, mcan3, n);
msk = [msk1; msk2; msk3];
hm = [hm1; hm2; hm3];
hr = [hr1; hr2; hr3];
q = [q1; q2; q3];
mhis = [mhis1; mhis2; mhis3];
return
```

The user must specify the maximum level of mask complexity (defined as the number of non-zero elements present in a mask) up to which the search is to be carried out, and she/he is also given the option of requesting a detailed listing of the analysis. In the current version of the software, a data model containing five variables recoded into three levels must not have more than 2500 rows. The inputs required by this funtion are: a recoded data matrix *rawrec*, an integer value *n*, and another integer value *r*.

The integer *n* specifies the maximum complexity level up to which the optimal mask evaluation should be carried out. In this analysis, we chose not to go beyond 5 because the higher complexity analyses generate too large Behavior Systems that are restrictive for the presently available tools. As has already been discussed in the fifteen-element, single output masks that we are using in this thesis, the relationships should reliably representable through not more than four input sampling variables.

The user may request a detailed report file which lists all intermediary results of the optimal mask evaluation at every complexity level. This is done by entering "1" in the function's input parameter r. Due to the sheer volume of this listing, it is recommended that r (or SAPS–II global variable *repo*) is normally set to "0".

The outputs of the function are five: msk, hm, hr, q, and $mhis$, giving in this way a summary of the search at different levels of mask complexity. A more complete listing, as mentioned above, is available by setting r to 1.

Output msk is a matrix containing the three optimal masks, concatenated to each other vertically from below. Vector hm is the entropy vector, containing the Shannon entropy relative to each of the optimal masks found at every complexity level, hr represents the normalized entropy reduction of the same masks, and q their quality factor. These three vectors can be divided into three equal parts, each of which refers to one of the masks in the same order as they appear in the msk matrix. The elements of each part refer to complexity levels 2, 3, 4, and 5 for each mask.

Matrix $mhis$ is the optimal mask history matrix containing all the sub-optimal masks found at all complexity levels considered; they are concatenated vertically from below for masks with different outputs, and horizontally from the right for increasing mask complexities.

Listed below are all the results outputted by the optimal mask analysis. The results were edited and renamed to improve readability and to allow possible future reference to the different parts of each output:

$$msk = [m_l;\ m_d;\ m_\gamma]$$

$$m_l = \begin{pmatrix} -1. & 0. & 0. & 0. & -2. \\ -3. & -4. & 0. & 0. & 0. \\ 0. & 0. & +1. & 0. & 0. \end{pmatrix}$$

$$m_d = \begin{pmatrix} -1. & 0. & 0. & 0. & 0. \\ -2. & -3. & 0. & -4. & 0. \\ 0. & 0. & 0. & +1. & 0. \end{pmatrix}$$

$$m_\gamma = \begin{pmatrix} 0. & 0. & 0. & 0. & 0. \\ -1. & 0. & 0. & 0. & -2. \\ 0. & 0. & 0. & 0. & +1. \end{pmatrix}$$

The Shannon entropy vector hm, the normalized entropy reduction vector hr, and the quality vector q will be concatenated ot each other from the right and will be displayed in one single matrix for each optimal mask. We called them hq_l, hq_d, and hq_γ:

$$\begin{array}{ccc} \text{hr} & \text{hm} & \text{q} \end{array}$$

$$hq_l = \begin{pmatrix} 0.1828 & 1.2953 & 0.0548 \\ 0.4584 & 0.8583 & 0.0917 \\ 0.6517 & 0.5520 & 0.0978 \\ 0.8326 & 0.2653 & 0.0999 \end{pmatrix}$$

$$hq_d = \begin{pmatrix} 0.1520 & 1.3441 & 0.0456 \\ 0.3187 & 1.0799 & 0.0637 \\ 0.5604 & 0.6967 & 0.0841 \\ 0.7737 & 0.3587 & 0.0928 \end{pmatrix}$$

$$hq_\gamma = \begin{pmatrix} 0.2289 & 1.2221 & 0.0687 \\ 0.4404 & 0.8870 & 0.0881 \\ 0.5201 & 0.7606 & 0.0780 \\ 0.6245 & 0.5951 & 0.0749 \end{pmatrix}$$

The last output of the function is the optimal mask history matrix $mhis$ which will be displayed here mask by mask instead of as one large matrix containing all the matrices, as is the output given by SAPS–II. Masks marked with an asterisk * are the optimal masks throughout all complexity levels:

$$mhis = \begin{pmatrix} m_{l2} & m_{l3} & m_{l4} & m_{l5} \\ m_{d2} & m_{d3} & m_{d4} & m_{d5} \\ m_{\gamma 2} & m_{\gamma 3} & m_{\gamma 4} & m_{\gamma 5} \end{pmatrix}$$

$$m_{l2} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & +1 & 0 & 0 \end{pmatrix}$$

$$m_{l3} = \begin{pmatrix} -1 & 0 & 0 & 0 & 0 \\ -2 & 0 & 0 & 0 & 0 \\ 0 & 0 & +1 & 0 & 0 \end{pmatrix}$$

$$m_{l4} = \begin{pmatrix} -1 & 0 & 0 & 0 & 0 \\ -2 & -3 & 0 & 0 & 0 \\ 0 & 0 & +1 & 0 & 0 \end{pmatrix}$$

$$m_{l5} = \begin{pmatrix} -1 & 0 & 0 & 0 & -2 \\ -3 & -4 & 0 & 0 & 0 \\ 0 & 0 & +1 & 0 & 0 \end{pmatrix} *$$

$$m_{d2} = \begin{pmatrix} -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & +1 & 0 \end{pmatrix}$$

$$m_{d3} = \begin{pmatrix} -1 & 0 & 0 & 0 & 0 \\ 0 & -2 & 0 & 0 & 0 \\ 0 & 0 & 0 & +1 & 0 \end{pmatrix}$$

$$m_{d4} = \begin{pmatrix} -1 & 0 & 0 & 0 & 0 \\ -2 & -3 & 0 & 0 & 0 \\ 0 & 0 & 0 & +1 & 0 \end{pmatrix}$$

$$m_{d5} = \begin{pmatrix} -1 & 0 & 0 & 0 & 0 \\ -2 & -3 & 0 & -4 & 0 \\ 0 & 0 & 0 & +1 & 0 \end{pmatrix} *$$

$$m_{\gamma 2} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & +1 \end{pmatrix}$$

$$m_{\gamma 3} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & -2 \\ 0 & 0 & 0 & 0 & +1 \end{pmatrix} *$$

$$m_{\gamma 4} = \begin{pmatrix} 0 & 0 & 0 & 0 & -1 \\ -2 & 0 & 0 & -3 & 0 \\ 0 & 0 & 0 & 0 & +1 \end{pmatrix}$$

$$m_{\gamma 5} = \begin{pmatrix} -1 & 0 & 0 & 0 & 0 \\ -2 & 0 & 0 & -3 & -4 \\ 0 & 0 & 0 & 0 & +1 \end{pmatrix}$$

Finally, the from-matrices used for the recoding of the Data Model used in the optimal mask analysis are saved along with the optimal masks in the file **fmasks**.dat for future use. The second best masks (with second highest quality factor) are saved in a separate file **amasks**.dat for a future study of their effectiveness as forecasting tools.

```
[>  m1 = msk(1:3, :);
[>  m2 = msk(4:6, :);
[>  m3 = msk(7:9, :);
[>  m1a = mhis(1:3, 11:15);
[>  m2a = mhis(4:6, 11:15);
[>  m3a = mhis(7:9, 11:15);
[>  save fl fd fga m1 m2 m3 >fmasks
[>  save m1a m2a m3a >amasks
```

Analysing the Optimal Masks

One way of checking if the masks really represent the system behavior is to recalculate them with another random input stream.

The ACSL simulation software allows the user to change the seed of the random number generator by setting the variable *seed* to the any desired value. It has a default value of 555 in the program, and the ACSL User's Manual (Mitchell and Gauthier, 1986, page 4-23) suggests that *seed* be set to an odd interger value in order to obtain a maximal length sequence, and the value should be large to avoid high correlation among the the first five to ten elements of the stream.

We ran the model once more, now with a different seed, as shown in the code below:

```
[>  acsl('set tmx=15000, inpt=2, seed=1177')
[>  [pde, ptr, l, d, ga] = start;
```

The recoding is done in exactly the same manner as in the previous section as shows the following Ctrl–C/SAPS–II code:

```
[>  inrec = [pde, ptr];
[>  clear pde ptr
[>  xraw = [l, d, ga];
[>  clear l d ga
[>  deff rekode
[>  [xrec, from] = rekode([xraw]);
```

The resulting from-matrices are very similar to the previous ones, such that all the level boundaries are more or less at the same points as before. Small differences are expected since the random stream has changed, but the states the input variables assume are still random, uniformly distributed as 1s, 2s or 3s, and the magnitude of the steps is still the same.

$$fl = \begin{pmatrix} 4.9314 & 4.9891 & 5.0107 \\ 4.9891 & 5.0107 & 5.0659 \end{pmatrix} \times 10^{+05}$$

$$fd = \begin{pmatrix} 3.2590 & 3.2937 & 3.3072 \\ 3.2937 & 3.3072 & 3.3392 \end{pmatrix} \times 10^{+04}$$

$$f\gamma = \begin{pmatrix} -0.0076 & -0.0012 & 0.0012 \\ -0.0012 & 0.0012 & 0.0081 \end{pmatrix}$$

Figure 4.4 below shows graphically the first 20 data points of the two inputs *pde* and *ptr* and also depicts the actually measured values of the three outputs l, d and γ compared to their recodings.

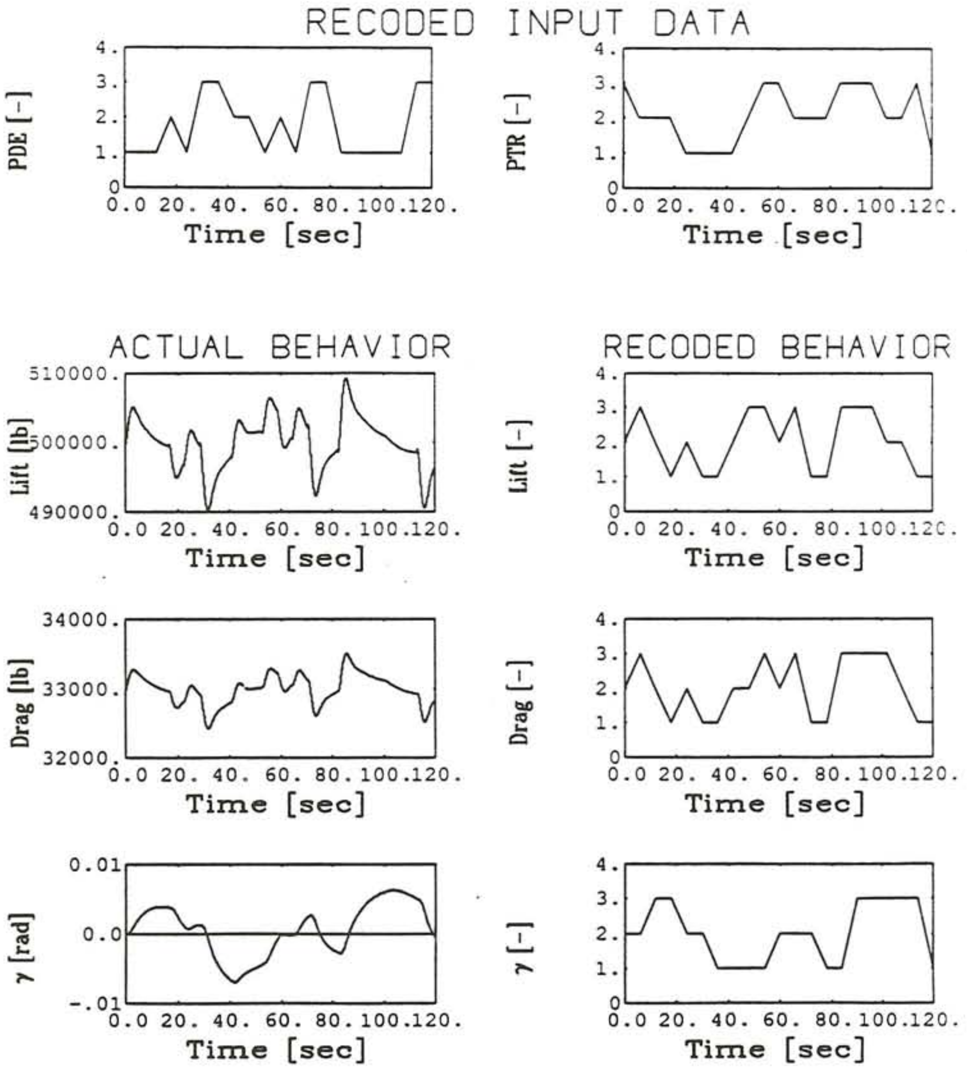

Figure 4.4 Recoding of the second data model

The second optimal mask analysis is run in the same manner as before, yielding very similar results, confirming the results of the first optimal mask analysis:

```
[>  deff omhis
[>  [msk, hm, hr, q, mhis] = omhis([inrec, xrec], 5, 0);
```

The optimal masks are exactly the same:

$$m_l = \begin{pmatrix} -1. & 0. & 0. & 0. & -2. \\ -3. & -4. & 0. & 0. & 0. \\ 0. & 0. & +1. & 0. & 0. \end{pmatrix}$$

$$m_d = \begin{pmatrix} -1. & 0. & 0. & 0. & 0. \\ -2. & -3. & 0. & -4. & 0. \\ 0. & 0. & 0. & +1. & 0. \end{pmatrix}$$

$$m_\gamma = \begin{pmatrix} 0. & 0. & 0. & 0. & 0. \\ -1. & 0. & 0. & 0. & -2. \\ 0. & 0. & 0. & 0. & +1. \end{pmatrix}$$

The normalized entropy reductions, Shannon entropies and the quality factors relative to the best masks at complexity levels 2 through 5 changed a little, since the Data Model has changed, but as its behavior has not changed, these factors are still about the same:

$$\begin{array}{ccc} \text{hr} & \text{hm} & \text{q} \end{array}$$

$$hq_l = \begin{pmatrix} 0.1997 & 1.2685 & 0.0599 \\ 0.4539 & 0.8656 & 0.0908 \\ 0.6534 & 0.5494 & 0.0980 \\ 0.8230 & 0.2806 & 0.0988 \end{pmatrix}$$

$$hq_d = \begin{pmatrix} 0.1497 & 1.3477 & 0.0449 \\ 0.3065 & 1.0991 & 0.0613 \\ 0.5565 & 0.7029 & 0.0835 \\ 0.7796 & 0.3493 & 0.0936 \end{pmatrix}$$

$$hq_\gamma = \begin{pmatrix} 0.2004 & 1.2674 & 0.0601 \\ 0.4060 & 0.9414 & 0.0812 \\ 0.4960 & 0.7989 & 0.0774 \\ 0.5990 & 0.6355 & 0.0719 \end{pmatrix}$$

And the best masks found at complexity levels 2 through 5 are (again "$*$" stands for optimal mask throughout all complexity levels):

$$mhis = \begin{pmatrix} m_{l2} & m_{l3} & m_{l4} & m_{l5} \\ m_{d2} & m_{d3} & m_{d4} & m_{d5} \\ m_{\gamma 2} & m_{\gamma 3} & m_{\gamma 4} & m_{\gamma 5} \end{pmatrix}$$

$$m_{l2} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & +1 & 0 & 0 \end{pmatrix}$$

$$m_{l3} = \begin{pmatrix} -1 & 0 & 0 & 0 & 0 \\ -2 & 0 & 0 & 0 & 0 \\ 0 & 0 & +1 & 0 & 0 \end{pmatrix}$$

$$m_{l4} = \begin{pmatrix} -1 & 0 & 0 & 0 & 0 \\ -2 & -3 & 0 & 0 & 0 \\ 0 & 0 & +1 & 0 & 0 \end{pmatrix}$$

$$m_{l5} = \begin{pmatrix} -1 & 0 & 0 & 0 & -2 \\ -3 & -4 & 0 & 0 & 0 \\ 0 & 0 & +1 & 0 & 0 \end{pmatrix}$$

$$m_{d2} = \begin{pmatrix} -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & +1 & 0 \end{pmatrix}$$

$$m_{d3} = \begin{pmatrix} -1 & 0 & 0 & 0 & 0 \\ -2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & +1 & 0 \end{pmatrix}$$

$$m_{d4} = \begin{pmatrix} -1 & 0 & 0 & 0 & 0 \\ -2 & -3 & 0 & 0 & 0 \\ 0 & 0 & 0 & +1 & 0 \end{pmatrix}$$

$$m_{d5} = \begin{pmatrix} -1 & 0 & 0 & 0 & 0 \\ -2 & -3 & 0 & -4 & 0 \\ 0 & 0 & 0 & +1 & 0 \end{pmatrix} *$$

$$m_{\gamma 2} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & +1 \end{pmatrix}$$

$$m_{\gamma 3} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & -2 \\ 0 & 0 & 0 & 0 & +1 \end{pmatrix} *$$

$$m_{\gamma 4} = \begin{pmatrix} 0 & 0 & 0 & 0 & -1 \\ -2 & 0 & 0 & -3 & 0 \\ 0 & 0 & 0 & 0 & +1 \end{pmatrix}$$

$$m_{\gamma 5} = \begin{pmatrix} -1 & 0 & 0 & 0 & 0 \\ -2 & 0 & 0 & -3 & -4 \\ 0 & 0 & 0 & 0 & +1 \end{pmatrix}$$

Note that the second best mask generating the fourth state variable d is slightly different: the second input sampling variable is the sixth element of the mask (numbering the elements of the mask from left to right, starting at the top left corner), instead of #7 as in the first analysis. Analysing the whole mask history (setting the **omhis** function's input parameter r (or the global variable *repo*) to 1, it is found that mask

$$\begin{pmatrix} -1 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & +1 & 0 \end{pmatrix}$$

was the second best mask at complexity level 3 , with quality factor .603, while mask

$$\begin{pmatrix} -1 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & +1 & 0 \end{pmatrix}$$

was the best mask at that level, with a quality factor of .637 ($\approx 6\%$ larger). Therefore it is perfectly understandable and acceptable that the mask histories are not exactly the same, since the differing masks have comparable quality factors.

4.5 Forecasting

Now let us apply the generative systems represented by the optimal masks just calculated, to see how well they are able to forecast future behavior of the same system.

To do so, we will use a less shaken data model to represent a normal flight in a more substantial way. A more realistic flight, but still a dynamic one, can be represented by the model being driven by harmonic functions of fairly long periods, which was implemented by sinusoidal functions affecting the same trim values $\delta_{e_{trim}}$ and T_{trim}. The amplitude of the functions is the same as the magnitude of the respective steps in the shaken flight: $\Delta\delta_e = .001$ rad and $\Delta T = 3000$ lb. In this flight phase the angular velocity of the driving functions *pde* and *ptr* can be reset with the parameters *wde* for the elevator deflexion and *wtr* for the thrust. However their default values: $wde = 0.1$ rad/sec and $wtr = 0.05$ rad/sec, yielding periods of about 63 and 126 sec. respectively, were used in the following run. The length of the flight will be 15,600 seconds, generating, at a sampling interval of 6 sec., a data model with 2601 recordings. The driving inputs *pde* and *ptr* are calculated as:

$$pde = \sin(wde * t)$$
$$ptr = \sin(wtr * t)$$

The following Ctrl–C/ACSL code sets the flight phase switch *inpt* to 3 (harmonic perturbations) and the maximum simulation time *tmx* to 15,600.

```
[>  acsl('set tmx = 15600, inpt = 3');
[>  [pde, ptr, l, d, ga] = start;
```

Recoding of the data model was done again into three levels for all variables. The input variables now need recoding as well, since their states are no longer discrete values. However, as both inputs are harmonic, their recoding can be done evenly in their range of angular variation:

$$\begin{array}{ll} \text{level1}: & [0,\ 2\pi/3)\ \ \text{rad} \\ \text{level2}: & [2\pi/3,\ 4\pi/3)\ \ \text{rad} \\ \text{level3}: & [4\pi/3,\ 2\pi \equiv 0)\ \ \text{rad} \end{array}$$

which is equivalent to the recoding performed with the Ctrl–C/SAPS–II user defined function **rekode** already discussed in the previous section.

The following Ctrl–C/SAPS–II code loads the user-defined recoding function **rekode** and recodes the inputs into three levels such that each level holds the same amount of data points. The from-matrices used to recode the input variables are represented by fin which is built such that

$$fin = \begin{pmatrix} fpde \\ fptr \end{pmatrix}$$

and the recoded input matrix is $inrec$, the first column of which holds the recoded data points of *pde* while the second column holds those of *ptr*.

