A NEW APPROACH TO THE FLIGHT CONTROL SYSTEM RECONFIGURATION FOLLOWING A BATTLE DAMAGE AND/OR A GENERIC FAILURE ON A CONTROL SURFACE

By

MARCELLO NAPOLITANO

Master of Science in Aeronautical Engineering
University of Naples
Naples, ITALY
1985

Submitted to the Faculty of the
Graduate College of the
Oklahoma State University
in partial fulfillment of
the requirements for
the Degree of
DOCTOR OF PHILOSOPHY
December, 1989

A NEW APPROACH TO THE FLIGHT CONTROL
SYSTEM RECONFIGURATION FOLLOWING
A BATTLE DAMAGE AND/OR A
GENERIC FAILURE ON A
CONTROL SURFACE

Thesis approved:

Robert L. Swain
Thesis Adviser

Steven G. Kraig

Mark T. Hagan

Gary E. Young

Norman N. Durham
Dean of the Graduate College

PREFACE

The aircraft flight control system reconfiguration problem following a damage or a generic failure on a main control surface has been considered in this study.

First, the estimated model of the damaged aircraft is obtained by using a Multiple Model Kalman Filtering approach. Then, a particular algorithm is applied to the flight control reconfiguration. The determination of the desired control law, which can adapt in a very short period of time to a major damage to a main control surface, is obtained by making use of the recent control and response time histories. In addition, a method is proposed to efficiently distribute the reconfiguration task among all the remaining healthy control surfaces. Furthermore, a particular approach is proposed in order to calculate a new set of feedback gains of the flight control system such that dynamic decoupling and desirable handling qualities are retained even after the damage.

The model estimation, the control algorithm and the feedback gains updating process have been codified in computer simulation programs for a 6 degrees of freedom aircraft model. The simulation results of the reconfiguration are presented.

At this point, I wish to thank the persons who assisted me in this study and during my coursework at Oklahoma State University. Particularly, I wish to express my deep gratitude to my adviser, Dr.

Robert L. Swaim, for his intelligent, expert and constant guidance and for his invaluable aid.

I am also grateful to the other committee members, Dr. Gary Young, Dr. Stephen Liang and Dr. Martin Hagan, for the invaluable assistance and expertise that they provided during my coursework. Special thanks are due to my family and to my wife for their moral support and constant encouragement.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

NOMENCLATURE

α = angle of attack	(deg)
β = sideslip angle	(deg)
u = forward perturbed speed (along X)	(ft sec^{-1})
p = roll rate	(deg sec^{-1})
q = pitch rate	(deg sec^{-1})
r = yaw rate	(deg sec^{-1})
θ = pitch attitude angle	(deg)
ϕ = bank roll angle	(deg)
W = airplane weight	(lbs)
ρ = air density	(slug ft^{-3})
VEL = airplane true speed	(ft sec^{-1})
MACH = Mach number	
h = altitude	(ft)
I_{xx}, I_{yy}, I_{zz} = moments of inertia around X,Y,Z axes respectively	(slug ft^2)
I_{xz} = products of inertia in the XZ plane	(slug ft^2)
c = wing mean geometric aerodynamic chord	(ft)
b = wing surface reference span	(ft)
S = wing surface reference area	(ft^2)
$X_{c.g.}$ = distance from the leading edge of c to the airplane center of gravity in tenths of c	
δ_{EL} = left elevator deflection angle	(deg)
δ_{ER} = right elevator deflection angle	(deg)
δ_{AL} = left aileron deflection angle	(deg)

δ_{AR} = right aileron deflection angle (deg)

δ_{SL} = left spoiler deflection angle (deg)

δ_{SR} = right spoiler deflection angle (deg)

δ_{CL} = left canard deflection angle (deg)

δ_{CR} = right canard deflection angle (deg)

δ_{R} = rudder deflection angle (deg)

C_T = thrust coefficient

C_D = drag coefficient

C_L = lift coefficient

C_Y = lateral force coefficient

C_l = rolling moment coefficient

C_m = pitching moment coefficient

C_n = yawing moment coefficient

C_{i_1} = steady state value of the coefficient of the i-th aerodynamic force or moment

with i = T,D,L,Y,l,m,n

C_{i_j} = general form of the aerodynamic stability and control derivatives : variation of the coefficient of the i-th aerodynamic force or moment with changes of the j-th variable

with i = T,D,L,Y,l,m,n

$j = \alpha,\dot{\alpha},q,u,\beta,p,r$, (Stability derivatives)
$\delta_E,\delta_A,\delta_S,\delta_C,\delta_R$ (Control derivatives)

X(k) = state vector at the time instant 'k'

U(k) = input vector at the time instant 'k'

W(k) = disturbance vector at the time instant 'k'

Z(k) = measurement vector at the time instant 'k'

V(k) = measurement error vector at the time instant 'k'

Y(k) = response vector at time instant 'k'

A = discrete-time state matrix

B = discrete-time control matrix

L = disturbance propagation matrix

C = observation matrix

Q = covariance matrix associated with the disturbance vector

R = covariance matrix associated with the measurement noise vector

X(t) = state vector at the time 't'

U(t) = input vector at the time 't'

A_C = continuous-time state matrix

B_C = continuous-time control matrix

n = order of the state vector

m = order of the input vector

l = order of the observation vector

N = number of discretized values of the normal force coefficient of the damaged control surface

CHAPTER I

INTRODUCTION

Reconfiguration Problem

For a military aircraft one of the biggest challenges is the design of a flight control system that allows the aircraft to recover from battle damages and/or generic failures.

Often the accident investigations report that there was a way in which the disaster could have been avoided if the proper actions had been taken in a timely fashion. However, the length of time when valid effective actions to save the aircraft could be taken is just a few seconds. Given the understandable panic during those moments, a pilot, even well trained, may not find the solution in time.

The integration and application of recent advances in failure detection, failure identification and control systems technologies may make it possible to detect and identify potentially catastrophic failures in the flight control system and the restructure the control system of the aircraft in real time in order, depending on the size of the problem, to continue the mission (or the flight) or to execute a safe landing.

The objective is to be able to control the aircraft immediately after battle damages and/or generic failures on a control surface. Note that this classification does not include unsolvable problems (e.g.

wings falling off) where the aircraft cannot be saved. Therefore, the flight control task is to utilize whatever control resources remain in order to regain control of the aircraft, to prevent further damage by excessive air loads, and to give the crew time to assess the options (Ref.[1]).

In the period after a damage on a control surface the following will be experienced:

1 - Altered trim conditions.

2 - Changes in the aerodynamic forces and moments.

3 - Changed control effectiveness.

4 - Altered aircraft dynamics.

5 - Changes, even losses, of control components, including sensors, communication devices, computers and actuators.

Reconfiguration Conditions and Requirements

In order to implement a reconfiguration strategy, we may introduce a variety of control surfaces (speed brakes, wing flaps, differential (even dihedral) canards, spoilers, rudder below fuselage) and thrust control mechanisms (differential thrust, thrust vectoring, canted engines), as shown in Ref.[12]. It is clear that, as the number of control surfaces and thrust control mechanisms available on the aircraft increases, lower degrees of performance loss occur after a damage.

The selection of the control surfaces and thrust control mechanisms to be used for the reconfiguration is a function of several factors: control effectiveness, increased aircraft complexity and costs, weight penalties, increased aerodynamic drag due to the increased wetted area, applicability depending on aircraft type. Fig. 1 shows a F-16 aircraft

with 9 independent control surfaces.

The following quantities, along with a fully operational flight computer, are assumed to be available for reconfiguration purposes:

1 - Actuator position for each actuator.

2 - Aircraft body angular and linear velocities in the three body axes.

3 - Aircraft attitude and angle of attack.

Essentially we can classify the failures and or battle damages of a control surface in two categories : that is locked and missing surface. Generally we can say that a locked surface corresponds to a failure in the control surface's actuator. A battle damage, instead, mainly implies missing surface or, more realistically, both missing and locked surface. Of course, in order to describe mathematically the model of the damaged aircraft, the behaviors of the aircraft due to a locked surface or a missing surface will be different.

In terms of requirements for aircraft survival and performance after a damage and/or failure on the control surfaces, two types of requirements were imposed by the military aircraft specifications (Ref.[19]).

First was the requirement to perform the mission and return home safely after the loss of one or more flight control system elements due to ballistic weapons. For such requirement the following groundrule was stated: after a single component failure or hit by the 37 mm HEJ projectile, the aircraft should (at minimum) be able to abort the mission, return to friendly territories, and perform a landing, without a significant increase susceptibility (which means retaining Terrain Following/ Terrain Avoidance capabilities). This groundrule is

consistent with the U.S. Air Force program goal of reducing the probability of kill per hit from 37 mm projectiles by one order of magnitude and it is reasonable when considering the expected high cost and relatively low procurement of an advanced future fighter (compared to the F-4 or F-16 procurements of about 4000 and 2000 aircrafts respectively).

Second was the requirement for the equipment to be designed from a spectrum of sufficiently reliable elements so that the probability of aircraft loss per mission due to a random damage and/or generic failure on a flight control system component is smaller than $1 * 10^{-7}$.

Reconfiguration Tasks

There are several approaches to the reconfiguration problem which have been introduced, implemented in software and more or less successfully tested. A list can be given as:

1 - Quantitative Feedback Theory (Ref.[3]-[6]).

2 - Direct Digital Output Feedback with a Linear Quadratic design procedure (Ref.[7]-[9]).

3 - Pseudo Inverse technique with the application of a Control Mixer (Ref.[10],[11]).

According to the way that we addressed the reconfiguration problem, the task of battle damage and/or generic failure accomodation can be broken into the following main tasks:

1 - EXECUTIVE CONTROL task, which provides essentially synchronization of the remaining tasks.

2 - FAILURE DETECTION task, which controls the aircraft behavior and detects significant abnormalities.

3 - FAILURE IDENTIFICATION task, which searches the cause or a set of probable causes.

4 - FAILED MODEL ESTIMATION task, which generates a mathematical model of the aircraft dynamics considered to reflect changes due to the damage.

5 - RECONFIGURATION LAW DESIGN task, which determines what actions should be taken in order to recover the damaged aircraft.

6 - FEEDBACK GAINS UPDATING task, which calculates a new set of feedback gains, in order to retain stability, dynamic decoupling and desirable handling qualities even after the damage.

Frequently, in the literature the FAILURE DETECTION task and the FAILURE IDENTIFICATION task are found combined in a unique task. Fig.2 shows a 'step-by-step' overview of the reconfiguration problem.

In the present work only the last three tasks of the overall reconfiguration problem are considered; therefore it is assumed that the occurrence of a damage on a control surface has been detected and that the damaged control surface has been identified.

Eventually, another task to be introduced is a PILOT ADVISORY function. While the computers of the flight control system are reconfiguring the aircraft, it would be desirable for the pilot to be able to see on a cockpit display which control surface has been damaged, what the flight control system is trying to achieve and what actions, if possible under current conditions, could eventually benefit the overall reconfiguration, for example reduction of speed or reduction of altitude.

CHAPTER II

AERODYNAMIC EFFECTS OF A DAMAGE AND/OR GENERIC FAILURE

Aerodynamic Considerations

As stated in the previous chapter, our objective is to identify and to control a system with changed dynamics. Regardless of the approach used to control such a system, a more efficient way to estimate the changed dynamics can be implemented if we have some knowledge on how the dynamics may actually change following a damage on a control surface.

The aerodynamic characteristics of a surface are expressed in terms of normal force, axial force and moment around some fixed points or axes. A damage on a control surface, which involves a missing part of it, implies changes in the aerodynamic characteristics of such surface. In order to evaluate these changes the following aerodynamic consideration can be made: the main control surfaces (typically ailerons, elevator and rudder) are not located in directly wetted area.

This implies that the aerodynamic drag exerted by the surface's deflection, which is related to the axial force, is negligible (Ref.[13]). On the other hand the aerodynamic moments of a control surface around various axes are just proportional, through the geometric parameters of the aircraft, to the normal force exerted on the

surface. Therefore, with the axial force negligible and with the aerodynamic moments a function of the normal force, we can say that the changed aircraft dynamics following a damage on a control surface is due essentially to an instantaneous change of the normal force coefficient of the damaged surface. Such coefficient is then used for the calculation of the corresponding non-dimensional stability and control derivatives.

Therefore, it would be very useful if we can implement a set of closed-form expressions of the non-dimensional aerodynamic stability and control derivatives as functions of the normal force coefficient of the control surface (C_{N_δ}) considered to be damaged; for example :

$$C_{m_\alpha} = C_{m_\alpha}(C_{N_\delta}) \ , \ C_{L_\alpha} = C_{L_\alpha}(C_{N_\delta}) \ , \ \ldots \tag{2.1}$$

$$C_{l_\beta} = C_{l_\beta}(C_{N_\delta}) \ , \ C_{n_\beta} = C_{n_\beta}(C_{N_\delta}) \ , \ \ldots \tag{2.2}$$

While there exist efficient analytical closed-form expressions for aerodynamic stability derivatives as functions of the normal coefficients of the control surfaces for conventional subsonic airplanes (Ref. [17]-[18]), an accurate wind-tunnel investigation and relative data correlation would be strongly needed for unconventional supersonic aircraft like the modern fighters.

This implies that, following this approach, the design of an aircraft reconfiguration system is not merely a control theory problem; it involves also a certain amount of aerodynamic investigations in order to develop these closed-loop forms to be used for the aircraft mathematical modeling.

Aircraft Model and Aerodynamic Data

In order to test the control techniques to be used for the reconfiguration in a computer simulation program, an aircraft model with relative aerodynamic characteristics, geometric and inertial data needed to be introduced.

The main candidates for the implementation of a reconfiguration system are, of course, military airplanes, especially fighters. But, since data from a military aircraft were not available to us, a civil aircraft has been considered, without any loss of generality. Such model of a business jet, shown in Fig. 3 (Ref.[13]) is assumed to have 9 independent control surfaces: left and right elevators, left and right ailerons, left and right spoilers, left and right canards and rudder. However, even if the aircraft aerodynamic, geometric and inertial data do not correspond to those of a fighter, the considered flight conditions are typical of an air combat scenario, that is high altitude and high subsonic Mach number. Also, the flight maneuvers, which will be later introduced and analyzed, are typical of a combat situation, with step inputs on elevators and ailerons.

The aerodynamic characteristics, the geometric and inertial data, and the flight conditions of the introduced aircraft model are reported in Table I.

Note that, at this point of the work, in order to introduce a reconfiguration approach, the chosen aircraft exhibits satisfactory handling qualities in terms of short period, phugoid, rolling, spiral and dutch-roll damping and natural frequencies. Therefore, there is no need for introducing a stability augmentation system (S.A.S.), whose feedback gains K essentially give rise to a closed-loop expression of

the state matrix $A_{C(C.L.)} = (A_C - B_C K)$. In a general case, which will be later considered, the closed loop characteristics of the aircraft with a S.A.S. will be discussed.

Also we assume that the considered aircraft model does not implement any Control Configured Vehicle (C.C.V.) function, such as gust or maneuver load alleviation systems, flight envelope limiting systems, etc.

As we have previously stated, it would be useful to have a set of closed-form expressions of the non-dimensional stability derivatives as functions of the normal force coefficient of the damaged control surface. Unfortunately, among the available data, we do not have the value of C_{N_δ}. However, as expected, we have the value of the control derivatives for each control surface. Therefore, in order to illustrate the introduced approach, instead of using C_{N_δ}, we are going to implement closed-form expressions of the stability derivatives as function of the control derivatives.

Since the introduced aircraft is a rather conventional subsonic airplane, Ref. [13],[17,] have provided sufficiently exact methods in order to obtain closed-form expressions for the non-dimensional stability derivatives.

Considering a damage on one of the elevator surfaces, the following stability derivatives would be affected by changes in the values of $C_{L_{\delta E}}$ and $C_{m_{\delta E}}$ (which is proportional to $C_{L_{\delta E}}$) due to the damage:

$$C_{L_\alpha},\ C_{m_\alpha},\ C_{L_{\dot{\alpha}}},\ C_{m_{\dot{\alpha}}},\ C_{L_q},\ C_{m_q}\ ;$$

Furthermore there will be an induced rolling moment.

In order to introduce a relation between the stability derivative C_{L_α} and the control derivative $C_{L_{\delta E}}$, recall the following expression (Ref. [13],[17]):

$$C_{L_\alpha} = C_{L_{\alpha WB}} + C_{L_{\alpha H}} \, q_H/q \, S_H/S \, (1 - d\varepsilon/d\alpha) \tag{2.3}$$

$$C_{L_{\delta E}} = C_{L_{\alpha H}} \, q_H/q \, S_H/S \, \tau_E \tag{2.4}$$

where $C_{L_{\alpha WB}}$ is the contribution to C_{L_α} from the (wing and body) of the aircraft. In other words it is the C_{L_α} of the aircraft if the horizontal tail is not considered; $C_{L_{\alpha H}}$ is the contribution to C_{L_α} from the horizontal tail.

q_H/q is the ratio between the dynamic pressure at the horizontal tail location and the nominal value of the dynamic pressure. Such ratio is less than 1 because of the loss of flow energy at the horizontal tail in the form of friction and separation drag of the wing surface; a typical value of q_H/q is around 0.9.

S_H/S is the ratio between the horizontal tail surface area and the wing surface area.

$d\varepsilon/d\alpha$ is the downwash effect induced on the horizontal tail by the wing-trailing-vortex system. Such effect is directly proportional to the wing sweep angle, to the wing taper ratio and inversely proportional to the wing aspect ratio.

τ_E is the angle of attack effectiveness of the elevator.

By solving for $C_{L_{\alpha H}}$ as function of $C_{L_{\delta E}}$ from Eq. (2.4):

$$C_{L_{\alpha H}} = \frac{C_{L_{\delta E}}}{q_H/q \, S_H/S \, \tau_E} \tag{2.5}$$

If we substitute Eq. (2.5) in Eq. (2.3) we have:

$$C_{L_\alpha} = C_{L_{\alpha WB}} + C_{L_{\delta E}} (1 - d\varepsilon/d\alpha)/\tau_E \tag{2.6}$$

From Ref. [13],[17] we have found that we can assume:

$$(1 - d\varepsilon/d\alpha) = \tau_E = 0.5 \tag{2.7}$$

Therefore, Eq. (2.6) will become:

$$C_{L_\alpha} = C_{L_{\alpha WB}} + C_{L_{\delta E}} \tag{2.8}$$

Hence, we have obtained a closed-form relation of the stability derivative C_{L_α} as a function of the control derivative $C_{L_{\delta E}}$.

Similarly, we may introduce a relationship between the stability derivative C_{m_α} and the control derivative $C_{m_{\delta E}}$, which is proportional to $C_{L_{\delta E}}$; in order to do so, recall the following expressions (Ref. [13],[17]:

$$C_{m_\alpha} = C_{L_{\alpha WB}} (X_{CG} - X_{AC_{WB}})/c +$$

$$- C_{L_{\alpha H}} q_H/q \; S_H/S \; (X_{AC_H} - X_{CG})/c \; (1-d\varepsilon/d\alpha) \tag{2.9}$$

$$C_{m_{\delta E}} = -C_{L_{\alpha H}} q_H/q \; S_H/S \; (X_{AC_H} - X_{CG})/c \; \tau_E \tag{2.10}$$

where $X_{AC_{WB}}$ and X_{AC_H} are the locations along the X axis of the aerodynamic center of the (wing + body) part of the aircraft and of the horizontal tail of the aircraft respectively. Generally, the aerodynamic center of an airfoil is defined as that point about which the pitching moment coefficient remains invariant with the angle of attack.

By solving for $C_{L_{\alpha H}}$ as function of $C_{m_{\delta E}}$ from Eq. (2.10):

$$C_{L_{\alpha H}} = \frac{-C_{m_{\delta E}}}{q_H/q \; S_H/S \; (X_{AC_H} - X_{CG})/c \; \tau_E} \tag{2.11}$$

If we substitute Eq. (2.11) into Eq. (2.9) we have:

$$C_{m_\alpha} = C_{L_{\alpha WB}} (X_{CG} - X_{AC_{WB}})/c + C_{m_{\delta E}} (1 - d\varepsilon/d\alpha)/\tau_E \tag{2.12}$$

By using the approximation introduced in Eq. (2.7) we have:

$$C_{m_\alpha} = C_{L_{\alpha WB}} (X_{CG} - X_{AC_{WB}})/c + C_{m_{\delta E}} \tag{2.13}$$

Hence, we have obtained a closed-form relation of the stability derivative C_{m_α} as function of the control derivative $C_{m_{\delta E}}$.

About the stability derivatives $C_{L_{\dot{\alpha}}}$ and $C_{m_{\dot{\alpha}}}$ from the 'lag of downwash' theory (Ref. [13]) we recall that there is a fixed relation between $C_{L_{\dot{\alpha}}}$, $C_{m_{\dot{\alpha}}}$ and $C_{L_{\delta E}}$. This can be seen in the following expressions from Ref. [13],[17]:

$$C_{L_{\dot{\alpha}}} = 2 \, C_{L_{\alpha H}} \, q_H/q \; S_H S \; X_H/c \; d\varepsilon/d\alpha \tag{2.14}$$

$$C_{m_{\dot{\alpha}}} = -2 \, C_{L_{\alpha H}} \, q_H/q \; S_H/S \; X_H^2/c^2 \; d\varepsilon/d\alpha \tag{2.15}$$

where $X_H = X_{AC_H} - X_{CG}$ (2.16)

If we substitute Eq. (2.5) into Eqs. (2.14) and (2.15), using the approximation in Eq. (2.7) and the numerical values introduced in Table I for the stability and control derivatives, we have:

$$C_{L_{\dot{\alpha}}} = C_1 * C_{L_{\delta E}} \tag{2.17}$$

$$C_{m_{\dot{\alpha}}} = C_2 * C_{L_{\delta E}} \tag{2.18}$$

where $C_1 = 3.957$, $C_2 = -12.051$.

Hence, we have obtained closed-form expressions of the stability derivatives $C_{L_{\dot{\alpha}}}$ and $C_{m_{\dot{\alpha}}}$ as function of the control derivative $C_{L_{\delta E}}$.

Finally, we need a similar expression for C_{L_q} and C_{m_q}. In order to do so, recall that for most airplanes the center of gravity is located somewhere on the wing mean aerodynamic chord. This implies that the contribution of the wing to C_{L_q} and C_{m_q} is small with respect to the contributions of the horizontal tail and of the canards. Therefore, from Ref. [13],[17], recall the following expressions:

$$C_{L_q} = 2\, C_{L_{\alpha H}}\, q_H/q\; S_H/S\; (X_{AC_H} - X_{CG})/c +$$

$$+ 2\, C_{L_{\alpha c}}\, q_c/q\; S_c/S\; (X_{CG} - X_{AC_c})/c \qquad (2.19)$$

$$C_{m_q} = -2.2\, C_{L_{\alpha H}}\, q_H/q\; S_H/S\; (X_{AC_H} - X_{CG})^2/c^2 +$$

$$- 2\, C_{L_{\alpha c}}\, q_c/q\; S_c/S\; (X_{CG} - X_{AC_c})^2/c^2 \qquad (2.20)$$

where the parameters $C_{L_{\alpha c}}$, q_c/q, S_c/S, X_{AC_c}, τ_c are related to the canards; they have the same meanings of the corresponding parameters introduced for the horizontal tail.

By using the geometric data and the values of the control derivatives relative to the canards introduced in Table I, using Eq. (2.5) for the canards, assuming $q_c/q = 1$, $\tau_c = 0.9$ and $(X_{CG} - X_{AC_c})/c = 2.368$ we have that:

$$2\, C_{L_{\alpha c}}\, q_c/q\; S_c/S\; (X_{CG} - X_{AC_c})/c = C_4 = 2.105 \qquad (2.21)$$

$$2\, C_{L_{\alpha c}}\, q_c/q\; S_c/S\; (X_{CG} - X_{AC_c})^2/c^2 = -C_6 = 4.984 \qquad (2.22)$$

Also, by substituting Eq. (2.11) into Eqs. (2.19), (2.20), we have:

$$C_{L_q} = (C_3 * C_{m_{\delta E}}) + C_4 \tag{2.23}$$

$$C_{m_q} = (C_5 * C_{m_{\delta E}}) + C_6 \tag{2.24}$$

By using the numerical values of Table I for C_{L_q} and C_{m_q} and the previously calculated values for C_4 and C_6 we have:

$$C_3 = -2.635 \tag{2.25}$$

$$C_5 = 9.977 \tag{2.26}$$

Therefore, we have obtained closed-form expressions of the stability derivatives C_{L_q} and C_{m_q} as function of the control derivative $C_{m_{\delta E}}$.

As final result, the following expressions for the stability derivatives corresponding to a damaged aircraft condition can be obtained:

$$dC_{L_\alpha} = C_{L_\alpha} - (C_{L_{\delta E}} - dC_{L_{\delta E}}) \tag{2.27}$$

$$dC_{m_\alpha} = C_{m_\alpha} - (C_{m_{\delta E}} - dC_{m_{\delta E}}) \tag{2.28}$$

$$dC_{L_{\dot{\alpha}}} = C_1 * dC_{L_{\delta E}} \tag{2.29}$$

$$dC_{m_{\dot{\alpha}}} = C_2 * dC_{L_{\delta E}} \tag{2.30}$$

$$dC_{L_q} = (C_3 * dC_{m_{\delta E}}) + C_4 \tag{2.31}$$

$$dC_{m_q} = (C_5 * dC_{m_{\delta E}}) + C_6 \tag{2.32}$$

with $C_1 = 3.957$, $C_2 = -12.051$, $C_3 = -2.635$,

$C_4 = 2.105$, $C_5 = 9.977$, $C_6 = -4.984$;

where the prefix 'd' indicates the value of the stability and control derivatives after the damage; as it can be seen, they are all functions of $dC_{L_{\delta E}}$ and $dC_{m_{\delta E}}$, which is proportional to $dC_{L_{\delta E}}$.

Of course, in the particular case when the damage doesn't imply a missing part of the control surface but only locked actuator's surface,

the values of the stability and control derivatives under nominal and damaged conditions will be coincident.

Such analytical closed forms of the non-dimensional aerodynamic characteristics are to be stored in the flight computer, ready to be used for on-line reconfiguration purposes.

State Variable Model of the Aircraft

The non-dimensional aerodynamic stability and control derivatives previously introduced are then combined with the flight conditions data, and with the geometric and inertial data of the aircraft in order to calculate the dimensional stability and control derivatives. The details of these calculations are shown in Ref.[13]. Generally we can say that these dimensional stability and control derivatives are proportional to the wing surface, to the dynamic pressure (which is given by $1/2*\rho*Vel^2$), to the wing aerodynamic chord and to the wing span. They are inversely proportional to the aircraft mass, to the velocity and to the inertial moments of the aircraft around the stability axes. These are the axes with respect to which the steady state values of the forward and vertical linear velocities are different than zero while the the steady state values of the lateral velocity and of the angular velocities are zero.

Following the Newtonian equations of the motion, such dimensional parameters are then linearly combined for calculating the elements of the A_C and B_C matrices of the continuous-time state variable model of the aircraft. The chosen state variables are : $\{\alpha, q, u, \theta, \beta, p, r, \phi\}$. The result is a set of 8 equations describing the dynamics of the aircraft linearized with respect to some equilibrium points; such points

are known as "trim conditions", and for each particular aircraft they are functions of the flight conditions.

Table II shows the state variable model of the aircraft with the numerical values of the matrices A_c and B_c relative to an undamaged nominal situation at the flight conditions reported in Table I.

CHAPTER III

DAMAGED MODEL ESTIMATION

Discrete-time State Variable Model

In this chapter a particular application of the Kalman Filter is introduced for the purpose of estimating the mathematical model of the aircraft considered to reflect the changed dynamic characteristics following the damage.