```
[>  deff rekode
[>  [inrec, fin] = rekode([pde, ptr]);
```

$$fin = \begin{pmatrix} -1.0000 & -0.4987 & 0.5011 \\ -0.4987 & 0.5011 & 1.0000 \\ -1.0000 & -0.4987 & 0.5002 \\ -0.4987 & 0.5002 & 1.0000 \end{pmatrix}$$

Recoding the output variables requires a little more insight. The inner limits of the recoding levels that were used to recode the shaken output data in the optimal mask analysis should be used here as well, since we are going to use those masks. As the driving step perturbations are longer now, the amplitudes of the responses must have incresed and therefore the outer limits of the from-matrices should be adjusted to the new data so that every measured data point would be within the limits of one level. This is very important, since the recoding function will not recognize values outside the limits specified in the from-matrices.

In the following experiment, we will recode the outputs of the generated (harmonic driven) data model with the from-matrices obtained for recoding the shaken data model, and apply the forecasting capabilities of SAPS–II to see how well the generative system can draw inferences about the system behavior.

The from-matrices used in the optimal mask analysis (fl, fd, and fga) can be recalled into the Ctrl–C workspace with the Ctrl–C **load** command. The following listing adjusts the outer limits of the optimal masks to the data model they will now be used for:

```
[>  load fl fd fga <fmasks
[>  fl(1, 1) = min([l; fl(1, 1)]);
[>  fl(2, 3) = max([l; fl(2, 3)]);
[>  fd(1, 1) = min([d; fd(1, 1)]);
[>  fd(2, 3) = max([d; fd(2, 3)]);
[>  fga(1, 1) = min([ga; fga(1, 1)]);
[>  fga(2, 3) = max([ga; fga(2, 3)]);
```

$$fl = \begin{pmatrix} 4.9685 & 4.9885 & 5.0110 \\ 4.9885 & 5.0110 & 5.0349 \end{pmatrix} \times 10^{+05}$$

$$fd = \begin{pmatrix} 3.2838 & 3.2936 & 3.3073 \\ 3.2936 & 3.3073 & 3.3298 \end{pmatrix} \times 10^{+04}$$

$$fga = \begin{pmatrix} -0.0052 & -0.0013 & 0.0013 \\ -0.0013 & 0.0013 & 0.0087 \end{pmatrix}$$

Matrix *xrec* below is the recoded outputs' matrix obtained recoding the raw output variables with the standard SAPS–II **recode** function with its *'domain'* option.

```
[>  lrec = recode(l, 'domain', fl, 1:3);
[>  drec = recode(d, 'domain', fd, 1:3);
[>  garec = recode(ga 'domain', fga, 1:3);
[>  xrec = [lrec, drec, garec];
[>  clear lrec drec garec
```

Now with the data model composed of the recoded data matrices *inrec* and *xrec* just evaluated and the system's optimal masks, we can use GSPS's inductive reasoning feature (SAPS–II **forecast** function) to see how well it guesses which states the output variables will assume in the ten following sampling intervals, given a certain set of inputs for those data points.

We will use the user defined Ctrl–C/SAPS–II function **frc** in the repeated forecasting procedure. This function was especially coded for this problem by Dr. Cellier in September 1987, and its first version was first published in (Cellier, 1987). It uses the standard SAPS–II **forecast** function three times for every time instant, once with each optimal mask, to forecast the state assumed by each output variable at that time instant. Forecasting will only be terminated when states for all three outputs have been forecast for all requested time instants. The minimum acceptable probability associated with the forecasting is set to zero, and a state is invented if its probability of occurrence is zero, i.e., if the state has never occurred before in the available data model.

The function requires 6 inputs: $past, inpt, m1, m2, m3$ and nl and has one output $frcst$. Matrix *past* contains the recoded data model of the system under consideration up to the time instant from which we want to start **forecasting** future behavior. Matrix *inpt* is a matrix containing the states of the inputs at the time instants we want the forecasting to take place and the number of rows of this matrix specifies how many time intervals ahead we want to forecast. Inputs $m1, m2$, and $m3$ are the optimal masks relative to the first, second and third output variables (l, d, and γ), respectively, and *nl* is an integer value which specifies the number of recoding levels the outputs were recoded into.

This last datum, *nl*, is needed for situations where forecasting is not possible and it is needed to invent a state (generate the state randomly). Such situations occur when, due to high optimal mask complexity and/or a too small data model, a mask reads a state that has never occurred before in the data model represented by the *past* matrix; the probability of this event to occur is zero and the repeated forecasting process cannot be continued. To avoid the abortion of the forecasting process, the new state is invented with the aid of a random number generator, and the forecasting will continue until all requested states have been computed as specified by the size of the *inpt* matrix. Listed below are both functions: the continuous forecasting function **frc** and its state-inventing function **manicure** (also written by Dr. Cellier, in September 1987).

```
//[frcst, p] = frc(r, inp, m1, m2, m3, nl)
//
//Forecast behavior of linear four variable system
//
pmin = 0;
p = zrow(1,3);
deff manicure
[n, m] = size(inp);
for i = 1: n, ...
  [row, col] = size(r); ...
  in = inp(i, :); ...
  fc = [in, zrow(1,3)]; ...
  fcc = [r; fc]; ...
  [ff1, p1] = forecast(fcc, m1, row, pmin); ...
  [ff1, p1] = manicure(ff1, r, 3, nl, p1); ...
```

```
   [ff2, p2] = forecast(fcc, m2, row, pmin); ...
   [ff2, p2] = manicure(ff2, r, 4, nl, p2); ...
   [ff3, p3] = forecast(fcc, m3, row, pmin); ...
   [ff3, p3] = manicure(ff3, r, 5, nl, p3); ...
   ff = [in, ff1(row+1, 3), ff2(row+1, 4), ff3(row+1, 5)]; ...
   r = [r; ff]; ...
   p = [p;[p1(row+1),p2(row+1),p3(row+1)]];...
   end
[mr, nr] = size(r);
frcst = r(mr-n+1: mr, :);
p = p(2:n+1,:);
return


// [f2, p] = manicure(f1, raw, nbr, nl, pp)
//
// If no forecast was possible invent one.
//
[nf, mf] = size(f1);
[nr, mr] = size(raw);
if nf=nr+1, f2 = f1; p = pp; ...
   else ...
   display('Warning, no forecast was possible'), ...
   location = [nr+1, nbr], ...
   invent = round((nl-.0002)*rand(1)+.50001); ...
   f2 = zrow(nr+1, mr); ...
   f2(1: nr, :) = raw; ...
   f2(nr+1, nbr) = invent; ...
   p=[pp; pp(nr)/nl];...
   end
return
```

We will use a data model with 2601 elements in each of the five variables out of which the first 2500 will represent the past behavior required by the forecasting procedure, which is close to the maximum number of samples that the **frc** function currently can handle. For the data model at hand, the number of samples is perfectly acceptable since the maximum number of distinct states of the system is $n = 3^5 = 243$ and therefore, in cases when the uncertainty is maximum (the probability of occurrence of all states is the same), we could have at least ten recordings of each state in a data model with 2601 sets of samples.

The Ctrl–C/SAPS–II code below loads the recoding function **frc** into the workspace, defines the *past* matrix as being composed of the first 2500 states of the recoded data matrix just evaluated. The *inpt* matrix is formed by the next ten states of the inputs, meaning that we want to forecast the states of the outputs ten sampling intervals ahead. Matrix *future* contains the actually measured states of the outputs relative to those time instants and will be of use in evaluating the quality of the forecasting, comparing measured and forecast values.

```
[>  deff frc
[>  load m1 m2 m3 <fmasks
[>  past = [inrec, xrec];
[>  past = past(1:2500, :);
[>  inpt = inrec(2501:2510, :);
[>  future = xrec(2501:2510, :);
[>  frcst = frc(past, inpt, m1, m2, m3, 3);
[>  error = frcst(:, 3:5)−future;
[>  infut = [inpt, future];
```

The optimal masks of the system $m1, m2$ and $m3$ were recalled to the workspace with the Ctrl–C **load** command and the Data Model was divided into past behavior and future behavior, data sample 2500 being the limit.

The output of function **frc**, *frcst*, has ten rows (relative to the ten forecast time instants) and five columns: the two first ones represent the input state variables (identical to *inpt*) and the next three columns represent the forecast outputs. Elements marked with an asterisk refer to states that had to be invented with the **manicure** function.

$$infut = \begin{pmatrix} 1. & 3. & 3. & 3. & 3. \\ 1. & 3. & 2. & 3. & 3. \\ 2. & 2. & 2. & 2. & 3. \\ 2. & 2. & 1. & 1. & 3. \\ 3. & 2. & 1. & 1. & 3. \\ 3. & 1. & 1. & 1. & 2. \\ 3. & 1. & 1. & 1. & 1. \\ 3. & 1. & 2. & 1. & 1. \\ 2. & 1. & 2. & 2. & 1. \\ 1. & 1. & 2. & 2. & 1. \end{pmatrix}$$

$$frcst = \begin{pmatrix} 1. & 3. & 3. & 3. & 3. \\ 1. & 3. & 2. & 3. & 3. \\ 2. & 2. & 2. & 3. & 3. \\ 2. & 2. & 1. & 1. & 3. \\ 3. & 2. & 1. & 1. & 3. \\ 3. & 1. & 1. & 1. & 2. \\ 3. & 1. & 1. & 1. & 1. \\ 3. & 1. & 2. & 1. & 1. \\ 2. & 1. & 2. & 1. & 1. \\ 1. & 1. & 2. & 3. & 1. \end{pmatrix}$$

$$error = \begin{pmatrix} 0. & 0. & 0. \\ 0. & 0. & 0. \\ 0. & 1. & 0. \\ 0. & 0.^{*} & 0. \\ 0. & 0. & 0. \\ 0. & 0. & 0. \\ 0. & 0. & 0. \\ 0. & 0. & 0. \\ 0. & -1. & 0. \\ 0. & 1.^{*} & 0. \end{pmatrix}$$

Note that, even with a data model more than ten times larger than the maximum possible number of combinations of the states, the forecasting mechanism failed twice to forecast the state of the fourth state variable, Drag. The states of the samples (2504, 4) and of (2510, 4) had to be guessed, and therefore, the meaning of the non-zero element in the *error* matrix (2510, 4) is different from the other two at (2503, 4) and (2409, 4), which are bad forecasts.

The unability of a mask to forecast a state is strongly related to the complexity of the mask used: the higher the mask complexity, the more likely will the mask read a set of states that has never occurred before in the recorded time history, and if one wants to continue the forecasting process, this state has to be guessed by some other means. Our program picks an arbitrary value within the allowed range using the routine **manicure**.

But this is not the reason why the forecasting mechanism could twice not forecast future states for the fourth state variable. Note that right before the unability to forecast, one time instant behind, in both of the cases there had occurred a bad forecasting (non-zero elements (2503, 4) and (2509, 4) in the *error* matrix). The fourth generative sampling variable of $m2$ is exactly the state of the fourth state variable (of the data matrix) one time instant behind of the forecasting time. So, what happens here is that the past bad forecasting affects the next forecasting in such a way that a totally unknown set of states appears in the behavior model, forcing the forecasting mechanism to guess, out of the blue sky, the next state of the fourth state variable.

Another forecasting was evaluated using the same data matrix, now fifty time instants ahead, from the 2551th data points up to the 2560th. This time the forecasting of the last two elements of the third state variable (Lift) was not possible, very likely, due to, previous poor forecastings of state variable 5 (γ).

```
[>  past = [inrec, xrec];
[>  past = past(1:2550, :);
[>  inpt = inrec(2551:2560, :);
[>  future = xrec(2551:2560, :);
[>  frcst = frc(past, inpt, m1, m2, m3, 3);
[>  error = frcst(:, 3:5)-future;
[>  infut = [inpt, future];
```

$$infut = \begin{pmatrix} 2. & 1. & 2. & 2. & 1. \\ 1. & 1. & 2. & 2. & 1. \\ 1. & 1. & 2. & 2. & 2. \\ 1. & 1. & 2. & 1. & 2. \\ 1. & 2. & 2. & 1. & 2. \\ 2. & 2. & 2. & 1. & 1. \\ 2. & 2. & 2. & 1. & 1. \\ 3. & 3. & 2. & 1. & 1. \\ 3. & 3. & 2. & 2. & 1. \\ 3. & 3. & 3. & 3. & 1. \end{pmatrix}$$

$$frcst = \begin{pmatrix} 2. & 1. & 2. & 2. & 1. \\ 1. & 1. & 2. & 2. & 1. \\ 1. & 1. & 2. & 2. & 2. \\ 1. & 1. & 2. & 1. & 2. \\ 1. & 2. & 2. & 1. & 2. \\ 2. & 2. & 2. & 1. & 2. \\ 2. & 2. & 2. & 1. & 3. \\ 3. & 3. & 2. & 1. & 3. \\ 3. & 3. & 1. & 2. & 2. \\ 3. & 3. & 1. & 3. & 1. \end{pmatrix}$$

$$error = \begin{pmatrix} 0. & 0. & 0. \\ 0. & 0. & 0. \\ 0. & 0. & 0. \\ 0. & 0. & 0. \\ 0. & 0. & 0. \\ 0. & 0. & 1. \\ 0. & 0. & 2. \\ 0. & 0. & 2. \\ -1.^{*} & 0. & 1. \\ -2.^{*} & 0. & 0. \end{pmatrix}$$

A new forecasting was evaluated, now with the sub-optimal mask $m3a$ related to the fifth state variable. Even having the optimal mask $m3$ been evaluated as the best generative mask to forecast the next state of γ, its efficiency is being questioned based on the poor forecasting results. In reality, the mask looks a little too simple, being of complexity 3, and not having much interaction between the variables: $m3$ suggests that the next future state of the fifth state variable depends only on the state of the first state variable (*pde*, the perturbation caused by a change in the trimmed value of the elevator deflexion) in the present time and on the state of the fifth state variable itself (γ) in the present time instant as well. The sub-optimal mask $m3a$ looks more complete since it includes the influence of Drag (state variable 4), and we know that changes in the Drag do affect the flight path angle (changes in Thrust affect Drag, and Drag affects flight path angle). The results of the forecasting using $m3a$ reinforces our belief that $m3a$ is a better forecasting tool for γ than $m3$, even though it has a lower quality factor associated with it.

```
[>  load m3a <amasks
[>  frcst = frc(past, inpt, m1, m2, m3a, 3);
[>  error = frcst(:, 3:5) − future;
```

$$m3a = \begin{pmatrix} 0. & 0. & 0. & 0. & -1. \\ -2. & 0. & 0. & -3. & 0. \\ 0. & 0. & 0. & 0. & +1. \end{pmatrix}$$

$$error = \begin{pmatrix} 0. & 0. & 0. \\ 0. & 0. & 0. \\ 0. & 0. & 0. \\ 0. & 0. & 0. \\ 0. & 0. & -1. \\ 0. & 0. & 0. \\ 0.^{*} & 0. & 0. \\ 0. & 0. & 0. \\ 0. & 0. & 0. \\ 0. & 0. & 0. \end{pmatrix}$$

Again it so happened that one state of the third state variable could not be forecast, but the reason most likely is that the state of the sample (2555, 5) was a bad forecast of the fifth state variable, making it impossible for $m1$ to predict (2557, 3). In the *error* matrix, (2557, 3) is a zero element, but it is still a guessed element and has 66% of chance of not matching the actual measured state of that variable.

The same experiments were made for a different data-model which was created by driving the model with still harmonic functions, but now with different frequencies: $wde = 0.15$ and $wtr = 0.10$, i.e., periods of 42 sec. for *pde* and of 63 sec. for *ptr*.

```
[> acsl('set inpt=3, tmx=15600, wde=0.15, wtr=0.1')
[> [pde,ptr,l,d,ga] = start;
[> deff rekode
[> [inrec, fin] = rekode([pde, ptr]);
[> load fl fd fga <fmasks
[> fl(1, 1) = min([l, fl(1, 1)]);
[> fl(2, 3) = max([l, fl(2, 3)]);
[> fd(1, 1) = min([d, fd(1, 1)]);
[> fd(2, 3) = max([d, fd(2, 3)]);
[> fga(1, 1) = min([ga, fga(1, 1)]);
[> fga(2, 3) = max([ga, fga(2, 3)]);
[> lrec = recode(l, 'domain', fl, 1:3);
[> drec = recode(d, 'domain', fd, 1:3);
[> garec = recode(ga, 'domain', fga, 1:3);
[> xrec = [lrec, drec, garec];
[> deff frc
[> load <fmasks
[> load <amasks
[> past = [inrec, xrec];
[> past = past(1:2500, :);
[> inpt = inrec(2501:2510, :);
[> future = xrec(2501:2510, :);
[> infut = [inpt, future];
```

$$infut = \begin{pmatrix} 3. & 1. & 1. & 1. & 3. \\ 3. & 1. & 1. & 1. & 2. \\ 3. & 2. & 1. & 1. & 1. \\ 2. & 2. & 2. & 2. & 1. \\ 1. & 3. & 3. & 3. & 1. \\ 1. & 3. & 3. & 3. & 2. \\ 2. & 3. & 3. & 3. & 3. \\ 3. & 3. & 2. & 2. & 3. \\ 3. & 2. & 1. & 2. & 2. \\ 3. & 1. & 1. & 2. & 2. \end{pmatrix}$$

The use of the optimal masks again leads to the same problem: bad forecastings of the fifth state variable at a certain time instant influence the forecasting of the third state variable two time instants ahead. $\gamma(2503)$ through $\gamma(2506)$ are all incorrect forecastings and jeopardize the forecasting of $l(2505)$ through $l(2508)$, and it even so happened that the states of $l(2505)$ and $l(2506)$ had to be invented since forecasting was impossible.

```
[>  frcst = frc(past, inpt, m1, m2, m3, 3);
[>  error = frcst(:, 3:5) - future;
```

$$frcst = \begin{pmatrix} 3. & 1. & 1. & 1. & 3. \\ 3. & 1. & 1. & 1. & 2. \\ 3. & 2. & 1. & 1. & 2. \\ 2. & 2. & 2. & 2. & 2. \\ 1. & 3. & 1. & 3. & 2. \\ 1. & 3. & 3. & 3. & 3. \\ 2. & 3. & 2. & 3. & 3. \\ 3. & 3. & 2. & 2. & 3. \\ 3. & 2. & 1. & 2. & 2. \\ 3. & 1. & 2. & 2. & 2. \end{pmatrix}$$

$$error = \begin{pmatrix} 0. & 0. & 0. \\ 0. & 0. & 0. \\ 0. & 0. & 1. \\ 0. & 0. & 1. \\ -2.^* & 0. & 1. \\ 0.^* & 0. & 1. \\ -1. & 0. & 0. \\ 0. & 0. & 0. \\ 0. & 0. & 0. \\ 1. & 0. & 0. \end{pmatrix}$$

The use of $m3a$ instead of $m3$ again improves the forecasting enough to ensure that it is the better forecasting tool, despite of its quality coefficient.

```
[>  frcst = frc(past, inpt, m1, m2, m3a, 3);
[>  error =frcst(:, 3:5) – future
```

$$frcst = \begin{pmatrix} 3. & 1. & 1. & 1. & 3. \\ 3. & 1. & 1. & 1. & 1. \\ 3. & 2. & 1. & 1. & 1. \\ 2. & 2. & 2. & 2. & 1. \\ 1. & 3. & 3. & 3. & 1. \\ 1. & 3. & 3. & 3. & 3. \\ 2. & 3. & 3. & 3. & 3. \\ 3. & 3. & 2. & 2. & 3. \\ 3. & 2. & 1. & 2. & 2. \\ 3. & 1. & 2. & 2. & 2. \end{pmatrix}$$

$$error = \begin{pmatrix} 0. & 0. & 0. \\ 0. & 0. & -1. \\ 0. & 0. & 0. \\ 0. & 0. & 0. \\ 0. & 0. & 0. \\ 0. & 0. & 1. \\ 0. & 0. & 0. \\ 0. & 0. & 0. \\ 0. & 0. & 0. \\ 1. & 0. & 0. \end{pmatrix}$$

The experiment was run once more using the same data model, but now forecasting the states the outputs would assume at the time instants 2491 through 2500, given the inputs.