The linearized aircraft dynamics can be described in the discrete form by the following equations :

$$X(k+1) = A\,X(k) + B\,U(k) + L\,W(k) \qquad (3.1)$$

$$Z(k) = C\,X(k) + V(k) \qquad (3.2)$$

where X(k) is a n-th dimension state vector,

U(k) is a m-th dimension control vector,

Z(k) is a l-th dimension observation vector,

V(k) is a l-th dimension measurement noise vector,

W(k) is a r-th dimension disturbance vector,

with $n = 8$, $m = 9$, $l = 6$, $r = 2$;

A and B are the discretized versions of the A_c and B_c matrices introduced in the previous chapter (with T = sampling period = 0.01 sec).

W(k) and V(k) can be considered mutually independent white noise random vectors with zero mean and known covariance matrices, Q and R,

respectively. While V(k) describes sensors' measurement errors with the values of R depending on the sensor's performance (usually given by the company producing the sensors), W(k) allows us to model atmospheric turbulence.

The matrix L reflects the propagation of the turbulence on the overall aircraft dynamics. Given that the atmospheric turbulence can be modeled as additional inputs α_g and β_g (where 'g' stands for 'gust'), L can be considered as an (8x2) matrix with columns corresponding to the α and β columns of the A matrix. The numerical values of the elements of the matrices A , B , L , C , Q , R are shown in Table III.

Multiple Model Kalman Filtering

Once we have the aerodynamic characteristics of the aircraft as functions of the normal force coefficients for each control surface, provided that the Failure Detection and Identification tasks are able to indicate which control surface has been damaged, we can discretize the value of the normal force coefficient of that particular control surface in a number N of values. Note that the particular case of a damage with a locked actuator but without missing part of the control surface has been considered by selecting the last of the N models as the normal undamaged aircraft model, with the nominal value of the normal force coefficient.

Therefore Eq. (3.1) and (3.2) will become:

$$X_i(k+1) = A_i X_i(k) + B_i U(k) + L_i W(k) \qquad (3.3)$$

$$Z_i(k) = C X_i(k) + V(k) \quad \text{with } i=1,\ldots,N \qquad (3.4)$$

The estimation task is to determine which one of the N models correctly characterizes the system (Ref. [15],[16],[21]). Let H_j be the

event that model 'j' is the most exact system characterization; H will be then a random variable with discrete values H_1, H_2,..., H_N; furthermore, let Y(k) be as:

$$Y(k) = \{U(0),U(1),....,U(k-1);Z(1),Z(2),....,Z(k)\}$$

then we define:

$$P_j(k) = P (H = H_j / Y(k))$$

which represents the probability that model 'j' is the correct system characterization, given measurements Y(k). Of course we are looking for the condition of one of the N probabilities associated with the N discretized models converging to 1, which physically means that the mathematical model associated with that particular probability closely describes the dynamics of the aircraft following the damage.

Therefore, using the Y(k) data as input, an iterative algorithm was needed in order to implement a recursive formula for the probabilities $P_i(k)$; the algorithm stops when one of the probabilities converges with a satisfactory accuracy to 1.

In order to solve this problem, a bank of N steady state Kalman Filters has been introduced; extensive use of the $MATRIX_X$ package has been made for calculating the Kalman Filter gains and covariances matrices while the remaining algorithm has been implemented in a Pascal program. The algorithm proceeds as in the following (Ref. [14],[15],[16],[21]):

STEP 1: A set of N steady state Kalman Filters is constructed for the N models with the relative gains and covariance matrices calculated with $MATRIX_X$.

STEP 2: For each of the N models the filter residuals are calculated by using:

$$r_i\ (k+1) = Z(k+1) - C\ \hat{X}_i\ (k+1/k) \tag{3.5}$$

STEP 3: The Bayesian probabilities are updated by using:

$$P_i(k+1) = \frac{\beta_i\ e^{-1/2\{r_i^T(k+1)\ S_i^{-1}\ r_i(k+1)\}} * [P_i(k)]}{\sum_{j=1}^{N} \beta_j\ e^{-1/2\ \{r_j^T(k+1)\ S_j^{-1} r_j(k+1)\}} * [P_j(k)]} \tag{3.6}$$

where S_i is the covariance matrix calculated from the Kalman Filter equations for each model and:

$$\beta_i = (2\pi)^{-\ell/2}\ \text{DET}\ [S_i]^{-1/2}$$

STEP 4: The convergence is checked for all the probabilities; if none of them converges with a sufficient accuracy to 1, the algorithm goes back to STEP 2, otherwise it exits the loop and the model associated with the probability which has converged represents the closest model characterization. Note that, in order to avoid false model estimation due to highly fluctuating probabilities, we may want to exit the algorithm only when a probability has converged to 1 with a sufficient accuracy (let's say (5-10)%), for a certain number of time steps (let's say 30-40).

Such algorithm has to be implemented on-line on the flight computer.

Note that the initial probability $P_i(1)$ can be chosen by using a statistic law to indicate that, following the damage, the normal force coefficient of the damaged surface is more likely to take on some particular range of values. In our case a binomial distribution with

$\rho=q=0.5$ has been introduced to simulate such behavior of the normal force coefficient following the damage.

Also note that in the approach used in this study, since we introduce steady-state Kalman Filters rather than time-varying Kalman Filters, the $P_i(k)$ are not exactly conditional probabilities (Ref.[15]).

Simulation Results

This approach for the estimation of the model of the damaged aircraft has been tested with data from a computer simulation, with randomly generated white noises V(k) and W(k), following a damage on the left elevator, at the flight conditions reported in Table I. The correspondent elevator's inputs are shown in Fig.4. Note that the left elevator has been damaged and it remains fixed at a -5^{o} deflection; therefore the aircraft is going to have a tendency to roll.

The values of the A and B matrices following the damage are shown in Table IV; of course, due to the damage, several elements of these matrices have different values than the ones reported in Table III. Particularly the damage involves a decrease of the value of the normal force coefficient from the nominal value of $C_{L_{\delta EL}} = 0.276$ to $C_{L_{\delta EL}} = 0.121$.

By discretizing the nominal value of the coefficient in a set of N values, the corresponding set of A and B matrices are constructed by the algorithm. Note that in the B matrix only the elements of the column corresponding to the left elevator change.

The parameter N plays a very important role. For high values of N, corresponding to an high modeling accuracy, long convergence times are expected for the probability corresponding to the model that closer

describes the damaged system; the reverse occurs for small values of N. For our purposes, the values of N=12 and N=23 have been considered. Table V shows the two sets of discretized values of the normal force coefficient of the left elevator relative to N=12 and N=23.

The simulation proceeds as the following: it starts with the aircraft flying under normal, nominal, undamaged conditions; then, at time = 1 sec., the damage occurs, causing an instantaneous change in the value of $C_{L_{\delta EL}}$; 4 seconds are assumed to be the difference in time between the instant when the damage occurs and the instant when the model estimation process starts; in these seconds we assume that the Failure Detection and Identification tasks are able to detect the occurrence of the damage and to indicate the damaged control surface; furthermore, during these seconds, we build the N models and the relative Kalman Filters structure. Therefore, the model estimation process starts at time = 5 sec.

When N=12, with the damaged dynamics numerically described by a model somewhere between model #5 and model #6, but closer to model #6, the probability corresponding to model #6 converges to 1, as shown in Fig.5, in a short amount of time, around 1.2 sec., with a time increment of 0.01 sec.

When N=23, with the damaged dynamics numerically described by a model somewhere between model #10 and model #11, but closer to model #11, the probability corresponding to model #11 converges to 1, as shown in Fig.6, in a longer time, around 4 sec., with the same time increment.

Furthermore, given that the calculations for the N steady state Kalman Filters have to be done before the iterative algorithm starts, with N=23 longer initial computational time has to be added to the

already longer convergence time of the algorithm. Therefore, a small increase in modeling accuracy is paid with a much longer pre-estimation computational and algorithm convergence time. On the other side we know that, since the length of time when valid effective actions to save the aircraft could be taken is just a few seconds, even one or two seconds can be a decisive matter.

Once we have shown that for N=23 we have a longer convergence time, let's consider N=12 and let's examine a condition when the value of the normal force coefficient changes, due to the damage, from $C_{L_{\delta EL}} = 0.276$ to $C_{L_{\delta EL}} = 0.114$. The relative mathematical model, in terms of matrices A and B, is still close to model #6, but is more in between model #6 and model #5 than the mathematical model shown in Table IV is, as it can be easily understood by looking at Table V. This implies higher values of the residuals in Eq.(3.5) and, therefore, a larger fluctuation of the probability associated with model #6. The result is shown in Fig. 7; the convergence to the right model still occurs but in a time longer than the one shown in Fig. 5, that is around 1.6 sec. instead of 1.2 sec.

Given that this algorithm has to be implemented on-line, for an accurate selection of a value for N we also have to consider the computational speed of the airborne computer. The role played by the various parameters for the selection of N is shown in Fig. 8. Note that 1 and 2 sec. of computational time for the steady state Kalman Filter gains and covariances are assumed for N=12 and N=23, respectively.

As a final remark, a value of N around 10-12 should provide a very acceptable modeling accuracy without paying an excessive price in terms of (initial computational time + convergence time), and the on-line implementation for N=10-12 models should be within the available

computational power and speed of the today's aircraft computers.

Probability Convergence Sensitivity Analysis

For the determination of a Kalman Filter structure for a generic system, the dynamical model parameters (matrices A,B, L and C), the noise properties in terms of correlation, the noise statistics (matrices Q and R) have to be specified. Generally speaking, since the system model is usually an approximation to a physical situation, the model parameters and noise statistics are seldom exact. In other words the system model used in constructing the filter is different from the real system that generates the measurements.

Sometimes such an approximation is intentional; for example, expecially for radar tracking application where a large number of states are involved, it may be desirable to use a system model of lower dimension than the dimension of the real system in order to gain computational speed and simplicity. However, it is clear that an inexact filter will degrade the filter performances. Suppose that the real system is described by :

$$X(k+1) = A\,X(k) + B\,U(k) + L\,W(k) + Bias \tag{3.7}$$

$$Z(k) = C\,X(k) + V(k) \tag{3.8}$$

with $W(k) = N\,[\bar{w},Q]$, $V(k) = N\,[\bar{v},R]$;

where the model used to describe the system is given by:

$$X_m(k+1) = A\,X_m(k) + B_m\,U(k) + L_m\,W_m(k) + Bias_m \tag{3.9}$$

$$Z_m(k) = C_m\,X_m(k) + V_m(k) \tag{3.10}$$

with $W_m(k) = N\,[\bar{w}_m,Q_m]$, $V_m(k) = N\,[\bar{v}_m, R_m]$

where Bias and $Bias_m$ conventionally account for various sources of errors, for example: non-linearities, reduction in system dimensions and

so on.

Using the equations of the model we build the Kalman Filter structure, that is a recursive or steady state expression for the gain matrix K_m and for the estimation error covariance matrix P_m. The problem is that the computed matrix P_m is not the estimation error covariance matrix because the filter model is different from the real model. Neither is this filter the minimum variance filter for the actual system described by Eqs. (3.7) and (3.8).

A measure of the filter performance is provided by the "actual estimation error covariance matrix" defined by:

$$P_a(k+1/k) = E\left[\tilde{X}(k+1/k)\ \tilde{X}(k+1/k)^T\right] \tag{3.11}$$

(or the relative steady state expression) where

$$\tilde{X}(k+1/k) = X(k+1) - \hat{X}(k+1/k) \tag{3.12}$$

Ref. [25] provides a general form of an algorithm for obtaining a recursive expression for P_a.

Let's go back now to our original problem, that is to determine which model among a set of N models more closely describes the real aircraft dynamics following the damage, given the measurements.

The next point to be investigated is how the damaged model estimation algorithm performs when some discrepancies (other than differences in the A,B and L matrices) occur between the modeled system and the real system, which means differences in the C,Q and R matrices.

We can state that the assumption of uncorrelated disturbance noises and measurement noises can be considered acceptable and that it is reasonable to think that the observation matrix C is the same for the real and for the modeled system (which is consistent with the original assumption of availability of operating sensors for reconfiguration

purposes from Chapter I). Also that the values of the elements of the matrix R are furnished with a sufficient accuracy by the company manifacturing the sensors. Therefore, the only parameters that in the real life occurrences can show remarkable discrepancies between the real system and the modeled system are the values of the elements of the disturbance covariance matrix Q and, moreover, the statistical structure itself of the random vector W(k).

Probability Convergence Sensitivity Analysis For Different Values of the Elements of the Q Matrix

We have previously considered the turbulence covariance matrix Q with the coefficients $q_{11} = q_{22} = 0.0005$ and $q_{12} = q_{21} = 0.0$. In terms of gust components we would have a vertical velocity component W_g whose trend is shown in Fig.9. As we can see, the aircraft is going through a gust with vertical velocities up to $\pm$35 ft/sec. The relative set of N Kalman Filters is designed for the same values of such matrix. Therefore, at this condition, $Q = Q_m$.

Let's consider now a turbulence with a covariance matrix Q with the coefficients $q_{11} = q_{22} = 0.002$ and $q_{12} = q_{21} = 0.$, while inside the Kalman Filter's structures we still have $q_{m11} = q_{m22} = 0.0005$ and $q_{m12} = q_{m21} = 0$. In terms of gust components we would have a vertical velocity component W_g whose trend is shown in Fig.10. As we can see,in this case the aircraft is going through a gust with vertical velocities up to $\pm$75 ft/sec. Such kind of turbulence can be found in low altitude storms; it is a turbulence whose peak values are much higher than the previous ones. Aircraft simulation data corrupted with such a turbulence have been fed to our set of N = 12 Kalman Filters and the previously

introduced damaged model estimation process has been attempted. The expected result is that the convergence to the right model should still occur because, given that the wrong turbulence modeling is common to all N models, there is always a model that better describes the real dynamics but the estimation performance should somehow deteriorate. The result, shown in Fig.11, is that the convergence to the right model still occurs, but in a longer time and with a more 'turbulent' trend.

The reason for such trend of the probability associated with the model that more closely describes the aircraft dynamics lies in the complex nature of Eq.(3.6). It is clear that turbulence higher than expected give rise to higher values of the residuals of Eq.(3.5). Therefore, for values of Q higher than the corresponding values of Q_m, we are going to have higher negative values of the exponent of the numerator of Eq.(3.6) and, thus, smaller values of the of the overall numerator; in other words this means that $Prob_i(k+1)$ is less correlated to the $Prob_i(k)$, which explains the big changes in the time of the probability and the 'turbulent' behavior shown in Fig.11.

From the Kalman Filter equations we also can say that higher values of the elements of Q_m, which is related to the modeled but not to the real turbulence, imply higher values of the estimation error covariance matrix P_m and, therefore, higher values of the residual covariance matrix S and, of course, smaller values of the inverse of such matrix. It may be thought that if we design the Kalman Filters for values of the elements of Q_m much greater than the ones that we may expect from the real life turbulence it may help during severe turbulence because we would have higher values of the numerator of Eq.(3.6); this is only partially true because such approach would also imply smaller values of

the constants β's and higher values of the residuals, which, again, will make the probabilities at the instant (k) and (k+1) less correlated and the trend still 'turbulent', as shown in Fig.12. After all, we only have one 'optimal' solution! In Fig.12 the following values for the elements of Q and Q_m were considered: $q_{m11} = q_{m22} = 0.0005$, $q_{11} = q_{22} = 0.0001$, with all the off-diagonal elements being zeros.

A more functional approach to the problem would be to implement in the flight computer a relation of the elements of Q as function of the altitude (starting from a certain altitude in order to avoid local ground effects) instead of using one configuration for the matrix Q valid for all flight conditions (as shown in Ref.[30]), then to multiply the so calculated values for a factor moderately bigger than 1 to account for particularly severe turbulence and , finally, to store such values in the Q_m matrix of the Kalman Filter structure. Such an approach should protect the estimation process from unexpected high levels of atmospheric turbulence without excessively deteriorating the performances of the estimation process, as shown in Fig.13. In this case the following values for Q and Qm were considered : $q_{m11} = q_{m22} = 0.0005$ and $q_{11} = q_{22} = 0.00033$. This result has to be compared with the nominal conditions ($Q=Q_m$) result shown in Fig.5.

Another obvious approach in order to reduce the effects of the turbulence would be to filter the probabilities calculated with the previously introduced algorithm. A second order probability filter can take the generic form:

$$Fpr_i\ (k+1) = c_1\ Fpr_i(k) + c_2\ Fpr_i(k-1) + c_3\ Pr_i(k+1) \qquad (3.13)$$

where $\sum_{i=1}^{3} c_i = 1$, $Fpr_i(k)$ is the filtered probability relative at the instant k. By adjusting the values of the c_i's we can change the

correlation between the probabilities at instant (k+1) and instant (k).

Note that this approach is particularly useful when the model estimation task is also coupled with a control task in order to reduce excessive excursions in control input activities (Ref.[14],[15],[16], [21]). In our case, where we are only concerned with the damaged model estimation task, such approach doesn't improve the performance of the estimation process. In fact it increases the convergence time as shown in Fig. 14 (relative to the same values of Q and Q_m as in Fig.13) for 3 different set of values of the constants c_i's. This is definitely an undesirable effect. As we can see in Fig.14, the convergence time increases with increasing values of c_1 and c_2, which are the coefficients relative to the filtered probability at the instants (k) and (k-1) with respect to the value of c_3, which is the coefficient relative to the unfiltered probability at the instant (k+1).

Probability Convergence Sensitivity Analysis For Correlated Disturbance Noise W(k)

Up to this point the assumption of atmospheric turbulence modeled with a white noise Gaussian random vector has been made. In the real life the components of the atmospheric turbulence may show some form of correlation (Ref.[27]). The result of such approximation is typically an underestimation of the peak gusts by the Gaussian model. A great amount of research has been devoted in the past for creating realistic non-Gaussian correlated models of atmospheric turbulence to be used in flight simulators instead of Gaussian white noise generated turbulence. Data analysis have allowed modeling such turbulences with autoregressive (AR) processes of the first or, at most, of the second order. However,

expecially at higher altitude, the atmospheric turbulence components are not highly correlated.

For our purposes, let's consider now our atmospheric turbulence model vector W(k) given by two AR(1) processes:

$$\alpha_g(k+1) = 0.5 \quad \alpha_g(k) + e_\alpha(k) \tag{3.14}$$

$$\beta_g(k+1) = 0.5 \quad \beta_g(k) + e_\beta(k) \tag{3.15}$$

where $e_\alpha(k)$ and $e_\beta(k)$ are white noise Gaussian with zero means and variances $\sigma_\alpha^2 = \sigma_\beta^2 = 0.0005$, which are the diagonal elements of the Q_m matrix previously introduced. Note that $\alpha_g(k)$ and $\beta_g(k)$ are still assumed to be mutually independent.

The vertical velocity compoment of the gust for the cases of correlated atmospheric turbulence are shown in Fig. 15 and Fig. 9. Of course the autocorrelation function associated with the data for uncorrelated gust velocity component will tend to the classical "impulse at the origin and zero elsewhere" typical of a white noise process, as shown in Fig.16; as a check we know that $\sigma_{w_g}^2 = (Vel)^2 \sigma_{\alpha_g}^2$ $= R_{w_g}(\tau = 0) = 227.8$ (App. B), which is in good agreement with the plotted data. Note that such autocorrelation function has been calculated for 2500 data points. As expected, the autocorrelation function associated with the data for correlated gust velocity component will tend to zero more gradually as shown in Fig.17 for the same number of data points.

Our aircraft system is therefore going to be excited by a non-white noise random input vector, and consequently the aircraft response will also show some correlation. At the same time, our Kalman Filters structure is designed for an atmospheric turbulence modeled as a white noise process. This will generate, therefore, an inconsistency. A way

to detect the presence of unmodeled or uncorrectly modeled disturbance noise in our system is to analyze the autocorrelation function of the residuals of the Kalman Filters of Eq.(3.5). If the atmospheric turbulence exciting our aircraft system is a white noise process correctly modeled in the Kalman Filters structure, the associated autocorrelation function of the residuals (for the α and β states) will tend to the "impulse" look, as shown in Fig.18. For the residuals of model #6; on the other side, if the atmospheric turbulence exciting our aircraft system is not a white noise process and, therefore, not correctly modeled in the Kalman Filters structure, the associated autocorrelation function of the residuals (for the α and β states) will decrease a little more gradually, as shown in Fig.19, for the residuals of model #6.

At this point let's analyze how this affects the performance of the model estimation process. The real mathematical model describing the aircraft system generating the dynamic data increases its order from 8 to 10 for modeling the correlation in the components $\alpha_g(k)$ and $\beta_g(k)$. The now increased order state variable model is shown in Table VI.

Dynamic simulation data from this system which at a certain time experiences a change in the dynamics due to the damage have been fed to our Kalman Filters structure and the damaged model estimation process has been attempted. The result is that, with a certain amount of surprise, the probability still converges to the right model (which is model #6), as shown in Fig.20, with approximatively the same convergence time shown in Fig.5, which was relative to nominal conditions of exactly modeled atmospheric turbulence in terms of covariance and color. Note that the peak shown in Fig.20 is not of particular concern to us because

by the time that such peak occurs we would have already exited the algorithm because we already had reached the convergence condition for same time. Next, we will try to understand the reasons for a behavior so relatively similar to the conditions of uncorrelated noise. Again, an explanation lies in the complex structure of Eq.(3.6). The covariance residual matrix S_{UNC} (or its inverse S_{UNC}^{-1}), where 'UNC' stands for uncorrelated atmospheric turbulence, relative to model #6, implemented in the algorithm up to this point, is associated with the estimation error covariance matrix P_{UNC}, relative to model #6, calculated by the Kalman Filter structure using the relation:

$$S_{UNC} = C*P_{UNC}*C^T + R \tag{3.16}$$

Such S_{UNC} and P_{UNC} are therefore relative to a 8-th order system with atmospheric turbulence correctly modeled as white noise processes. Next, we would like to calculate the "true" estimation error covariance matrix, relative to model #6, for a 10-th order system with a 8-th order filter and the associated covariance residual matrix, P_{COR} and S_{COR}, where 'COR' stands for correlated atmospheric turbulence. In order to do so, let's recall the system equations and the Kalman Filter equations; note that, without any loss of generality, the deterministic input U(k) is not considered; also note that the matrices A and L are relative to model #6.

$$X(k+1) = A\,X(k) + L\,W(k) \tag{3.17}$$

$$Z(k) = C\,X(k) + V(k) \tag{3.18}$$

$$\hat{X}(k+1/k) = A\,[I - KC]\,\hat{X}(k/k-1) + AK\,Z(k) \tag{3.19}$$

where the gain matrix K is calculated by the Kalman Filter and it is relative to model #6. By using Eq.(3.18), Eq.(3.19) will become:

$$\hat{X}(k+1/k) = A\,[I - KC]\,\hat{X}(k/k-1) + AK\,(C\,X(k) + V(k)) \tag{3.20}$$

If we introduce $\varepsilon(k+1) = X(k+1)-\hat{X}(k+1/k)$, we have:

$$\varepsilon(k+1) = AX(k) + L\,W(k) - A\,\hat{X}(k/k-1) + AKC\,\hat{X}(k/k-1)$$
$$- AKC\,X(k) - AK\,V(k) \qquad (3.21)$$

Next, we can introduce a new augmented state variable vector:

$$\begin{bmatrix}\varepsilon(k+1)\\ W(k+1)\\ V(k+1)\end{bmatrix} = \begin{bmatrix}A\ (I_8-KC) & L & -AK\\ 0_{(2x8)} & E & 0_{(2x6)}\\ 0_{(6x8)} & 0_{(6x2)} & 0_{(6x6)}\end{bmatrix} \begin{bmatrix}\varepsilon(k)\\ W(k)\\ V(k)\end{bmatrix}$$

$$X_{AUG}(k+1) \qquad A_{AUG} \qquad X_{AUG}(k)$$

$$+ \begin{bmatrix}0_{(8x2)} & 0_{(8x6)}\\ I_2 & 0_{(2x6)}\\ 0_{(6x2)} & I_6\end{bmatrix} \begin{bmatrix}e_W(k)\\ e_V(k)\end{bmatrix} \qquad (3.22)$$

$$L_{AUG} \qquad W_{AUG}(k)$$

Note that the order of X_{AUG} is (8+2+6) = 16;

the order of W_{AUG} is (2+6) = 8;

the size of A_{AUG} is (16x16);

the size of L_{AUG} is (16x8).

Also note that V(k), the measurements noise, is still a white noise Gaussian random vector with covariance matrix R.

Q_{AUG}, which is the covariance matrix for W_{AUG}, C_{AUG} and E will be given by :

$$Q_{AUG} = \begin{bmatrix} Q & 0_{(2x6)}\\ 0_{(6x2)} & R\end{bmatrix} \qquad (3.23)$$

$$C_{AUG} = \begin{bmatrix} C & 0_{(6x8)} \end{bmatrix} \tag{3.24}$$

$$E = \begin{bmatrix} 0.5 & 0.0 \\ 0.0 & 0.5 \end{bmatrix} \tag{3.25}$$

The diagonal elements of the matrix E are the coefficients of the two AR(1) process which are modeling the elements of vector W(k), which are $\alpha_g(k)$ and $\beta_g(k)$. Therefore we have a new set of state variable equations:

$$X_{AUG}(k+1) = A_{AUG}\, X(k) + L_{AUG}\, W_{AUG}(k) \tag{3.26}$$

$$Z(k) = C_{AUG}\, X_{AUG}(k) + V(k) \tag{3.27}$$

The steady state estimation error covariance matrix for correlated (but not modeled by the Kalman Filters structure) atmospheric turbulence and the residuals covariance matrix can be found from :

$$P_{COR} = A_{AUG}\, P_{COR}\, A_{AUG}^T + L_{AUG}\, Q_{AUG}\, L_{AUG}^T \tag{3.28}$$

$$S_{COR} = C_{AUG}\, P_{COR}\, C_{AUG}^T + R \tag{3.29}$$

Note that P_{COR} and S_{COR} are respectively the "true" estimation error covariance matrix and "true" residual covariance matrix for a 10-th order system modeled with a 8-th order filter. These matricial manipulations have been implemented in a user defined function on $MATRIX_x$ and the results in terms of S^{-1}, which is the matrix playing a key rule in Eq.(3.6), are shown in Table VII. Note that the unmodeled correlated turbulence does not involve very big changes in the S matrix and the β costant, with respect to S_{UNC} and β_{UNC}. On the other hand, by looking at the trends of the uncorrelated and correlated vertical velocity components of the gust in Fig. 15 and Fig. 9, we don't expect the magnitudes of the residuals of the Kalman Filter for model #6 to be too different between the two cases.

Therefore, this similarity of the residuals and the closeness between S_{UNC}^{-1} and S_{COR}^{-1}, β_{UNC} and β_{COR} justify the not too different trend of the probability convergence shown in Fig.20 and Fig.5. The bigger correlation between the probabilities in following instants shown in Fig.20 with respect to the nominal conditions is surely related to the correlation of the turbulence components which influence the correlation of the probability troughout the correlation of the filter residuals.

The shown robustness of the Multiple Model Kalman Filtering approach to non-Gaussian correlated noise is consistent with what it is stated in Ref.[21].

Essentially, up to this point, we have shown that a moderate correlation in the atmospheric turbulence, if it really exists, doesn't deteriorate the performance of the model estimation process. Again, let's point out that correlation in the components of the atmospheric turbulence is not very high and surely decreases with altitude because it is mostly due to ground effects. Once it occurs that we have to perform a model estimation process in such atmospheric conditions, the question is the following: Could we be able to model such correlation on-line, before implementing the Kalman Filters structure?

The previously carried analysis was essentially done off-line with the residuals of an already designed Kalman Filters structure. In real life we should first estimate such correlation and then design an appropriate Kalman Filters structure ; however, in order to do so, there are three main problems. First, it may be impossible, in those instants when the aircraft is drastically changed its dynamics due to the damage, to extract from the sensors data the turbulence components in order to

analyze them; second, whatever method we would eventually use to determine the order and then the coefficients of the correlation, we are going to need to collect a certain amount of data and then we have to analyze them with some computationally time consuming algorithms; all of this has to be done while the damaged aircraft is "waiting" to be reconfigured and we know that time is a key point for the success of the reconfiguration; third, if eventually we could come up with an increased order system to account for correlated turbulence, then the computaional time to design the Kalman Filters will exponentially increase.