```
[>   past = past(1:2490, :);
[>   inpt = inrec(2491:2500, :);
[>   future = xrec(2491:2500, :);
[>   infut = [inpt, future];
[>   frcst = frc(past, inpt, m1, m2, m3a, 3);
[>   error = frcst(:, 3:5) − future;
```

$$infut = \begin{pmatrix} 1. & 1. & 2. & 2. & 2. \\ 1. & 1. & 2. & 2. & 3. \\ 2. & 2. & 2. & 1. & 3. \\ 3. & 2. & 1. & 1. & 2. \\ 3. & 3. & 1. & 1. & 1. \\ 3. & 3. & 2. & 2. & 1. \\ 2. & 3. & 3. & 3. & 1. \\ 1. & 2. & 3. & 3. & 3. \\ 1. & 2. & 3. & 3. & 3. \\ 2. & 1. & 2. & 2. & 3. \end{pmatrix}$$

$$frcst = \begin{pmatrix} 1. & 1. & 2. & 2. & 2. \\ 1. & 1. & 2. & 2. & 3. \\ 2. & 2. & 2. & 1. & 3. \\ 3. & 2. & 1. & 1. & 2. \\ 3. & 3. & 1. & 1. & 1. \\ 3. & 3. & 2. & 2. & 1. \\ 2. & 3. & 3. & 3. & 1. \\ 1. & 2. & 3. & 3. & 2. \\ 1. & 2. & 3. & 3. & 3. \\ 2. & 1. & 2. & 2. & 3. \end{pmatrix}$$

$$error = \begin{pmatrix} 0. & 0. & 0. \\ 0. & 0. & 0. \\ 0. & 0. & 0. \\ 0. & 0. & 0. \\ 0. & 0. & 0. \\ 0. & 0. & 0. \\ 0. & 0. & 0. \\ 0. & 0. & -1. \\ 0. & 0. & 0. \\ 0. & 0. & 0. \end{pmatrix}$$

Similar results were observed setting $inpt = 4$ to drive the model with random steps (negative, zero or positive) with a random duration between 2 and 4 time intervals (of 6 seconds). When using these driving functions, there were times when the use of the sub-optimal mask $m3a$ in place of $m3$ worsened the forecasting, but as this situation was fairly rare and the forecasting would turn out to be good again some time intervals later, we chose the set $m1$, $m2$, $m3a$ to represent the generative system of our model.

CHAPTER 5
THE ONLINE MONITORING SYSTEM

5.1 The Experimental Setup

This chapter describes how the generative models represented by the optimal masks can be used as an on-line monitoring tool for the overall system. The approach taken is based on the forecasting capability of the GSPS methodology to first determine that an accident has occurred and then hypothesize about the nature of the accident.

An on-line monitoring system was built using the generative models to forecast the future behavior of the source system and compare the inductively expected states of the variables with the actually measured ones (e.g. those stemming from sensors). Small errors in the forecasting process are expected when the system faces a situation that had no precedents in the available time history (e.g. the system is suddenly excited in a different way than it had ever been before). Most of these small errors will be swallowed by the recoding process. From time to time, an error may come through. However, such errors will soon disappear as the new situation goes by. A true accident, however, will quickly be sensed since the forecasting error will be sensibly higher and will tend to stay so if the original system has undergone a structural change.

After a high frequency transient phase that the system will certainly go through following the accident, given that the new system is still stable, a new data model will start being built to gather information about the new system. Once enough information has been collected, a new generative model should take charge and the on-line monitoring process is continued.

Originally we had planned to make our methodology come up, on-line, with a new generative model for the damaged aircraft. This idea proved to be unrealistic due to the amount of data required for the optimal mask analysis. As discussed previously in chapter 3, the number of recordings necessary to perform a meaningful optimal masks analysis in a system depends on the number of variables present in the Behavior Model matrix and the number of levels these variables are recoded into. Assuming that both of these quantities are known, the time required to collect this data still depends on the sampling rate to be used which, in turn, is related to the (structurally determined) eigenvalues of the linearized system. For the original cruise flight model, it was shown that a minimum of 1215 recordings were necessary for the optimal mask analysis. The sampling interval for that model was calculated to be 6 seconds, which means that we would have had to "shake" the broken airplane for a little over two hours. After the accident, depending on the sampling rate required for the new system, this time may increase even more.

Therefore, the current study is limited to distinguishing between different types of *prerecorded* "accidents". A set of generative models, each representing one type of accident, can be stored in a "mask library" for future use. These accidents do not necessarily have to be catastrophes: they may as well be models of different flight phases of the aircraft, or other expected structural changes in the original model. A rule-base could easily manage the library, issuing, for example, priority lists as to which masks to try out first. In the library, each generative model should be stored with instructions for recoding the data (number of levels and the inner limits of these levels — the outer limits are data dependent) and the sampling interval to be used in the data.

With all this information at hand, given that an accident has occurred, all we need after the stabilization of the system is some data describing the behavior of the after-the-accident model.

The on-line monitoring system has the main function of detecting a structural change in the monitored system. It works exactly in the same manner as the **forecasting** capability of GSPS was used in the previous chapter to check the validity of the set of optimal masks. The methodology now presented differs from the previous one mainly in how the data is pre-treated before it can be qualitatively analysed. This data "massaging" was needed to extend the applicability of the analysis to practically all possible changes in the behavior of the variables.

5.2 Normalized Qualitative Data

The GSPS methodology is sensitive to the levels of magnitude that each variable under observation assumes throughout its time history. The range enclosed within each level is entirely problem dependent, and should be defined by the observer according to her/his interest in the variables. This is a critical phase in the qualitative modeling process, and special attention should be given to choosing the number of levels necessary and to the definition of their limits. A thorough knowledge about the system and its variables is needed in order to define the recoding levels in a meaningful way; experts' opinions are a good source for the information needed in this phase.

The use of the recoding function **rekode** which defines the limits of the inner level (assuming that three levels are enough to recode the data), such that each level contains the same number of recordings is meaningful when it is applied to data generated by a system that is exerted equally in every possible way. Setting *inpt* $= 2$ in our model forces the perturbations in the control variables to change at every communication interval with a randomly negative, zero, or positive step. In a sufficiently long "shaken" flight, the system is exerted such that the generated data model is as rich in information as it could possibly be, and as there are about the same amount of negative, zero, and positive perturbations throughout the time history under observation, the response of each output should be oscillating around its steady state tendency (see figure 4.2).

The steady state tendency of most variables is a constant except for, e.g., outputs of open integrators, as are the horizontal displacement x and the altitude h in our model. A flight with constant climb (or descent) rate is easily achieved by simply changing the thrust from its reference value that characterizes the (levelled) reference flight condition, and, obviously, the horizontal displacement cannot stabilize at a constant value.

It is also desirable to allow changes in the steady state values of the variables, so that different reference flight conditions can be monitored and analysed with a same set of generative models. In order to allow a more general use of the from-matrices, the steady state value of the variables should be nullified, since it holds no useful information for the analysis. If, however, the observer does have interest in it, then it should be included as a separate variable in the data model and analysed accordingly.

5.3 A New Source Model

The source model used until now

$$S_{old}: \ \{pde \ \ ptr \ \ l \ \ d \ \ \gamma\}$$

was first believed to fully describe the system behavior, based on the fact that the aerodynamic force response of the aircraft and the flight path angle should fully describe the trajectory of our model. That is perfectly true, but it is not just in the trajectory that we are interested but in as much behavioral information as we can possibly get from just three variables. As the lift and the drag are highly positively correlated as can be easily seen in figures 4.3 and 4.4, the behavioral information they provide to the generative system is mostly redundant, and therefore, for the qualitative analyses performed in this chapter, we will substitute the drag d by the amplitude of the longitudinal velocity vector, v. Hence,

$$S_{new}: \ \{pde \ \ ptr \ \ l \ \ v \ \ \gamma\}$$

is the new source model whose generative model was evaluated using the code below:

```
[>  acsl('set tmx = 15000, inpt = 2')
[>  [pde, ptr, l, v, ga] = start;
[>  inrec = [pde, ptr];
[>  xraw = [l, v, ga];
[>  mean = average(xraw, 2501);
[>  xdev = xraw - ones(xraw)*diag(mean);
[>  clear xraw pde ptr l d ga
[>  deff rekode
[>  [xrec, f] = rekode(xdev);
[>  deff omhis
[>  [mb4, hm, hr, q, mhis] = omhis([inrec, xrec], 5, 0);
[>  save f mb4 >temp
```

The optimal mask analysis was performed on a data model with 2500 elements in each variable, sampled at every 6 seconds. The communication interval (ACSL parameter *cint*) was not reset in this run since we wanted to use its default value $cint = 6$ automatically set by our program. Note that the recoded data model is obtained applying the function **rekode** to the deviations of the output variables from their mean values. The from-matrices f that were used for the optimal mask analysis must still have their external limits extended a little beyond the maximum and minimum values reached by the variables in a not so shaken flight so that all data generated by the models B4 in all its flight phases can be recoded by the same from matrices. To do so, we saved the current from-matrices f and the optimal masks $mb4$ temporarily in the file **temp.dat** and will now rerun the model with the fligh phase switch *inpt* set to 4.

```
[>  acsl('set('tmx = 15000, inpt=4')
[>  [l, v, ga] = start;
[>  mean = average([l, v, ga]);
[>  xdev = [l, v, ga] - ones(2501, 3)*diag(mean);
[>  mx = max(xdev);
[>  mn = min(xdev);
[>  load <temp
[>  fb4 = f;
[>  for i = 1:3, fb4(2*(i-1)+1, 1) = mn(i);...
[>       fb4(2*i, 3) = mx(i);...
[>       end
[>  sintb4 = 6;
[>  save sintb4 fb4 mb4 >fmb4
```

The from-matrices and the optimal masks that represent the new generative model B4 are shown below:

$$fb4_l = \begin{pmatrix} -1.1000 & -0.1101 & 0.1059 \\ -0.1101 & 0.1059 & 1.1000 \end{pmatrix} \times 10^4$$

$$fb4_v = \begin{pmatrix} -7.5000 & -0.7844 & 0.7376 \\ -0.7844 & 0.7376 & 7.5000 \end{pmatrix}$$

$$fb4_\gamma = \begin{pmatrix} -0.0120 & -0.0013 & 0.0013 \\ -0.0013 & 0.0013 & 0.0120 \end{pmatrix}$$

$$mb4_l = \begin{pmatrix} -1. & 0. & 0. & 0. & -2. \\ -3. & -4. & 0. & 0. & 0. \\ 0. & 0. & +1. & 0. & 0. \end{pmatrix}$$

$$mb4_v = \begin{pmatrix} -1. & 0. & 0. & 0. & 0. \\ 0. & -2. & 0. & -3. & 0. \\ 0. & 0. & 0. & +1. & 0. \end{pmatrix}$$

$$mb4_\gamma = \begin{pmatrix} 0. & 0. & 0. & 0. & 0. \\ -1. & 0. & 0. & -2. & -3. \\ 0. & 0. & 0. & 0. & +1. \end{pmatrix}$$

Mask $mb4_v$ is, in reality, the second best mask according to its quality factor but proved to give better forecasting results in different sets of data. The sampling interval, which is the third quantity that specifies the generative model FMB4, has not changed from the previous source model to the present one: $sintb4$ is still 6 seconds.

5.4 The Broken Model B747

An unpredicted structural change in the longitudinal flight model was simulated changing the original airplane parameters to the ones relative to a power approach configuration of a B747. The emergency is just theoretical but the new model fits very well our purposes since it is structurally different from the original one.

The values for the parameters are listed below and are based on the ones listed in "Aircraft Handling Qualities Data" page 217 (Heffley, 1972). From now on, we shall refer to this model as B747:

W	$= 564{,}000.0$	$[lb]$
v_0	$= 278$	$[ft/sec]$

I_y	$= 30{,}500{,}000.0$	$[slug\ ft^2]$
S	$= 5{,}500.0$	$[ft^2]$
ρ	$= 0.0024$	$[slug/ft^3]$
α_0	$= 0.0995$	$[rad]$
δ_{e_0}	$= -0.0366$	$[rad]$
C_{L_0}	$= 1.11$	$[\ ]$
C_{D_0}	$= 0.102$	$[\ ]$
C_{L_α}	$= 5.7$	$[1/rad]$
C_{D_α}	$= 0.66$	$[1/rad]$
C_{M_α}	$= -1.26$	$[1/rad]$
$C_{L_{\dot\alpha}}$	$= -6.7$	$[1/\frac{rad}{s}]$
$C_{M_{\dot\alpha}}$	$= -3.2$	$[1/\frac{rad}{s}]$
C_{L_q}	$= 5.4$	$[1/\frac{rad}{s}]$
C_{M_q}	$= -20.8$	$[1/\frac{rad}{s}]$
$C_{L_{\delta_e}}$	$= 0.338$	$[1/rad]$
$C_{M_{\delta_e}}$	$= -1.34$	$[1/rad]$

The sampling interval for the B747 model above was evaluated based on the slowest time constant of its linearized model whose eigenvalues were calculated as being $-0.0461 \pm 0.1596i$ and $-0.4600 \pm 0.6876i$. The two time constants of this system are $\tau_1 = 22$ seconds and $\tau_2 = 2$ seconds and the suggested sampling interval has, therefore, almost doubled for this model: $t_{s_{B747}} = 11$.

A graphical view of the accident occurring in a steady cruise flight of the original model (B4) is given by figures 5.1 . Figure 5.1a below depicts the transients of the control variables δ_e and Tr (in solid line) and their reference values $\delta_{e_{trim}}$ and Tr_{trim} (in dashed lines).

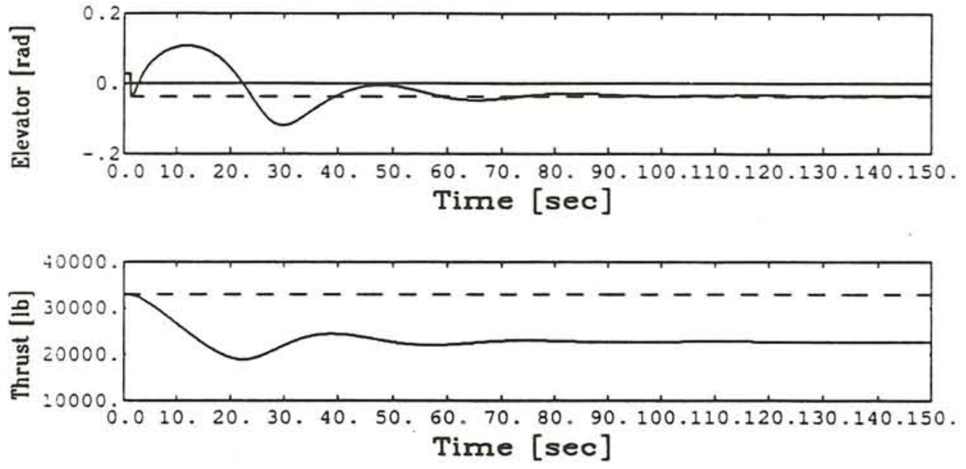

Figure 5.1a Control variables and their reference values in the accident

Note that as the reference value of the elevator deflexion is changed from $\delta_{e_0} = 0.027886$ in the B4 model to $\delta_{e_0} = -0.0366$, the variable follows this change and stabilizes at the new reference value. The thrust, on the other hand, drops to about $\frac{2}{3}$ of its original value due to a steep decrease in the velocity (characteristic of the B747 model) while its reference value remains unchanged.

Next, figure 5.1b shows the effect of the accident on the output variables when they all go through transients of extremely large amplitudes stabilizing after about 100 seconds at their new steady states. Lift and drag are both affected, whereby the drag is increased considerably (not shown in the figure). The velocity drops to about half of its original value and the flight path angle stabilizes at a slightly negative value. All these new settings describe an airplane decelerating and descending.

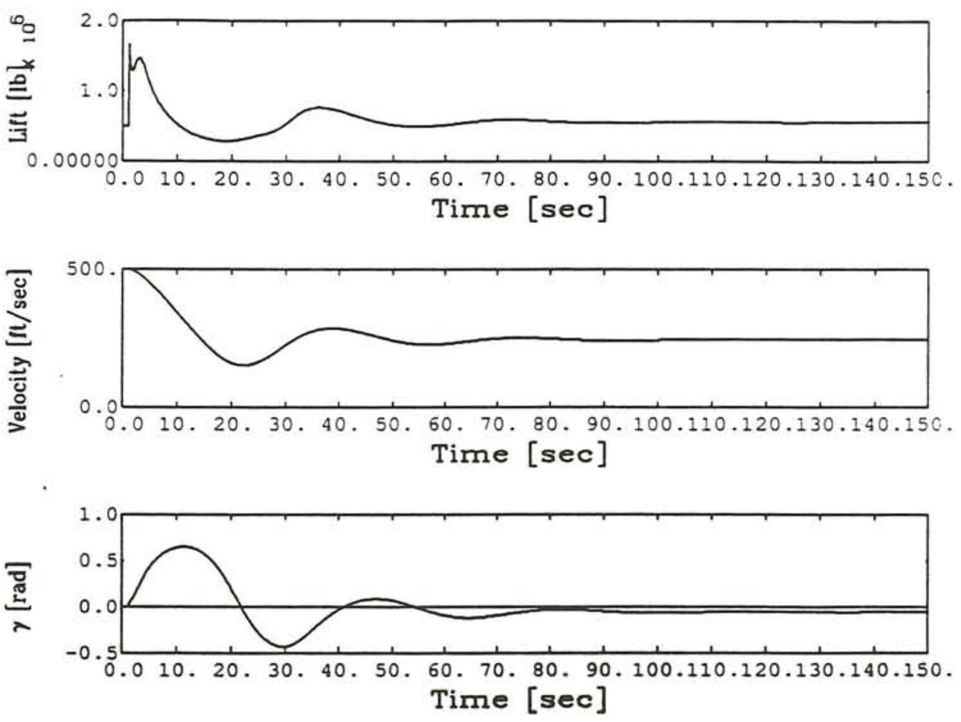

Figure 5.1b Output variables' transients in the accident

The simulation is continued beyond the stabilization of all variables to perform a step response analysis on the system. The simulation is run twice with the same settings for the same phase, only to change the step perturbation. The results are depicted in figure 5.2 below:

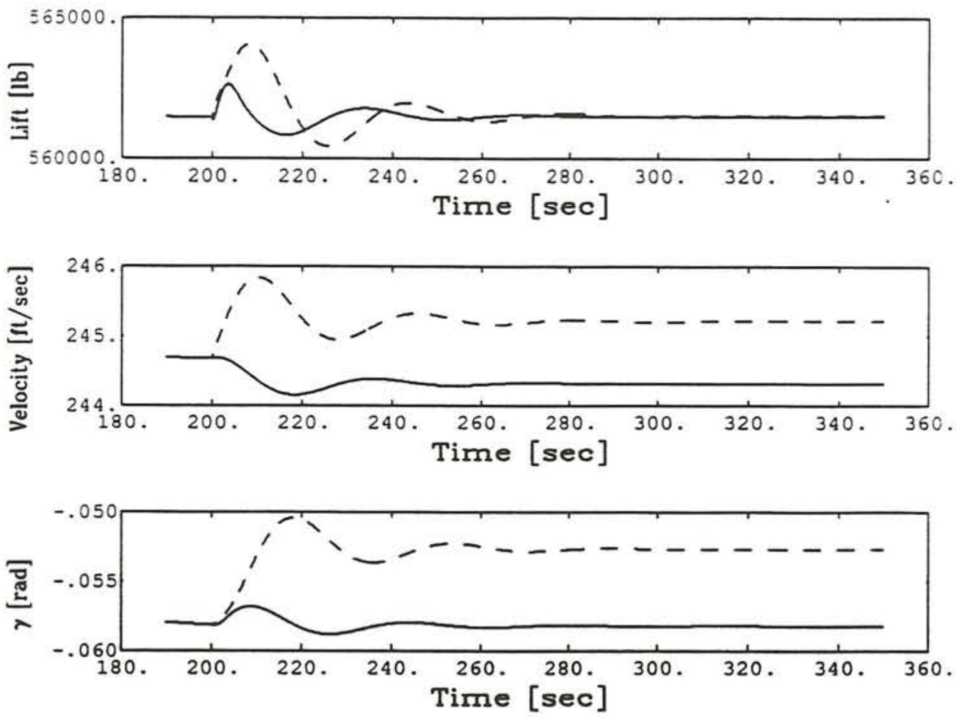

Figure 5.2: Step response of the B747 model. The trajectories displayed in solid lines represent the response of the system to a step change of −.001 radians in the elevator deflexion at fixed throttle.The trajectoris in dashed lines refer to the opposite situation: a step change of 3000 lb. in the reference value of the thrust at fixed elevator deflexion.

The qualitative model FMB747 was evaluated in an analogous manner as the FMB4 with the difference that the flight is started with the original model (B4) and changed to the B747 model in the tenth communication interval. This procedure is necessary because every flight has to be started trimmed to avoid the initial iterative trimming phase of the model which would change the initial conditions differently for every case. As all parameters and initial conditions were carefully chosen for the B4 model so that it would start perfectly trimmed, for the sake of uniformity and easier comparison of different flights, we shall start all flights with it, and simulate an accident right after the tenth communication interval and then discard the first one hundred data points where most of the transients lay. The following code was used in place of the seven first lines of code that was used for the evaluation of FMB4:

```
[>  cint = 11;
[>  tbreak = 10*cint;
[>  tmx = 2600*cint;
[>  c2alist('tmx, cint, tbreak')
[>  acsl('set inpt = 2')
[>  do B747
[>  [pde, ptr, l, v, ga] = start;
[>  inrec = [pde, ptr];
[>  inrec = inrec(101 : 2601, :);
[>  xraw = [l, v, ga];
[>  xraw = xraw(101 : 2601, :);
 ⋮
```

The Ctrl–C do-procedure file **b747.ctr** sets all the new values for the parameters that are to be changed in a B4/B747 accident scheduled at time *tbreak* = 10∗cint = 10×11 = 110 seconds. The from-matrices and optimal masks relative to the generative model B747 evaluated in this way are listed below:

$$fb747_l = \begin{pmatrix} -1.6029 & -0.2137 & 0.2113 \\ -0.2137 & 0.2113 & 1.3981 \end{pmatrix} \times 10^4$$

$$fb747_v = \begin{pmatrix} -5.9153 & -1.0733 & 1.0150 \\ -1.0733 & 1.0150 & 6.1723 \end{pmatrix}$$

$$fb747_\gamma = \begin{pmatrix} -0.0278 & -0.0050 & 0.0050 \\ -0.0050 & 0.0050 & 0.0290 \end{pmatrix}$$

$$mb747_l = \begin{pmatrix} 0. & 0. & 0. & 0. & 0. \\ -1. & 0. & 0. & -2. & -3. \\ 0. & 0. & +1. & 0. & 0. \end{pmatrix}$$

$$mb747_v = \begin{pmatrix} 0. & 0. & 0. & 0. & 0. \\ -1. & -2. & 0. & 0. & -3. \\ 0. & 0. & 0. & +1. & 0. \end{pmatrix}$$

$$mb747_\gamma = \begin{pmatrix} -1. & 0. & 0. & 0. & 0. \\ -2. & -3. & -4. & 0. & 0. \\ 0. & 0. & 0. & 0. & +1. \end{pmatrix}$$

5.5 The Mask Library

Three other sets of optimal masks representing different structural changes of the original B4 model were evaluated and saved for future use. The files containing the generative models were named **fm***model-name* and carry the **.dat** default file extension attributed by Ctrl–C to its data files. Each of these files contains at least three elements: the sampling interval, the set of optimal masks, and the from-matrices relative to a data set that describes the behavior of the model.

The set of optimal masks may include second best masks which were used in place of the optimal masks (rated based on the quality factor) when their forecasting capability proved to be superior. The set of from-matrices was computed with the function **rekode** applied to each model's shaken ($inpt = 2$) data, and then adjusted to normal data ($inpt = 4$) since less shaken data tend to have larger magnitudes.

Model B5

We will call B5 the flight model with the following changes in the parameters of the original model (B4):

$$C_{D_\alpha} = 0.5 \quad [1/\text{rad}]$$

$$\begin{aligned} C_{L_\alpha} &= 4.0 \quad [1/\text{rad}] \\ C_{M_\alpha} &= -1.0 \quad [1/\text{rad}] \\ K_\theta &= 0.12 \ [\ \ ] \end{aligned}$$

which completely modify the influence that the angle of attack has on the aerodynamic response of the aircraft. Its effect on the lift coefficient is increased by almost 100%, its effect on the drag coefficient is decreased by about 25%, while its effect on the pitching moment coefficient is increased by about 25%. The feedback gain K_θ is dropped to about half of its original value so that the B5 system would stabilize satisfactorily.

The eigenvalues of the equivalent linearized model (calculated about the B5 model's stabilized steady state condition) were calculated to be:

$$\lambda_{B5} = \begin{pmatrix} -0.0262 + 0.0696i \\ -0.0262 - 0.0696i \\ -0.5284 + 0.9631i \\ -0.5284 - 0.9631i \end{pmatrix}$$

having time constants (rounded to the nearest integer) of 38 and 2 seconds. The step response of the system to the same step perturbations that were used for models B4 and B747 previously ($\Delta\delta_{e_{trim}} = -.001$ and $\Delta \mathrm{Tr}_{\mathrm{trim}} = 3000$, is depicted in figures 5.3 . Figure 5.3a shows the effect of the individually applied perturbations on their respective control variables.

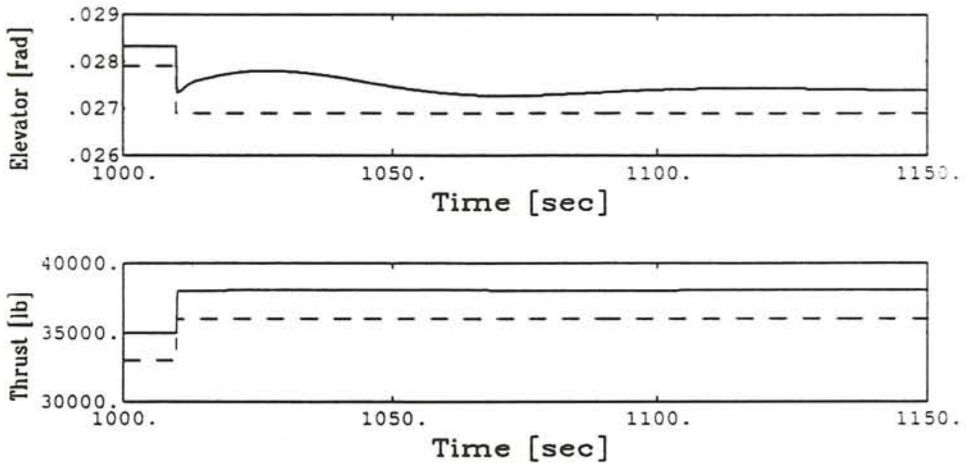

Figure 5.3a: Effect of individually applied step perturbations (dashed lines) on their respective control variable (solid lines) in the model B5.

Figure 5.3b shows the step response of the model B5 to the individually applied step perturbations. The system is visibly slower and more damped than the previous B4 and B747 models.

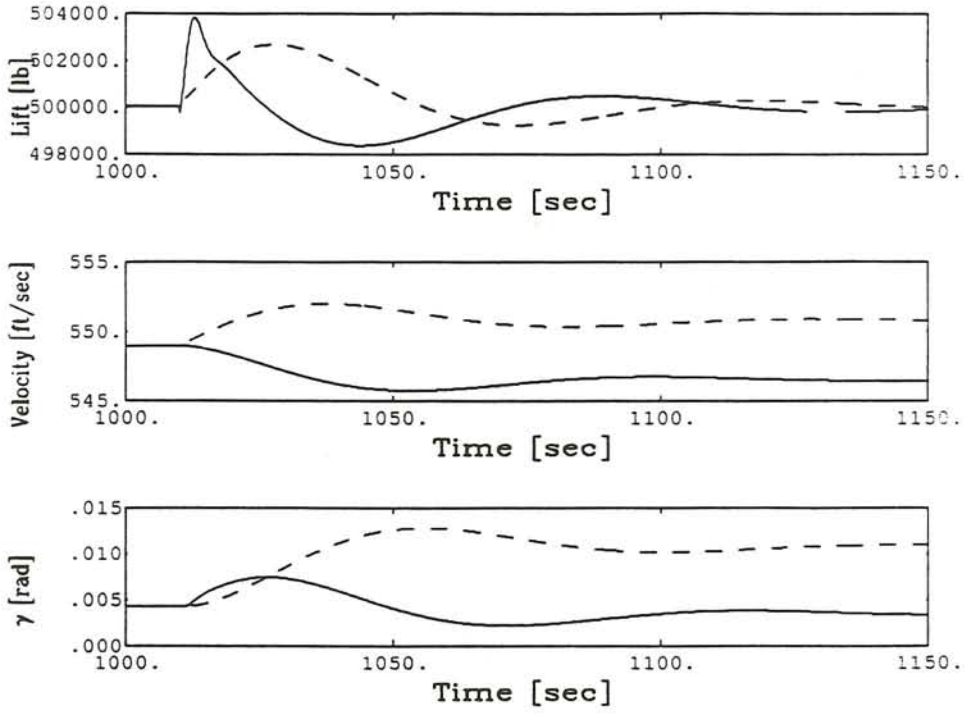

Figure 5.3b: Step response of the B5 model. The curves in solid lines represent the response of the system to a step change of −.001 radians in the elevator deflexion at fixed throttle, and the ones in dashed lines refer to the opposite situation: a step change of 3000 lb. in the reference value of the thrust at fixed elevator deflexion.

The same procedure described for the FMB747 model was again followed to evaluate the qualitative model FMB5 which is represented by the from-matrices $fb5$, the optimal masks $mb5$ and the sampling interval $sintb5 = 19$.

$$fb5_l = \begin{pmatrix} -1.4733 & -0.1603 & 0.1487 \\ -0.1603 & 0.1487 & 1.3025 \end{pmatrix} \times 10^4$$

$$fb5_v = \begin{pmatrix} -7.9056 & -1.5244 & 1.4346 \\ -1.5244 & 1.4346 & 8.1471 \end{pmatrix}$$

$$fb5_{\gamma} = \begin{pmatrix} -0.0183 & -0.0029 & 0.0027 \\ -0.0029 & 0.0027 & 0.0196 \end{pmatrix}$$

$$mb5_{l} = \begin{pmatrix} 0. & 0. & 0. & 0. & 0. \\ 0. & -1. & 0. & 0. & -2. \\ 0. & 0. & +1. & 0. & 0. \end{pmatrix}$$

$$mb5_{v} = \begin{pmatrix} -1. & 0. & 0. & -2. & 0. \\ -3. & -4. & 0. & 0. & 0. \\ 0. & 0. & 0. & +1. & 0. \end{pmatrix}$$

$$mb5_{\gamma} = \begin{pmatrix} 0. & 0. & 0. & 0. & 0. \\ -1. & -2. & 0. & -3. & 0. \\ 0. & 0. & 0. & 0. & +1. \end{pmatrix}$$

Model B13

This model is characterized by a much more damped step response to the same step inputs. The amplitude of the oscillations is considerably higher as can be seen in figure 5.4.

The effect of the angle of attack on the lift coefficient has been slightly increased, as compared to the original model B4. A most significant change was made in the effect of the angle of attack on the drag coefficient which has now almost trippled from its original value, and a considerable change occurred, as well, in the effect of the elevator deflexion on the pitching moment coefficient, which makes this model now more sensitive to this control variable. Again satisfactory stability was achieved with a new value for the feedback gain K_{θ}. The changed parameters are:

$$\begin{aligned} C_{D_{\alpha}} &= 0.7 \quad [1/\text{rad}] \\ C_{L_{\alpha}} &= 6.0 \quad [1/\text{rad}] \\ C_{M_{\delta_e}} &= -2.0 \quad [1/\text{rad}] \\ K_{\theta} &= 0.1 \quad [\ ] \end{aligned}$$

The eigenvalues of the equivalent linearized model (calculated about the B13 model's stabilized steady state condition) were calculated to be:

$$\lambda_{B13} = \begin{pmatrix} -0.0322 + 0.0000i \\ -0.0937 - 0.0000i \\ -0.6994 + 0.9452i \\ -0.6994 - 0.9452i \end{pmatrix}$$

Note that the system has only one oscillatory mode now, the fastest one. The phugoid mode has split into two exponential decays, out of which the slowest one has a time constant of about 31 seconds.

Figures 5.4 below depict the effect of the perturbations on the control variables and the step response of the system to the individually applied perturbations.

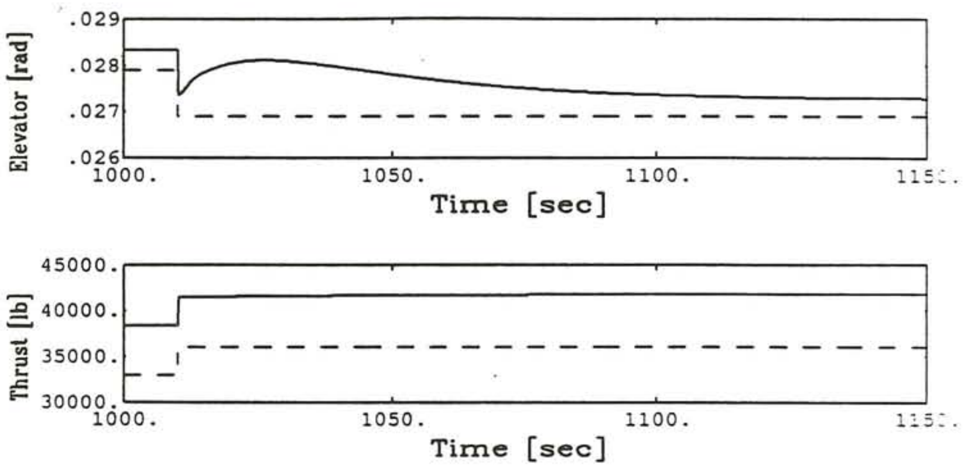

Figure 5.4a: Effect of individually applied step perturbations (dashed lines) on their respective control variable (solid lines) in the B13 model.

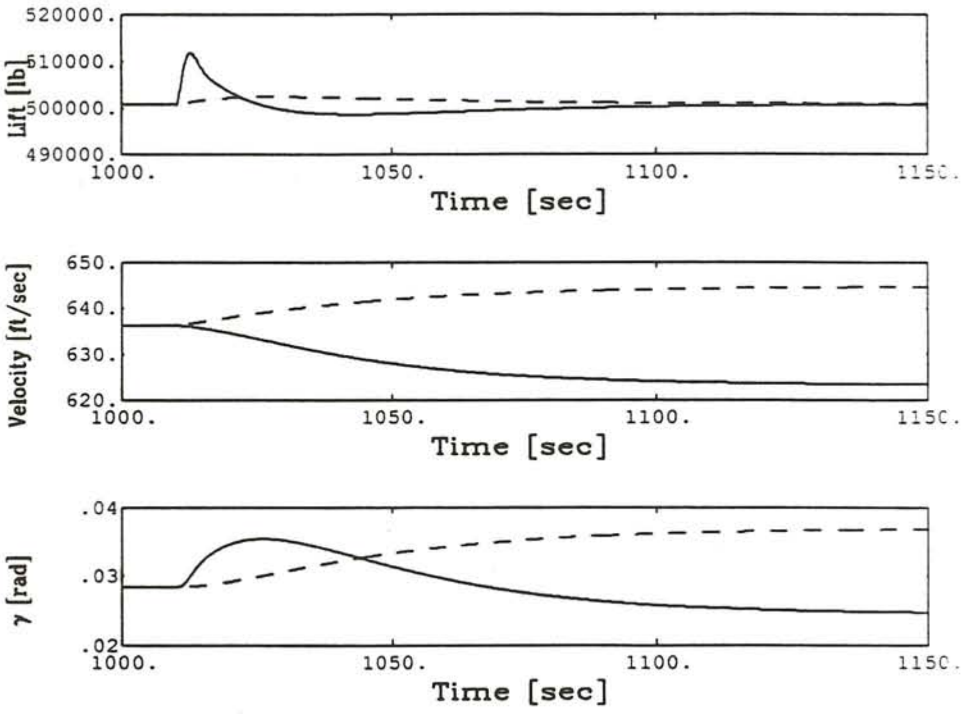

Figure 5.4b: Step response of the B13 model. The curves in solid lines represent the response of the system to a step change of −.001 radians in the elevator deflexion at fixed throttle, and the ones in dashed lines refer to the opposite situation: a step change of 3000 lb. in the reference value of the thrust at fixed elevator deflexion.

The generative model FMB13 is represented by:

$$fb13_l = \begin{pmatrix} -2.5914 & -0.1058 & 0.0961 \\ -0.1058 & 0.0961 & 2.4755 \end{pmatrix} \times 10^4$$

$$fb13_v = \begin{pmatrix} -18.0545 & -2.6505 & 2.5067 \\ -2.6505 & 2.5067 & 18.1259 \end{pmatrix}$$

$$fb13_\gamma = \begin{pmatrix} -0.0227 & -0.0036 & 0.0036 \\ -0.0036 & 0.0036 & 0.0216 \end{pmatrix}$$

$$mb13_l = \begin{pmatrix} 0. & 0. & 0. & 0. & 0. \\ 0. & -1. & 0. & 0. & -2. \\ 0. & 0. & +1. & 0. & 0. \end{pmatrix}$$

$$mb13_v = \begin{pmatrix} 0. & 0. & 0. & 0. & 0. \\ -1. & -2. & 0. & -3. & 0. \\ 0. & 0. & 0. & +1. & 0. \end{pmatrix}$$

$$mb13_\gamma = \begin{pmatrix} 0. & 0. & 0. & 0. & 0. \\ -1. & 0. & 0. & -2. & 0. \\ 0. & 0. & 0. & 0. & +1. \end{pmatrix}$$

Model B14

This model was created with the intention of lessening as much as possible the influence of the elevator deflexion on the pitching moment coefficient. The model is characterized by the following changed parameters:

$$\begin{aligned} C_{M_{\delta_e}} &= -0.5 \quad [1/\text{rad}] \\ K_\theta &= 0.4 \quad [\ ] \end{aligned}$$

The eigenvalues of the equivalent linearized model (calculated about the B14 model's stabilized steady state condition) were calculated to be:

$$\lambda_{B14} = \begin{pmatrix} -0.0406 + 0.0740i \\ -0.0406 - 0.0740i \\ -0.5109 + 0.7413i \\ -0.5109 - 0.7413i \end{pmatrix}$$

and the slowest time constant present in the system is about 25 seconds.

Figures 5.5 below depict the effect of the perturbations on the control variables in this model and its step response to the individually applied perturbations which is very similar to the response of the model B5, with the difference that the model B14 is slightly more damped and somewhat slower.

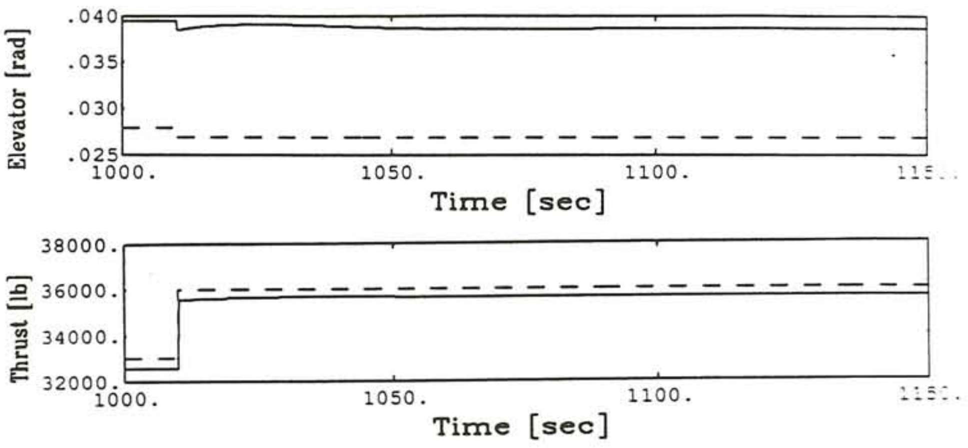

Figure 5.5a: Effect of individually applied step perturbations (dashed lines) on their respective control variable (solid lines) in the B14 model.

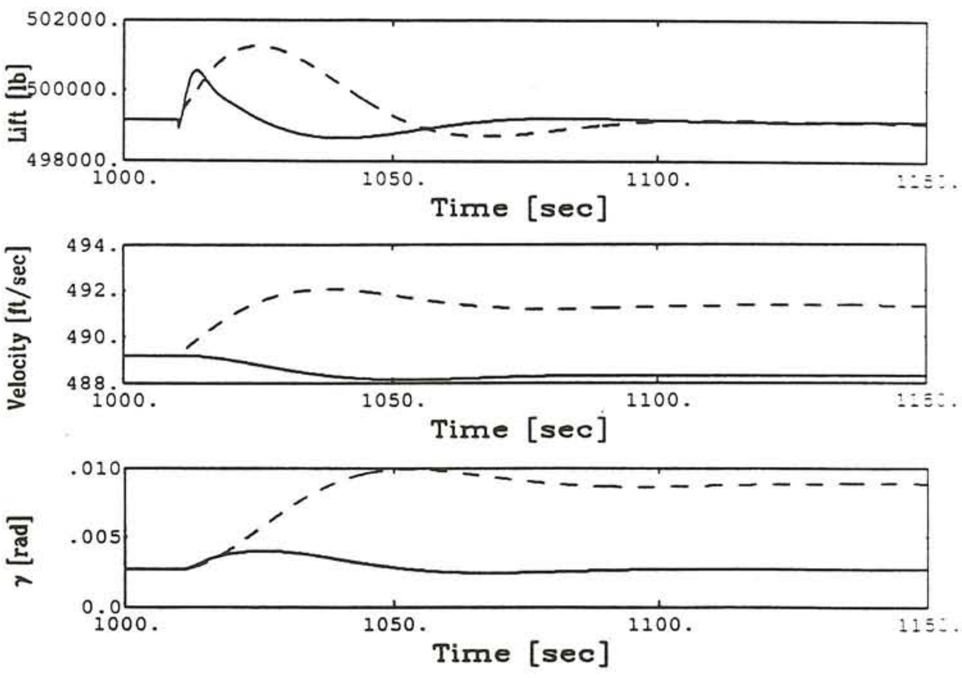

Figure 5.5b: Step response of the B14 model. The curves in solid lines represent the response of the system to a step change of −.001 radians in the elevator deflexion at fixed throttle, and the ones in dashed lines refer to the opposite situation: a step change of 3000 lb. in the reference value of the thrust at fixed elevator deflexion.

The model's from-matrices and optimal masks are:

$$fb14_l = \begin{pmatrix} -0.6079 & -0.1082 & 0.0988 \\ -0.1082 & 0.0988 & 0.6602 \end{pmatrix} \times 10^4$$

$$fb14_v = \begin{pmatrix} -4.9056 & -0.97329 & 0.9774 \\ -0.9732 & 0.9774 & 4.6891 \end{pmatrix}$$

$$fb14_\gamma = \begin{pmatrix} -0.0112 & -0.0018 & 0.0017 \\ -0.0018 & 0.0017 & 0.0107 \end{pmatrix}$$

$$mb14_l = \begin{pmatrix} 0. & 0. & 0. & 0. & 0. \\ 0. & -1. & 0. & 0. & -2. \\ 0. & 0. & +1. & 0. & 0. \end{pmatrix}$$

$$mb14_v = \begin{pmatrix} 0. & 0. & 0. & 0. & 0. \\ 0. & -1. & 0. & 0. & 0. \\ 0. & 0. & 0. & +1. & 0. \end{pmatrix}$$

$$mb14_\gamma = \begin{pmatrix} 0. & 0. & 0. & 0. & 0. \\ 0. & -1. & 0. & -2. & 0. \\ 0. & 0. & 0. & 0. & +1. \end{pmatrix}$$

5.6 Detection of the Accident

The following quantitative simulation describes a flight in which an accident is scheduled to happen in the first sampling instant after time $t = 6000$ seconds. The simulation was carried out with the flight phase switch *inpt* set to 4, which drives the model with randomly changing perturbations in the control variables of the aircraft. The perturbation steps are randomly negative, zero or positive and change at random time intervals uniformly distributed between 15 and 25 seconds (default values). The Ctrl–C/ACSL code used to run the ACSL model and generate the quantitative data is:

```
[>  cint = 1;
[>  load fb4 mb4 sintb4 <fmb4
[>  tbreak = 1000*sintb4;
[>  tmx = 1000*sintb4 + 600 + 19*500;
[>  c2alist('tmx, cint, tbreak')
[>  acsl('set inpt=4')
[>  do B747
[>  [pde, ptr, l, v, ga] = start;
```

The generative model relative to the original airplane is represented by

$$FMB4 : \{mb4,\ sintb4,\ fb4\}$$

where *mb*4 is composed of the three best masks relative to the behavior of that system, *sintb*4 represents the sampling interval the masks *mb*4 expect the data to have been sampled with, and *fb*4 defines the recoding levels the masks are able to recognize. Matrix *fb*4 is composed of the three from-matrices that were used to recode the standardized errors of the raw data in the optimal mask analysis that yielded *mb*4.

The ACSL simulation's communication interval *cint* had to be set to one in this run because that is the greatest common divisor of all sampling intervals of the models presented in the previous section (6, 11, 19, 15, and 12 seconds) and the data model generated by this simulation run should fit every model.

The total simulation time required is such that the outputted time history yields 1000 data points prior to the accident (6000 seconds), 600 seconds for stabilization after the accident, and 500 data points even for the slowest broken model in the mask library (*sintb*5 = 19 seconds), which adds up to 16,100 seconds.

The Ctrl–C/ACSL interface command **c2alist**('tmx, cint, tbreak') makes all these three variables common to both Ctrl–C and ACSL workspaces. B747 in the second line from the bottom of the code above is defined as a Ctrl–C do-procedure which executes the Ctrl–C commands contained in the file B747.ctr that sets the new values for the parameters that are to be changed when the accident takes place. These values were listed along with the description of the model B747 in a previous section of this chapter.

Figure 5.6 below shows the behavior of the output variables in the neighborhood of the accident. Note the high amplitude transients caused by the accident and, later in time, the large difference in the mean value of the variables before and after the accident.

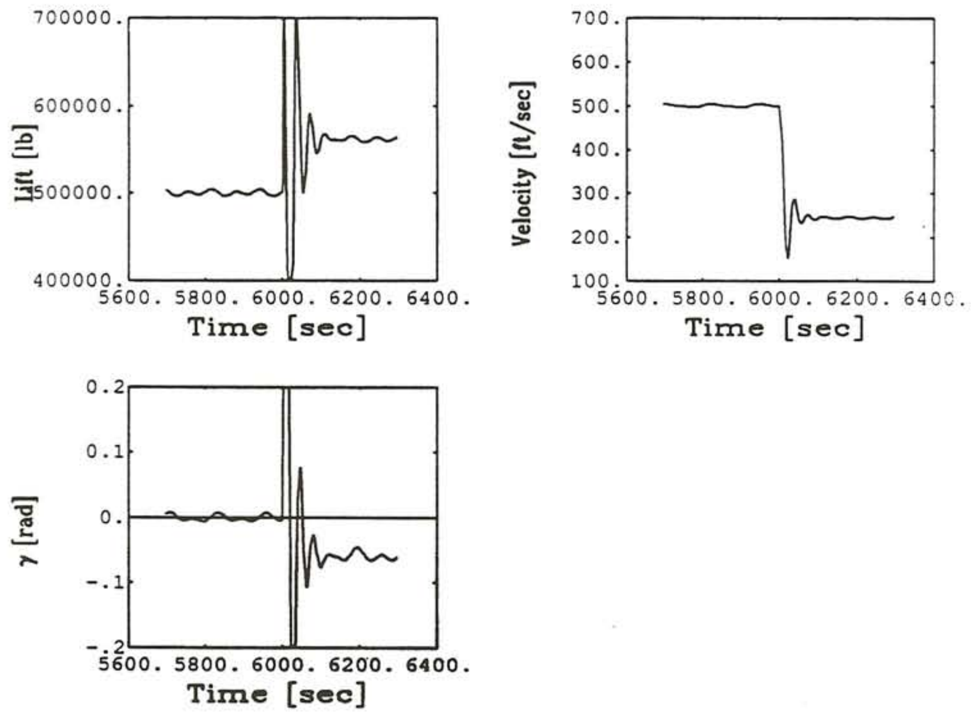

Figure 5.6: Output variables in their raw configuration in the neighborhood of the accident

The quantitative data obtained from this simulation was transformed into qualitative data as required by the online monitoring system. The whole time history can be separated into three separate phases, namely:

1. from $t = 0$ to $t = 6001$, normal flight
2. from $t = 6001$ to $t = 6600$, accident transients and stabilization
3. from $t = 6601$ to $t = 16100$, after-the-accident flight

Due to the size of the data model generated, in this first phase of the analysis we will discard all data posterior to time $t = 6120$ seconds to save room in the working space. The whole time history is automatically saved in a file named RRR.dat by the Ctrl–C/ACSL interface after each **start** command. If the ACSL simulation is not rerun with different settings, the raw data model can be loaded back into the Ctrl–C workspace any time with the Ctrl–C **load** command with its **-a** option switch.

The following code starts creating the data model necessary for the qualitative analysis. Matrix *inrec* contains the input variables *pde* and *ptr* sampled every six seconds and already (naturally) recoded. Matrix *xraw* contains the output variables *l*, *v*, and *ga* sampled correspondingly but still in their raw form. The raw data model so formed ([inrec, xraw]) is 1030 elements long and represents the first 6174 seconds of flight.

```
[>  sample = sintb4/cint;
[>  inrec = [pde, ptr];
[>  inrec = inrec(1:sample:1030*sample,:);
[>  xraw = [l, v, ga];
[>  xraw = xraw(1:sample:1030*sample,:);
[>  clear pde ptr l v ga
```

Now the output variables of the data model still have to have their mean value subtracted and then be recoded into three levels with the *fb*4 from-matrices adjusted to the available data. This task is carried out by the following code:

```
[>  //calculate the mean value of the data model
[>  //prior to the accident
[>  mean = average(xraw(1: 1000, :), 1000);
[>  xdev = xraw − ones(xraw)*diag(mean);
[>  clear mean xraw
[>  //adjust the from-matrices to the current data (xdev)
[>  deff fadjust
[>  f = fadjust(fb4, xdev);
[>  for i=1:3, xrec(:,i) = recode(xdev(:,i),'domain', f(2*(i-1)+1 : 2*i, :), 0:4);
[>  clear xdev
[>  //the final recoded data model is [inrec, xrec]
```

The recoding process of the output variables includes a preprocessing phase where the mean values of these variables were subtracted in order to extend the applicability of the from-matrices to different sets of data. Variations of the mean value, if of interest to the observer, should be represented by an additional variable if the number of recoding levels is to be kept as small as possible (3).

Note that the mean value used to compute *xdev* in the previous code is a global mean relative to the whole data sample available before the accident (1000 data points). This procedure is valid for this data model because all its mean values are constant in time, but had the variables had a mean value changing in time, a moving average based on data samples as large as possible should have been used.

The deviations of the variables from their global mean value are then recoded into three levels which are defined by the same from-matrices that were used in the optimal mask evaluation. These from-matrices can be applicable to any data model that represents the deviations of the raw data from their mean values if they are first adjusted to this data model.

By adjusting the from-matrices to the data model, we mean creating two additional levels, namely levels 0 and 4, that will represent the data points that are out of the boundaries specified by the original from-matrices: below level 1 or adove level 3. These additional levels are needed now for the detection of the accident and also later, when we will try match the masks existent in the library with the after-the-accident data. The existence of data in these two new levels strongly hints to the existence of problems in the qualitative analysis and should trigger a warning so that, if the occurrence of such recoded data persists, the alarm should be triggered. The adjusting of the from-matrices is carried out by the Ctrl–C user-defined function **fadjust** listed below:

```
//[f2] = fadjust(f1, x)
f2 = [zrow(6, 1), f1, zrow(6, 1)];
xx = [x; [f1(1, 1), f1(3, 1), f1(5, 1)]; [f1(2, 3), f1(4, 3), f1(6, 3)]];
for i = 1:3, ...
  f2(2*(i-1)+1, 1) = min(xx(:, i)); ...
  f2(2*i, 1) = f1(2*(i-1)+1, 1) + eps; ...
  f2(2*(i-1)+1, 5) = max(xx(:, i)); ...
  f2(2*i, 5) = f1(2*(i-1)+1, 3) + eps; ...
end
return
```

A continuous forecasting process was performed on the qualitative data in the neighborhood of the point where the accident took place: from the 991-st data point (t = 5940 seconds) through the 1030-th data point (t = 6174 seconds). The past behavior information required by the forecasting process is given by the time history matrix *past*. The states of the input variables concatenated from the right by the (actually measured) states of the output variables relative to the time instants to be forecast form the $future$ matrix.

The following code performs the forecasting of the three output variables in the neighborhood of the accident, compares the results of the forecasting to their actually recorded states, and runs the results of this comparison through an error filter that will trigger an alarm if the mean value of the error is too high.

```
[>  past = [inrec(1: 990, :), xrec(1: 990, :)];
[>  future = [inrec(991: 1030, :); xrec(991: 1030, :)];
[>  deff ennustus
[>  [error, bounds] = ennustus(past, future, mb4, 3);
[>  deff filter
[>  alarm = filter(error, bounds);
```

The user-defined function **ennustus** (the Finnish word for *forecasting*) performs the forecasting in a similar manner as did the function **frc** used in the previous chapter. The forecasting process was improved with the inclusion of actually measured data to the past behavior matrix during the continuous forecasting process. Erroneously forecast and guessed states, when **manicure** had to be called, would both contribute to future bad forecastings and guessings since exactly those data points are the ones that the masks use as basic information to generate the states in the next time instant. **Ennustus** simply forecasts the triple of outputs and uses them to build its first output matrix: the forecasting error matrix. The forecast triple is not concatenated to the top of the past behavior matrix where the actually measured states relative to that time instant are now placed.

Ennustus also builds a second output matrix *bounds* which has the same dimensions as *error* and shows if the actually measured states of the output variables being forecast have been recoded into the saturation levels or not. The elements of *bounds* assume one of the three values: "-1" for negative saturation, "0" for variables within normal recoding levels, and "1" for positive saturation.

The output *error* of the function is the error matrix associated with the forecasting, i.e., the difference between the (recoded) actually measured states and their forecast equivalents.

The third input *mask* required by the function refers to a matrix which contains all three optimal masks concatenated to each other from below as one single input. In the present forecasting problem, the requied input is:

$$mask = mb4 = [mb4_l;\ mb4_v;\ mb4_\gamma]$$

The second input *future* required by the function contains all five variables of the data model relative to the time period to be forecast: the states assumed by the input variables and the respective actually measured states of all three output variables.

As in **frc**, the input *past* refers to the past behavior matrix up to the present (the time instant from when on forecasting is started), and the last input *nl* is an integer representing the number of levels present in the data matrix (it is assumed that all output variables have been recoded into the same number of levels). Both functions use this last input to be able to realistically guess (randomly) a future state if the behavior read by the mask has no precedents, having therefore zero probability of occurring. In such cases, the system invents randomly a state and gives it $1/nl$ probability of occurrence. In the current analysis *nl* is set to 3 since that is the actual number of recoding levels in the model - levels 0 and 4 should not be included in the list of possible guessings. The sequence of Ctrl–C/SAPS–II commands that form the **ennustus** function is listed below:

```
[>  //[error, bounds] = ennustus(past, future, mask, nl)
[>  deff manicure
[>  [nn, mm] = size(future);
[>  prob = zrow(1, 3);
[>  frcst = zrow(1, 3);
[>  pmin = 0
[>    for i = 1: nn, ...
[>      [row, col] = size(past); ...
[>      fcc = [past; [future(i, 1:2), zrow(1, 3)]]; ...
[>      [ff1, p1] = forecast(fcc, mask(1: 3, :), row, pmin); ...
[>      [ff1, p1] = manicure(ff1, past, 3, nl, p1); ...
[>      [ff2, p2] = forecast(fcc, mask(4: 6, :), row, pmin); ...
[>      [ff2, p2] = manicure(ff2, past, 4, nl, p2); ...
[>      [ff3, p3] = forecast(fcc, mask(7: 9, :), row, pmin); ...
[>      [ff3, p3] = manicure(ff3, past, 5, nl, p3); ...
[>      past = [past; future(i, :)]; ...
[>      frcst = [frcst; [ff1(row+1, 3), ff2(row+1, 4),ff3(row+1, 5)]]; ...
[>      prob = [prob; [p1(row+1), p2(row+1), p3(row+1)]]; ...
[>    end
[>  prob = prob(2: nn+1, :);
[>  frcst = frcst(2: nn+1, :);
[>  error = frcst − future(:, 3:5);
[>  bounds = round(0.49*(future(:, 3:5) -2*ones(frcst)))
[>  return
```

The following listing shows the results of the forecasting mechanism in the neighborhood of the accident: the forecasting error matrix *error* (columns 3, 4, and 5) and the *alarm* function (column 6). Column 1 represents the sampling points, and column 2 represents the support variable *t*. Superscripts * denote guessed states. All data after time 6006 seconds (exclusive) are saturated.

```
[>  step = [991: 1030]';
[>  t = [6*990: 6: 6*1029]';
[>  results = [step, t, error, alarm]

RESULTS          =

 991.    5940.        0.    0.    0.          0.
 992.    5946.        0.    0.    0.          0.
 993.    5952.       −1.    0.    0.          0.
 994.    5958.        0.    0.    0.          0.
 995.    5964.        0.    0.    1.          0.
 996.    5970.        0.    0.    0.          0.
 997.    5976.        0.    0.    0.          0.
 998.    5982.        0.    0.    0.          0.
 999.    5988.        0.    0.    0.          0.
1000.    5994.        0.    0.    0.          0.
1001.    6000.        0.    0.   −1.          0.
1002.    6006.        0.    0.    0.          0.
1003.    6012.       −2.    3.   −1.          1.
1004.    6018.       −3.    1.*  −1.*         1.
1005.    6024.        1.*   0.    0.          1.
1006.    6030.        0.    0.    0.          1.
1007.    6036.        0.    0.    4.          1.
```

1008.	6042.	−3.*	2.*	2.*	1.
1009.	6048.	−1.*	3.*	0.	1.
1010.	6054.	0.	0.	0.	1.
1011.	6060.	−1.*	0.	−2.*	1.
1012.	6066.	−2.*	2.*	3.*	1.
1013.	6072.	−2.	0.	3.	1.
1014.	6078.	−3.*	2.*	0.	1.
1015.	6084.	0.	0.	0.	1.
1016.	6090.	0.	0.	0.	1.
1017.	6096.	0.	0.	0.	1.
1018.	6102.	−3.*	1.	0.	1.
1019.	6108.	0.	0.	0.	1.
1020.	6114.	0.	0.	0.	1.
1021.	6120.	−3.*	0.	0.	1.
1022.	6126.	−1.*	−2.*	0.	1.
1023.	6132.	0.	0.	0.	1.
1024.	6138.	0.	0.	0.	1.
1025.	6144.	−3.*	0.	3.*	1.
1026.	6150.	−3.*	0.	0.	1.
1027.	6156.	0.	0.	0.	1.
1028.	6162.	−3.*	0.	0.	1.
1029.	6168.	0.	0.	0.	1.
1030.	6174.	0.	0.	0.	1.

The intentionally long forecasting error history above shows not only the successful detection of the accident right when its effects reach the output variables, but also the inherent learning capability of the methodology. The forecasting errors grow immediately at the detection of the accident related transients, but after the stabilization of the output variables outside of the normal recoding level boundaries, the online monitoring device sees constant (saturated) trajectories which it will be pefectly able to forecast correctly.

Figures 5.7 show all five qualitative data vectors in the neighborhood of the accident compared with their quantitative equivalents.

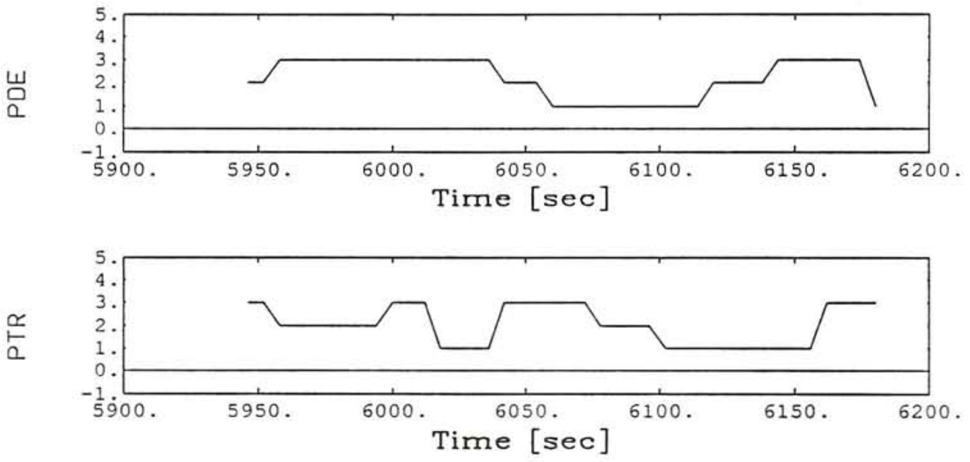

Figure 5.7a: Random perturbation inputs in the neighborhood of the accident.

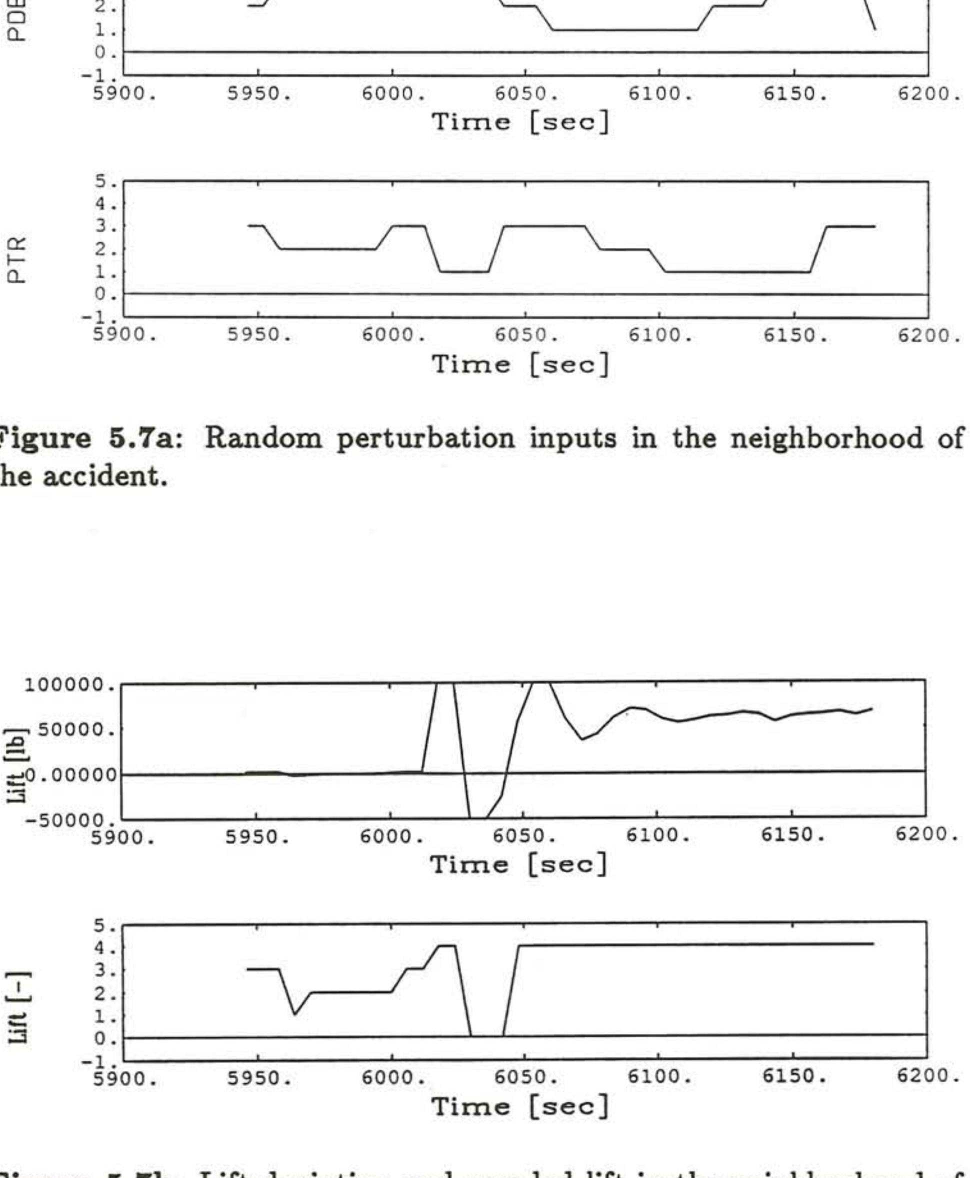

Figure 5.7b: Lift deviation and recoded lift in the neighborhood of the accident.

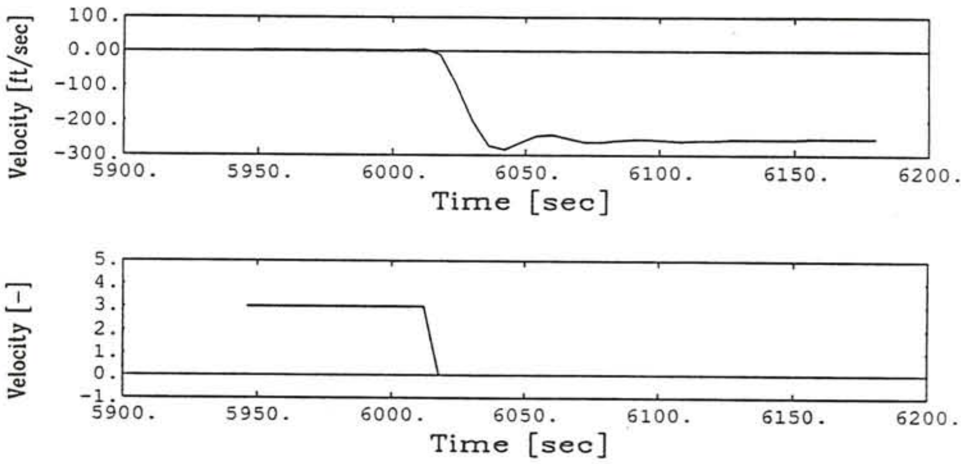

Figure 5.7c: Velocity deviation and recoded velocity in the neighborhood of the accident.

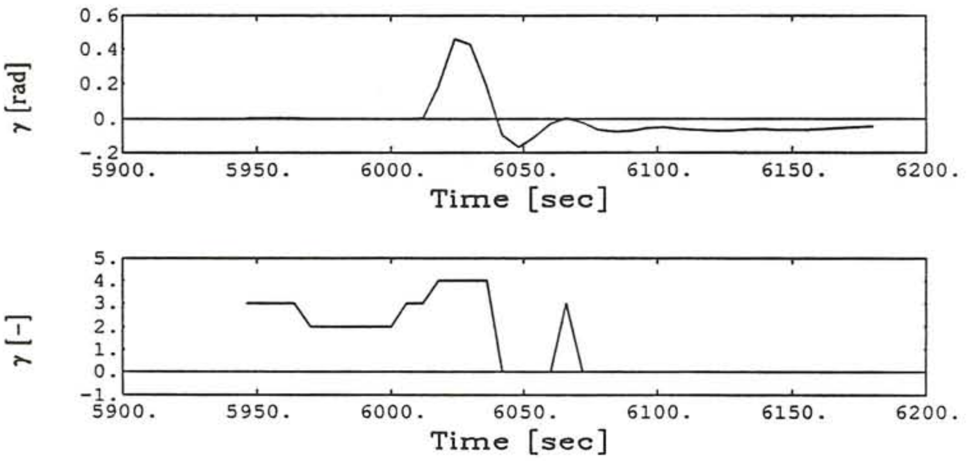

Figure 5.7d: Flight path angle deviation and recoded flight path angle in the neighborhood of the accident.

The inductive reasoning of our monitoring system gets tricked quite fast in the present situation because the behavior read by the masks first has no precedents, and all there is to do is guess that state and add the new information to its memory. With the recoding process saturated, the three output variables will not change their states, and, as the forecasting process is continued, it does not take long, before SAPS–II learns that the behavior of the outputs is not changing. After a short while, the masks that are less dependent on the input variables start being forecast perfectly right: variable v is always 0 and variable γ is always 4, no matter what values the inputs assume. Mask $mb4_l$ is not so easily influenced by the problem, but will certainly also learn that the output l will not change from the value 4.

The occurrence of saturated variables is detected by the *filter* function through the *bounds* matrix as explained below. In this way, even for a clean, almost zero forecasting error matrix, the alarm is triggered as soon as the presence of saturated variables is detected.

The error filter function **filter** was implemented through the code listed below. It calculates first the moving average based on samples of 3 elements of the sum of the absolute values of the errors at every time instant. This moving average is reconstructed to the size of the original forecast data by simply making its two first elements equal to the first element of the actual moving average vector. Next, the resulting vector is normalized with the value 4, which is the maximum value of the sum of the absolute values of the errors in three consecutive rows of the error matrix for which the *alarm* function is still not triggered. Finally, the alarm is made more sensitive with the factor 1.49 and rounded to the nearest integer (0 or 1). It is compared with the vector *satrec* which initially is a zero vector of the size of alarm and assumes the value "1" when more than two states at that time instant, or in the two previous ones are out of bounds, i.e., have been recoded into levels 0 or 4.

```
[>  //[alarm] = filter(error, bounds)
[>  abserr = abs(error);
[>  sum = abserr*ones(3, 1);
[>  smoothed = average(sum, 3);
[>  smoothed = [smoothed(1)*ones(2, 1); smoothed];
[>  normed = smoothed/4;
[>  alarm = round(1.49*normed);
[>  [m, n] = size(alarm);
[>  for i = 1:3, if alarm(i) > 1, alarm(i) = 1;
[>  absbou = abs(bounds);
[>  alarm = max([alarm, absbou]')';
[>  return
```

Figure 5.8 below depicts the output *alarm* of the error filter which is triggered three time instants after the forecasting process starts failing in its inferences about the future behavior of the system.

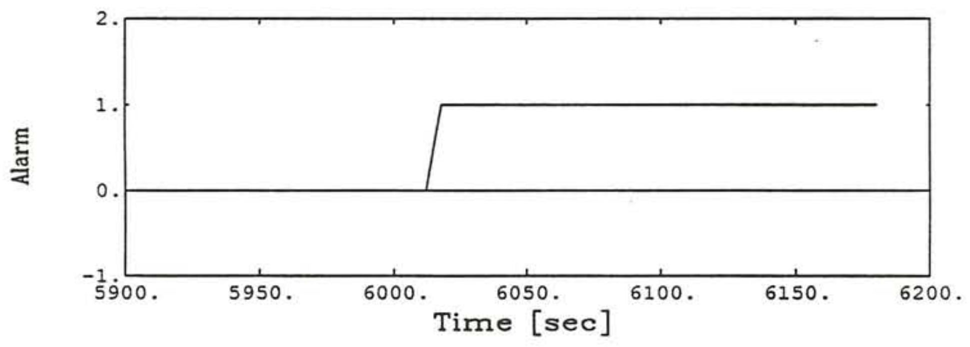

Figure 5.8 Output *alarm* of the on-line monitoring system

5.7 Recognition of the Type of Accident

After the accident related transients have stabilized, a new data model starts being collected so that different generative models that have been previously stored in the mask library can be tried out on this data, trying to identify the type of accident as soon as possible.

The data model describing the whole flight is still stored in the **rrr**.dat which can be reloaded into the Ctrl–C workspace with the **load** command using its **-a** option (load from an ACSL-output file). The loaded variable is a matrix whose columns are formed by the outputs requested from the last ACSL simulation run in the same order, from left to right, as they were written inside the square brackets in the last **start** command.

The following code first loads the time history of the whole flight, discards all data previous to the stabilization of the variables after the accident, and separates the already recoded inputs (*inrec*) from the raw outputs (*xraw*). Matrix *xdev* is then created by subtracting the mean values of the variables of *xraw* from them as has previously been discussed. Finally, as this same data will be used for all generative models separately, *inrec* and *xdev* are saved into the file **temp**.dat for future reference.

```
[>  load data <rrr -a
[>  data = data(7001: 16101, :);
[>  inrec = data(:, 1:2);
[>  xraw = data(:, 3:5);
[>  clear data
[>  [m, n] = size(xraw);
[>  mean = average(xraw, m);
[>  xdev = xraw − ones(xraw)*diag(mean);
[>  clear m n mean xraw
[>  save inrec xdev >temp
```

The generative models (sampling interval, from matrices and optimal masks) have been saved in the files **fmb4**.dat, **fmb747**.dat, **fmb5**.dat, **fmb13**.dat and **fmb14**.dat and form the mask library. Each generative model can be called back into the workspace with the Ctrl–C **load** command.

Now we want to apply every generative model to the after-the-accident data model and based on their forecasting efficiency on that data, we can decide which qualitative model fits best the type of malfunction the original model has experienced. Each set of optimal masks requires its own sampling interval and recoding levels for the recoding of the data model. After the data model has been recoded for the generative model, the forecasting efficiency of the masks is tested with a continuous forecasting process over 10 steps. The forecasting error matrix *error* is passed through the **filter** function explained in the previous section of this same chapter which, if triggered, rejects the generative system.

Let us first try out the generative model of the original flight model *fmb4*. The following code loads the generative model, samples the data with its sampling interval (*sintb4* = 6 seconds) and recodes the still non-recoded outputs using the adjusted from-matrices represented by *f*. From practical experience, we learned that a minimum data model size (past behavior information) of 200 elements is required for the methodology to yield meaningful results for this problem, and that is the number we used in all the following forecastings in addition to the 10 future states that are to be forecast.

```
[> load sintb4 fb4 mb4 <fmb4
[> load inrec xdev <temp
[> xdev = xdev(1: sintb4: sintb4*210, :);
[> inrec = inrec(1: sintb4: sintb4*210, :);
[> deff fadjust
[> f = fadjust(fb4, xdev);
[> for i=1:3, xrec(:,i) = recode(xdev(:,i), 'domain', f(2*(i-1)+1: 2*i, :), 0:4);
```

Prior to recoding the quantitative data *xdev*, the code above adjusts the from matrices of the generative system *fmp4* to the data. In this analysis as we try to apply any mask to any data to try to recognize the type of accident that occurred, the two additional external recoding levels (levels 0 and 4) created by the function **fadjust** are very important. Occurrence of recoded data in these levels must be detected externally and used as an additional information to reject the generative model. Displayed below are the adjusted from-matrices that compose the matrix f such that

$$f = \begin{pmatrix} f_l \\ f_\upsilon \\ f_\gamma \end{pmatrix}$$

$$f_l = \begin{pmatrix} -1.7772 & -1.1000 & -0.1101 & 0.1059 & 1.1000 \\ -1.1000 & -0.1101 & 0.1059 & 1.1000 & 1.5610 \end{pmatrix} \times 10^4$$

$$f_\upsilon = \begin{pmatrix} -7.5000 & -7.5000 & -0.7844 & 0.7376 & 7.5000 \\ -7.5000 & -0.7844 & 0.7376 & 7.5000 & 7.5000 \end{pmatrix}$$

$$f_\gamma = \begin{pmatrix} -0.0233 & -0.0120 & -0.0013 & 0.0013 & 0.0120 \\ -0.0120 & -0.0013 & 0.0013 & 0.0120 & 0.0207 \end{pmatrix}$$

where the leftmost and the rightmost columns of each matrix represent the two new levels created for the out-of-bounds data.

Data matrices *past* and *future* are built with the recoded data model, and the forecasting is performed with the already discussed **ennustus** function using the *mb*4 optimal masks.

```
[>  past = [inrec, xrec];
[>  past = past(1: 200, :);
[>  future = [inrec, xrec];
[>  future = future(201: 210, :);
[>  deff ennustus
[>  errorb4 = ennustus(past, future, mb4, 3)
[>  deff filter
[>  alarmb4 = filter(error)
```

Analysing the recoded data matrix, we found a very high incidence (about 50%) of 0s and 4s in the third output variable γ_{rec}, about 15% in the first variable l_{rec} and about 5% in v_{rec}. This fact by itself already practically discards the generative model B4 as a possible candidate to characterize the after-the-accident model, a decision which is strengthened by the results of the continuous forecasting process over 10 data points:

$$error_{b4} = \begin{pmatrix} 0. & 0. & 1.^{\dagger} \\ 0. & -1. & -1. \\ -2.^{*} & -2. & 0. \\ 1. & -1. & 0.^{\dagger} \\ 1.^{\dagger} & 0. & 0. \\ 0.^{*} & -1. & 1.^{\dagger} \\ -2. & -2.^{*} & 0.^{\dagger} \\ 0. & 0. & -1. \\ 0.^{*} & 0. & -1.^{\dagger} \\ 2. & 1. & -2.^{*\dagger} \end{pmatrix}$$

where the superscript * means that the forecasting error in question is relative to a guessed state, when no forecasting was possible, and the superscript † denotes forecasting errors relative to out-of-bounds states, recoded to level 0 or 4. Note that the best forecast variable is actually γ which contains the most out-of-bounds recoded states, and all but one of the successfully forecast states of γ are actually saturated values. This fact only reinforces the need for the monitoring of saturating recoded variables and the influence their occurrence should have on the acceptance or rejection of a certain generative system as the right match for the behavior described by the data. Below, our *alarm* function definitely rejects the generative model:

$$alarm_{b4} = \begin{pmatrix} 1. \\ 1. \\ 1. \\ 1. \\ 1. \\ 1. \\ 1. \\ 1. \\ 1. \\ 1. \end{pmatrix}$$

As a next try, the generative model $fmb5$ seems to be a good pick. The code used is basically the same as the previous one, but is repeated below for the sake of clarity:

```
[>  load sintb5 fb5 mb5 <fmb5
[>  load inrec xdev <temp
[>  xdev = xdev(1: sintb5: sintb5*210, :);
[>  inrec = inrec(1: sintb5: sintb5*210, :);
[>  f = fadjust(fb5, xdev);
[>  for i=1:3, xrec(:,i) = recode(xdev(:,i), 'domain', f(2*(i-1)+1: 2*i, :), 0:4);
[>  past = [inrec, xrec];
[>  past = past(1: 200, :);
[>  future = [inrec, xrec];
[>  future = future(201: 210, :);
[>  errorb5 = ennustus(past, future, mb5, 3)
[>  alarmb5 = filter(error)
```

The adjusted from-matrices were extended to slightly different values now since the sampled data model changed with the new sampling interval $sintb5 = 19$ seconds. The three from-matrices are displayed below:

$$f_l = \begin{pmatrix} -1.5489 & -1.4733 & -0.1603 & 0.1487 & 1.3025 \\ -1.4733 & -0.1603 & 0.1487 & 1.3025 & 1.3025 \end{pmatrix} \times 10^4$$

$$f_v = \begin{pmatrix} -7.9056 & -7.9056 & -1.5244 & 1.4346 & 8.1471 \\ -7.9056 & -1.5244 & 1.4346 & 8.1471 & 8.1471 \end{pmatrix}$$

$$f_\gamma = \begin{pmatrix} -0.0227 & -0.0183 & -0.0029 & 0.0027 & 0.0196 \\ -0.0183 & -0.0029 & 0.0027 & 0.0196 & 0.0253 \end{pmatrix}$$

The recoded data *xrec* does not reach the saturation regions of the from-matrix so often this time. This is understandable and expected, since by comparing *fmb*5 and *fmb*747 from-matrices we see that there is not a very large difference in their original external limits.

Two states had to be guessed in the forecasting process and they are marked with the superscript * in the $error_{b5}$ matrix below. The elements with the superscript † refer to elements that had been recoded into the saturation levels.

$$error_{b5} = \begin{pmatrix} 0. & 0. & 0. \\ 0. & -2.^* & 0. \\ 2. & -1. & 1. \\ 0. & 0. & -1.^\dagger \\ 1. & -1. & 0. \\ -2. & -2.^* & 0. \\ 0. & 0. & 0. \\ 2. & 1. & 0. \\ 0. & 0. & 0. \\ 1.^\dagger & 0. & 0. \end{pmatrix}$$

Note the excellent forecasting of the third output variable. One explanation for this behavior is the fact that the sampling interval used by the B5 model is almost double of *sintb*747, i.e., both masks "see" almost the same dynamics, *sintb*5 being able to read every other data point as compared to *mb*747. The strongest generating sampling variable of the mask $mb747_\gamma$ (the one that appears throughout its mask history) is $pde(t - sintb747)$ to generate $\gamma(t + sintb747)$, $sintb747 = 11$ seconds. In the $mb5$ mask, the generating sampling variable $pde(t)$ also influences $v(t + sintb5)$, $sintb5 = 19$ seconds, and even in the $mb5_\gamma$ mask history, this variable appears in every single suboptimal mask except for the mask of complexity two which has the lowest quality factor. As $sintb5 \approx 2 \times sintb747$, there is a good possibility that mask $mb5$ is actually fitting the data, at least up to some quality. Analogous forecastings in different sections of the data model do not yield so low error matrices, but this particular case is a very good illustration of how the methodology can lead to precipitated erroneous conclusions.

The $alarm_{b5}$ vector below rejects the generative model, but it is influenced also by the two guessed states, each causing an error with an absolute value of 2. Had both guesses been correct, yielding zero forecasting error, the vector would not have been so clearly triggered. However, due to the occasional saturation of the variables in the recoding process, the generative system should not be accepted.

$$alarm_{b5} = \begin{pmatrix} 1. \\ 1. \\ 1. \\ 1. \\ 1. \\ 1. \\ 1. \\ 1. \\ 0. \\ 1. \end{pmatrix}$$

The next qualitative model to be tried on the data represents the model B13. The adjusted from-matrices required by the model are:

$$fb13_l = \begin{pmatrix} -2.5914 & -2.5914.0 & -0.1058 & 0.0961 & 2.4755 \\ -2.5914 & -0.1058.2 & 0.0961 & 2.4755 & 2.4755 \end{pmatrix} \times 10^4$$

$$fb13_v = \begin{pmatrix} -18.0545 & -18.0545 & -2.6505 & 2.5067 & 18.3080 \\ -18.0545 & -2.6505 & 2.5067 & 18.3080 & 18.3080 \end{pmatrix}$$

$$fb13_\gamma = \begin{pmatrix} -0.0240 & -0.0227 & -0.0036 & 0.0036 & 0.0222 \\ -0.0227 & -0.0036 & 0.0036 & 0.0222 & 0.0222 \end{pmatrix}$$

The resulting recoded data model has a very peculiar and interesting characteristic: the second output variable is about 80% of the times recoded into the level 2. This happens due to the recoding levels imposed by $fb13_v$ whose inner level's limits are almost the double of $fb747_v$'s which situates them at about half of the maximum variation of v_{b747}. This is an excelent example of how non-varying states can fool this methodology. The longer a variable remains in one level, the higher will the methodology set its probability of so remaining, and even worse: the successful forecasting of saturated variables or variables with overdimensioned recoding levels is independent of the mask used (with time, basically any mask will give clean zero forecasting errors). Therefore, monitoring the recoded data model for the occurrence of saturated states is not enough: the monitoring has to be extended to non-varying states as well.

As expected, the second variable was forecast 100% right but even so, due to the the very poor forecasting of the two other variables, the alarm is again triggered without any need for the information about the non-variance of v_{rec}.

$$error_{b13} = \begin{pmatrix} 0. & 0. & -1. \\ -2. & 0. & 1. \\ 0. & 0. & 1. \\ -1. & 0. & 1. \\ 1. & 0. & 2. \\ 1. & 0. & 2. \\ 0. & 0. & 2. \\ 1. & 0. & 0. \\ 0. & 0. & 2. \\ 0. & 0. & 1. \end{pmatrix}$$

$$alarm_{b13} = \begin{pmatrix} 1. \\ 1. \\ 1. \\ 1. \\ 1. \\ 1. \\ 1. \\ 1. \\ 1. \\ 0. \end{pmatrix}$$

The next qualitative model to be tested on the data is B14. As can be seen from its adjusted from-matrices displayed below, all variations of the variables relative to this model are smaller than the dynamics of the data under study (compare with $fb747$).

$$fb14_l = \begin{pmatrix} -1.7772 & -0.6079 & -0.1082 & 0.0988 & 0.6602 \\ -0.6079 & -0.1082 & 0.0988 & 0.6602 & 1.1746 \end{pmatrix} \times 10^4$$

$$fb14_v = \begin{pmatrix} -5.4727 & -4.9056 & -0.9732 & 0.9774 & 4.6891 \\ -4.9056 & -0.9732 & 0.9774 & 4.6891 & 4.6891 \end{pmatrix}$$

$$fb14_\gamma = \begin{pmatrix} -0.0240 & -0.0112 & -0.0018 & 0.0017 & 0.0107 \\ -0.0112 & -0.0018 & 0.0017 & 0.0107 & 0.0253 \end{pmatrix}$$

The recoded data model, as expected saturates from time to time. Variables l_{rec} and γ_{rec} have an incidence of saturation of about 25%, while v_{rec} shows only about 5% saturation. However, this will not account for the decision of rejecting the qualitative model since the *alarm* is triggered by the visibly poor forecasts of all variables. Again superscripts † identify the forecasting errors relative to saturated data points.

$$error_{b14} = \begin{pmatrix} -2. & -2. & -1.^\dagger \\ -1. & -1. & 0. \\ 0. & 0. & 2. \\ 1. & 0. & -1. \\ 0. & 1. & -1. \\ 1. & 1. & -1. \\ 0. & 2. & -1. \\ 1.^\dagger & 1. & 0. \\ -1.^\dagger & -2. & 1.^\dagger \\ -1.^\dagger & -1. & -1.^\dagger \end{pmatrix}$$

$$alarm_{b14} = \begin{pmatrix} 1. \\ 1. \\ 1. \\ 1. \\ 1. \\ 1. \\ 1. \\ 1. \\ 1. \\ 1. \end{pmatrix}$$

And finally, the last generative model (B747) should now recognize the data matching first the from-matrices such that no, or practically no data is recoded into the saturation levels 0 and 4, and then with low forecasting errors.

The adjusted from-matrices are:

$$fb747_l = \begin{pmatrix} -1.6029 & -1.6029 & -0.2137 & 0.2113 & 1.3981 \\ -1.6029 & -0.2137 & 0.2113 & 1.3981 & 1.3981 \end{pmatrix} \times 10^4$$

$$fb747_v = \begin{pmatrix} -5.9153 & -5.9153 & -1.0733 & 1.0150 & 6.1723 \\ -5.9153 & -1.0733 & 1.0150 & 6.1723 & 6.1723 \end{pmatrix}$$

$$fb747_\gamma = \begin{pmatrix} -0.0278 & -0.0278 & -0.0050 & 0.0050 & 0.0290 \\ -0.0278 & -0.0050 & 0.0050 & 0.0290 & 0.0290 \end{pmatrix}$$

There is no occurrence of saturated states in the recoded data model and the forecasting process presents one guessed state due to the still small data size (200 points), taking into account that the mask that could not forecast the guessed state is of complexity five ($mb747_\gamma$). The complexity of the two other masks of the system is four which also makes it more difficult to forecast having a past behavior matrix of only 200 elements.

$$error_{b747} = \begin{pmatrix} 0. & -1. & 0. \\ 0. & 0. & -2.^* \\ 0. & 0. & -1. \\ -1. & 0. & -1. \\ 1. & 0. & 0. \\ 0. & 0. & 0. \\ 1. & 0. & -1. \\ 0. & -1. & 0. \\ -1. & 0. & -1. \\ 0. & 0. & 0. \end{pmatrix}$$

$$alarm_{b747} = \begin{pmatrix} 0. \\ 0. \\ 0. \\ 1. \\ 0. \\ 0. \\ 0. \\ 0. \\ 1. \\ 0. \end{pmatrix}$$

Recognition of a B4/B13 accident

Considering now the occurrence of a B4/B13 accident (the original B4 model turns into the model B13 after the accident) the recognition of the type of accident will follow the same steps as in the previous section. The two models present fairly similar behaviors, with the major difference that the broken model is very much slower and all variables have much higher amplitude transients.

The data model is created in a similar way only that now, as we are not interested in the detection of the accident, *tbreak* is set to one second, to create only data relative to the after-the-accident behavior of the aircraft. The code below is basically the same as what was used in the previous simulation to create the B4/B747 accident's data model:

```
[>  acsl('set tmx = 6000, cint=1, tbreak=1, inpt=4')
[>  do B13
[>  [pde, ptr, l, v, ga] = start;
[>  inrec = [pde(1001: 6000), ptr(1001: 6000)];
[>  xraw = [l(1001: 6000), v(1001: 6000), ga(1001:6000)];
[>  [m, n] = size(xraw);
[>  mean = average(xraw, m);
[>  xdev = xraw − ones(xraw)*diag(mean);
[>  clear m n mean xraw
[>  save inrec xdev >temp
```

The same procedure was followed to try to match qualitative and quantitative models. Model $fmb13$ clearly recognizes the data with a fairly clean forecasting error matrix, and without a single non-zero element in the *alarm* vector.

$$error_{b13} = \begin{pmatrix} 0. & 0. & 0. \\ 0. & 0. & 1. \\ 0. & 0. & 1. \\ 0. & 0. & 0. \\ -1. & 0. & 1. \\ 0. & 0. & 0. \\ 0. & -1. & 0. \\ -2. & 0. & 0. \\ 0. & 0. & -1. \\ 0. & 0. & 0. \end{pmatrix}$$

$$alarm_{b13} = \begin{pmatrix} 0. \\ 0. \\ 0. \\ 0. \\ 0. \\ 0. \\ 0. \\ 0. \\ 0. \\ 0. \end{pmatrix}$$

All other qualitative models require recoding levels that saturate (some more, some less) with the higher amplitude responses of the B13 model. Most of them have visibly poor forecasting capabilities on the data and are immediately rejected by the *alarm* signal.

Model B5 is the only model that has a very good forecasting error matrix which would not have triggered the alarm when passed through the **filter** function. The signal $alarm_{b5}$ is triggered by the saturated states of v and the generative model $fmb5$ is rejected. The good forecasting capability of $mp5$ on this B13 data may be explained, in addition to the presence of saturated recoded data points, by the fact that both systems have fairly comparable sampling intervals (19 and 16 seconds) and their optimal masks are similar as well. More recoding levels or even a larger source model may be necessary to better distinguish between these two models. In the present case, the problem is taken care of by the constant presence of recoded data in the saturated levels.

$$error_{b5} = \begin{pmatrix} 0. & 0.^{\dagger} & 0. \\ 0. & 0.^{\dagger} & 0. \\ -1. & 0.^{\dagger} & 0. \\ 1. & 0.^{\dagger} & 0. \\ -1. & 0.^{*} & 0.^{*} \\ 0. & -1.^{*} & 0. \\ 0. & 0. & -1. \\ 0. & 1. & 0. \\ 0. & 1.^{\dagger} & -1. \\ -1. & 1.^{\dagger} & 0. \end{pmatrix}$$

$$alarm_{b5} = \begin{pmatrix} 1. \\ 1. \\ 1. \\ 1. \\ 0. \\ 0. \\ 0. \\ 0. \\ 1. \\ 1. \end{pmatrix}$$

Recognition of a B4/B5 accident

Model B5 is characterized by the most oscillatory response to step inputs among all five models presented as can be seen from the figures 5.3. It is also the slowest model, with its linearized model's slowest time constant being around 19 seconds.

We performed a new mask library consultation to try to recognize a B5 data model obtained and processed in the same way as in the previous experiment.

The data is satisfactorily recognized by the $fmb5$ generative model which requires the recoding levels represented by the $fb5$ from-matrices, extension of which to the maximum and minimum recorded states in the present data model is represented by the f matrices below:

$$f_l = \begin{pmatrix} -1.4733 & -1.4733 & -0.1603 & 0.1487 & 1.3025 \\ -1.4733 & -0.1603 & 0.1487 & 1.3025 & 1.3025 \end{pmatrix} \times 10^4$$

$$f_v = \begin{pmatrix} -7.9056 & -7.9056 & -1.5244 & 1.4346 & 8.1471 \\ -7.9056 & -1.5244 & 1.4346 & 8.1471 & 8.1471 \end{pmatrix}$$

$$f_\gamma = \begin{pmatrix} -0.0183 & -0.0183 & -0.0029 & 0.0027 & 0.0196 \\ -0.0183 & -0.0029 & 0.0027 & 0.0196 & 0.0196 \end{pmatrix}$$

Note that as the matrices have been adjusted to a data model generated by the B5 model itself, the external recoding levels (0 and 4) are widthless, i.e., the maximum and minimum recorded values of each variable are smaller or equal to the external limits of their respective $fb5$ from-matrix, and therefore there will be no elements of $xrec$ recoded into the saturated levels 0 and 4.

The qualitative model $fmb5$ cannot be promptly validated by the methodology since the alarm signal is triggered for one time instant. The forecasting is not perfect (around 60% of right forecasts in all variables, but each miss is just a forecast to the adjacent level (all non-zero errors have absolute value 1). The forecasting error matrix $error_{b5}$ and the alarm signal $alarm_{b5}$ are displayed below:

$$error_{b5} = \begin{pmatrix} -1. & 1. & 0. \\ 0. & 0. & -1. \\ 0. & 0. & 0. \\ 0. & 0. & 0. \\ 0. & 0. & 1. \\ 0. & 0. & 0. \\ 1. & 0. & 1. \\ 0. & -1. & -1. \\ 0. & -1. & 0. \\ -1. & 0. & 0. \end{pmatrix}$$

$$alarm_{b5} = \begin{pmatrix} 0. \\ 0. \\ 0. \\ 0. \\ 0. \\ 0. \\ 0. \\ 0. \\ 1. \\ 0. \end{pmatrix}$$

Applying the qualitative model $fmb747$ to the B5 data model yields very good forecasting results for the variable γ_{rec} which was successfully forecast 80% of the times. There are two plausible reasons for $mb747_\gamma$ being able to do such a good job. The first one is that, analysing the inner recoding levels for γ of these two systems, we note that by the use of $fb747_\gamma$ on γ created by a B5 model, we are extending this recoding level from $\approx \pm 0.0028$ to ± 0.0050, which increases the occurrence of 2's in γ_{rec}. This fact added to the difference in the time constants associated with both models (B5 is almost twice as slow as B747) helps the generation of a data model with longer strings of data recoded into the same level, and that is what happened with γ_rec in the $future$ matrix: the eight first elements of the last column of $future$ are 2's.

$$future = \begin{pmatrix} 3. & 3. & 3. & 3. & 2. \\ 3. & 1. & 2. & 3. & 2. \\ 3. & 1. & 1. & 2. & 2. \\ 3. & 2. & 2. & 3. & 2. \\ 3. & 1. & 2. & 3. & 2. \\ 3. & 1. & 2. & 2. & 2. \\ 1. & 1. & 3. & 2. & 2. \\ 1. & 1. & 2. & 1. & 2. \\ 3. & 1. & 1. & 1. & 1. \\ 2. & 3. & 2. & 1. & 1. \end{pmatrix}$$

The second reason is related to the masks $mb5_\gamma$ and $mb747_\gamma$. Masks $mb5$ have its windows 19 seconds apart whereas $mb747$ masks' windows are 11 seconds apart, meaning that through every second window of $mb747$, an observer is able to see a similar behavior as he can see through adjacent windows of $mb5$. Element (-1) in $fmb747_\gamma$ must be seeing about the same behavior as element (-1) in $fmb5_\gamma$, and if its influence is strong in both translational rules, both masks will forecast well in data generated by any of the two models.

Distinguishing one model from the other may require longer and/or several forecasting analyses. In this case, the *alarm* signal is triggered, and the model $fmb747$ is rejected considering the sporadic saturation of v_{rec} and slight overdimensioning of the level 2 of $fb747_\gamma$.

$$error_{b747} = \begin{pmatrix} 0. & 0. & 0. \\ 1. & 0. & 0. \\ 1. & -1. & -1. \\ -1. & -2. & 0. \\ 0. & 0. & 0. \\ 0. & -1. & 0. \\ -2. & -1. & 0. \\ 0. & 0. & 0. \\ 0. & 0. & 1. \\ 1. & 1. & 0. \end{pmatrix}$$

$$alarm_{b747} = \begin{pmatrix} 0. \\ 0. \\ 0. \\ 1. \\ 1. \\ 1. \\ 0. \\ 0. \\ 0. \\ 0. \end{pmatrix}$$

The generative model $fmb13$ turns out to be a problem when used in this data: its forecasting is very good due to the fact that its sampling interval $sintb13$ is comparable with $sintb5$ (16 and 19) and both models have very similar optimal masks. Longer forecasting data is needed to distinguish between the two qualitative models when using B5 or B13 data. Here the ten-step forecasting error matrix is very convincingly showing that the accident is of B13-type. It may even be that more recoding levels are required to distinguish B5 data from B13 data. The one fact that gives a hint which may lead to the rejection of the model is that the central recoding limit of v is apparently overdimensioned (more than 50% of the data was recoded into level 2 with the same first random perturbation driving the model).

$$error_{b13} = \begin{pmatrix} 0. & 0. & 0. \\ 0. & 0. & 0. \\ 0. & 0. & -1. \\ 0. & 0. & 0. \\ 1. & 0. & 0. \\ 0. & -1. & 1. \\ 0. & -1. & 0. \\ 0. & 0. & 0. \\ -1. & 0. & 0. \\ 0. & 0. & 0. \end{pmatrix}$$

$$alarm_{b13} = \begin{pmatrix} 0. \\ 0. \\ 0. \\ 0. \\ 0. \\ 0. \\ 0. \\ 0. \\ 0. \\ 0. \end{pmatrix}$$

The generative model $fmb14$ recodes 10 - 15% of the data into the saturation levels. Its forecasting results are bad enough to leave no doubt about the decision to reject the model.

$$error_{b14} = \begin{pmatrix} -1. & 1. & -1. \\ 0. & 1. & -1. \\ 0. & 0. & 1. \\ -1.^{\dagger} & 0. & 1. \\ 0. & 0. & 0. \\ 1.^{\dagger} & 2. & 0. \\ 0. & 0. & 0. \\ 0. & 1. & 0. \\ 0. & 1. & -1. \\ 0. & 0. & 0. \end{pmatrix}$$

$$alarm_{b14} = \begin{pmatrix} 1. \\ 1. \\ 1. \\ 1. \\ 0. \\ 1. \\ 0. \\ 0. \\ 0. \\ 0. \end{pmatrix}$$

Recognition of the B4/B14 accident

Generative model $fmb14$, as expected, forecasts very well on the data created by the quantitative model B14. The forecasting error matrix and the output $alarm$ of the filter applied to it are displayed below:

$$error_{b14} = \begin{pmatrix} -1. & 0. & 0. \\ 0. & 0. & 0. \\ 0. & 0. & 0. \\ -1. & -1. & 0. \\ 0. & 0. & 0. \\ 0. & 1. & 0. \\ 0. & 0. & 0. \\ 1. & 0. & 0. \\ 1. & 0. & 0. \\ 0. & 0. & 0. \end{pmatrix}$$

$$alarm_{b14} = \begin{pmatrix} 0. \\ 0. \\ 0. \\ 0. \\ 0. \\ 0. \\ 0. \\ 0. \\ 0. \\ 0. \end{pmatrix}$$

The use of the generative model $fmb747$ on the data relative to the model B14 creates a problem in the recoding process of the variables l and γ. The limits of the inner recoding level of these two variables are much higher in the $fb747$ from-matrix than in the $fb14$. The limits relative to the variable l are twice as large in the $fb747$ from-matrix than in the $fb14$ matrix, and the limits relative to γ are three times as large. The overdimensioning of the inner level of these two variables causes the recoding of a large number of elements into the level 2, which makes it "easy" for any mask to forecast that the next state of these variables will be 2.

Even with the forecasting being acceptable enough since the alarm signal is only triggered for two consecutive time instants, the generative model is rejected due to the excessive occurrence of 2's in l_{rec} (62%) and γ_{rec} (68%). The elements marked with a ‡ in the *error* matrix below refer to forecasting errors of elements that had been recoded into the level 2. Superscript * denotes a guessed element.

$$error_{b747} = \begin{pmatrix} -1. & 0. & 0.^{\ddagger} \\ 0.^{\ddagger} & 0. & 0.^{\ddagger} \\ 0. & -1. & -1.^{\ddagger} \\ 0.^{\ddagger} & -1. & 0.^{\ddagger} \\ 0.^{\ddagger} & 0. & 0.^{\ddagger} \\ 0.^{\ddagger} & -1. & -1.^{\ddagger} \\ 0.^{\ddagger} & 1. & -1.^{\ddagger} \\ 0.^{\ddagger} & 0. & -1.^{*\ddagger} \\ 1. & 0. & 1. \\ 0. & 0. & 0. \end{pmatrix}$$

$$alarm_{b747} = \begin{pmatrix} 0. \\ 0. \\ 0. \\ 0. \\ 0. \\ 0. \\ 0. \\ 0. \\ 1. \\ 1. \\ 0. \end{pmatrix}$$

Generative model $fmb5$ applied to the data yields fairly good results, mainly on variable l. There is no saturation or visible overdimensioning of the inner recoding level. The *alarm* signal is triggered for three time instants and the model is not accepted since $fmb13$ has given better results. The model should not be totally rejected, but put aside and given high priority to be tried in a new consultation to the library.

$$error_{b5} = \begin{pmatrix} 0. & 0. & 0. \\ 0. & -1. & 0. \\ 0. & 0. & 0. \\ 0. & 0. & 0. \\ 0. & 0. & 1. \\ 0. & -2. & -1. \\ 1. & 1. & 1. \\ 0. & -1. & -1. \\ 0. & 0. & 0. \\ 0. & 0. & 0. \end{pmatrix}$$

$$alarm_{b5} = \begin{pmatrix} 0. \\ 0. \\ 0. \\ 0. \\ 0. \\ 0. \\ 1. \\ 1. \\ 1. \\ 0. \end{pmatrix}$$

Generative model B13 is rejected based on the visible overdimensioning of the inner recoding level of the variable v (77% of the elements of v_{rec} are 2s) and not so badly, but of the variable γ as well (49%). The forecasting error matrix $error_{b13}$ and its associated alarm signal are displayed below, where elements marked with a ‡ refer to foreasting errors associated with elements recoded into the overdimensioned levels, and elements marked with * refer to guessed states.

$$error_{b13} = \begin{pmatrix} 0. & 0.^{\ddagger} & 0.^{\ddagger} \\ -1. & 0.^{\ddagger} & 0.^{\ddagger} \\ 2. & 0.^{\ddagger} & -1. \\ 0. & 0. & 0.^{\ddagger} \\ -1. & -1.^{*\ddagger} & 0. \\ 0. & 0.^{\ddagger} & 0.^{\ddagger} \\ -1. & 0.^{\ddagger} & 0.^{\ddagger} \\ 1. & 0.^{\ddagger} & 0.^{\ddagger} \\ 0. & 0.^{\ddagger} & 0.^{\ddagger} \\ 0. & 0. & -1. \end{pmatrix}$$

$$alarm_{b13} = \begin{pmatrix} 0. \\ 0. \\ 0. \\ 0. \\ 1. \\ 0. \\ 0. \\ 0. \\ 0. \\ 0. \end{pmatrix}$$

CHAPTER 6
CONCLUSION

The main concern of this work has been to utilize the inductive reasoning capabilities inherent to the presented methodology to include learning capabilities to the decision making process of highly automated systems.

This thesis shows how qualitative simulation can be successfully applied for on-line monitoring of systems. The methodology described is able to create a very accurate inductive reasoning device that is able to forecast with extremely low uncertainty the near future behavior of the monitored system. Such a device, having learned the system behavior, is able to recognize this behavior in a given Data Model and, as shown in the experiments described in chapter 5 of this work, a structural failure in the mother-system is easily detected with it.

Still making use of the inductive reasoning capabilities inherent to the methodology, it is shown how it is possible to recognize the behavioral pattern of a certain Data Model with different prerecorded Generative Models. Once the behavioral pattern of the system outputs is recognized, it should be perfectly feasible to build a new control stategy regarding the new, structurally changed mother-system. This may be done using predefined control laws or building the control stategy on-line, once the most difficult part, the recognition of the behavioral pattern of the system after the unpredicted structural failure, has been accomplished.

The speed of the inductive reasoning process is system dependent since all non-prerecorded knowledge has to be learned from the actual behavior of the monitored system. In emergency situations, this learning process may be too slow for the urgent global decisions that have to be taken since the response of the system to many different sets of inputs has to be analysed and repeated several times to extract the behavioral information. In this thesis, a preliminary study has demosntated the feasibility of structure recognition. The implemented recognizer is able to correctly distinguish between several prerecorded Generative Systems, representing, for example, the most likely types of accidents that may happen.

Of crucial importance in the methodology is the qualitative modeling. First, it is essential to choose a meaningful source model such that the problem to be analysed is well represented. Then the number of recoding levels to be used for the variables must be selected, and limits for each level must be determined. More research is needed to develop tools to support the user in selecting these parameters in an optimal manner.

Data manipulation prior to the recoding process is another point to be carefully considered. E.g., standardized deviations of the data from their mean values could be used in the raw Data Model such that the Generative Models could be used in a more general way, applicable to any Data Model.

APPENDIX

ACSL PROGRAM LISTING