These real life factors combined with the fact that correlation, if any, doesn't deteriorate too much the model estimation process performance lead to the conclusions that, for our purposes, the components of the atmospheric turbulence can be modeled with sufficient accuracy as white noise Gaussian random vectors.

Probability Convergence Sensitivity Analysis for Nonlinear Damaged Aircraft Dynamics

As far as the nonlinearity of the real dameged aircraft system is concerned, the issue is whether the tracking error from the Kalman Filter corresponding to the linearized model closest to the true, nonlinear system is markedly smaller than the errors from filters based on 'more distant' models.

Given that the residual covariance matrices S's are fixed for each model, the values of the exponents of the Eq. (3.6) and therefore the probability will depend upon the values of the residuals; higher values of the residuals (which is typical from nonlinear system response), with fixed S^{-1}'s, will reduce the tracking capabilities of the approach and

it will make more difficult to distinguish among the models.

Consequently, from a numerical simulation point of view, we are going to have trends similar to the ones shown for the probability convergence sensitivity analysis to higher than expected turbulence. Intuitively, we can say that the performance of the model estimation process will depend upon how 'far apart' the different models are. However, future considerations introduced in Chapter IV will show us that this problem is not of particular concern to us.

Conclusions

In this chapter an approach for estimating the mathematical model of the aircraft reflecting the changes due to the damage using Multiple Model Kalman Filters is proposed. The advantage of this approach is the relative simplicity since we can take advantage of the particular way th t the elements of the matrices A and B may change following a damage on a control surface. This could make such an approach particularly attractive for practical implementation. The rule played by the number N of models to be implemented has been outlined; a choice of N around 10-12 has been assumed to represent an acceptable trade-off point between modeling accuracy and (Kalman Filters design computational time + probability convergence time). The effects of atmospheric turbulence with intensities higher than expected on the model estimation process are considered; it is suggested to implement an expression of the values of Q_m as function of the flight conditions, mostly of the altitude, rather then using only one atmospheric turbulence model for the design of the Kalman Filters structure. Finally, the effects of correlated components of atmospheric turbulence on the model estimation process are

considered. The usually small amount of correlation, the almost unchanged model estimation performance, the impossibility of extracting atmospheric turbulence data from the sensors in the instants immediately following a damage, the excessive price in terms of computational time for calculating and modeling such correlation bring us to the conclusion that it is reasonable to model the atmospheric turbulence components as white noise Gaussian random processes.

CHAPTER IV

RECONFIGURATION ALGORITHM

Statement of a Control Problem

In the previous chapter we have introduced an approach in order to determine the mathematical model of the aircraft which reflects the changed dynamics due to the damage. Now we need an algorithm which, given the models of the nominal undamaged and damaged aircraft system and the time histories of the deterministic control surfaces input, is able to find the equivalent input that will make the damaged aircraft system respond as the undamaged system would under normal circumstances.

Introduction to the Reconfiguration Algorithm

The reconfiguration algorithm used for our purposes was introduced in Ref.[1]. In order to present such an approach, let's consider a 2-nd order linear time invarient controllable system. In the continuous-time state variable form we have:

$$\dot{X}_1 = X_2 \tag{4.1}$$

$$\dot{X}_2 = a_1 X_2 + a_2 X_1 + a_3 u \tag{4.2}$$

where u is the control variable, required to be bounded; a_1, a_2 and a_3 are constant coefficients.

Our goal is to determine a method of computing u which will make the system defined by Eqs.(4.1) and (4.2) to behave like the ideal

system described, in the same state variables form, by:

$$\dot{Y}_1 = Y_2 \tag{4.3}$$

$$\dot{Y}_2 = b_1 Y_2 + b_2 Y_1 + b_3 u_m \tag{4.4}$$

where u_m is the control variable for the ideal system, which is also required to be bounded; b_1, b_2 and b_3 are also constant coefficients. By integrating Eq.(4.2) between the time t=nh and t=(n+1)h, assuming that u has a value u(n) over this interval, we will have:

$$X_2(n+1) - X_2(n) = A + B\,u(n) \tag{4.5}$$

where $X_2(n)$ and u(n) are used to denote $X_2(nh)$ and u(nh) and A and B are:

$$A = a_1 \int_{nh}^{(n+1)h} X_2(t)dt + a_2 \int_{nh}^{(n+1)h} X_1(t)dt \tag{4.6}$$

$$B = a_3 h \tag{4.7}$$

The desired change in X_2 over the time interval is defined as Z(n). If the actual change has to be the same as the desired change, from Eq.(4.5) we have:

$$u(n) = [Z(n) - A]/B \tag{4.8}$$

Once we determine A and B, the control required to make the actual change in X_2 equal to the desired change can be provided by Eq.(4.8). Note that the two terms in A will not generally be constant and they will depend on the varying values of X_1 and X_2. At this point we assume that A and B both are approximately constant over a small group of intervals. Therefore, by applying Eq.(4.5) on these preceding intervals we have:

$$X_2(n) - X_2(n-1) = A + B\,u(n-1) \tag{4.9}$$

$$X_2(n-1) - X_2(n-2) = A + B\,u(n-2) \tag{4.10}$$

At the beginning of the n-th interval, where u(n) is to be

computed, all the terms in Eqs.(4.9) and (4.10) will be known from measurements (or from computer simulations), except A and B. These can be determined by elimination between the two equations:

$$A = \frac{u(n-2)[X_2(n)-X_2(n-1)] - u(n-1)[X_2(n-1)-X_2(n-2)]}{u(n-2) - u(n-1)} \tag{4.11}$$

$$B = \frac{2\,X_2(n-1) - X_2(n-2) - X_2(n)}{u(n-2) - u(n-1)} \tag{4.12}$$

The last parameter that we need to calculate is the desired change Z(n). For example, suppose that the system is responding to a constant step input of magnitude M, then the desired change in X_2 over the n-th interval is taken as:

$$Z(n) = h\,[b_1\,X_2(n) + b_2\,X_1(n) + b_3\,M] \tag{4.13}$$

Substituting this value of Z(n) into Eq. (4.8), together with the values of A and B previously calculated, we will have a control algorithm which will make the real system behave like the ideal one would at the same point under normal circumstances. By accomplishing this in succeeding time intervals, then the real system will be reconfigured.

The introduced methodology looks surprisingly simple and, at the same time, efficient. However, there are two problems that need to be kept in mind:

1 - The resulting value of u(n) might not be bounded.

2 - u(n) might be too close to u(n-1) and numerical problems may arise for the computation of u(n+1). This is because we deal with small denominators in Eqs.(4.11) and (4.12).

Characteristics of the Reconfiguration Approach

Before going to the details of the application of the method to flight control reconfiguration, some further considerations need to be made. According to the way that the algorithm is going to be used, this can essentially be classified as a "model following" problem, where the model of the nominal aircraft can be constructed on-line (see Fig.2) by using the aerodynamic, inertial, geometric characteristics stored in the flight computers along with the closed-loop characteristics, if a S.A.S. or a C.C.V. function is implemented, and the flight data (velocity, dynamic pressure, air density) coming from the operational sensors. The dynamic model of the real damaged aircraft is also constructed on-line by using the previously introduced Multiple Model Kalman Filters method.

As it can be seen from Eqs.(4.9)-(4.11), there is a 2 sample instants delay associated with the calculation of the required control input at time 'nh', u(n).

In order to illustrate the approach we have considered a linear time-invariant system. But, without any loss of generality, we could consider a nonlinear system as long as we are able to simulate it (Ref.[1]). In fact Eqs.(4.8),(4.11) and (4.12) still hold; the only difference lies in Eq.(4.13). Of course, given that the damaged and nominal nonlinear systems have to be simulated on-line, the dynamic simulations of such systems are going to be computationally more time consuming.

Note, that using such approach, the model estimation task and the control task are totally separated, with the noise problem faced in the model estimation part. The dynamic simulation by the reconfiguration algorithm is essentially deterministic, without any disturbance. A

previous attempt to use in the reconfiguration algorithm some computer simulated noised data representing the damaged aircraft dynamic response has given rise to an unacceptable control activity, in terms of actuator frequencies and unrealistic maximum and minimum angular deflections of the control surfaces.

Application of the Method to Aircraft Reconfiguration

The technique introduced for a simple single-input 2nd order system has been used for a much more complex multi-input aircraft system. Again, the linearized models of the damaged and nominal aircraft dynamics can be calculated using the procedures described in Ref.[13]. Note that, in order to evaluate the performance of the reconfiguration algorithm only, we assume that the damaged aircraft model has already been estimated and that the damaged aircraft dynamics is 'exactly' described by one of those N models of the Multiple Model Kalman Filters structure. This is a reasonable assumption if a sufficiently high number N of models is implemented, that is N = 10-15.

For a realistic simulation we have to consider the following possibilities for a damaged control surface:

1 - Control surface that remains fixed at a particular position.

2 - Control surface that jams to the maximum angular deflection.

3 - Control surface that jams to the minimum angular deflection.

As it was introduced in Chapter II, the aircraft is assumed to have 9 independent control surfaces:

- Right and left elevators (max. and min. defl. of $\pm 25^{o}$).

- Right and left spoilers (max. and min. defl. of $\pm 45^{o}$).

- Right and left ailerons (max. and min. defl. of $\pm 25^{o}$).
- Right and left canards (max. and min. defl. of $\pm 25^{o}$).
- Rudder (max. and min. defl. of $\pm 30^{o}$).

A key factor for a successful application of this reconfiguration technique consists of being able to implement on-line a very small value of the reconfiguration step 'h', introduced in Eq. (4.6); that is, at each interval of time that the reconfiguration method is applied. Such value is a function of the reconfiguration algorithm complexity and available computational power of the airborne computer.

On the other side, values of h too small may potentially give rise to a problem because u(n-2) might be too close to u(n-1), as previously mentioned. In order to avoid this problem a control is performed in the code such that in case of a 'flat' input, that is u(n-2)=u(n-1), a minimum fixed quantity is assumed for (u(n-2)-u(n-1)).

Given that a linearized set of equations represents the best compromise between modeling accuracy and low computational time for modeling a 6 degrees-of-freedom aircraft, our next goal is to minimize the reconfiguration algorithm complexity. In order to do so, the following consideration can be made. Because of the nature of the aircraft dynamics, such system is a completely state controllable system. From a mathematical point of view this implies that:

$$\text{rank } W_c = \text{rank } [B, AB, \ldots, A^{n-1}B] = n \qquad (4.14)$$

where 'n' is the order of the system, as it has been proved by using MATRIX_X.

From a practical point of view this means that it is enough to find the control inputs that reconfigure some of the states to be sure that all the states are reconfigured. A typical choice would be to

reconfigure p,q,r, which are the angular velocities around the stability axes X_s, Y_s, Z_s.

Once that the amount of control input needed for the reconfiguration of a state is calculated by using Eq.(4.8), after we have obtained for each time interval the quantities Z(n), A and B, the next step is to distribute efficiently from an aerodynamic point of view this control input among all the remaining available healthy independent control surfaces. The approach that has been implemented is that each control surface contributes to the reconfiguration with an amount, W_i, proportional to its effectiveness:

$$W_i = \frac{\text{Control effectiveness of the i-th control surface}}{\text{Sum of control effectiveness of all surfaces}} \tag{4.15}$$

For example, in order to reconfigure the q state after the damage on the left elevator, we have that the remaining right elevator will furnish W_{ER} of the amount of needed control, where W_{ER} is given by:

$$W_{ER} = \frac{\text{abs}(C_{m_{\delta ER}})}{\text{SUM}(\text{abs}(C_{m_{\delta i}}))} \tag{4.16}$$

with $\text{SUM}(\text{abs}(C_{m_{\delta i}})) = \text{abs}(C_{m_{\delta ER}}) + 2\ \text{abs}(C_{m_{\delta SL}}) + 2\ \text{abs}(C_{m_{\delta CL}})$.

The p and r states are ordinarily reconfigured by using, respectively, the ailerons and the rudder.

Such approach will avoid the saturation of a particular control surface used for the reconfiguration leaving some angular deflection margins to be used in the following flight manoeuvers by the pilot or by the stability augmentation systems (S.A.S.) or control configured vehicle (C.C.V.) functions implemented on the aircraft.

Reconfiguration Simulation Results

The introduced algorithm to be used for the aircraft reconfiguration has been implemented in a Pascal program.

Let's recall one more time that, for the time being, we are still assuming the aircraft being originally 'naturally' stable with satisfactory handling qualities and, therefore, S.A.S. or any form of C.C.V. functions are not considered. This has only been done for simplicity purposes and it does not affect the performance of the reconfiguration algorithm because the algorithm only uses the mathematical models of the damaged and of the nominal aircraft, regardless if they represent an open-loop or a closed-loop dynamics. However, we may want to keep this in mind for future comments.

The dynamics of the chosen aircraft model is simulated at the flight conditions shown in Table I, that is high altitude and high subsonic Mach number, which is typical of an air combat scenario. The introduced maneuvers are also typical of a duel situation, with step inputs on elevators and ailerons, as shown in Fig.4 and Fig.21.

The damage is simulated to occur at time = 1 sec.; consistently with what we have done in the previous chapter, we assume that (4-4.5) sec. is the time needed by the Failure Detection and Identification tasks and for the design of the Kalman Filters structure while (1.5-2) sec. is the time needed for the convergence of the probability associated with the model that more closely describes the damaged aircraft dynamics. Therefore, the total time span between the instant when the damage occurs and when the reconfiguration algorithm takes over is assumed to be 6 sec. Two different cases have been considered.

Case 1 (Fig.22-Fig.24) is relative to a situation where a damage on

the left elevator implies a reduction of around 1/2 of its control effectiveness (from $C_{L_{\delta_{EL}}} = 0.278$ to 0.121), with the deflection remaining fixed at the position at the instant of the damage, that is -5^{o}.

The task of the damaged surface is distributed, using the criteria introduced in Eq.(4.16), among the remaining half elevator, the right and left spoilers, the right and left canards. Particularly, using the data in Table I, we have:

$$W_{ER} = 41.86\% \tag{4.17}$$

$$W_{CL} = W_{CR} = 20.35\% \tag{4.18}$$

$$W_{SL} = W_{SR} = 8.7\% \tag{4.19}$$

Fig.22-Fig.24 show that the introduced algorithm, which takes over at time = 7 sec., achieves in a very short amount of time an accurate reconfiguration for the angle of attack, for the pitching angular velocity and properly counteracts the rolling moment induced by the damaged left elevator. Fig. 25 shows the associated right elevator and canards deflection inputs calculated by the algorithm. In such figure the opposite sign of the deflection of the canards and the right elevator is due to the fact that they are located in opposite positions with respect to the center of gravity. Also note that the absolute values of these deflections happen to be very similar because of the values of W_{CL}, WCR, and W_{ER} in Eqs. (4.17, 4.18) and because of the values of $C_{m_{\delta CL}}$, $C_{m_{\delta CR}}$, $C_{m_{\delta ER}}$ in Table I.

It is important to notice expecially from Fig.22 and Fig.23 how the characteristics of the aircraft dynamic response change following the damage. In fact the damage on the left elevator, which, as we have previously stated, involves a reduction of the value of $C_{L_{\delta E}}$ and,

consequently, a reduction of the value of $C_{m_{\delta E}}$, influences moreless strongly the values of C_{m_α}, C_{m_q}, $C_{m_{\dot{\alpha}}}$, and C_{L_α}, as shown in Chapt. II (Ref.[13] and Ref.[17]-[18]), which then cause mostly a change in the short period characteristics of the aircraft. This can be shown in the following expressions, valid for the short period approximation in the longitudinal dynamic stability (Ref.[13]):

$$\omega_{n_{s.p.}} = \left((Z_\alpha M_q / U_1) - M_\alpha \right)^{1/2} \tag{4.20}$$

$$\zeta_{s.p.} = \frac{-(M_q + Z_\alpha / U_1 + M_{\dot{\alpha}})}{2\,\omega_{n_{s.p.}}} \tag{4.21}$$

Z_α, M_q, M_α, $M_{\dot{\alpha}}$ are the dimensional stability derivatives containing, respectively, C_{L_α}, C_{m_q}, C_{m_α}, $C_{m_{\dot{\alpha}}}$, U_1 is the steady state forward speed. This can be seen in Fig.26, where the short period characteristics are plotted in the s-plane; the reason that we go back to the continuous time domain is that this is the plane where the conditions in terms of handling qualities are usually assigned. For normal damaged conditions the poles obtained by using the EIG command in $MATRIX_X$ are shown with the symbol '+'. The ones calculated by using Eqs. (4.20) and (4.21) are shown by using the symbol '*'. Given that the short period poles are complex conjugates (or, at least, this is the way that they normally are) only the positive one is shown. Under damaged conditions, the positions of the poles calculated by using $MATRIX_X$ and Eqs.(4.20)-(4.21) are shown, respectively, with the symbols '-' and 'x'. The closeness of these poles for both conditions proofs the validity of the short period approximation. Table VIII shows instead the effects of the damage on the natural frequency and on the

damping of the short period mode; in the Table both the results using $MATRIX_X$ and using Eqs.(4.20) and (4.21) are reported.

This discussion brings up the issue that the damage may decrease the stability, in this case the longitudinal stability, and we can even reach the point where the aircraft (regardless if it is designed 'naturally' stable for those particular flight conditions or it is 'artificially' stabilized with a S.A.S.) becomes unstable after the damage.

In this extreme case time is really a key factor because if the reconfiguration algorithm is applied too late, that is the unstable dynamic trend is already well developed, there may not be enough control authority to bring back the aircraft to perform with the desired dynamic characteristics because the aircraft may have already gone into some unrecoverable flight conditions.

However, back to our case, as it can be seen in the enclosed figures, the application of the reconfiguration algorithm 'forces' the damaged aircraft to give the same dynamic response of the nominal aircraft. No attempt has been made, up to this point, in order to restore desirable handling qualities. If a S.A.S. is implemented on the aircraft, like it is always the case for fighters, this could be achieved by redesign of the feedback gains. However, this is going to be the topic of the next chapter.

Also, in Fig.22-24, by looking at the values of the transients of the plotted parameters we can see that we are around the limits of the linearity assumption.

Case 2 (Fig. 27-30) is relative to a similar situation. The only difference is that, following the damage, the left elevator jams to the

minimum angular deflection, that is -25^{o}. Note that this is a very severe condition. Even in this case, an acceptable reconfiguration is achieved for the angle of attack and for the pitching and rolling angular velocities; note that the induced rolling moment is in this case a not negligible side-effect. By looking at the relative figures we can clearly see the 2 instants delay nature of the reconfiguration algorithm. The differences in Fig.27-29 between the undamaged and the reconfigured responses are bigger than the respective differences shown in Fig.22-24, because of the higher amount of aerodynamic forces exerted, during that 2 instants delay, by the damaged surface jammed to the minimum deflection.

However, the magnitude of the parameters plotted in Fig.27-29 leads us to some considerations; in fact such magnitudes are surely outside the limits of the linearity assumption. For those values of the angle of attack and of the angular velocities the linear plant models are definitively no longer valid, given that the aircraft aerodynamics becomes highly non-linear.

Now, recall that we have considered an open loop aircraft, without any form of S.A.S. or C.C.V. functions implemented in the flight control system. Today's generation fighters are equipped with fly-by-wire systems in which control surface deflections are not commanded directly by the pilot but are generated by the flight control computers in order to achieve commanded states or to achieve some desired dynamic response.

Therefore, in the instants following a damage on a control surface when the aircraft is experiencing substantial linear and rotational impulses, these damage induced disturbances to the aircraft would be opposed by the closed-loop control laws. In other words, the C.C.V.

functions (for example yaw, pitch, roll dampers, maneuver and gust load alleviation, flight envelope limiting,...) would be using multiple control surfaces, even if not for reconfiguration purposes, that would keep the values of α,p,q and all other state variables at lower magnitudes than the ones shown in Fig. 27-29. Even if not numerically simulated in this study, the dynamic characteristics of the aircraft modified by the sophisticated C.C.V. functions are considered in our overall approach to the reconfiguration problem, shown in Fig.2, under the name of closed-loop characteristics to be introduced for the construction of both the damaged and the nominal mathematical models of the aircraft. Hence, for a modern C.C.V. fighter, the magnitudes of the states after a damage would be much lower than the ones shown in Fig. 27-29 and, most likely, within the limits of the linearity assumption.

Looking back to Chapter III, this surely helps the damaged model estimation process to perform within the basic assumption that the aircraft dynamics for the N models can be described by N sets of linearized equations.

After these reflections concerning the nature of the aircraft transient response immediately after the damage it is important to point out that other main factors for the reconfiguration success are the maximum angular velocity of the actuators and the maximum number of impulses that can be sent to such actuators in a time unit. In the reconfiguration algorithm the maximum allowed actuator angular speed was 200^{o}/sec, which is around the average of today's actuators.

This brings up the problem that the advantages of a reconfiguration technique can be experienced only if the remaining actuators implemented are strong and quick enough.

Conclusions

In this chapter the algorithm to be used for the reconfiguration has been introduced, outlining some of its characteristics and some of the problems associated with it.

We have illustrated the particular way that such algorithm has been used, that is in a "model following" mode, with the damaged and nominal aircraft system models being deterministic and constructed on-line using the flight data and the closed-loop characteristics.

We also have pointed out that such algorithm can be applied to nonlinear aircraft models, but this would require much longer computational time in the simulation. A key point for the success of the reconfiguration is to implement a small value of the reconfiguration step 'h'.

Also, a particular procedure in order to reduce the algorithm complexity has been introduced, that is reconfiguring only some key states, by taking advantage of the controllability of the aircraft system. Furthermore, a method is proposed to efficiently distribute the reconfiguration task among all the remaining healthy control surfaces.

Then, the simulation results for a typical combat inputs in a typical combat scenario have been shown, discussing some important points, like the change in the stability characteristics of the aircraft following the damage, the validity of the linearity assumption and the need for strong and quick actuators to be driven by this reconfiguration algorithm.

CHAPTER V

REDESIGN OF THE FEEDBACK GAINS BY EIGENSPACE ASSIGNMENT

Introduction

Up to this point, we have introduced a Multiple Model Kalman Filtering approach for the estimation of the mathematical model of the aircraft considered to reflect the damage and a reconfiguration algorithm which "forces" the damaged aircraft to behave like the nominal one would under normal circumstances. In other words, by using the introduced reconfiguration algorithm, we merely calculate a set of "compensating" inputs.

The chosen aircraft model is considered to be, at that particular flight condition, 'naturally' stable and satisfying some particular requirements for handling qualities in terms of rigid body characteristics (that is, short period, phugoid, dutch-roll, spiral and rolling modes); therefore a S.A.S. has not been considered; also, the model is assumed to implement none of the C.C.V. functions (pitch, roll and yaw damping, gust and load alleviation, flight envelope limiting, etc...).

However, for military aircraft, expecially fighters, strict requirements in terms of maneuvrability make the resulting aircraft 'naturally' unstable throughout all its flight envelope (altitude vs.

Mach number). Therefore, some form of S.A.S. needs to be implemented in order to satisfy the handling qualities requirements. On top of the S.A.S. we may then design other control systems that implement other C.C.V. functions.

Therefore, in order to reflect this real life situation, we are going to modify the given aircraft model by making it 'naturally' unstable; then, by using MATRIXx, a feedback control can be designed to make the closed-loop aircraft satisfy the handling qualities requirements. Next, once the damaged model has been estimated, we would like the Feedback structure to be redesigned in order to accomodate the changed dynamics of the controlled aircraft system.

Consider now the conservative case when the damaged control surface, besides being used by the pilot, has a feedback control input to the surface. In this case, by redesigning the feedback structure, we not only want to guarantee the nominal undamaged satisfactory handling qualities but also would like to remove the unavoidable coupling between the longitudinal and lateral-directional dynamics that the damage on the control surface has generated.

It may be thought that the reconfiguration has been already accomplished with the introduced algorithm, and this is true. However, we may look at the problem as in the following: After the damage occurs, once the damaged model estimation has been succesfully accomplished, let the reconfiguration algorithm perform the initial heaviest load of the reconfiguration itself, that is to bring back the aircraft from whatever conditions it had reached immediately following the damage to the desired states. In the same time, on a parallel computational line, let the feedback structure be redesigned for that particular flight

condition and for all the flight envelope with the goal of retaining desirable handling qualities and removing the damage-induced dynamic coupling.

Once this has been done, the closed-loop characteristics of the aircraft need to be modified such to reflect the redesign of the feedback structure and, consequently, we can update the damaged aircraft mathematical model used in the reconfiguration algorithm. In other words, the reconfiguration, initially obtained by using only the algorithm previously introduced, is now accomplished by using both the Feedback Structure and the reconfiguration algorithm itself, as it can be seen in Fig.2.

From a flight management point of view this is surely a more functional approach. In fact it wouldn't make sense to maintain for the rest of the flight the original feedback structure for a system that changed its characteritics.

Statement of the Problem

For the design of a S.A.S. the most common approach is given by the state feedback technique. Generally, given a dynamic system described in the discrete-time form by:

$$X(k+1) = A\,X(k) + B\,U(k) \tag{5.1}$$

The control input vector U(k) can be expressed as:

$$U(k) = U_F(k) + U_p(k) \tag{5.2}$$

where $U_F(k)$ indicates control inputs used by the state feedback while $U_p(k)$ indicates pilot deflection inputs. Using the state feedback approach we have:

$$U_F(k) = K\,X(k) \tag{5.3}$$

where K is a (mxn) matrix of feedback gains. By using Eq.(5.1), Eq.(5.3) becomes:

$$X(k+1) = [\ A + B_F K\ ]\ X(k) \ + B_P\ U_P(k) \qquad (5.4)$$

where B_F is relative to $U_F(k)$; B_P is relative to $U_P(k)$; $[A + B_F K] = A_{C.L.}$ = closed-loop state matrix. Note that $U_F(k)$ and $U_P(k)$ may have some common elements.

An aircraft cannot be considered to be a single-input system, given that the control surfaces are at least 3. However, for most aircraft configurations, under design conditions, the longitudinal and the lateral-directional dynamics can be decoupled. Therefore, we can split the design of an aircraft S.A.S. into the longitudinal part and the lateral-directional part, each one using a single control input (usually elevator or canards in the longitudinal case and rudder and/or ailerons in the lateral directional case). Note that this is not necessarily true for particular aerodynamic configurations, for example, aircraft with wings or horizontal tails with high dihedral angle (Ref.[32]) or with oblique wings (Ref.[15],[16]).

In order to show the implementation of this design philosophy, some of the original aerodynamic data have been changed in order to simulate a 'naturally' unstable aircraft, which requires a S.A.S. in order to be stable. Particularly, the following data from Table I have been modified:

C_{m_α} = -0.75 --> 0.1

$C_{m_{\dot{\alpha}}}$ = -6.7 --> -5.36

C_{m_q} = -20.15 --> -16.12

C_{L_α} = 5.9 --> 5.605

This reduction of the aerodynamic data values has been made empirically

but it reflects the fact that an 'artificially' stable aircraft usually has smaller wings and a very reduced size of the horizontal tail (Ref.[31]). This can be seen in Fig. 31. The short-period characteristics relative to the aircraft without a S.A.S. show that the handling qualities are unacceptable. In fact we have:

$$\text{POLE } 1_{S.P.} = -2.1890; \qquad \text{POLE } 2_{S.P.} = -0.0962$$

$$\text{POLE } 1_{P.} = 0.1708 + i\ 0.1112; \qquad \text{POLE } 2_{P.} = 0.1708 - i\ 0.1112$$

As we can see, the short period poles are no longer complex conjugates, and the phugold has become unstable. However, the S.A.S. has to guarantee satisfactory flying qualities throughout all the flight envelope. This is obtained with a form of gain scheduling. From an implementation point of view this means that the flight envelope is subdivided into a certain number of regions within which the aircraft exhibits satisfactory flying qualities with that particular set of gains. Fig. 32 shows a typical flight envelope. Twenty-eight different regions have been introduced where the actual eigenvalues are within a limited range from the desired eigenvalues. For example the flight condition A_4V_3 is relative to a condition with altitude at level 4, that is between 25,000 and 35,000 ft, and with Mach number at level 3, that is between 1.0 and 1.3.