```
PROGRAM AIRCRAFT LONGITUDINAL STABILITY STUDY

INITIAL
       "-------ACSL CONSTANTS-----------------------------------"
       CONSTANT       TMX    =  12000.0     ,  ...
                      CINT   =      6.0     ,  ...
                      SEED   =    555.0     ,  ...
                      TBREAK =  30000.0

       "-------INITIALIZE RANDOM NUMBER GENERATOR SEED---------"
       UNIFI(SEED)

       "-------AIRPLANE CONSTANTS-------------------------------"
       CONSTANT       IY    = 27000000.0    ,  ...
                      CBAR =        27.3    ,  ...
                      WT    =   500000.0    ,  ...
                      G     =       32.2    ,  ...
                      S     =     6000.0    ,  ...
                      RO    =         0.0012  $"ALTITUDE 20000 FT"

       "-------NEED MASS FROM WEIGHT IN LBS--------------------"
       MASS    = WT/G

       "-------AERODYNAMIC CONSTANTS---------------------------"
       CONSTANT       CMZ  =   0.039        ,  ...
                      CLZ  =   0.5455       ,  ...
                      CDZ  =   0.036667     ,  ...
                      CMAL =  -0.74         ,  ...
                      CLAL =   5.2          ,  ...
                      CDAL =   0.26         ,  ...
                      CMDE =  -1.4          ,  ...
                      CLDE =   0.36         ,  ...
                      CDDE =   0.0          ,  ...
                      CMAD =  -8.0          ,  ...
                      CLAD =   2.0          ,  ...
                      CMQ  = -22.0          ,  ...
                      CLQ  =   5.5

       "-------INITIAL CONDITIONS-------------------------------"
       CONSTANT       VZ   =    500.1375    ,  ...
                      HZ   =  20000.0       ,  ...
                      XZ   =      0.0       ,  ...
                      QZ   =      0.0       ,  ...
                      THZ  =     -0.000055  ,  ...
                      GAZ  =      0.0       ,  ...
                      ALZ  =     -0.000055  ,  ...
```

```
                DEZ =      0.027886   , ...
                TRZ =   33005.5

"-------FEEDBACK GAINS---------------------------------"
CONSTANT        KTH   =     0.25       , ...
                KU    =    40.0

"-------CONSTANTS FOR THE TRIMMING PHASE---------------"
CONSTANT        ERMX   =     0.1        , ...
                K1     =     0.02       , ...
                K2     =    -1.0        , ...
                K3     =     3.0        , ...
                KEVD   =     1.0        , ...
                KEQD   =    10.0        , ...
                KEGAD  =   100.0        , ...
                DETRIM =     0.027886   , ...
                THTRIM =    -0.000055   , ...
                TRTRIM = 33005.5        , ...
                UTRIM  =   500.1375

LOGICAL         START
                START = .FALSE.

"-------SET DEFAULT FLIGHT PHASE = UNDRIVEN CASE-------"
CONSTANT        INPT =  0.0

"-------INPT = 1 DEFAULT PARAMETERS--------------------"
CONSTANT        DDE1 =    -0.001       , ...
                DDE2 =     0.001       , ...
                DDE3 =     0.0         , ...
                TDE1 =    10.0         , ...
                TDE2 =   250.0         , ...
                TDE3 =     0.0         , ...
                DTR1 =  3000.0         , ...
                DTR2 =     0.0         , ...
                DTR3 =     0.0         , ...
                TTR1 =   500.0         , ...
                TTR2 =     0.0         , ...
                TTR3 =     0.0

"-------INPT = 2 AND INPT = 4 DEFAULT PARAMETERS-----"
CONSTANT        TPULSE =    0.0        , ...
                DDE    =    0.001      , ...
                DTR    = 3000.0        , ...
                SINT   =    6.0        , ...
                DELTA  =   20.0        , ...
                DELTA1 =    0.9        , ...
                DELTA2 =    1.1

"-------INPT = 3 DEFAULT PARAMETERS-----------------"
CONSTANT        WDE    =         0.10      , ...
                WTR    =         0.05      , ...
                CLZNEW =         0.5455    , ...
                CLALNW =         5.2       , ...
```

```
                                   CLDENW  =        0.36      ,   ...
                                   CLADNW  =        2.0       ,   ...
                                   CLQNEW  =        5.5       ,   ...
                                   CMZNEW  =        0.039     ,   ...
                                   CMALNW  =       -0.74      ,   ...
                                   CMDENW  =       -1.4       ,   ...
                                   CMADNW  =       -8.0       ,   ...
                                   CMQNEW  =      -22.0       ,   ...
                                   CDZNEW  =        0.036667  ,   ...
                                   CDALNW  =        0.26      ,   ...
                                   CDDENW  =        0.0       ,   ...
                                   ALZNEW  =       -0.000055  ,   ...
                                   THZNEW  =       -0.000055  ,   ...
                                   GAZNEW  =        0.0       ,   ...
                                   DEZNEW  =        0.027886  ,   ...
                                   TRZNEW  =    33005.5       ,   ...
                                   VZNEW   =      500.1375    ,   ...
                                   IYNEW   = 27000000.0       ,   ...
                                   SNEW    =     5000.0       ,   ...
                                   WTNEW   =   500000.0       ,   ...
                                   CBARNW  =       27.3       ,   ...
                                   KTHNEW  =        0.25      ,   ...
                                   KUNEW   =       40.0       ,   ...
                                   DDENEW  =        0.001     ,   ...
                                   DTRNEW  =     3000.0       ,   ...
                                   SINTNW  =        6.0

     I1..CONTINUE
END  $"OF INITIAL"


DYNAMIC

DERIVATIVE

       "-------ELEVATOR CONTROL-----------------------------"
       DE = DETRIM + KTH*(TH - THTRIM)

       "-------THRUST CONTROL-------------------------------"
       TR = TRTRIM + KU*(U - UTRIM)

       "-------TANGENTIAL VELOCITY--------------------------"
       U = V*COS(AL)

       "-------ANGLE OF ATTACK -----------------------------"
       AL = TH - GA

       "-------DRAG COEFFICIENT ----------------------------"
       CD = CDZ + CDAL*AL + CDDE*DE

       "-------LIFT COEFFICIENT-----------------------------"
       CL = CLZ + CLAL*AL + CLDE*DE + ...
            (CBAR/(2*V))*(CLAD*(Q - GAD) + CLQ*Q)
```

```
        "-------PITCH MOMENT COEFFICIENT---------------------"
        CM = CMZ + CMAL*AL + CMDE*DE + ...
             (CBAR/(2*V))*(CMAD*(Q - GAD) + CMQ*Q)

        "-------DYNAMIC PRESSURE-----------------------------"
        QP = 0.5*RO*(V**2)

        "-------DRAG AND LIFT--------------------------------"
        D  = QP*S*CD
        L  = QP*S*CL

        "-------PITCHING MOMENT------------------------------"
        M  = QP*S*CBAR*CM

        "-------FLIGHT PATH RATE-----------------------------"
        GAD = (QP*S*(CLZ + AL*CLAL + DE*CLDE + ...
              (CBAR/(2*V))*(CLAD + CLQ)*Q) - WT*COS(GA) + ...
              TR*SIN(AL))/(MASS*V + 0.25*RO*S*V*CBAR*CLAD)

        "-------LONGITUTINAL ACCELERATION--------------------"
        VD = (TR*COS(AL) - D - WT*SIN(GA))/MASS

        "-------PITCH RATE DERIVATIVE------------------------"
        QD = M/IY

        "-------VERTICAL VELOCITY----------------------------"
        HD = V*SIN(GA)

        "-------HORIZONTAL VELOCITY--------------------------"
        XD = V*COS(GA)

        "--------PITCH RATE----------------------------------"
        Q  = INTVC(QD, QZ)

        "--------PITCH ANGLE---------------------------------"
        TH = INTEG(Q, THZ)

        "--------LONGITUDINAL VELOCITY-----------------------"
        V  = INTVC(VD, VZ)

        "--------FLIGHT PATH ANGLE---------------------------"
        GA = INTVC(GAD, GAZ)

        "--------ALTITUDE------------------------------------"
        H  = INTVC(HD, HZ)

        "--------HORIZONTAL DISTANCE TRAVELLED---------------"
        X  = INTEG(XD, XZ)

END  $"OF DERIVATIVE"
```

```
  "======================INITIAL TRIM PHASE==========="

  "--------ONCE FLYING, SKIP ITERATION-----------------"

  IF(START .EQ. .TRUE.) GO TO D1
  "--------COMPUTE WEIGHTED ERROR FROM TRIM------------"
  ERROR = (KEVD*VD)**2 + (KEQD*QD)**2 + (KEGAD*GAD)**2

  "--------IF WITHIN TOLERANCE, START FLIGHT-----------"
  START   = ERROR .LE. ERMX
  IF(START .EQ. .TRUE.) GO TO D1

  "--------COMPUTE NEW TRIAL VALUES--------------------"
  GAZ       = GAZ + K1*VD
  THZ       = THZ + K2*GAD
  DEZ       = DEZ + K3*QD

  "--------SAVE TRIM VALUES FOR FDBK-------------------"
  THTRIM = THZ
  UTRIM  = U

  "--------DISPLAY ITERATION INFORMATION---------------"
  LINES(1)
  PRINT 97, VD, QD, GAD
97..FORMAT(4H VD ,E10.4,4H QD ,E10.4,5H GAD ,E10.4)
  LINES(1)
  PRINT 98, GAZ, THZ, DEZ, ERROR
98..FORMAT(5H GAZ ,E12.4,5H THZ ,E12.4,5H DE  ,E12.4)
  LINES(1)
  PRINT 99, TRZ, ERROR
99..FORMAT(5H TRZ ,E12.4,5H ERR ,F10.1)

  "--------RETURN TO RESTART ITERATION ----------------"
  GO TO I1
D1..CONTINUE


  "========FIVE DIFFERENT DRIVING FUNCTIONS============"

   "INPT = 0  :  UNDRIVEN FLIGHT (DEFAULT)"
   "INPT = 1  :  STEP RESPONSE ANALYSIS"
   "INPT = 2  :  SHAKEN FLIGHT, RANDOM STEP PERTURBATIONS"
   "INPT = 3  :  NORMAL FLIGHT, HARMONIC PERTURBATIONS"
   "INPT = 4  :  NORMAL FLIGHT, RANDOM STEP PERTURBATIONS"


 IF(INPT .EQ. 0.0) GO TO C0
 IF(INPT .EQ. 1.0) GO TO C1
 IF(INPT .EQ. 2.0) GO TO C2
 IF(INPT .EQ. 3.0) GO TO C3
 IF(INPT .EQ. 4.0) GO TO C4


C0..CONTINUE
```

```
 "-------UNDRIVEN FLIGHT : UNDISTURBED CRUISE FLIGHT---"
 DETRIM = DEZ
 TRTRIM = TRZ
 IF(T .GE. TBREAK) GO TO B1
 GO TO E1

C1..CONTINUE
 "-------STEP RESPONSE ANALYSIS WITH THREE STEPS-------"
 DETRIM = DEZ + DDE1*STEP(TDE1) + ...
          DDE2*STEP(TDE2) + DDE3*STEP(TDE3)
 TRTRIM = TRZ + DTR1*STEP(TTR1) + ...
          DTR2*STEP(TTR2) + DTR3*STEP(TTR3)
 IF(T .GE. TBREAK) GO TO B1
 GO TO E1

C2..CONTINUE
 "-------SHAKEN FLIGHT---------------------------------"
 "-------PERTURBATIONS ARE CHANGED WHEN T = TPULSE----"
 IF(T .GE. TPULSE) GO TO P1
 GO TO E1
P1..CONTINUE
 "-------NEXT PERTURBATION IS SCHEDULED AND-----------"
 "-------THE MAGNITUDE OF THE PRESENT-----------------"
 "-------PERTURBATION IS CHOSED RANDOMLY--------------"
 TPULSE = TPULSE + SINT*UNIF(DELTA1, DELTA2)
 PDE = INT(UNIF(1,3.9999))
 PTR = INT(UNIF(1,3.9999))
 DETRIM = DEZ + (PDE - 2)*DDE
 TRTRIM = TRZ + (PTR - 2)*DTR
 IF(T .GE. TBREAK) GO TO B1
 GO TO E1

C3..CONTINUE
 "-------NORMAL FLIGHT: HARMONIC PERTURBATIONS--------"
 PDE = SIN(WDE*T)
 DETRIM = DEZ + PDE*DDE
 PTR = SIN(WTR*T)
 TRTRIM = TRZ + PTR*DTR
 IF(T .GE. TBREAK) GO TO B1
 GO TO E1

C4..CONTINUE
 "-------NORMAL FLIGHT: STEP PERTURBATIONS------------"
 "-------PERTURBATIONS ARE CHANGED WHEN T = TPULSE----"
 IF(T .GE. TPULSE) GO TO P2
 GO TO E1
P2..CONTINUE
 "-------NEXT PERTURBATION IS SCHEDULED AND-----------"
 "-------THE MAGNITUDE OF THE PRESENT-----------------"
 "-------PERTURBATION IS CHOSED RANDOMLY--------------"
 TPULSE = TPULSE + DELTA*UNIF(DELTA1, DELTA2)
 PDE = INT(UNIF(1,3.9999))
 PTR = INT(UNIF(1,3.9999))
 DETRIM = DEZ + (PDE - 2)*DDE
```

```
        TRTRIM = TRZ + (PTR - 2)*DTR
        IF(T .GE. TBREAK) GO TO B1
        GO TO E1

        "-------START USING BROKEN MODEL PARAMETERS----------"
       B1..CONTINUE
        CLZ  = CLZNEW
        CLAL = CLALNW
        CLDE = CLDENW
        CLAD = CLADNW
        CLQ  = CLQNEW
        CMZ  = CMZNEW
        CMAL = CMALNW
        CMDE = CMDENW
        CMAD = CMADNW
        CMQ  = CMQNEW
        CDZ  = CDZNEW
        CDAL = CDALNW
        CDDE = CDDENW
        ALZ  = ALZNEW
        THZ  = THZNEW
        GAZ  = GAZNEW
        DEZ  = DEZNEW
        TRZ  = TRZNEW
        VZ   = VZNEW
        IY   = IYNEW
        S    = SNEW
        WT   = WTNEW
        CBAR = CBARNW
        KTH  = KTHNEW
        KU   = KUNEW
        DDE  = DDENEW
        DTR  = DTRNEW
        SINT = SINTNW
        GO TO E1

       E1..CONTINUE

        "-------STOPPING CRITERION: MAX SIMULATION TIME------"
        TERMT(T .GE. TMX)

  END  $"OF DYNAMIC"
  END  $"OF PROGRAM"
```

REFERENCES

Ali, M. and Scharnhorst, D. A. (1985). "Sensor-Based Fault Diagnosis in a Flight Expert System", IEEE 1985 Conference on Artificial Intelligence Applications, Miami, FL, USA, pp 49–54.

Ali, M. et al. (1986). "A Flight Expert System (FLES) for On-Board Fault Monitoring and Diagnosis", Proc. SPIE Int. Soc. Opt. Eng. (USA), **635**, pp 58–61 (1986). (Applications of Artificial Intelligence III, Orlando, FL, USA, 1–3 April 1986.

Cellier, F. E. and Yandell, D. W. (1987). "SAPS–II: A New Implementation of the Systems Approach Problem Solver", *International J. of General Systems*, **13**(4), pp 307–22.

Cellier, F.E. (1987). "Qualitative Simulation of Technical Systems Using the General System Problem Solving Framework", *International J. of General Systems*, **13**(4), pp 333–44.

Chu, Y. et al (1980). "Modeling Operator Information Handling Tasks in Supervisory Control of Multiple-Process Systems", Proceedings of the International Conference on Cybernetics and Society, Boston, MA, USA, 8–10 Oct. 1980 (New York, USA: IEEE 1980), pp 391–5.

Cross, S. E. (1984). "Towards an Expert System Architecture for Flight Domain Applications", Proceedings of the IEEE 1984 National Aerospace and Electronics Conference. NAECON 1984. (IEEE Cat. No. 84CH2029–7), Dayton, OH, USA, 21–25 May 1984 (New York, USA: IEEE 1984) **2** pp 784–8.

Etkin, B. (1972). *Dynamics of Atmopheric Flight*, John Wiley & Sons, Inc.

Etkin, B. (1982). *Dynamics of Flight*, John Wiley & Sons, Inc.

Etkin, B. and Zhu, S. (1982). *Control Logic for Landing-Abort Autopilot Mode – UTIAS Report No. 258*, Institute for Aerospace Studies, University of Toronto, Canada.

Hacker, T. (1970). *Flight Stability and Control*, American Elsevier Publishing Company, Inc.

Heffley, R. K. et al. (1972). *Aircraft Handling Qualities Data – Report NASA–CR–2144*, Systems Technology, Inc.

Hess, R. A. (1984). "Unifying Approach to Human Pilot Modelling", Proceedings of the Ninth Triennial World Congress of IFAC, Budapest, Hungary, 2–6 July 1984, **5** pp 2609–13.

Irving, F. G. (1966). *An Introduction to the Longitudinal Static Flight of Low-Speed Aircraft*, Pergamon Press, London, England.

Kleinman, D. L. et al. (1983). "Modeling Human Decisionmaking in Emergency State Power Distribution", IEEE 1983 Mediterranean Electrotechnical Conference **2** D1.10 .

Klir, G. J. (1985). *Architecture of Systems Problem Solving*, Plenum Press, New York.

Law, A. M. and Kelton, W. D. (1982). *Simulation Modeling and Analysis*, McGraw Hill, New York.

Levine, R. D. and Tribus, M. (1978). *The Maximum Entropy Formalism*, The MIT Press, Cambridge, MA.

Mitchell, E. E. L. and Gauthier, J. S. (1986). *ACSL: Advanced Continuous Simulation Language – User/Guide Reference Manual*, Mitchell & Gauthier, Assoc., Concord, MA.

Mooij, H. A. (1985). *Criteria for Low-Speed Longitudinal Handling Qualities of Transport Aircraft with Closed-Loop Flight Control Systems*, Martins Nijhoff Publishers for The National Aerospace Laboratory NLR, Dordrecht, The Netherlands.

Morgan, P. D. (1983). "Command Decision Makers and their Modes of Interaction", NASA Report AD P002882, USA.

Morgan, P. D. (1985). "Modelling the Decision Maker Within a Command System", International Conference on Advances in Command, Control and Communication Systems: Theory and Applications (Conf. Publ. No. 247), Bournemouth, England, 16–18 April 1985 (London, England: IEE 1985) pp 65–9.

Riley, V. (1985). "Monitoring the Monitor: Some Possible Effects of Embedding Human Models in Highly Automated Manned Systems", *IEEE 1985 Proceedings of the International Conference on Cybernetics and Society (Cat. No. 85CH2253-3) Tucson, AZ, USA, 12-15 Nov. 1985* (New York, USA: IEEE 1985) pp 6–9.

Systems Control Technology (1986). *CTRL-C, A Language for the Computer-Aided Design of Multivariable Control Systems, User's Guide*, Systems Control Technology, Palo Alto, CA.

Trankle, T. L. and Markosian, L. Z. (1984). "Expert System for Control System Design", International Conference – Control 85 (Conf. Pub. No. 252), Cambridge, England, 9–11 July 1985 (London, England: IEE 1985), **2** pp 495–9.

Trankle, T. L. et al. (1986). "Expert System Architecture for Control System Design", Proceedings of the 1986 American Control Conference (Cat. No. 86CH2336–6), Seattle, WA, USA, 18–20 June 1986 (New York, USA: IEEE 1986) **2** pp 1163–9.

Vesanterä, P. J. and Cellier, F. E. (1989). "Building Intelligence into an Autopilot – Using Qualitative Simulation to Support Global Decision Making", *Simulation*, Jan 1989.