At this point we recall that, because of the damage, the longitudinal and lateral directional dynamics are no longer decoupled. This implies that, if we want to modify or, more realistically, redesign the flight control system in order to retain desirable flying qualities even after the damage, the approach of using 2 separated single-input subsystems is no longer valid.

Control of a Multi-Input Linear Time-Invariant System

For the design of a feedback structure for a linear (or linearized) time-invariant multi-input system, two different approaches can be followed :

1 - Linear Quadratic Optimal Control method.

2 - Eigenstructure Assignment technique (a more general definition for the pole assignment approach).

Optimal state feedback controllers, which minimize certain costs associated with control, can simultaneously provide control laws by solution of the Riccati equation, assuming that the system is controllable. One difficulty associated with this approach has its origin in the nature of the optimal control approach. By optimizing certain costs, the control designer has no direct control over the closed-loop system eigenvalues and eigenvectors (which, in our case, are directly related to the aircraft handling qualities). Furthermore, Optimal State Feedback controllers may require excessively long computations. Therefore, the eigenvalues/eigenvectors technique seems to better fit our needs of redesigning a flight control system which retains desirable flying qualities.

In the single-input case a unique feedback matrix is required to obtain desired eigenvalues. With each eigenvalue, there is an associated eigevector which is also unique (Ref. [33]). However, for the multi-input case there are an infinite number of feedback matrices which can assign specified eigenvalues. Also, with each feedback matrix, there is a new set of associated eigenvectors. Since the eigenvectors also affect the time response, it is important to assign

both the eigenvalues and the associated eigenvectors. Therefore, for the multi-input systems, there are a number of degrees of freedom given by free parameters (Ref.[34]-[36]).

A considerable amount of research effort has been devoted in the past years to investigate the relationships between the specified closed-loop eigenspace structure and the weighting matrices of the performance index of an optimal linear quadratic control for a single-input system (Ref.[37]) or, more recently, for a multiple-input system (Ref.[38], [39]).

A very interesting approach is to utilize the extra degrees of freedom for the selection of a set of eigenvectors from the allowable ones for a robust eigenspace structure assignment, where for 'robustness' the insensitivity of the desired eigenvalues to perturbations in the components of the matrices A and B is intended. Several methods have been introduced in order to realize a robust control. Remarkable theoretical results have been shown in Ref.[40]. The design of a robust eigenstructure assignment has been tried with an iterative approach, with eigenvalues assigned on the real axis only (Ref.[41]), or arbitrarily assigned in the complex plane (Ref.[42]). Furthermore, one more point which has been investigated is the coupling of a state feedback controller with a state estimation structure, in the case when not all the states are directly available for feedback (Ref.[44]).

But the robustness of the controller is not the only goal that can be achieved with the extra-degrees of freedom furnished by the analytical nature of the multi-input system control problem. Another desirable feature is to try to use these extra degrees of freedom to

obtain the most desirable response (Ref.[45]-[48]). In this case the eigenstructure assignment problem reduces itself to an optimization problem.

Feedback Structure Redesign

Our objective is to redesign the feedback structure such that desirable flying qualities and dynamic uncoupling can be achieved even with an aircraft with damaged control surfaces.

With the selection of the eigenvalues, we have essentially the possibility to assign the poles of the system. The dynamic response is also affected by the locations of the zeros and, by merely using eigenvalues, we do not have the possibility to change the zeros. From a mathematical point of view, the dynamic coupling can be seen in the values of the eigenvectors of the damaged closed-loop state matrix, which causes an interference between the longitudinal and the lateral-directional modes.

A logical solution to this problem would be to assign, in addition to the desired eigenvalues, a set of desirable eigenvectors which reflect uncoupling between these modes. In other words, it would be desirable to assign a set of eigenvectors reflecting the original, nominal, undamaged decoupled dynamics. In order to do so, the approach introduced in Ref. [45]-[48] to eliminate structural interferences of elastic modes with rigid modes is used.

For an n-th order observable and controllable system with m control inputs and l measurements available for direct output feedback, we can exactly assign MAX(m,l) eigenvalues and MIN(m,l) of the associated eigenvectors (Ref.[44]). We have an aircraft with 9 independent control

surfaces but a reduced number of control inputs are implemented in a feedback structure, that is around 3-4. Note that this implies that the matrix B_F considered in this study is only a part of the original matrix B. Therefore, we have $m < l$. If we desire to specify more than m eigenvectors, the best achievable result can be some least-squares fit to the desired eigenvectors (Ref.[45]).

Assume that $l=n=8$, that is all the states, measured by appropriate sensors, are directly available for feedback purposes. This is within our original assumption for implementing a reconfiguration strategy. Therefore, we have:

$$Z(k) = M\ X(k) \tag{5.5}$$

where M is an (8x8) identity matrix.

In order to determine K, note that the augmented closed-loop system eigenvalues (Γ_i) and eigenvectors (Ω_i) are related by the following relation:

$$(A + B_F KM)\ \Omega_i = \Gamma_i\ \Omega_i \quad \text{for } i = i,\ldots, n=8 \tag{5.6}$$

Introduce a set of m-dimensional vectors W_i:

$$W_i = KM\ \Omega_i \qquad \text{for } i = i,\ldots, n=8 \tag{5.7}$$

Given that $m < n$, the best achievable eigenvectors for each of the $l=n$ modes will be obtained by minimizing the following modes' cost function:

$$J_i = 1/2\ (\Omega_{ai} - \Omega_{di})^{*T} Q_i\ (\Omega_{ai} - \Omega_{di}) \tag{5.8}$$

where Ω_{ai} = achievable eigenvector associated with Γ_i;

Ω_{di} = desired eigenvector associated with Γ_i;

Q_i = i-th (nxn) symmetric positive semi-definite weighting matrix for the eigenvectors;

*T indicates conjugate transpose.

Such a cost function represents the error between the achievable eigenvector and some desired eigenvector, weighted by the matrix Q_i. It can be shown (Ref.[45]) that the W_i that minimizes J_i is given by:

$$W_i^{*T} = \Omega_{ai}^{*T} Q_i L_i [L_i^{*T} Q_i L_i]^{-1} \quad (5.9)$$

for i=1,...,l=n=8, where $L_i = (\Gamma_i I_n - A)^{-1} B_F$.

Note that in order for the inverse of Eq.(5.9) to exist we must have:

$$\text{Rank } [Q_i] > \text{Rank } [B_F] \quad (5.10)$$

Also we can see that such approach can be extended for synthesizing reduced order feedback control laws, depending on the dimension l of the vector Z(k). Once we have found Wi, the relation that allows us to calculate the achievable eigenvectors is given by:

$$(\Gamma_i I_n - A)\ \Omega_{ai} = B_F W_i \text{ for } i=1,...,n=8 \quad (5.11)$$

which gives:

$$\Omega_{ai} = (\Gamma_i I_n - A)^{-1}\ B_F W_i \text{ for } i=1,...,n=8 \quad (5.12)$$

Finally, the control gain matrix is given by:

$$K = W\ [MV]^{-1} \quad (5.13)$$

where W = matrix of concatinated W_i vectors = $[W_1,...,W_l]$;

V = matrix of concatinated achievable Ω_{ai} eigenvectors

= $[\Omega_{a1},....,\Omega_{al}]$.

Results of The Application of The Method To The S.A.S. Redesign For A Damaged Aircraft

The introduced method for the redesign of the feedback structure of the S.A.S. has been codified and tested by using several MATRIXx user-defined functions.

The following study has been done in the continuous-time domain.

There is no particular reason for this other than the fact that the handling qualities conditions are mostly assigned in the s-domain. A similar study can be done in the discrete-time domain. The handling qualities conditions can be obtained in the z-domain by using the relation $z = e^{sT}$, where T is the sampling period. Note that, following this approach, we should select a value of T such that all the desired pole locations are within the unit circle.

For this study we have used the previously introduced flight conditions (Altitude=40,000 ft. , Mach number=0.7); of course, the feedback structure redesign has to be performed for all the regions of the flight envelope.

In order to show the results of the application of the introduced methods the following steps have been followed:

STEP 1:

At the introduced flight conditions the design of the feedback structure is performed for the undamaged, 'naturally' unstable aircraft with the previously reported data. The selected values for the eigenvalues are taken from Ref.[13]. Note that such design could be performed separately for the longitudinal and lateral-directional dynamics because there was no coupling introduced by the values of the B matrix. Also note that only the elevators and the rudder control inputs are considered in this design. The calculated feedback gains relative to the rudder control input are almost null because the bare-air frame lateral-directional handling qualities are already acceptable. The resulting closed-loop A matrix has all the eigenvalues located in the desired positions, which means desirable handling qualities.

The resulting eigenvectors reflect a desirable uncoupling between the longitudinal and lateral-directional dynamics. Besides this, there is nothing unique in this set of eigenvectors. Ref.[48] contains several considerations regarding the selection of a desirable matrix of eigenvectors. For our purposes it is enough to achieve a dynamic uncoupling. However, particular relationships between the rigid dynamic modes may be investigated. The selected eigenvalues and the calculated eigenvectors are assumed to be our desired eigenstructure. Such eigenstructure, the associated feedback gains, the closed-loop A matrix and the B matrix are shown in Table IX .

STEP 2:

The previously calculated feedback gains are now used with the damaged, unstable aircraft with the same control inputs. Note that the rolling moment induced by the damage on the left elevator causes a certain amount of coupling. This can be seen in the values of the closed-loop A matrix as well as in the eigenvectors shown in Table X. Also, the eigenvalues reported in the same table show some remarkable differences with respect to the ones shown in Table IX. Particularly from Table X, we see that the phugoid has become unstable and that the short period characteristics are not satisfactory. This proves the need for a complete redesign of the feedback structure. The resulting eigenstructure, the closed-loop A matrix, the B matrix and the feedback gains are shown in Table X.

STEP 3:

Finally, given our desired eigenstructure and given the data for the damaged unstable aircraft, the redesign of the feedback

structure is now done. From what we have said in the introduction of the method recall that we can assign precisely MAX(m,l) desired eigenvalues and MIN(m,l) desired eigenvectors. We have selected l=n=8. So, by increasing the value of m, we can achieve closed-loop dynamics whose eigenvectors are closer to the desired set of eigenvectors; we can still assign l=n=8 desired eigenvalues. Also, we should achieve a closed-loop A matrix similar to the one shown in Table IX.

On the other side, for simplicity purposes, we do not want to introduce in the feedback structure too many deflection inputs, because our goal is only to remove the coupling between the longitudinal and the lateral-directional dynamics. Of course, in the selection of the feedback control inputs, we have excluded the damaged left elevator. By using m=3 control inputs, that is δ_{ER},δ_{CL} and δ_R, we already achieve our goals, as shown in Table XI, where the resulting eigenvalues, eigenvectors, closed-loop A matrix, B matrix and feedback structure are shown. Even better results are obtained, as expected, for m=4 control inputs, that is δ_{ER},δ_{CL},δ_{CR},δ_R. Table XII shows the resulting eigenvalues, eigenvectors, closed-loop A matrix, B matrix and feedback structure. Note that the absolute value of the feedback gains are lower than the ones shown in Table XI, which means that the task of removing the coupling has been more widely spread among the deflection inputs. Also, in Table XII, we can see that the achieved closed-loop A matrix, besides having all the desired eigenvalues, is almost coincident to the one relative to undamaged conditions shown in Table IX.

State Estimation with Eigenstructure Assignment

At this point go back to our original assumption that all the states are directly available for feedback, which means that l=n=8; so, we can assign 8 eigenvalues. When this assumption is no longer valid, we would have $l < n = 8$ and we would be able to assign a reduced number of eigenvalues. This can be a serious problem, since we have no control of all the eigenvalues, some can take on unstable values, like positive values in the s-domain.

In order to assign all the n eigenvalues, an effective approach would be to introduce an Observer or a Kalman Filter to estimate the unmeasured states. However, two points need to be considered when state estimation is coupled with eigenstructure assignment. They are the effects on the transient response due to a pilot input and the effects on the controller dynamics itself.

The effects on the aircraft transient response due to a pilot input can be analyzed by considering the augmented system transfer functions. Given the general state variable form of a continuous-time system:

$$\dot{X}(t) = A\,X(t) + B\,U(t) + L\,W(t) \tag{5.14}$$

$$Z(t) = M\,X(t) + V(t) \tag{5.15}$$

with

$$Y(t) = C\,X(t) \tag{5.16}$$

$$U(t) = U_F(t) + U_p(t) \tag{5.17}$$

$$U_F(t) = K\,\hat{X}(t) \tag{5.18}$$

$$\dot{\hat{X}}(t) = A\,\hat{X}(t) + B\,U(t) + F\,(Z(t) - M\,\hat{X}(t)) \tag{5.19}$$

where F is a matrix of estimator gains. Using the state estimation error:

$$e(t) = X(t) - \hat{X}(t) \tag{5.20}$$

the correspondent augmented system can be written as:

$$\begin{bmatrix}\dot{X}(t)\\ \dot{e}(t)\end{bmatrix} = \begin{bmatrix}(A + B_F K) & -B_F K\\ 0 & (A - FM)\end{bmatrix} \begin{bmatrix}X(t)\\ e(t)\end{bmatrix} +$$

$$\begin{bmatrix}B_P\\ 0\end{bmatrix} U_P(t) + \begin{bmatrix}L & 0\\ 0 & -F\end{bmatrix} \begin{bmatrix}W(t)\\ V(t)\end{bmatrix} \qquad (5.21)$$

Taking the Laplace transform of Eq. (5.21) we have:

$$\begin{bmatrix}X(s)\\ e(s)\end{bmatrix} = \begin{bmatrix}(sI_n - (A + B_F K)) & B_F K\\ 0 & (sI_n - (A - FM))\end{bmatrix}^{-1} *$$

$$\begin{bmatrix}B_P\\ 0\end{bmatrix} U_P(s) + \begin{bmatrix}L & 0\\ 0 & -F\end{bmatrix} \begin{bmatrix}W(s)\\ V(s)\end{bmatrix} \qquad (5.22)$$

$$\text{with } Y(s) = [C \;\; 0] \begin{bmatrix}X(s)\\ e(s)\end{bmatrix} \qquad (5.23)$$

From Eq. (5.23) we can see that the transfer functions between Y(s)and $U_P(s)$ are given by:

$$Y(s) = C\,[sI_n - (A + B_F k)]^{-1} B_P\, U_P(s) \qquad (5.24)$$

which is the same as it would be with a feedback structure but without state estimation. Therefore, with the redesigned feedback structure, as implemented, the closed-loop system transient response due to any pilot input is independent of the state estimator dynamics, regardless if we implement an Observer or a Kalman Filter.

On the other side, the state estimator will affect the controller dynamics. In order to show this relation, recall that, from Eq. (5.19),

we have:

$$\hat{X}(s) = (sI_n - A - B_F K + FM)^{-1}(B_p U_p(s) + F Z(s)) \qquad (5.25)$$

which will change Eq. (5.18) to:

$$U_F(s) = K [sI_n - A_{con}]^{-1} (B_p U_p(s) + F Z(s)) \qquad (5.26)$$

where $A_{con} = A + B_F K - FM$ (5.27)

Therefore, the estimator gain matrix F should also be selected to avoid undesirable controller characteristics, such as unstable eigenvalues of the matrix A_{con}. It should not be difficult to realize that this condition is not very restrictive. In other words, if the matrix (A - FM) has to have stable eigenvalues, regardless if F comes from an Observer or a Kalman Filter, most likely the matrix ($A + B_F K - FM$) also should have stable eigenvalues. However, a full address of this problem has not been considered in this study.

Conclusions

In this chapter a particular approach has been introduced in order to redesign the feedback structure of the flight control system to accomodate the changed dynamics due to damage.

The approach is based on Eigenstructure assignment. Through such assignment we would like to have closed-loop aircraft dynamics with desirable handling qualities and without the longitudinal and lateral-directional coupling induced by the damage. Such desirable uncoupling can be assigned with a particular eigenvectors matrix where the longitudinal modes are not influenced by the lateral modes and vice versa, which means that the elements of the upper right and lower left blocks of the desired eigenvectors matrix are zeros.

The introduced method allows us to assign MAX(m,1) eigenvalues and

MIN(m,l) eigenvectors. If we want to assign the overall eigenstructure, the best achievable result is a least squares fit to the desired eigenvectors (Ref.[45]).

For l=n=8, the result is that the desirable uncoupling is achieved with the implementation of a limited number of control inputs in the feedback structure (m = 3 or 4). The closed-loop characteristics to be used for the construction of the mathematical model of the damaged aircraft can, therefore, be updated. The resulting closed-loop A matrix, besides having the eigenvalues coincident with the desirable eigenvalues, is very similar to the closed-loop A matrix of the undamaged aircraft. Such feedback structure redesign has to be achieved on-line by the airborne computer for all the flight regions of the flight envelope. The difference with the previous reconfiguration tasks is that, for this particular phase, time is not a crucial factor. In fact, while a computational line is redesigning the feedack structure, on another parallel line the algorithm introduced in Chapter IV is taking care of the reconfiguration. Also, if not all the states are available for feedback purposes (l<n), the effects of adding a state estimation structure on the feedback structure have been illustrated.

As final comment, it should be outlined that, one more time, $MATRIX_X$ has shown to be an invaluable tool for computational tasks involving heavy matricial manipulations.

CHAPTER VI

CONCLUDING REMARKS

Summary , Conclusions and Recomendations

In this study the overall problem of flight control system reconfiguration following a damage and/or a generic failure on a control surface has been considered. Given that the problem has been broken down into the tasks shown in Fig.2, new approaches have been proposed for:

Damaged model estimation task;

Reconfiguration law design task;

Redesign of the feedback structure task.

The main characteristic of the approach used in this study is that the model estimation task and the control task are totally separated. The reason is that the presence of noises, both atmospheric turbulence and measurement errors, do not allow a successful implementation of the reconfiguration algorithm without a previous model estimation due to resulting unacceptable control activities. Therefore, the noise problem is faced in the model estimation task; the result is a deterministic damaged aircraft mathematical model whose dynamics tend to match, using the reconfiguration algorithm, the dynamics of the nominal aircraft mathematical model, also deterministic, built with computer stored data. Furthermore, a method is proposed to redesign the feedback structure.

For the damaged model estimation task, there is a unique characteristic in the introduced approach. We formulated the problem from two different perspectives. One from an aerodynamic point of view and one from an estimation point of view. More precisely, the aerodynamics furnish precious hints to the estimation part on how the system has changed following the damage. Furthermore, the Multiple Model Kalman Filters structure has shown to be a remarkable robust environment for the model estimation process.

For the reconfiguration law design task, the introduced algorithm can be classified as a 2 instants delay matching technique, potentially able to handle nonlinear problems, with an enbodied method to distribute the reconfiguration task among all the remaining healthy control surfaces.

For the redesign of the feedback structure, the main idea behind the introduced approach is to use the extra degrees of freedom furnished by the analytical nature of the control of a multi-input linear time-invariant system problem in order to restore desirable closed-loop handling qualities and to remove the damage generated dynamic coupling. For the considered aircraft model, such goal is easily achieved with a not too complex feedback structure (m=3 or m=4).

At this point it would be proper to include a critical analysis of the work presented in this study and to outline some limitations of the approaches used. A certain number of comments should be made in order to justify some assumptions, or some real life factors, which may eventually play a rule in the reconfiguration problem. Therefore, review the assumptions made and recall what actions should be taken if they are no longer valid.

First, we assumed the reconfiguration problem to be solved for a rigid aircraft. This is because the most likely aircraft for a reconfiguration strategy are fighters, which given their size and weight distribution, can be reasonably approximated as rigid bodies. Of course, the rigid body assumption can not longer be valid for large size vehicles, like bombers or transport aircraft. For these the state variables vector needs to include a certain number of flexible modes, causing an increase in the system order and an increase in computational time. Note that, eventually the damage generated interference between rigid and elastic modes may represent a serious threat to the success of the reconfiguration.

Next, in constructing the aircraft mathematical model, no attempt has been made to model the actuator dynamics. This has only been done for simplicity purposes and the results do not loose generality. A mathematical model including the actuator dynamics would have an order higher than the one used in this study and higher computational times but will add no conceptual difficulty.

In Chapter III and IV no S.A.S. or C.C.V. functions were considered to be implemented on the aircraft. This assumption was made for simplicity purposes and caused no loss of generality. Fortunally it is the real life implementation of these systems that makes reconfiguration possible. Thanks to these closed-loop systems (Stability Augmentation Systems, Dampers, Gust and Load Alleviation Systems, Flight Envelope Limiting Systems, Ride Quality Control Systems ...), the validity of the linearity assumption is saved for the case when the damaged control surface jams to the minimum and maximum deflection. This is extremely important to us because the damaged model estimation via Multiple Model

Kalman Filters is based on the linear model assumption.

A possible limitation of this approach, particularly related to the damaged model estimation part, is that the software needs a data base which is aircraft dependent. This is because the closed form expressions for the stability derivatives as a function of the normal force coefficient for each control surface depend on the aircraft aerodynamics. Also the computer stored data should include the weight, inertial and geometric characteristics of the considered aircraft. However, this is not major problem.

Another possible limitation of the approach is it does not consider the possibility of the occurrence of a damage simultaneously on more than one control surface.

However, we are still able to handle multiple damage as long as it occurs with a certain time difference because the aircraft data would be updated after each damaged model estimation. Of course, the probability to achieve a full and accurate reconfiguration decreases with the number of damaged control surfaces.

In this study we have reported different steps of the overall reconfiguration process following a damage on the left elevator, which along with the right elevator, are the main longitudinal control surfaces. We have not considered any damage and/or generic failure on a lateral-directional control surface, that is rudder and ailerons.

The reason for this is that, by analyzing a reconfiguration following a damage on an elevator surface, we have considered the most conservative case. In fact, in aircraft dynamics, a damage on a longitudinal control surface involves more aerodynamic coupling with the lateral directional dynamics than a damage on a lateral-directional

control surface involves with the longitudinal dynamics.

The physical explanation for this is that the point of application of the normal force exerted by the ailerons is usually located closer to the center of gravity location along the mean aerodynamic chord, such that very little pitching moment is exerted. The aerodynamic force generated by the rudder has a null component along the normal direction which means a null pitching moment. Note that a reconfiguration following a damage on the rudder cannot be attempted if on the aircraft there is not at least one control surface, other than the rudder, with control authority around the yaw axis. The most suitable solution would be a pair of canards with high dihedral angle.

From a mathematical point of view a damage on a lateral-directional control surface implies a small (or almost null, especially for a damage on the ailerons) changes in the elements of the A matrix. On the other hand, for a damage on the longitudinal control surface, we have shown quite remarkable changes in some of the elements of the A matrix. However, the introduced approaches for the different reconfiguration steps still hold.

One more point that needs to be mentioned is that the probability for a successful reconfiguration with the introduced approaches is strongly dependent on the available computational power and speed. We have discussed this aspect in Chapter III, where the computer speed affects the selection of the number of models N and in Chapter IV, where the computer speed affects the selection of the reconfiguration step h. However, this is true for all other reconfiguration methods. The reconfiguration needs to be sized on the computer performance; a slow flight computer may hurt probabilities of success, expecially when the

damaged aircraft dynamics turns out to be unstable.

The author is not aware of the current status of the art of flight computers. It is well known that their performance improves drastically year by year. From the studies reported in Ref. ([15],[16],[20],[21], [22]) the proposed approaches seem to be within today's computer capabilities.

The main advantage of the reconfiguration algorithm introduced in Chapter IV is that it considers the real life occurrence that, following the damage on the control surface, not only the elements of the B matrix but also some elements of the A matrix change and this is particularly true for a damage on the longitudinal control surfaces. With the approach introduced in Chapter V we have shown that it is possible to achieve a closed-loop A matrix almost coincident to the one relative to the undamaged conditions. Thus, there is no longer going to be a real need for an accurate but more sophisticated algorithm like the one introduced in Chapter IV. Instead, it would be more appropriate to use simpler approaches like the Control Mixer Concept using a Pseudo Inverse Technique (Ref.[10],[11]), which assumes the state matrix A being the same for nominal and damaged conditions. Now, if we had an extremely fast and powerful flight computer, the sequence of the reconfiguration tasks introduced in Fig. 2 may be modified such that the redesign of the feedback structure is attempted immediately after the damaged model estimation. Then the Control Mixer Concept takes the place of the reconfiguration algorithm introduced in Chapter IV. However, this suggestion is valid only if the redesign of the feedback structure may be accomplished in a very short amount of time, in the order of a fraction of a second. Otherwise, the advantages of implementing a

simpler reconfiguration algorithm are not worth the trouble of delaying the reconfiguration itself.

This work does not claim to bring revolutionary trends in the flight control reconfiguration problem. The main concept introduced here is that we should not look at this problem only as a control theory problem. It is the author's personal belief that, given the complexity of the aircraft system, given the extremely wide ranges of flight conditions and associated noises, given the unpredictable number of potential failures or damages, one fixed controller structure robust enough to any change in the dynamics due to a damage on any control surface is a fiction and it always will be so.

On the other hand, it does not make sense, on the today sophisticated flying machines, to implement only a few precomputed procedures for emergency conditions; for example, implementing effective procedures for handling an engine failure during takeoff. But, from an aerodynamic and flight dynamics perspective, we may have a view point that, combined with the tools offered by control theory, may guarantee us a greater flexibility and adaptability.

As final comment, the author hopes that the overall relative simplicity of the damaged model estimation and feedback structure redesign methods, along with the accuracy and efficiency of the reconfiguration algorithm, may be the key factors for successful real life on-line implementation of these approaches to the aircraft flight control reconfiguration problem.

SELECTED BIBLIOGRAPHY

1. Robinson, A.C., "Totally Robust Control - A New Concept for Design of Flight Control Systems", AIAA Paper 85-1974, Proceedings of the Guidance and Control Conference, Snowmass, CO, Aug. 1985, pp. 741-745.

2. Handelman, D.A. and Stengel, R.F., "A Theory for Fault Tolerant Flight Control Combining Expert System and Analytical Redundancy Concepts", AIAA Paper 86-2092, Proceedings of the Guidance and Control Conference, Williamsburg, VA, Aug. 1986, pp. 375-384.

3. D'Azzo, J.J. and Eslinger, R.A., "Multivariable Control Law Design for the AFTI/F-16 with a Failed Control Surface", Proceedings of the NAECON Conference, Dayton, OH, May 1985, pp. 453-456.

4. Horowitz, I., Arnold, P.B. and Houpis, C.H., "YF-16 CCV Flight Control System Reconfiguration Design Using Quantitative Feedback Theory", Proceedings of the NAECON Conference, Dayton, OH, May 1985, pp. 578-585.

5. Lear Siegler, Inc., Astronics Div., "Reconfiguration Strategies for Aircraft with Flight Control System Subjected to Actuator Failure/Surface Damage", AFWAL-TR86-3110, WPAFB, OH, March 1987.

6. "Multivariable Control Systems", AFWAL-TR-83-3093, OH, Sept. 1983.

7. Looze, D.P., Weiss, J.L., Eterno, J.S. and Barrett, N.M., "An Automatic Redesign Approach for Restructurable Control Systems", Proceedings of the NAECON Conference, Dayton, OH, May 1985, pp. 570-576.

8. Weiss, J.L., Looze, D.P. and Eterno, J.S., "Simulation results of Automatic Restructurable Flight Control System Concept", AIAA Paper 86-2032, Proceedings of the Guidance and Control Conference, williamsburg, VA, Aug. 1986, pp. 190-197.

9. Eterno, J.S. and Weiss, J.L., "Reconfigurable Multivariable Control Law for a Commercial Airplane Using a Direct Digital Output Feedback Design", NASA-TM-85759, Oct. 1985.

10. Rattan, K.S., "Evaluation of a Control Mixer Concept for Reconfiguration of a Flight Control System", Proceedings of the NAECON Conference, Dayton, OH, May 1985, pp. 560-569.

11. Raza, S.J. and Silverthorn, J.T., "Use of the PseudoInverse for Design of a Reconfigurable Flight Control System", AIAA Paper 85-1900, Proceedings of the Guidance and Control Conference, Snowmass, CO, Aug. 1985, pp. 349-356.

12. Weinstein, W., Posingies, W., Eslinger, R.A. and Gross, H.N., "Control Reconfigurable Combat Aircraft - Flight Control System Development", AIAA Paper 86-2236, Proceedings of the Guidance and Control Conference, Williamsburg, VA, Aug. 1986, pp. 772-783.

13. Roskam,J., Flight Dynamics of Rigid and Elastic Airplanes - Part 1, Roskam Aviation and Enginnering Corporation, 1982.

14. Montgomery, R.C. and Caglayan, A.K. "Failure Accomodation in Digital Flight Control Systems by Bayesian Decision Theory", Journal of Aircraft, Vol. 13, No. 2, Feb. 1976, pp.69-75.

15. Athans, M.,Castanov, D.,Dunn, K.P., Greene, C.S. Lee, W.H., Sandell, N.R. Jr. and Willsky, A.S. "The Stochastic Control of the F-8C Aircraft Using a Multiple Model Adaptive Control (MMAC) Method-Part I : Equilibrium Flight", IEEE Transactions on Automatic Control, Vol. AC-22, No. 5, Oct. 1977, pp. 768-780.

16. Hagan, M.T. and Goodner, C.E. "Aircraft Model-Follower Design : Phase V", School of Electrical and Computer Engineering, Oklahoma State University (prepared for Flight Controls, Boeing Military Airplane Co., Wichita, KS), Jan. 1988.

17. Roskam, J., Methods for Estimating Stability and Control Derivatives of Conventional Subsonic Airplanes, Roskam Aviation and Engineering Corporation, 1973.

18. Hoak, D.E., Ellison, D.E. et al, USAF Stability and Control Datcom, Flight Control Division, Air Force Flight Dynamics Laboratory, Wright-Patterson Air Force Base, OH.

19. "Control Reconfigurable Combat Aircraft Developement", AFWAL-TR-87-3011, WPAFB, OH, May 1987.

20. Kaufman, W.G.H. and Roy, R. "Multiple Model Adaptive Control Procedure for Blood Pressure Control", IEEE Transactions on Biomedical Engineering, Vol. BME-33, No. 1, Jan. 1986.

21. Willsky, A.S. "Failure Detection in Dynamic Systems" from AGARD-LS-109 "Fault Tolerance Design and Redundancy Management Techniques", 1980.

22. Cunningham, T.P. "Failure Management Techniques for High Survivability" from AGARD-LS-109 "Fault Tolerance Design and Redundancy Management Techniques", 1980.

23. Mendel, J.M., Lessons in Digital Estimation Theory, Prentice Hall, Inc., 1986.

24. Gelb, A., Applied Optimal Estimation, M.I.T. Press, 1974.

25. Jazwinski, A.H., Stochastic Processes and Filtering Theory, Academic Press, 1970.

26. Ogata, K., Discrete-Time Control Systems, Prentice Hall, Inc., 1987.

27. "Flight in Turbulence", AGARD-CP-140, 1973.

28. Reeves, P.M., Campbell, G.S., Ganzer, V.M. and Joppa, R.G. "Development and Application of a Non-Gaussian Atmospheric Turbulence Model for Use in Flight Simulators", NASA-CR-2451, Sept. 1974.

29. Roskam,J., Flight Dynamics of Rigid and Elastic Airplanes - Part 2, Roskam Aviation and Enginnering Corporation, 1982.

30. Dutton, J.A., et al. "Statistical Properties of Turbulence for Aerospace Vehicles Design", NASA-CR-1889, Aug. 1971.

31. Napolitano, M. "C.C.V. (Control Configured Vehicle)", MAE 5010.2 "Guidance and Control of Aerospace Vehicles" Final Report, Oklahoma State University, Spring 1988.

32. "Digital Flight Control Systems for Tactical Fighters", AFFDL-TR-74-69, OH, July 1974.

33. Mayne, D.Q. and Murdoch, P. "Modal Control of Linear Time Invariant Systems", International Journal of Control, Vol. 11, No.2, 1970, pp. 223-227.

34. Moore, B.C. "On the Flexibility Offered by State Feedback in Multi-Variable Systems Beyond Closed Loop Eigenvalues Assignment", IEEE Transactions on Automatic Control, Oct. 1986, pp.682-691.

35. D' Azzo, J.J. and Houpis, C.H., Linear Control System Analysis and Design, McGraw-Hill, 1981.

36. Porter, B. and D'Azzo, J.J. "Algorithm for Closed-Loop Eigenstructure Assignment by State Feedback in MultiVariable Linear Systems", International Journal of Control, Vol.27, No.3, 1978, pp. 487-492.

37. Fallside, F. and Seraij, H. "Direct Design Procedure for Single Input Feedback Systems", International Journal of Control, Vol. 32, 1971, pp. 95-106.

38. Harvey, C.A. and Stein, G. "Quadratic Weights for Asymptotic Regulator Properties", IEEE Transactions on Automatic Control, Vol. AC-23, No. 3, June 1978.

39. Stein, G. "Generalized Quadratic Weights for Asymptotic Regulator Properties", IEEE Transactions on Automatic Control, Vol. AC-24, No. 4, Aug. 1979.

40. Kautsky, J., Nichols, N.K. and Van Dooren, P. "Robust Pole Assignment in Linear State Feedback", International Journal of Control, Vol. 41, 1985, pp. 1128-1138.

41. Arbel, A. and Rath, O. "An Iterative Pole Placement Algorithm", International Journal of Control, Vol. 41, 1985, pp. 929-939.

42. Hassan, M.M. and Amin, M.H. "Recursive Eigenstructure Assignment Linear Systems", International Journal of Control, Vol. 45, 1987, pp. 291-310.

43. Chu, K.E. "A Pole Assignment Problem for 2-Dimensional Linear Discrete Systems", International Journal of Control, Vol. 43, 1985, pp. 957-964.

44. Srinathkumar, S. "Eigenvalue/Eigenvector Assignment Using Output Feedback", IEEE Transactions on Automatic Control, Vol. AC-23, No. 1, Feb. 1978.

45. Cunningham, T.B. "Eigenspace Selection Procedures for Closed-Loop Response Shaping with Modal Control", Proceedings of the IEEE Conference on Decision and Control, Albuquerque, NM, Dec. 1980.

46. Garrard, W. and Liebst, B. "Active Flutter Suppression Using Eigenspace and Linear Quadratic Design Techniques", AIAA Paper 83-2222, Proceedings of the Guidance and Control Conference, Gatlinburg, Tennessee, Aug. 1983.

47. Andry, A.N., Shapiro, E.Y. and Chung, J.C. "On Eigenstructure Assignment for Linear Systems", IEEE Transactions on Aerospace and Electronics Systems, Vol. AES-19, No. 5, Sept. 1983.

48. Schmidt, D.K. and Davidson, J.B. "Flight Control Law Synthesis for an Elastic Vehicle By Eigenspace Assignment", AIAA Paper 85-1898, Proceedings of the Guidance, Navigation and Control Conference, Snowmass, CO, Aug. 1985.

APPENDICES

APPENDIX A

TABLES

TABLE I

FLIGHT CONDITIONS, AERODYNAMIC, INERTIAL, GEOMETRIC DATA

VEL = 675.0 ft/sec^{-1}

MACH = 0.7

h = 40,000.0 ft

ρ = 0.000588 $slug\ ft^{-3}$

W = 13,000.0 lbs

I_{xx} = 28,800.0 $slug\ ft^2$

I_{yy} = 18,800.0 $slug\ ft^2$

I_{zz} = 47,000.0 $slug\ ft^2$

I_{xz} = 1,350.0 $slug\ ft^2$

c = 7.04 ft

b = 34.2 ft

S = 232.0 ft^2

$X_{c.g.}$ = 0.315

α_s = 2.7 deg
where 's' stands for 'stability axes'

$C_{D_u} = 0.0$,	$C_{D_1} = 0.0330$,	$C_{T_u} = 0.0$,	$C_{T_1} = 0.0330$
$C_{D_\alpha} = 0.3$,	$C_{D_{\delta EL}} = 0.0$,	$C_{D_{\delta ER}} = 0.0$,	$C_{D_{\delta CL}} = 0.0$
$C_{D_{\delta CR}} = 0.0$,	$C_{D_{\delta SL}} = 0.0$,	$C_{D_{\delta SR}} = 0.0$	
$C_{L_u} = 0.4$,	$C_{L_1} = 0.410$,	$C_{L_\alpha} = 5.9$,	$C_{L_{\dot\alpha}} = 2.20$
$C_{L_q} = 6.11$,	$C_{L_{\delta EL}} = 0.276$,	$C_{L_{\delta ER}} = 0.276$,	$C_{L_{\delta CL}} = 0.2$
$C_{L_{\delta CR}} = 0.2$,	$C_{L_{\delta SL}} = 0.15$,	$C_{L_{\delta SR}} = 0.15$	
$C_{m_u} = 0.050$,	$C_{m_1} = 0.007$,	$C_{m_{Tu}} = -0.0034$	$C_{m_{T1}} = -0.007$
$C_{m_\alpha} = -0.75$,	$C_{m_{T\alpha}} = 0.0$,	$C_{m_{\dot\alpha}} = -6.7$	$C_{m_q} = -20.15$

$C_{m_{\delta EL}} = -0.72$, $C_{m_{\delta ER}} = -0.72$ $C_{m_{\delta CL}} = -0.35$ $C_{m_{\delta CR}} = -0.35$

$C_{m_{\delta SL}} = -0.15$ $C_{m_{\delta SR}} = -0.15$

$C_{Y_\beta} = -0.730$, $C_{Y_p} = 0.0$, $C_{Y_r} = 0.4$, $C_{Y_{\delta AL}} = 0.0$

$C_{Y_{\delta AR}} = 0.0$, $C_{Y_{\delta R}} = 0.138$

$C_{l_\beta} = -0.132$, $C_{l_p} = -0.45$, $C_{l_r} = 0.163$, $C_{l_{\delta AL}} = 0.089$

$C_{l_{\delta AR}} = 0.089$, $C_{l_{\delta R}} = 0.0172$

$C_{n_\beta} = 0.127$, $C_{n_{T\beta}} = 0.0$, $C_{n_p} = 0.008$, $C_{n_r} = -0.2412$

$C_{n_{\delta AL}} = -0.0086$, $C_{n_{\delta AR}} = -0.0086$, $C_{n_{\delta R}} = -0.0747$

TABLE II

CONTINUOUS-TIME STATE VARIABLE MODEL OF THE AIRCRAFT FOR UNDAMAGED NOMINAL CONDITIONS

$$\begin{bmatrix} \dot{\alpha} \\ \dot{q} \\ \dot{u} \\ \dot{\theta} \\ \dot{\beta} \\ \dot{p} \\ \dot{r} \\ \dot{\Phi} \end{bmatrix} = \begin{bmatrix} -0.6752 & 0.9951 & -0.0002 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ -8.4535 & -1.6274 & 0.0009 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 8.4608 & 0.0 & 0.0 & -32.1756 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 1.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & -0.0832 & 0.0 & -0.9988 & 0.0477 \\ 0.0 & 0.0 & 0.0 & 0.0 & -4.9803 & -0.4369 & 0.1550 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 2.8191 & 0.0004 & -0.1365 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 1.0 & 0.0 & 0.0 \end{bmatrix} * \begin{bmatrix} \alpha \\ q \\ u \\ \theta \\ \beta \\ p \\ r \\ \Phi \end{bmatrix} +$$

A_c matrix (8 x 8)

$$\begin{bmatrix} -0.032 & -0.032 & -0.017 & -0.017 & -0.023 & -0.023 & 0.0 & 0.0 & 0.0 \\ -8.838 & -8.838 & -1.742 & -1.742 & 4.068 & 4.068 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.017 \\ 3.387 & -3.387 & 3.084 & -3.084 & 2.894 & -2.894 & 3.386 & 3.386 & 0.628 \\ 0.032 & -0.032 & 0.03 & -0.03 & 0.028 & -0.028 & -0.162 & -0.162 & -1.68 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \end{bmatrix} * \begin{bmatrix} \delta_{EL} \\ \delta_{ER} \\ \delta_{SL} \\ \delta_{SR} \\ \delta_{CL} \\ \delta_{CR} \\ \delta_{AL} \\ \delta_{AR} \\ \delta_{R} \end{bmatrix}$$

B_c matrix (8 x 9)

TABLE III

MATRICES OF THE DISCRETE-TIME STATE VARIABLE MODEL OF THE AIRCRAFT FOR UNDAMAGED NOMINAL CONDITIONS

$$A = \begin{bmatrix} 0.9929 & 0.0098 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ -0.0836 & 0.9834 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0844 & -0.0012 & 1.0 & -0.3218 & 0.0 & 0.0 & 0.0 & 0.0 \\ -0.0004 & 0.0099 & 0.0 & 1.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.999 & 0.0 & -0.01 & 0.0005 \\ 0.0 & 0.0 & 0.0 & 0.0 & -0.0496 & 0.9956 & 0.0018 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0282 & 0.0 & 0.8985 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & -0.0002 & 0.01 & 0.0 & 1.0 \end{bmatrix}$$

$$B = \begin{bmatrix} -0.0008 & -0.0008 & -0.0003 & -0.0003 & -0.0004 & -0.0004 & 0.0 & 0.0 & 0.0 \\ -0.0876 & -0.0876 & -0.0173 & -0.0173 & -0.0403 & -0.0403 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ -0.0004 & -0.0004 & -0.0001 & -0.0001 & -0.0002 & -0.0002 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0003 \\ 0.034 & -0.034 & 0.031 & -0.031 & 0.029 & -0.029 & 0.0338 & 0.0338 & 0.0062 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & -0.0016 & -0.0016 & -0.0168 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0002 & 0.0002 & 0.0 \end{bmatrix}$$

$$L = \begin{bmatrix} 0.9929 & 0.0 \\ -0.0836 & 0.0 \\ 0.0844 & 0.0 \\ -0.0004 & 0.0 \\ 0.0 & 0.9990 \\ 0.0 & -0.0496 \\ 0.0 & 0.0282 \\ 0.0 & -0.0002 \end{bmatrix}$$

$$C = \begin{bmatrix} 1.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 1.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 1.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 1.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 1.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 1.0 & 0.0 \end{bmatrix}$$

$$Q = \begin{bmatrix} 0.0005 & 0.0 \\ 0.0 & 0.0005 \end{bmatrix}$$

$$R = \begin{bmatrix} 0.0006 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0015 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.1000 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0006 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0245 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0015 \end{bmatrix}$$

TABLE IV

DISCRETE-TIME A AND B MATRICES OF THE STATE VARIABLE OF THE AIRCRAFT FOLLOWING THE DAMAGE

$$A = \begin{bmatrix} 0.9933 & 0.0099 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ -0.0348 & 0.9875 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0843 & -0.0012 & 1.0 & -0.3218 & 0.0 & 0.0 & 0.0 & 0.0 \\ -0.0002 & 0.0099 & 0.0 & 1.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.999 & 0.0 & -0.01 & 0.0005 \\ 0.0 & 0.0 & 0.0 & 0.0 & -0.0496 & 0.9956 & 0.0018 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0282 & 0.0 & 0.8985 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & -0.0002 & 0.01 & 0.0 & 1.0 \end{bmatrix}$$

$$B = \begin{bmatrix} -0.0003 & -0.0008 & -0.0003 & -0.0003 & -0.0004 & -0.0004 & 0.0 & 0.0 & 0.0 \\ -0.0379 & -0.0876 & -0.0173 & -0.0173 & -0.0403 & -0.0403 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ -0.0002 & -0.0004 & -0.0001 & -0.0001 & -0.0002 & -0.0002 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0003 \\ 0.015 & -0.034 & 0.031 & -0.031 & 0.029 & -0.029 & 0.0338 & 0.0338 & 0.0062 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & -0.0016 & -0.0016 & -0.0168 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0002 & 0.0002 & 0.0 \end{bmatrix}$$

TABLE V

DISCRETIZED VALUES OF THE NORMAL FORCE COEFFICIENT OF THE LEFT ELEVATOR RELATIVE TO N=12 AND N=23

$(C_{N_{\delta EL}})$ after the damage = 0.121

	N = 12		N = 23
	1 - 0.0		1 - 0.0
			2 - 0.0125
	2 - 0.0251		3 - 0.0251
			4 - 0.0376
	3 - 0.0502		5 - 0.0502
			6 - 0.0627
	4 - 0.0753		7 - 0.0753
			8 - 0.0878
	5 - 0.1004		9 - 0.1004
0.121 ->		0.121 ->	10 - 0.1129
	6 - 0.1254		11 - 0.1254
			12 - 0.1379
	7 - 0.1505		13 - 0.1505
			14 - 0.1630
	8 - 0.1756		15 - 0.1756
			16 - 0.1881
	9 - 0.2007		17 - 0.2007
			18 - 0.2132
	10 - 0.2258		19 - 0.2258
			20 - 0.2383
	11 - 0.2509		21 - 0.2509
			22 - 0.2634
	12 - 0.2760		23 - 0.2760

TABLE VI

INCREASED ORDER STATE VARIABLE MODEL FOR
MODELING ATMOSPHERIC TURBULENCE
WITH AR(1) PROCESSES
$(\alpha_w(k+1), \beta_w(k+1))$

$$
\begin{bmatrix} \alpha(k+1) \\ q(k+1) \\ u(k+1) \\ \theta(k+1) \\ \beta(k+1) \\ p(k+1) \\ r(k+1) \\ \Phi(k+1) \\ \alpha_w(k+1) \\ w(k+1) \end{bmatrix}
=
\left[\begin{array}{ccc|cc}
 & & & & \\
 & & & & \\
 & & & & \\
 & A & & & L \\
 & & & & \\
 & & & & \\
 & & & & \\
\hline
0.0 & \cdots & 0.0 & 0.5 & 0.0 \\
0.0 & \cdots & 0.0 & 0.0 & 0.5
\end{array}\right]
\begin{bmatrix} \alpha(k) \\ q(k) \\ u(k) \\ \theta(k) \\ \beta(k) \\ p(k) \\ r(k) \\ \Phi(k) \\ \alpha_w(k) \\ \beta_w(k) \end{bmatrix}
+
$$

$$\begin{bmatrix} 0.0 & | & 0.0 \\ \vdots & | & \vdots \\ 0.0 & | & 0.0 \\ \hline 1.0 & | & 0.0 \\ 0.0 & | & 1.0 \end{bmatrix} \begin{bmatrix} e_\alpha(k) \\ e_\beta(k) \end{bmatrix} + \begin{bmatrix} & & \\ & B & \\ & & \\ \hline 0.0 & \cdots & 0.0 \\ 0.0 & \cdots & 0.0 \end{bmatrix} \begin{bmatrix} \delta_{EL}(k) \\ \delta_{ER}(k) \\ \delta_L(k) \\ \delta_{SR}(k) \\ \delta_{CL}(k) \\ \delta_{CR}(k) \\ \delta_{AL}(k) \\ \delta_{AR}(k) \\ \delta_R(k) \end{bmatrix}$$

$$Z(k) = \begin{bmatrix} & & | & 0.0 & 0.0 \\ & & | & \vdots & \vdots \\ & C & | & \vdots & \vdots \\ & & | & \vdots & \vdots \\ & & | & 0.0 & 0.0 \end{bmatrix} \begin{bmatrix} \alpha(k) \\ q(k) \\ u(k) \\ \theta(k) \\ \beta(k) \\ p(k) \\ r(k) \\ \Phi(k) \\ \alpha_W(k) \\ \beta_W(k) \end{bmatrix} + V(k)$$

TABLE VII

COMPARISON BETWEEN S^{-1}_{UNC} (INVERSE OF THE RESIDUAL COVARIANCE MATRIX FOR UNCORRELATED TURBULENCE CORRECTLY MODELED) AND S^{-1}_{COR} (INVERSE OF THE RESIDUAL COVARIANCE MATRIX FOR CORRELATED TURBULENCE UNCORRECTLY MODELED)

$$S^{-1}_{UNC} = \begin{bmatrix} 523.151 & 33.887 & -1.129 & 0.0 & 0.0 & 0.0 \\ 33.887 & 656.989 & 0.498 & 0.0 & 0.0 & 0.0 \\ -1.129 & 0.498 & 9.749 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 515.100 & 2.817 & -26.034 \\ 0.0 & 0.0 & 0.0 & 2.817 & 40.719 & 0.983 \\ 0.0 & 0.0 & 0.0 & -26.034 & 0.983 & 655.746 \end{bmatrix}$$

$$S^{-1}_{COR} = \begin{bmatrix} 539.301 & 21.843 & -0.741 & 0.0 & 0.0 & 0.0 \\ 21.843 & 659.608 & 0.331 & 0.0 & 0.0 & 0.0 \\ -1.741 & 0.331 & 9.843 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 534.583 & 1.803 & -16.673 \\ 0.0 & 0.0 & 0.0 & 2.817 & 40.747 & 0.695 \\ 0.0 & 0.0 & 0.0 & -16.673 & 0.695 & 659.121 \end{bmatrix}$$

$\beta_{UNC} = 4342.6$

$\beta_{COR} = 4543.8$

TABLE VIII

COMPARISON BETWEEN THE SHORT PERIOD NATURAL FREQUENCIES AND DAMPINGS AT NOMINAL AND DAMAGED CONDITIONS USING EQS.(4.20), (4.21) AND MATRIX$_x$

Nominal Conditions

$M_\alpha = -8.7281$

$M_{\dot{\alpha}} = -0.4066$

$M_q = -1.2228$

$Z_\alpha = -458.34$

SHORT PERIOD characteristics using:

	Eqs.(4.20)-(4.21)	MATRIX$_x$
$\omega_{n_{S.P.}}$ =	3.091	3.062
$\zeta_{S.P.}$ =	0.373	0.376

Damaged Conditions

$M_\alpha = -3.7339$

$M_{\dot{\alpha}} = -0.2917$

$M_q = -0.9567$

$Z_\alpha = -444.27$

SHORT PERIOD characteristics using:

	Eqs.(4.20)-(4.21)	MATRIX$_x$
$\omega_{n_{S.P.}}$ =	2.089	2.063
$\zeta_{S.P.}$ =	0.456	0.462

TABLE IX

CLOSED-LOOP A MATRIX, B_F MATRIX, EIGENVALUES, EIGENVECTORS AND FEEDBACK GAINS FOR NOMINAL, UNDAMAGED,'NATURALLY' UNSTABLE AIRCRAFT

```
<> ac

 AC          =

  -0.6768   0.9939  -0.0002   0.0000   0.0000   0.0000   0.0000 0.0000
  -8.4604  -1.6258   0.0008  -0.0015   0.0000   0.0000   0.0000 0.0000
   8.4608   0.0000   0.0000 -32.1756   0.0000   0.0000   0.0000 0.0000
   0.0000   1.0000   0.0000   0.0000   0.0000   0.0000   0.0000 0.0000
   0.0000   0.0000   0.0000   0.0000  -0.0832   0.0000  -0.9988 0.0477
   0.0000   0.0000   0.0000   0.0000  -4.9804  -0.4369   0.1551 0.0000
   0.0000   0.0000   0.0000   0.0000   2.8192   0.0003  -0.1366 0.0000
   0.0000   0.0000   0.0000   0.0000   0.0000   1.0000   0.0000 0.0000


<> b

 B           =

   -0.0633    0.0000
  -17.6583    0.0000
    0.0000    0.0000
    0.0000    0.0000
    0.0000    0.0000
    0.0000    0.6278
    0.0000   -1.6802
    0.0000    0.0000
<>
```

```
EIVEC       =

   1.0D+02 *

    COLUMNS    1 THRU    4
  -0.0007 - 0.0011i    0.0000 - 0.0000i    0.0000 - 0.0001i    0.0045 + 0.0029i
   0.0034 - 0.0014i    0.0000 - 0.0000i    0.0002 - 0.0004i    0.0063 - 0.0143i
   0.0030 - 0.0086i   -0.0085 - 0.0231i    0.6970 - 1.7544i    0.0369 - 0.0118i
  -0.0008 - 0.0009i   -0.0001 + 0.0000i    0.0049 + 0.0019i    0.0036 + 0.0036i
   0.0000 + 0.0000i    0.0000 + 0.0000i    0.0000 + 0.0000i    0.0000 + 0.0000i
   0.0000 + 0.0000i    0.0000 + 0.0000i    0.0000 + 0.0000i    0.0000 + 0.0000i
   0.0000 + 0.0000i    0.0000 + 0.0000i    0.0000 + 0.0000i    0.0000 + 0.0000i
   0.0000 + 0.0000i    0.0000 + 0.0000i    0.0000 + 0.0000i    0.0000 + 0.0000i

    COLUMNS    5 THRU    8
   0.0000 + 0.0000i    0.0000 + 0.0000i    0.0000 + 0.0000i    0.0000 + 0.0000i
   0.0000 + 0.0000i    0.0000 + 0.0000i    0.0000 + 0.0000i    0.0000 + 0.0000i
   0.0000 + 0.0000i    0.0000 + 0.0000i    0.0000 + 0.0000i    0.0000 + 0.0000i
   0.0000 + 0.0000i    0.0000 + 0.0000i    0.0000 + 0.0000i    0.0000 + 0.0000i
   0.0019 + 0.0017i    0.0001 + 0.0029i   -0.0001 + 0.0000i    0.0001 - 0.0000i
  -0.0063 + 0.0040i    0.0081 - 0.0024i   -0.0095 + 0.0000i   -0.0003 + 0.0000i
   0.0030 - 0.0031i   -0.0049 + 0.0003i    0.0008 - 0.0000i    0.0020 - 0.0000i
   0.0025 + 0.0036i    0.0012 + 0.0048i    0.0188 + 0.0000i    0.0412 + 0.0000i
<>
```

```
<> eival

 EIVAL       =

  -1.1510 + 2.8598i    SHORT PERIOD (+)
  -0.0003 + 0.0890i    PHUGOID (+)
  -0.0003 - 0.0890i    PHUGOID (-)
  -1.1510 - 2.8598i    SHORT PERIOD (-)
  -0.0701 + 1.6858i    DUTCH ROLL (+)
  -0.0701 - 1.6858i    DUTCH ROLL (-)
  -0.5085 + 0.0000i    ROLLING
  -0.0080 + 0.0000i    SPIRAL
<> k

 K           =

  -0.5568   -0.0183    0.0000   -0.0001  0.0000    0.0000    0.0000    0.0000
   0.0000    0.0000    0.0000    0.0000  0.0000    0.0000   -0.0001    0.0000
<>
```

TABLE X

CLOSED-LOOP A MATRIX, B_F MATRIX, EIGENVALUES, EIGENVECTORS AND FEEDBACK GAINS FOR DAMAGED, 'NATURALLY' UNSTABLE AIRCRAFT WITH THE FEEDBACK GAINS RELATIVE TO THE UNDAMAGED CONDITIONS

```
<> adamcl

 ADAMCL    =

  -0.6493   0.9954 -0.0002   0.0000  0.0000  0.0000  0.0000 0.0000
  -0.7206  -1.4800  0.0008  -0.0011  0.0000  0.0000  0.0000 0.0000
   8.4608   0.0000  0.0000 -32.1756  0.0000  0.0000  0.0000 0.0000
   0.0000   1.0000  0.0000   0.0000  0.0000  0.0000  0.0000 0.0000
   0.0000   0.0000  0.0000   0.0000 -0.0832  0.0000 -0.9988 0.0477
  -1.0716  -0.0353  0.0000  -0.0002 -4.9804 -0.4369  0.1551 0.0000
  -0.0102  -0.0003  0.0000   0.0000  2.8192  0.0003 -0.1366 0.0000
   0.0000   0.0000  0.0000   0.0000  0.0000  1.0000  0.0000 0.0000
```

```
<> bdam

 BDAM       =

   -0.0454    0.0000
  -12.6809    0.0000
    0.0000    0.0000
    0.0000    0.0000
    0.0000    0.0000
   -1.9244    0.6278
   -0.0184   -1.6802
    0.0000    0.0000
<>
```

```
 EIVECDAM    =

    1.0D+02 *

     COLUMNS    1 THRU    4
    0.0000 + 0.0000i   -0.0001 - 0.0009i    0.0000 - 0.0000i   -0.0053 + 0.0003i
    0.0000 - 0.0000i    0.0007 + 0.0003i    0.0000 - 0.0000i    0.0025 + 0.0038i
    0.0000 - 0.0000i   -0.0019 - 0.0097i    0.0000 - 0.0000i   -0.0573 - 0.0032i
    0.0000 + 0.0000i   -0.0003 - 0.0005i    0.0000 + 0.0000i   -0.0032 - 0.0013i
   -0.0019 + 0.0018i    0.0000 - 0.0000i    0.0022 + 0.0019i    0.0000 + 0.0001i
   -0.0034 - 0.0066i    0.0008 - 0.0007i    0.0037 - 0.0076i   -0.0031 + 0.0051i
    0.0028 + 0.0033i    0.0000 - 0.0000i   -0.0031 + 0.0038i   -0.0001 - 0.0002i
   -0.0038 + 0.0022i   -0.0008 + 0.0001i    0.0044 + 0.0024i   -0.0003 - 0.0046i

     COLUMNS    5 THRU    8
    0.0000 + 0.0000i    0.0000 + 0.0000i    0.0004 + 0.0003i    0.0000 + 0.0000i
    0.0000 - 0.0000i    0.0000 + 0.0000i    0.0005 + 0.0003i    0.0000 + 0.0000i
    0.0000 - 0.0000i    0.0194 + 0.0842i    1.2951 + 0.6485i    0.0000 - 0.0000i
    0.0000 + 0.0000i    0.0003 - 0.0001i   -0.0023 + 0.0047i    0.0000 + 0.0000i
   -0.0001 + 0.0001i    0.0000 - 0.0000i    0.0000 - 0.0000i    0.0001 - 0.0000i
   -0.0057 + 0.0101i    0.0000 - 0.0001i   -0.0007 - 0.0007i   -0.0004 + 0.0000i
    0.0005 - 0.0009i    0.0000 + 0.0000i    0.0003 - 0.0003i    0.0025 + 0.0000i
    0.0112 - 0.0199i   -0.0004 + 0.0003i    0.0061 - 0.0061i    0.0524 - 0.0000i
<>
```

```
<> eivaldam

 EIVALDAM    =

   -0.0701 + 1.6858i    DUTCH ROLL (+)
   -1.0662 + 0.7332i    SHORT PERIOD (+)
   -0.0701 - 1.6858i    DUTCH ROLL (-)
   -1.0662 - 0.7332i    SHORT PERIOD (-)
   -0.5085 - 0.0000i    ROLLING
    0.0016 + 0.1157i    PHUGOID (+)
    0.0016 - 0.1157i    PHUGOID (-)
   -0.0080 + 0.0000i    SPIRAL
<> k

 K           =

   -0.5568   -0.0183    0.0000   -0.0001  0.0000    0.0000    0.0000    0.0000
    0.0000    0.0000    0.0000    0.0000  0.0000    0.0000   -0.0001    0.0000
<>
```

TABLE XI

CLOSED LOOP A MATRIX, B_F MATRIX, EIGENVALUES, EIGENVECTORS AND FEEDBACK GAINS FOR DAMAGED, 'NATURALLY' UNSTABLE AIRCRAFT WITH REDESIGNED FEEDBACK STRUCTURE ($m=3$: $\delta_{ER}, \delta_{CL}, \delta_R$)

```
<> redadamcl

 REDADAMCL  =

    COLUMNS    1 THRU    4
  -0.8335 + 0.0000i    0.9931 - 0.0000i   -0.0002 + 0.0000i   -0.0001 - 0.0000i
  -8.3387 + 0.0000i   -1.4691 - 0.0000i    0.0007 + 0.0000i   -0.0056 - 0.0000i
   8.4608 + 0.0000i    0.0000 + 0.0000i    0.0000 + 0.0000i  -32.1756 + 0.0000i
   0.0000 + 0.0000i    1.0000 + 0.0000i    0.0000 + 0.0000i    0.0000 + 0.0000i
   0.0001 + 0.0000i    0.0000 - 0.0000i    0.0000 - 0.0000i    0.0000 - 0.0000i
  -0.0159 - 0.0000i    0.0005 + 0.0000i    0.0000 - 0.0000i   -0.0002 + 0.0000i
  -0.0068 - 0.0000i   -0.0012 + 0.0000i    0.0000 + 0.0000i    0.0001 + 0.0000i
   0.0000 + 0.0000i    0.0000 + 0.0000i    0.0000 + 0.0000i    0.0000 + 0.0000i

    COLUMNS    5 THRU    8
   0.0000 - 0.0000i    0.0000 + 0.0000i    0.0000 - 0.0000i    0.0000 + 0.0000i
   0.0000 - 0.0000i    0.0000 + 0.0000i    0.0000 + 0.0000i    0.0000 - 0.0000i
   0.0000 + 0.0000i    0.0000 + 0.0000i    0.0000 + 0.0000i    0.0000 + 0.0000i
   0.0000 + 0.0000i    0.0000 + 0.0000i    0.0000 + 0.0000i    0.0000 + 0.0000i
  -0.0832 + 0.0000i    0.0000 - 0.0000i   -0.9988 - 0.0000i    0.0477 + 0.0000i
  -4.9804 + 0.0000i   -0.4369 - 0.0000i    0.1551 + 0.0000i    0.0000 - 0.0000i
   2.8192 - 0.0000i    0.0003 + 0.0000i   -0.1366 + 0.0000i    0.0000 - 0.0000i
   0.0000 + 0.0000i    1.0000 + 0.0000i    0.0000 + 0.0000i    0.0000 + 0.0000i
<>

<> bred

 BRED       =

  -0.0317   -0.0228    0.0000
  -8.8380    4.0685    0.0000
   0.0000    0.0000    0.0000
   0.0000    0.0000    0.0000
   0.0000    0.0000    0.0167
  -3.3876    2.8941    0.6278
  -0.0324    0.0277   -1.6802
   0.0000    0.0000    0.0000
<>
```

```
REDEIVEC    =

   1.0D+02 *

    COLUMNS    1 THRU    4
   0.0010 + 0.0008i   -0.0017 - 0.0013i    0.0000 + 0.0000i    0.0000 + 0.0000i
  -0.0027 + 0.0025i   -0.0032 + 0.0054i    0.0000 - 0.0000i    0.0000 + 0.0000i
   0.0004 + 0.0091i   -0.0154 + 0.0026i    0.0000 - 0.0000i    0.0000 - 0.0000i
   0.0011 + 0.0005i   -0.0012 - 0.0016i    0.0000 + 0.0000i    0.0000 + 0.0000i
   0.0000 - 0.0000i    0.0000 + 0.0000i   -0.0023 - 0.0012i    0.0020 + 0.0022i
   0.0000 + 0.0000i    0.0000 + 0.0000i    0.0051 - 0.0054i    0.0045 - 0.0071i
   0.0000 + 0.0000i    0.0000 + 0.0000i   -0.0022 + 0.0037i   -0.0035 + 0.0034i
   0.0000 + 0.0000i    0.0000 - 0.0000i   -0.0033 - 0.0029i    0.0041 + 0.0028i

    COLUMNS    5 THRU    8
   0.0000 + 0.0000i    0.0000 - 0.0000i    0.0001 + 0.0000i    0.0000 - 0.0000i
   0.0000 - 0.0000i    0.0000 - 0.0000i    0.0006 + 0.0000i    0.0000 - 0.0000i
   0.0000 + 0.0000i    0.0468 - 0.0078i    2.4069 + 0.1434i    0.0000 + 0.0000i
   0.0000 + 0.0000i    0.0000 - 0.0001i   -0.0003 + 0.0067i    0.0000 + 0.0000i
   0.0000 + 0.0001i    0.0000 + 0.0000i    0.0000 - 0.0000i    0.0001 - 0.0000i
   0.0015 + 0.0094i    0.0000 + 0.0000i    0.0000 - 0.0000i   -0.0003 + 0.0000i
  -0.0001 - 0.0008i    0.0000 + 0.0000i    0.0000 - 0.0000i    0.0020 - 0.0000i
  -0.0029 - 0.0186i    0.0000 + 0.0000i    0.0000 - 0.0000i    0.0412 - 0.0000i
<>

  REDEIVAL    =

   -1.1510 + 2.8598i    SHORT PERIOD (+)
   -1.1510 - 2.8598i    SHORT PERIOD (-)
   -0.0701 + 1.6858i    DUTCH ROLL (+)
   -0.0701 - 1.6858i    DUTCH ROLL (-)
   -0.5085 + 0.0000i    ROLLING
   -0.0003 + 0.0890i    PHUGOID (+)
   -0.0003 - 0.0890i    PHUGOID (-)
   -0.0080 + 0.0000i    SPIRAL
 <> redk

  REDK        =

    COLUMNS    1 THRU    4
   3.5953 - 0.0000i    0.0544 + 0.0000i    0.0000 - 0.0000i    0.0013 + 0.0000i
   4.2021 - 0.0000i    0.0637 + 0.0000i    0.0000 - 0.0000i    0.0015 + 0.0000i
   0.0039 + 0.0000i    0.0007 - 0.0000i    0.0000 - 0.0000i   -0.0001 - 0.0000i

    COLUMNS    5 THRU    8
   0.0000 + 0.0000i    0.0000 - 0.0000i    0.0000 - 0.0000i    0.0000 + 0.0000i
   0.0000 + 0.0000i    0.0000 - 0.0000i    0.0000 + 0.0000i    0.0000 - 0.0000i
   0.0000 + 0.0000i    0.0000 - 0.0000i    0.0001 - 0.0000i    0.0000 + 0.0000i
 <>
```

TABLE XII

CLOSED LOOP A MATRIX, B_F MATRIX, EIGENVALUES, EIGENVECTORS AND FEEDBACK GAINS FOR DAMAGED, 'NATURALLY' UNSTABLE AIRCRAFT WITH REDESIGNED FEEDBACK STRUCTURE ($m=4$: $\delta_{ER}, \delta_{CL}, \delta_R, \delta_R$)

```
<> redadamcl

 REDADAMCL  =

    COLUMNS   1 THRU   4
  -0.6768 + 0.0000i   0.9939 - 0.0000i  -0.0002 + 0.0000i   0.0000 - 0.0000i
  -8.4604 + 0.0000i  -1.6258 - 0.0000i   0.0008 + 0.0000i  -0.0015 - 0.0000i
   8.4608 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i -32.1756 + 0.0000i
   0.0000 + 0.0000i   1.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i
   0.0000 - 0.0000i   0.0000 + 0.0000i   0.0000 - 0.0000i   0.0000 + 0.0000i
   0.0000 - 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i
   0.0000 + 0.0000i   0.0000 - 0.0000i   0.0000 + 0.0000i   0.0000 - 0.0000i
   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i

    COLUMNS   5 THRU   8
   0.0000 + 0.0000i   0.0000 - 0.0000i   0.0000 - 0.0000i   0.0000 + 0.0000i
   0.0000 - 0.0000i   0.0000 + 0.0000i   0.0000 - 0.0000i   0.0000 + 0.0000i
   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i
   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i
  -0.0832 + 0.0000i   0.0000 - 0.0000i  -0.9988 + 0.0000i   0.0477 - 0.0000i
  -4.9804 - 0.0000i  -0.4369 + 0.0000i   0.1551 + 0.0000i   0.0000 - 0.0000i
   2.8192 - 0.0000i   0.0003 + 0.0000i  -0.1366 - 0.0000i   0.0000 + 0.0000i
   0.0000 + 0.0000i   1.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i
<>

<> bred

 BRED       =

  -0.0317   -0.0228   -0.0228    0.0000
  -8.8380    4.0685    4.0685    0.0000
   0.0000    0.0000    0.0000    0.0000
   0.0000    0.0000    0.0000    0.0000
   0.0000    0.0000    0.0000    0.0167
  -3.3876    2.8941   -2.8941    0.6278
  -0.0324    0.0277   -0.0277   -1.6802
   0.0000    0.0000    0.0000    0.0000
<>
```

```
 REDEIVEC    =

   1.0D+02 *

    COLUMNS    1 THRU    4
  -0.0012 - 0.0003i   -0.0001 - 0.0022i    0.0000 + 0.0000i    0.0000 + 0.0000i
  -0.0002 + 0.0037i    0.0063 + 0.0009i    0.0000 - 0.0000i    0.0000 - 0.0000i
  -0.0071 + 0.0057i    0.0118 - 0.0102i    0.0000 + 0.0000i    0.0000 - 0.0000i
  -0.0011 - 0.0005i   -0.0005 - 0.0020i    0.0000 + 0.0000i    0.0000 + 0.0000i
   0.0000 - 0.0000i    0.0000 + 0.0000i   -0.0002 + 0.0026i    0.0026 - 0.0013i
   0.0000 - 0.0000i    0.0000 - 0.0000i    0.0073 - 0.0014i    0.0017 + 0.0083i
   0.0000 - 0.0000i    0.0000 - 0.0000i   -0.0043 - 0.0001i   -0.0020 - 0.0044i
   0.0000 + 0.0000i    0.0000 + 0.0000i    0.0007 + 0.0044i    0.0048 - 0.0012i

    COLUMNS    5 THRU    8
   0.0000 + 0.0000i    0.0001 + 0.0001i    0.0000 + 0.0000i    0.0000 + 0.0000i
   0.0000 + 0.0000i    0.0005 + 0.0004i    0.0000 + 0.0000i    0.0000 + 0.0000i
   0.0449 + 0.0150i    1.9576 + 1.4313i    0.0000 - 0.0000i    0.0000 - 0.0000i
   0.0000 + 0.0001i    0.0040 - 0.0054i    0.0000 - 0.0000i    0.0000 - 0.0000i
   0.0000 - 0.0000i    0.0000 + 0.0000i    0.0000 + 0.0001i    0.0001 + 0.0000i
   0.0000 - 0.0000i    0.0000 - 0.0000i   -0.0022 + 0.0093i   -0.0003 - 0.0001i
   0.0000 - 0.0000i    0.0000 + 0.0000i    0.0002 - 0.0008i    0.0018 + 0.0007i
   0.0000 - 0.0000i    0.0000 + 0.0000i    0.0043 - 0.0183i    0.0383 + 0.0151i
<>
<> redeival

 REDEIVAL    =

  -1.1510 - 2.8598i    SHORT PERIOD (-)
  -1.1510 + 2.8598i    SHORT PERIOD (+)
  -0.0701 - 1.6858i    DUTCH ROLL (-)
  -0.0701 + 1.6858i    DUTCH ROLL (+)
  -0.0003 - 0.0890i    PHUGOID (-)
  -0.0003 + 0.0890i    PHUGOID (+)
  -0.5085 + 0.0000i    ROLLING
  -0.0080 - 0.0000i    SPIRAL
<> redk

 REDK        =

    COLUMNS    1 THRU    4
   1.6728 - 0.0000i    0.0546 + 0.0000i    0.0000 - 0.0000i    0.0002 + 0.0000i
   0.9770 - 0.0000i    0.0448 + 0.0000i    0.0001 - 0.0000i    0.0001 + 0.0000i
  -0.9811 + 0.0000i   -0.0191 - 0.0000i    0.0000 - 0.0000i   -0.0001 - 0.0000i
   0.0000 - 0.0000i    0.0000 + 0.0000i    0.0000 - 0.0000i    0.0000 + 0.0000i

    COLUMNS    5 THRU    8
   0.0000 - 0.0000i    0.0000 - 0.0000i    0.0000 + 0.0000i    0.0000 - 0.0000i
   0.0000 - 0.0000i    0.0000 + 0.0000i    0.0000 + 0.0000i    0.0000 - 0.0000i
   0.0000 + 0.0000i    0.0000 - 0.0000i    0.0000 - 0.0000i    0.0000 + 0.0000i
   0.0000 + 0.0000i    0.0000 - 0.0000i    0.0001 + 0.0000i    0.0000 - 0.0000i
<>
```

APPENDIX B

FIGURES

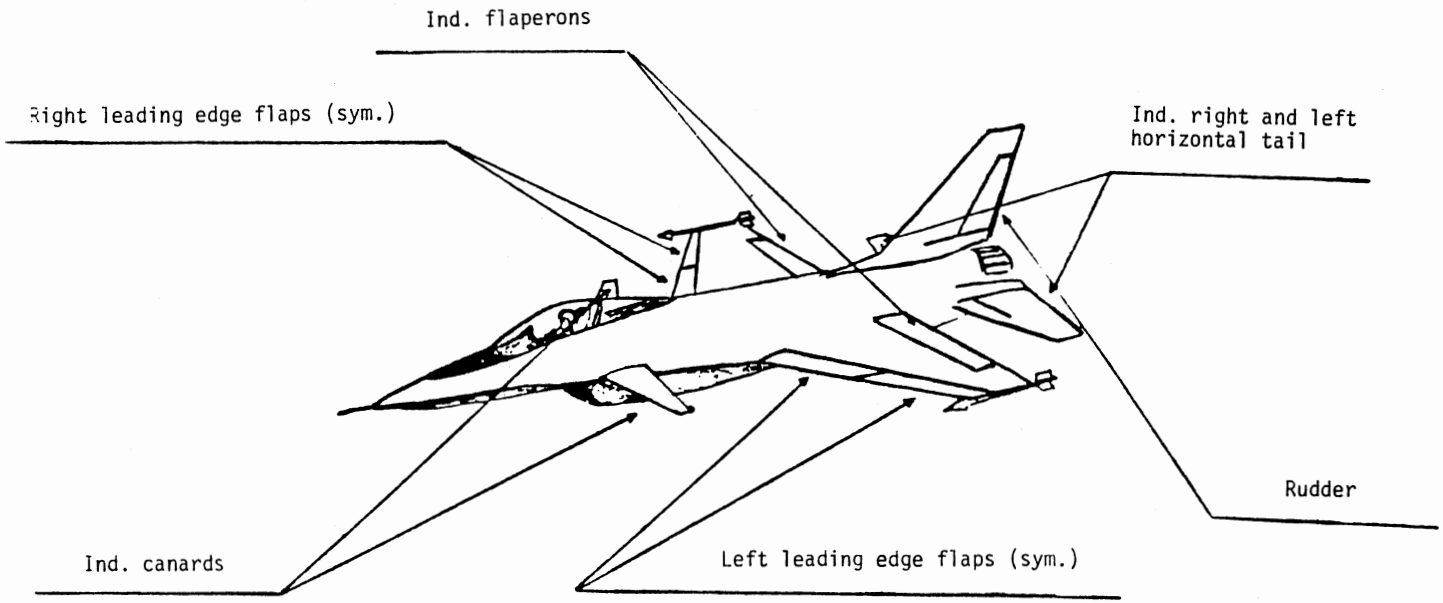

Figure 1. F-16 aircraft with independent control surfaces.

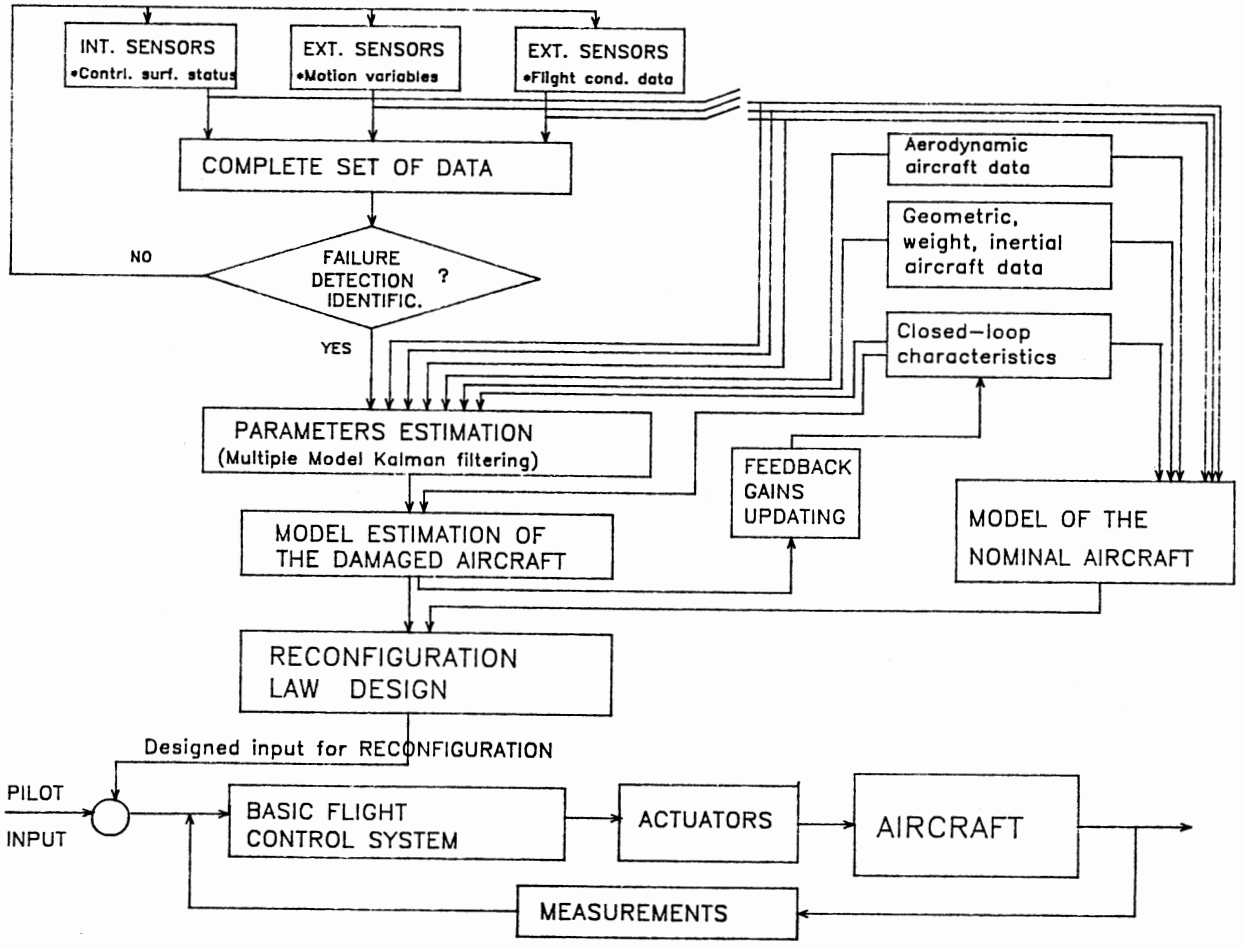

Figure 2. Step-by-step overview of the Reconfiguration Problem.

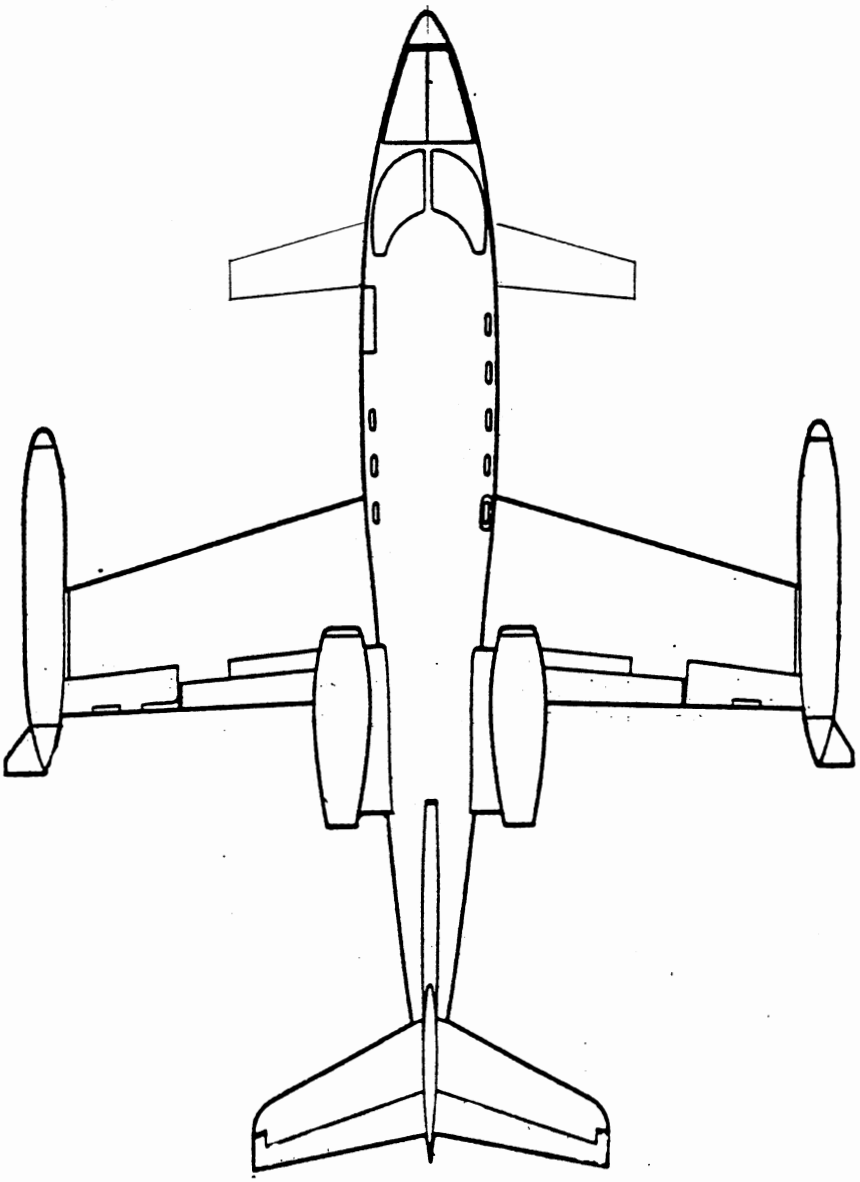

Figure 3. Used aircraft model.

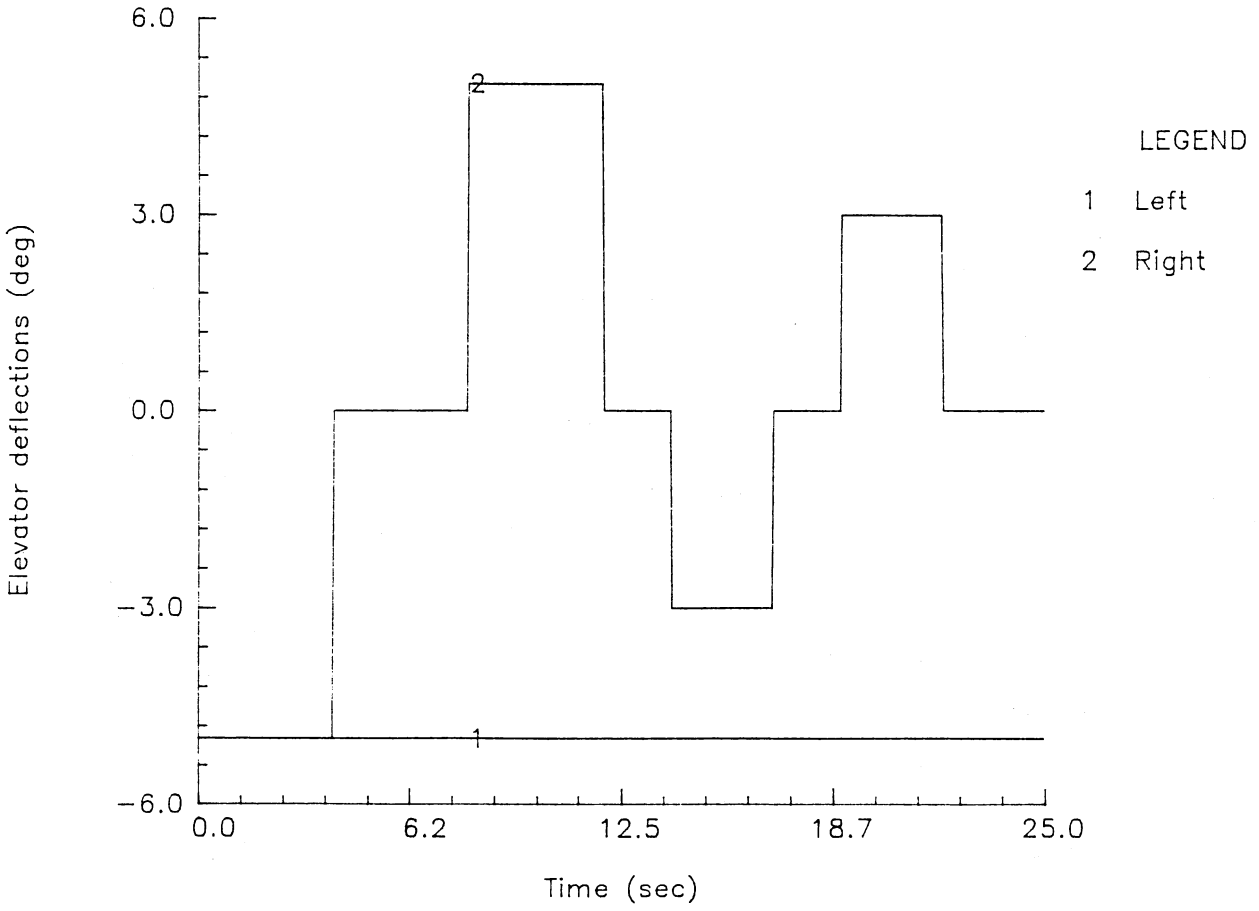

Figure 4. Right and left (damaged) elevator inputs.

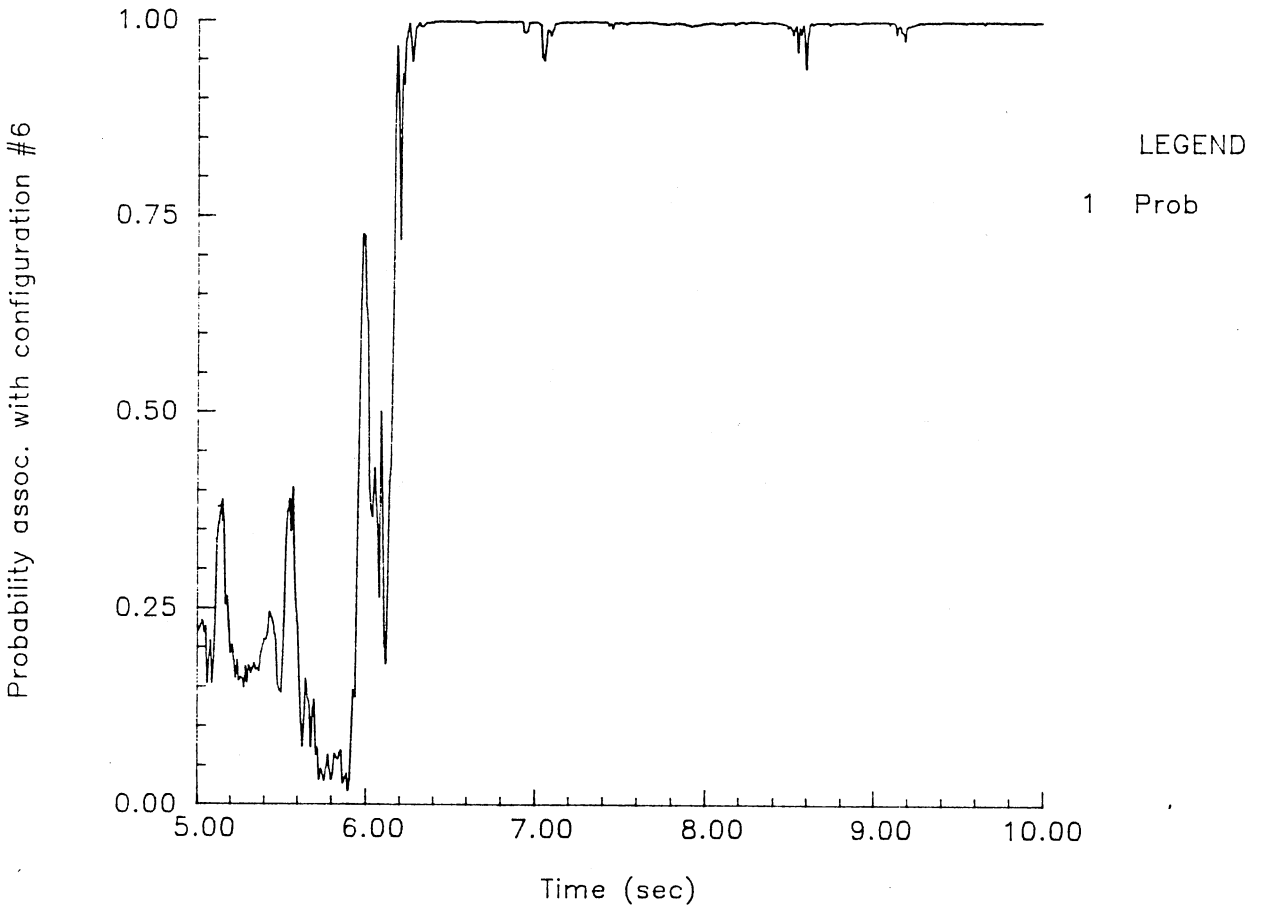

Figure 5. Probability convergence to the closest model ($N = 12$, $C_{L_{\delta EL}} = 0.121$).

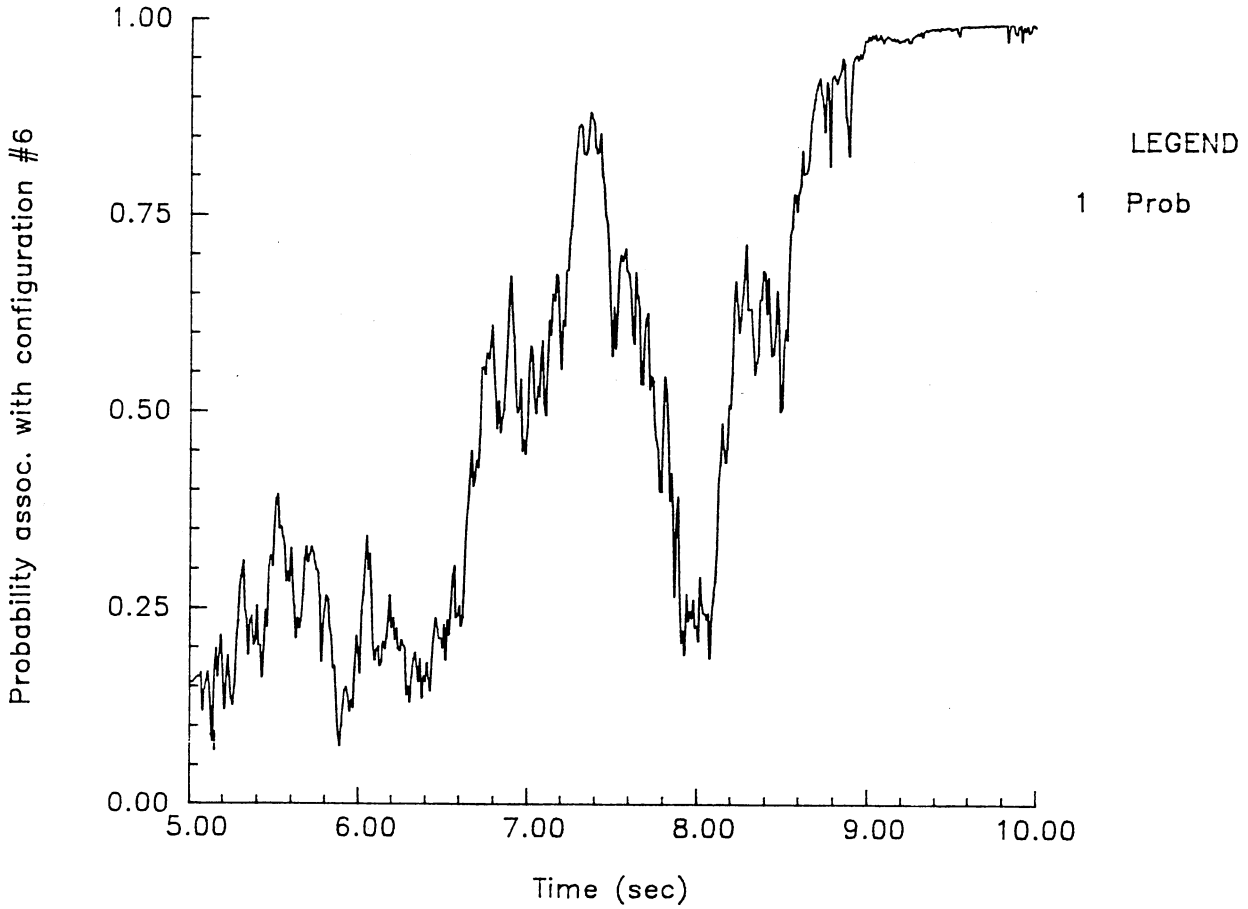

Figure 6. Probability convergence to the closest model ($N = 23$, $C_{L_{\delta EL}} = 0.121$).

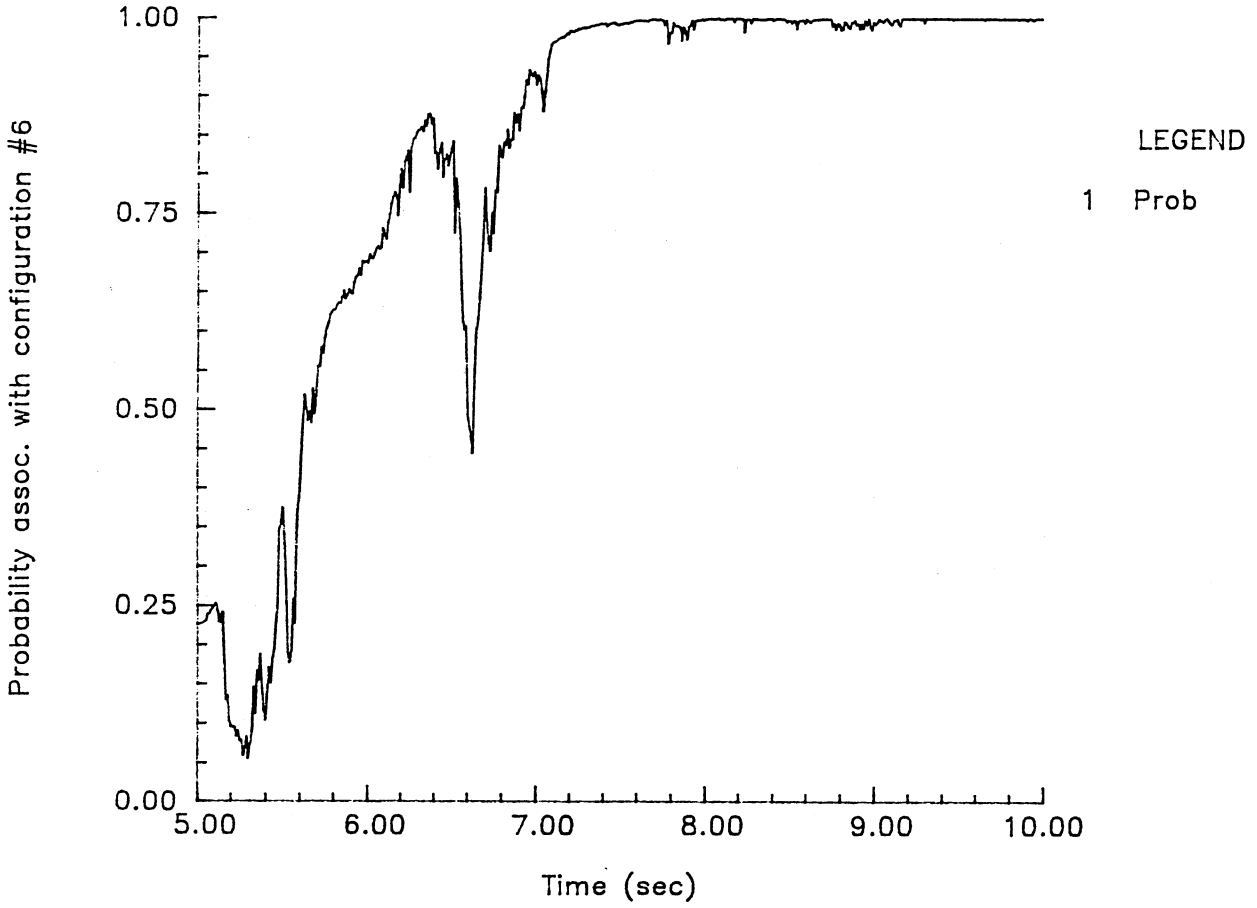

Figure 7. Probability convergence to the closest model ($N = 12$, $C_{L_{\delta EL}} = 0.114$).

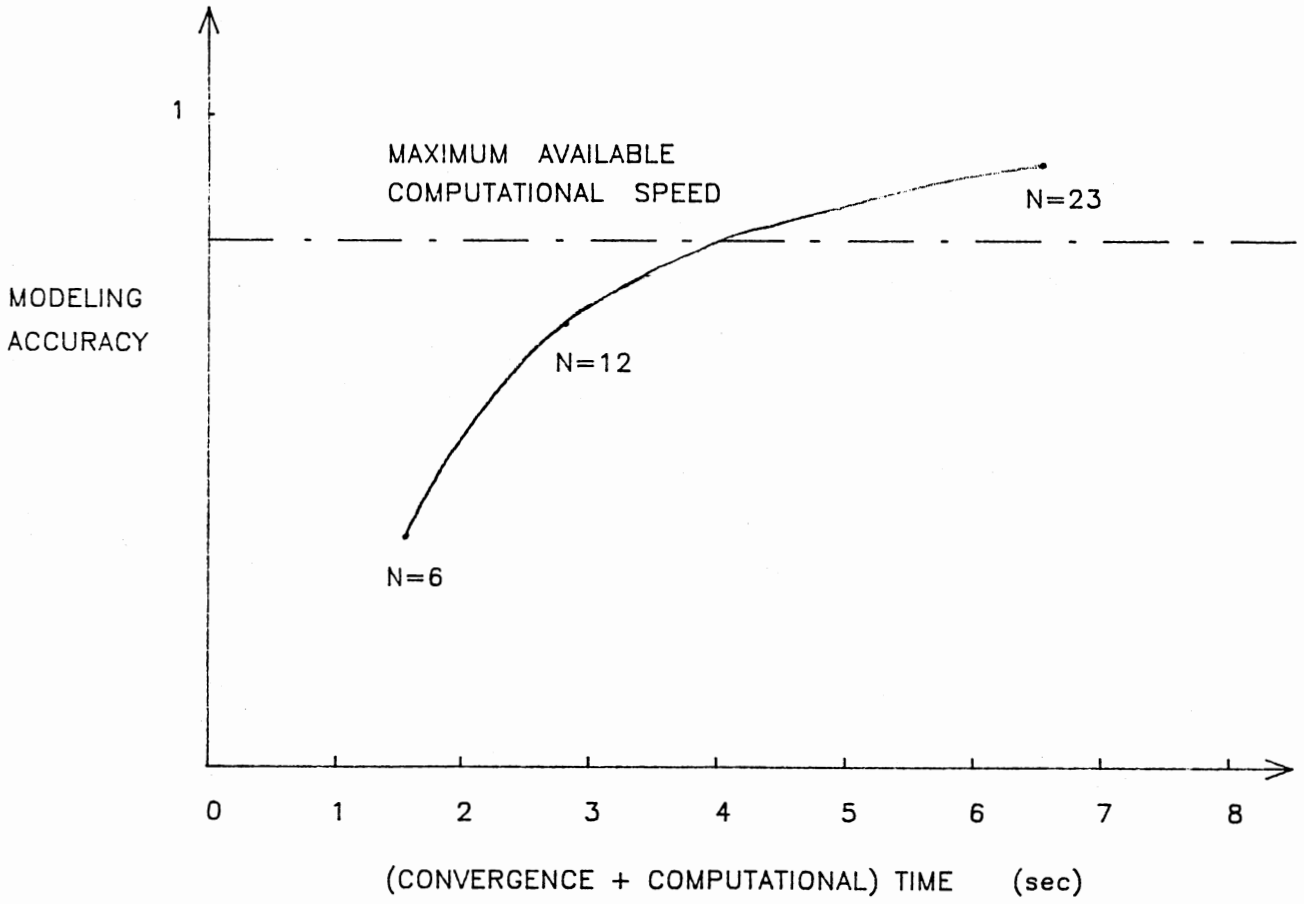

Figure 8. Parameters for the selection of the value of N.

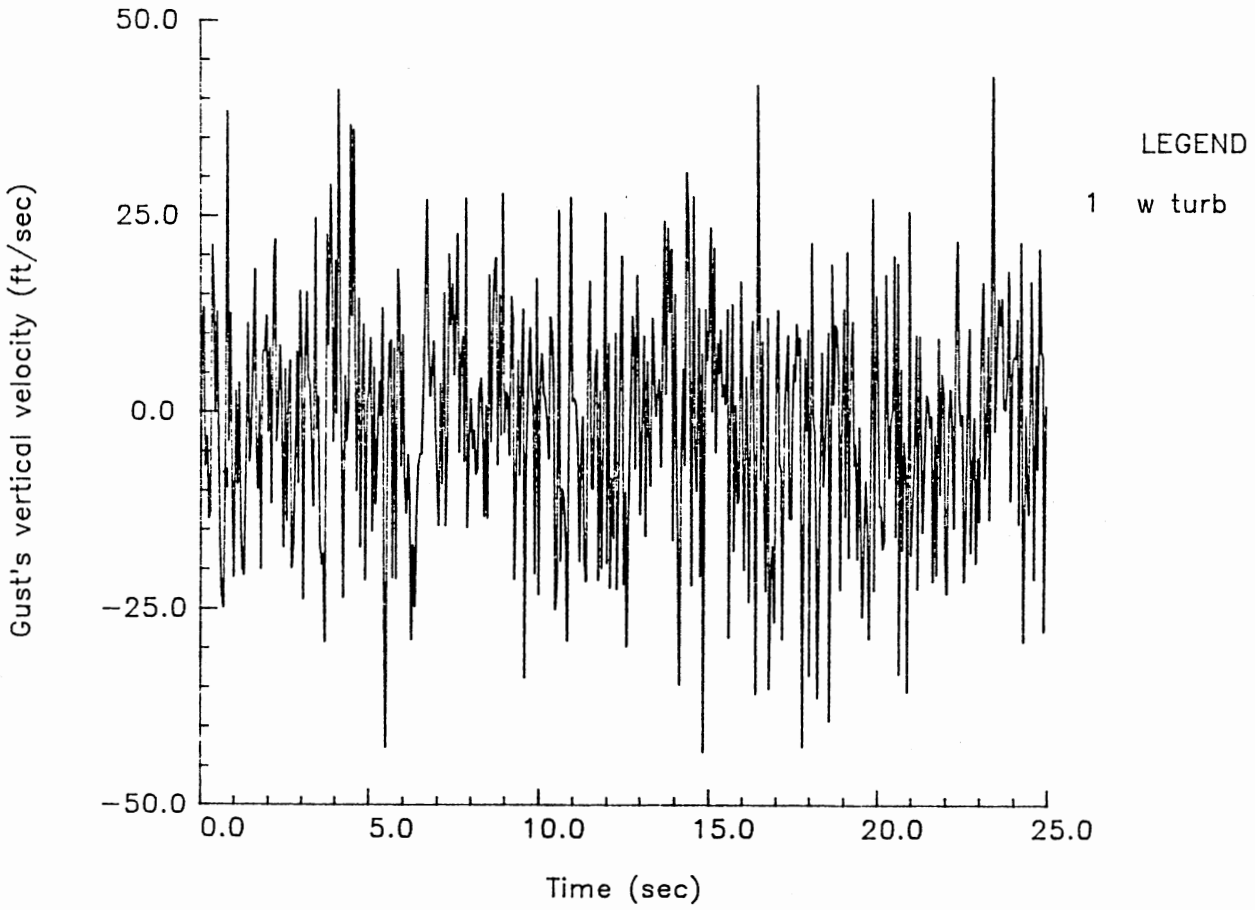

Figure 9. Gust's vertical velocity ($\alpha_g(k)$ = white noise, $\sigma^2_\alpha = 0.0005$).

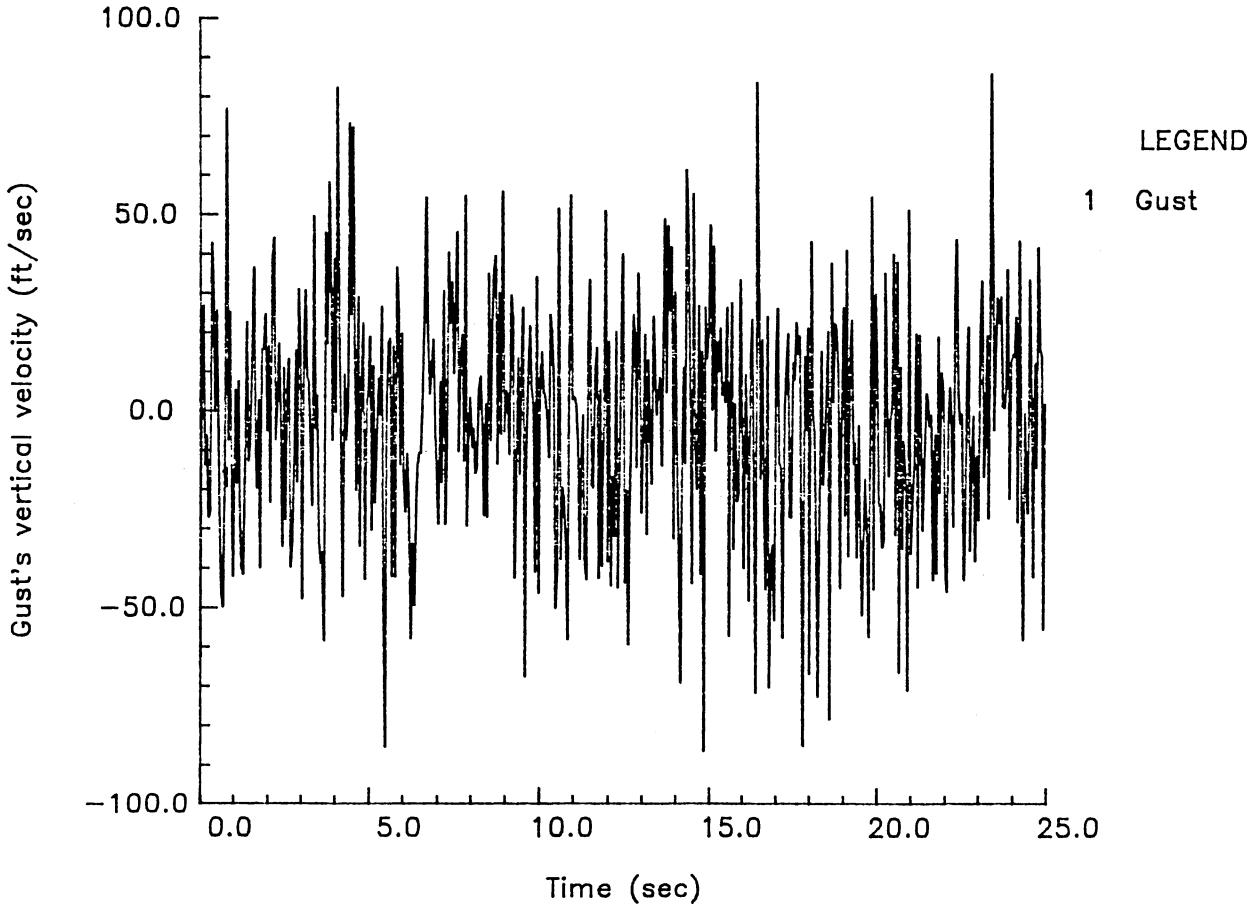

Figure 10. Gust's vertical velocity ($\alpha_g(k)$ = white noise, $\sigma^2_\alpha = 0.002$).

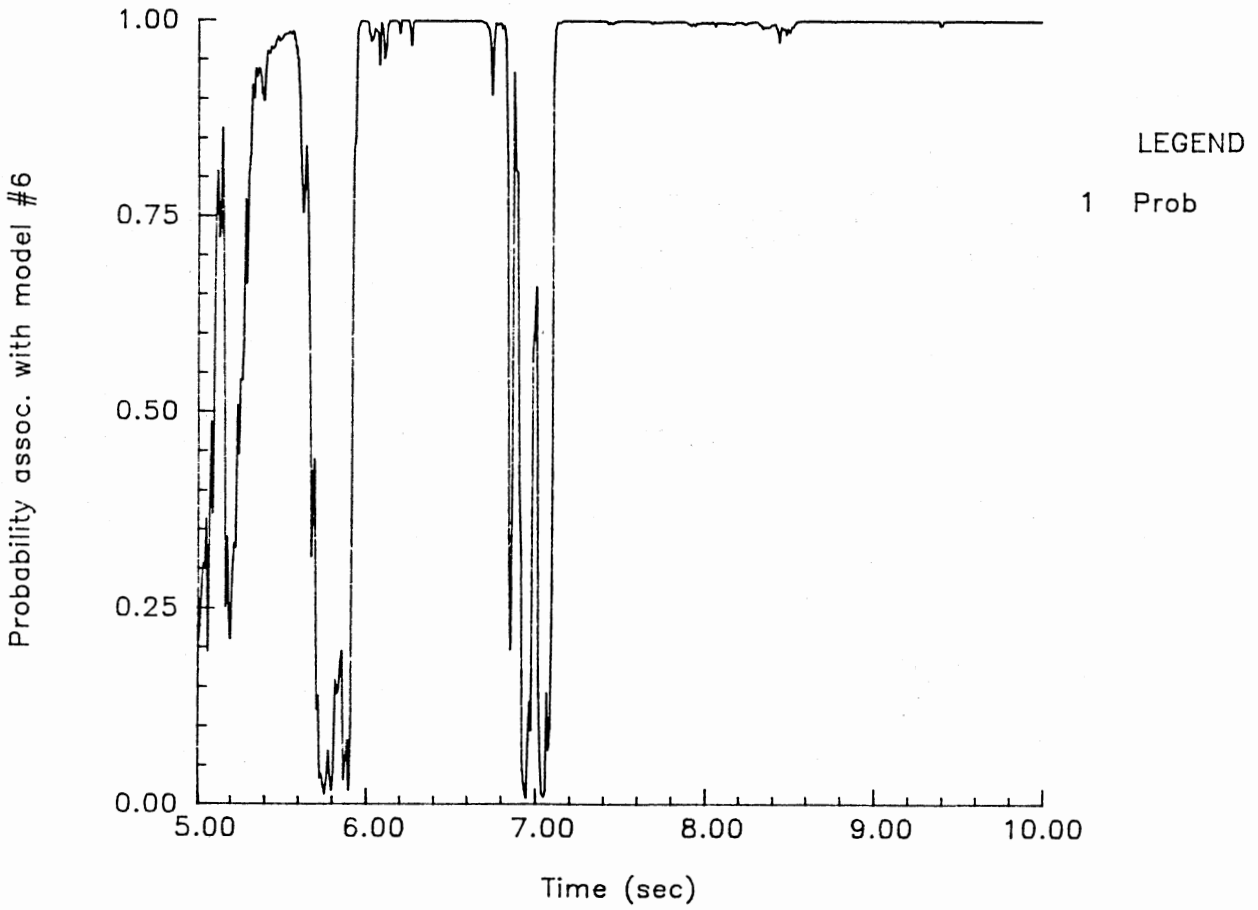

Figure 11. Probability convergence to the closest model ($N = 12$, $C_{L_{\delta EL}} = 0.121$) for different values of Q and Q_m

$$Q = \begin{bmatrix} 0.002 & 0 \\ 0 & 0.002 \end{bmatrix} \quad Q_m = \begin{bmatrix} 0.0005 & 0 \\ 0 & 0.0005 \end{bmatrix}$$

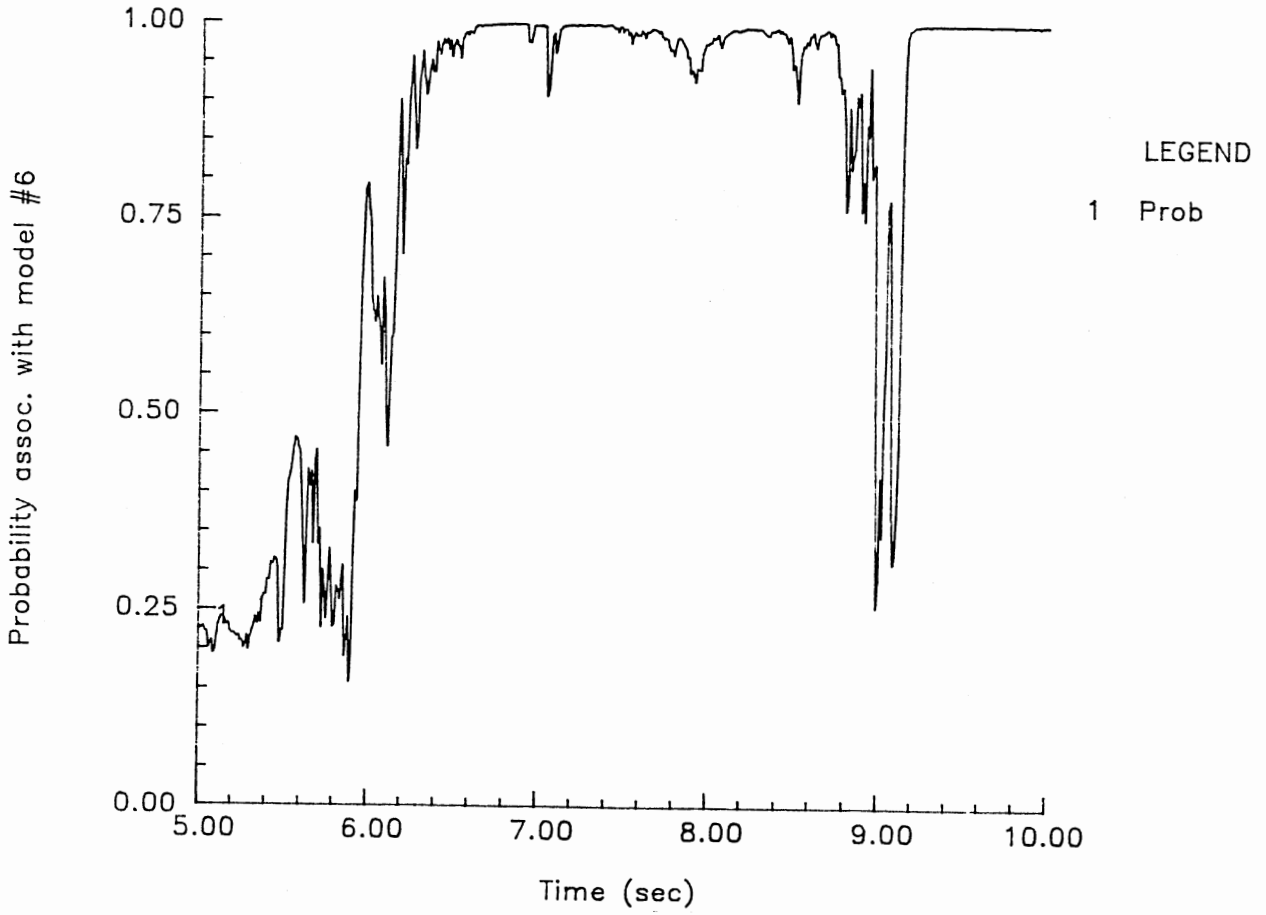

Figure 12. Probability convergence to the closest model ($N = 12$, $C_{L_{\delta EL}} = 0.121$) for different values of Q and Q_m

$$Q = \begin{bmatrix} 0.0001 & 0 \\ 0 & 0.0001 \end{bmatrix} \quad Q_m = \begin{bmatrix} 0.0005 & 0 \\ 0 & 0.0005 \end{bmatrix}$$

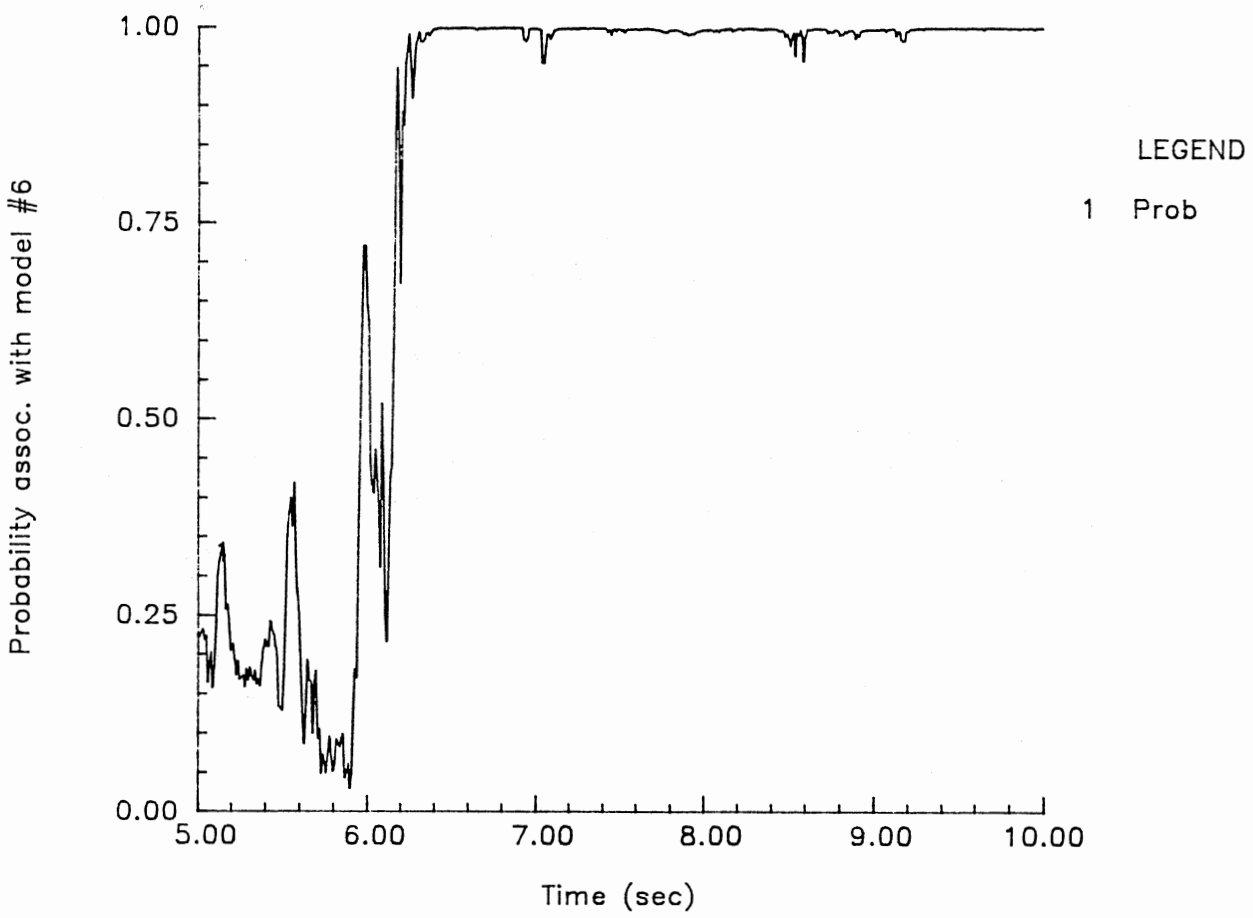

Figure 13. Probability convergence to the closest model ($N = 12$, $C_{L_{\delta EL}} = 0.121$) for different values of Q and Q_m

$$Q = \begin{bmatrix} 0.00033 & 0 \\ 0 & 0.00033 \end{bmatrix} \quad Q_m = \begin{bmatrix} 0.0005 & 0 \\ 0 & 0.0005 \end{bmatrix}$$

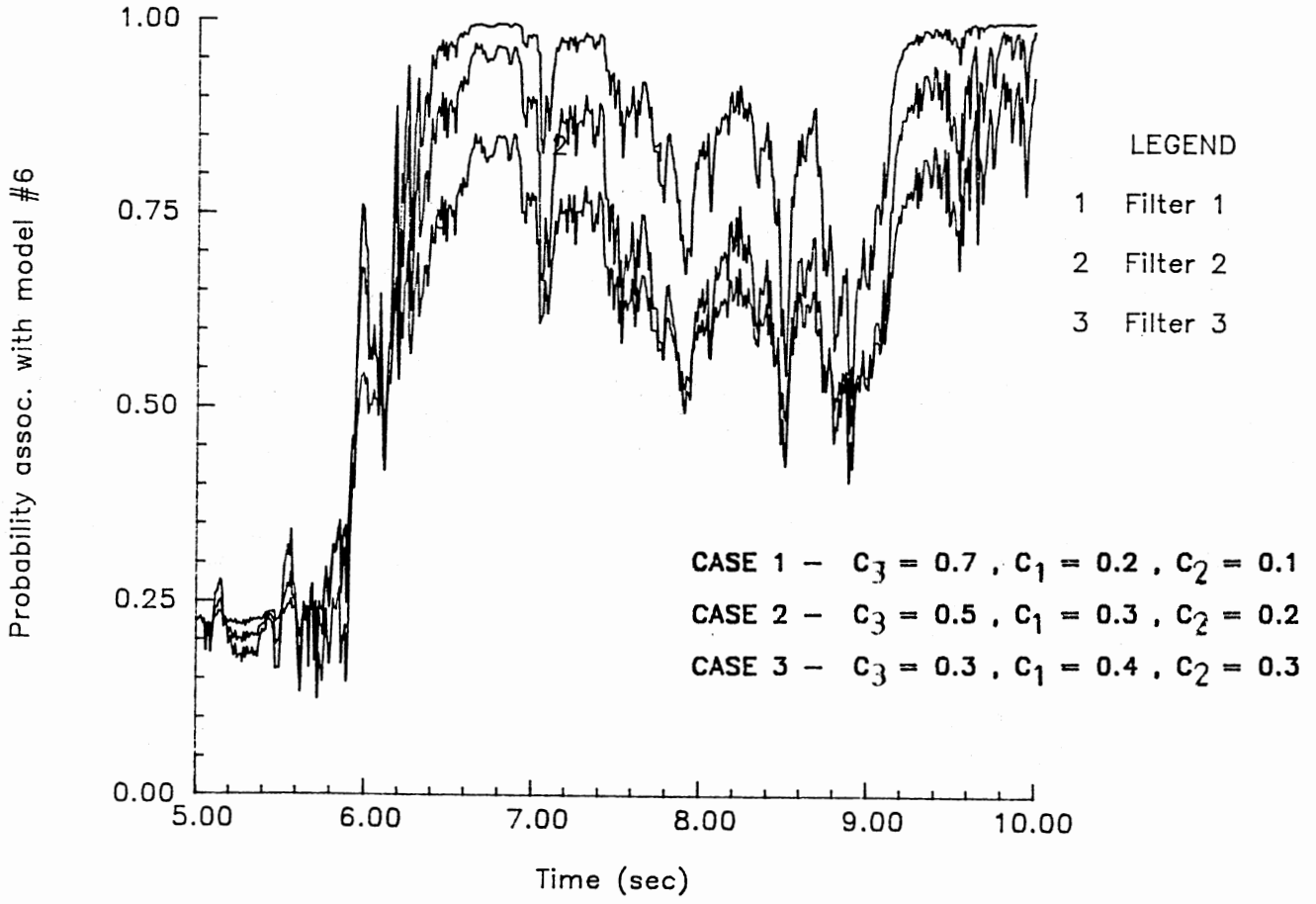

Figure 14. Filtered probability convergence to the closest model for different values of the filter constants.

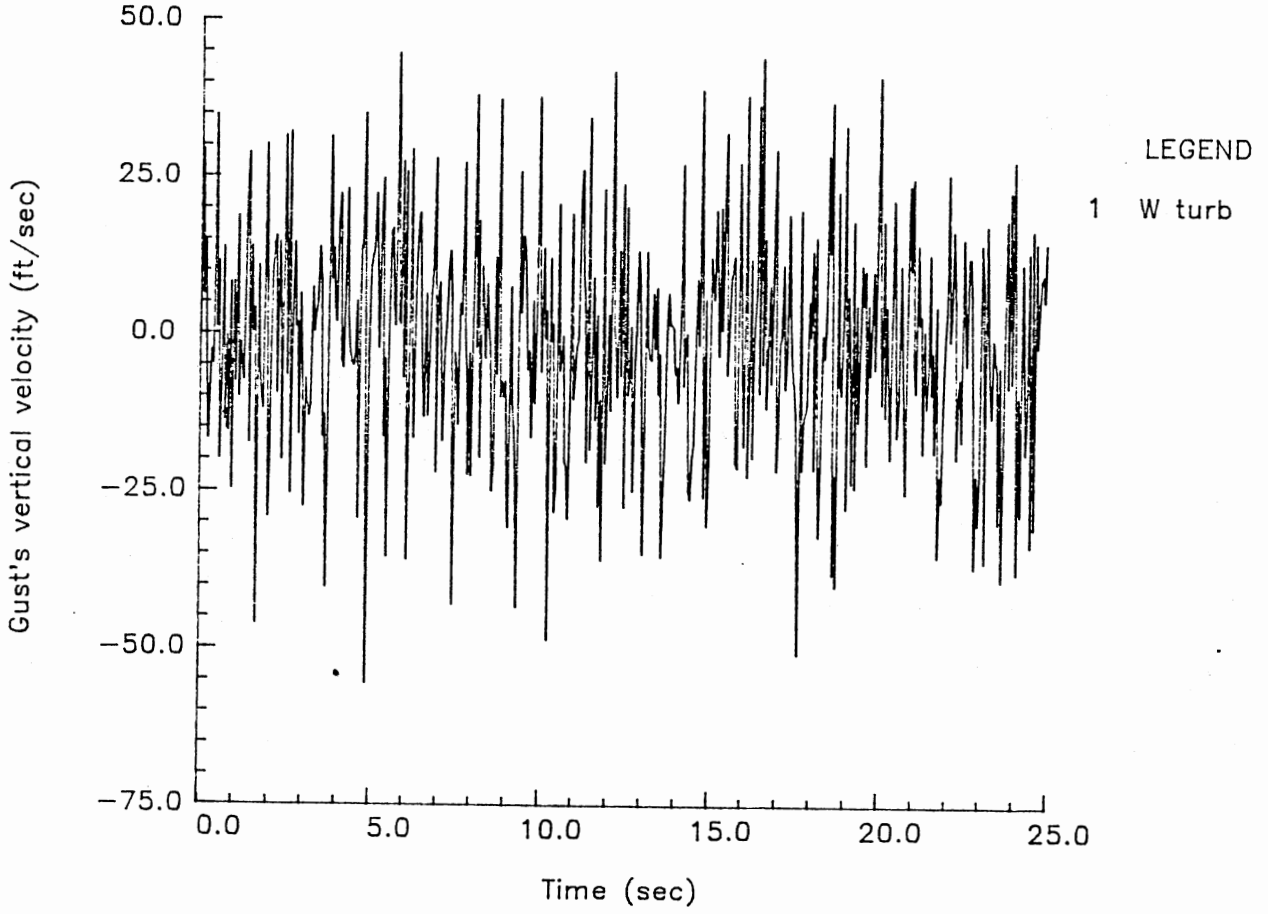

Figure 15. Gust's vertical velocity ($\alpha_g(k+1) = 0.5\ \alpha_g(k) + e_\alpha(k)$ with $\sigma^2_e = 0.0005$).

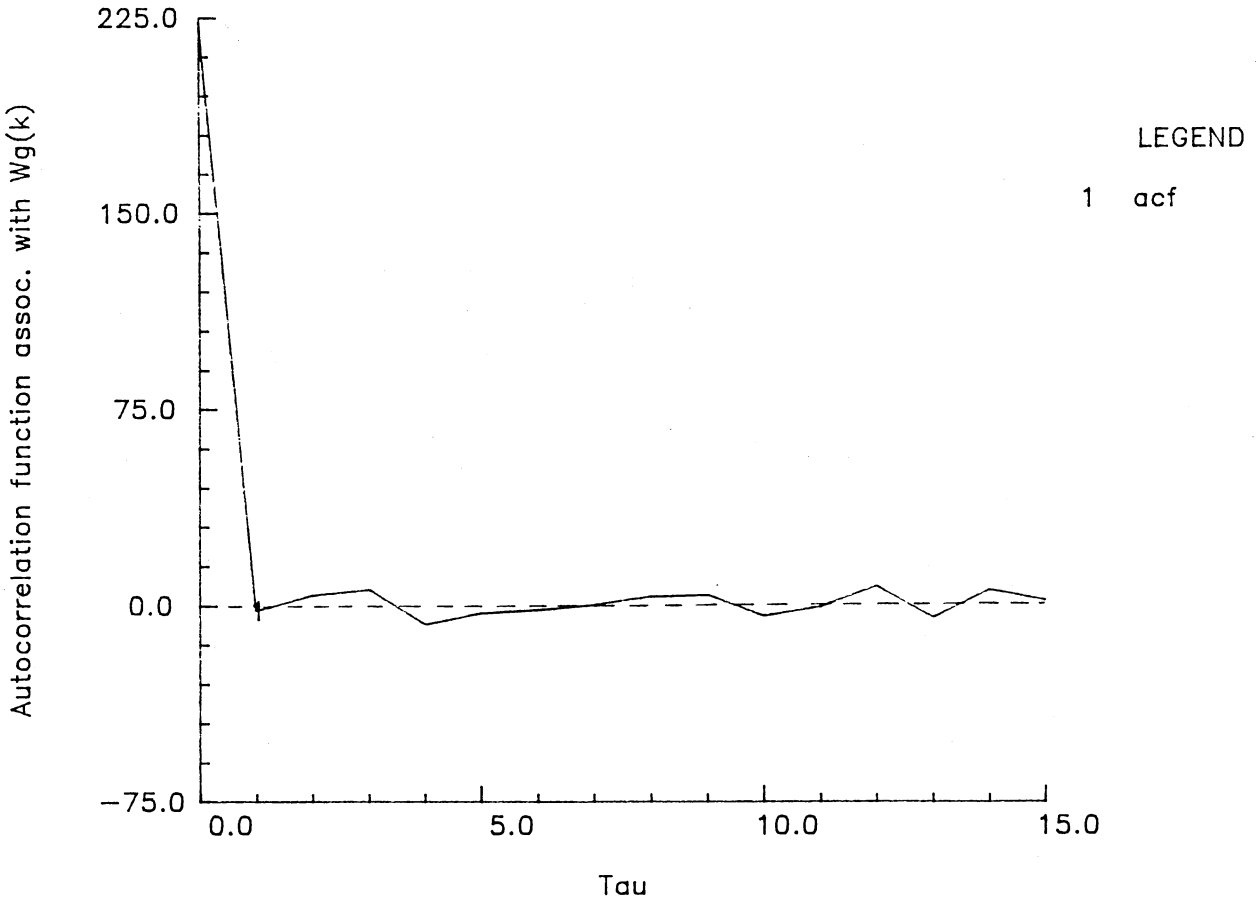

Figure 16. Autocorrelation function of the gust's vertical velocity ($\alpha_g(k)$ = white noise, σ^2_α = 0.0005)

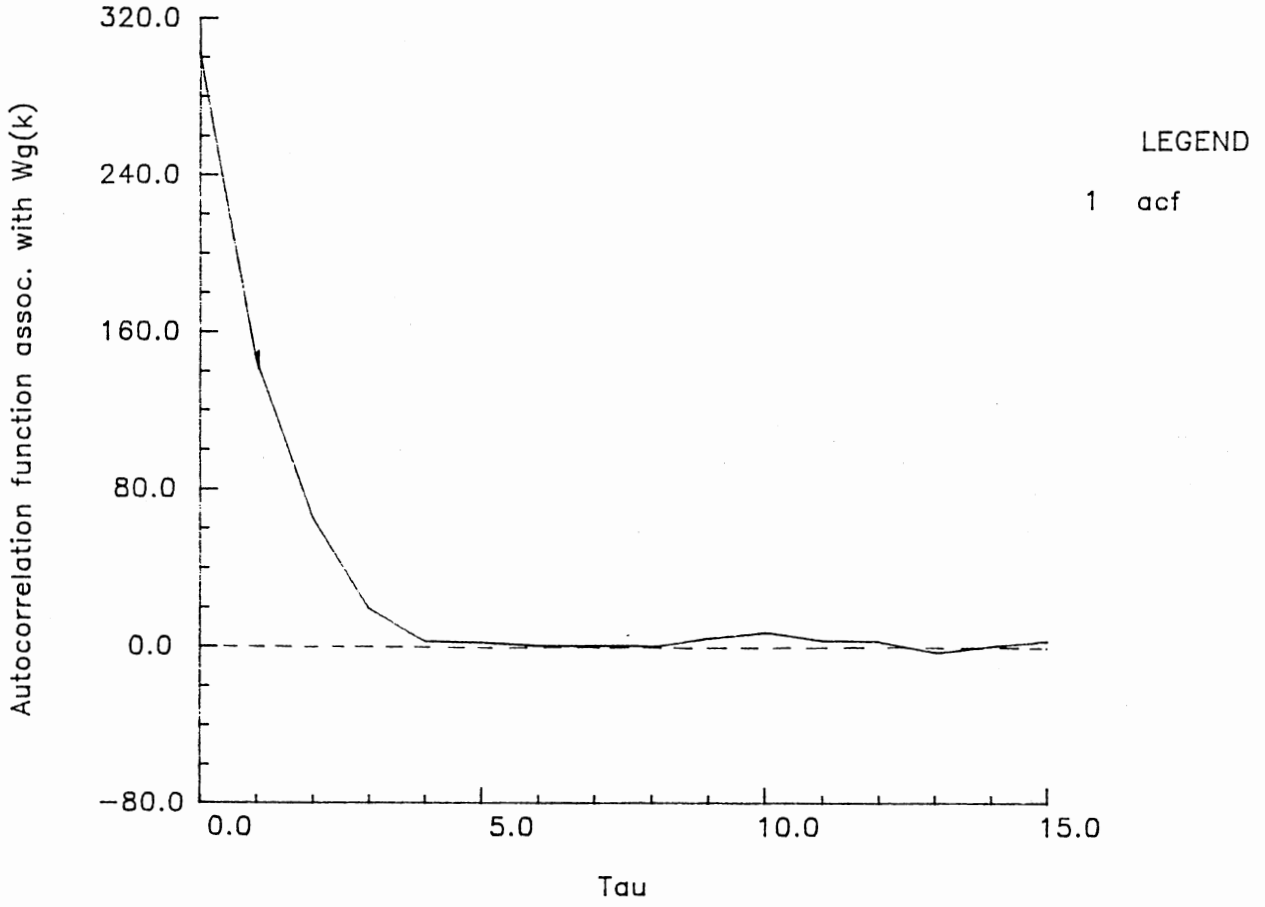

Figure 17. Autocorrelation function of the gust's vertical velocity ($\alpha_g(k+1) = 0.5\ \alpha_g(k) + e(k)$, $\sigma^2_e = 0.0005$).

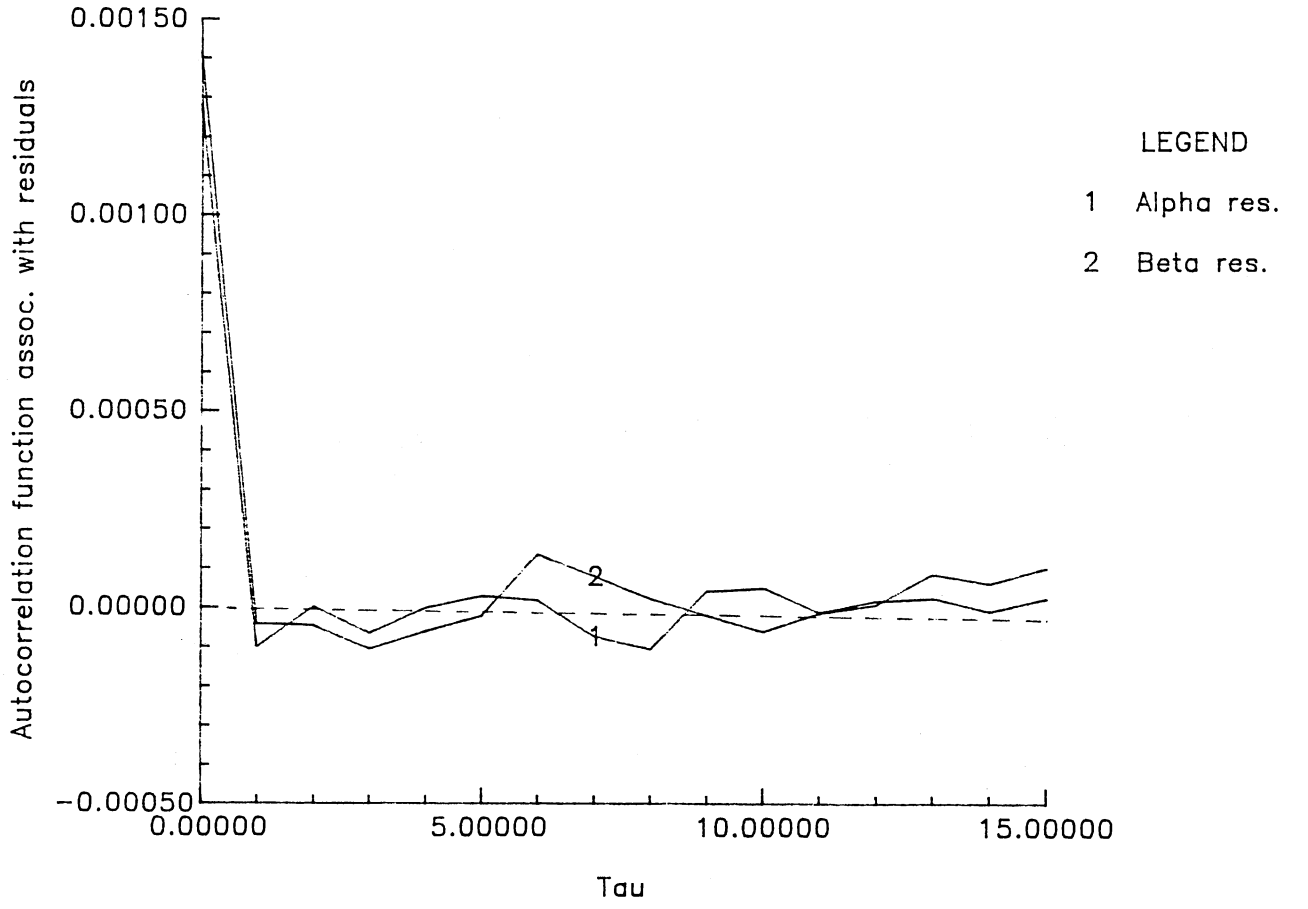

Figure 18. Autocorrelation function of the residuals for the states α and β associated with uncorrelated atmospheric turbulence.

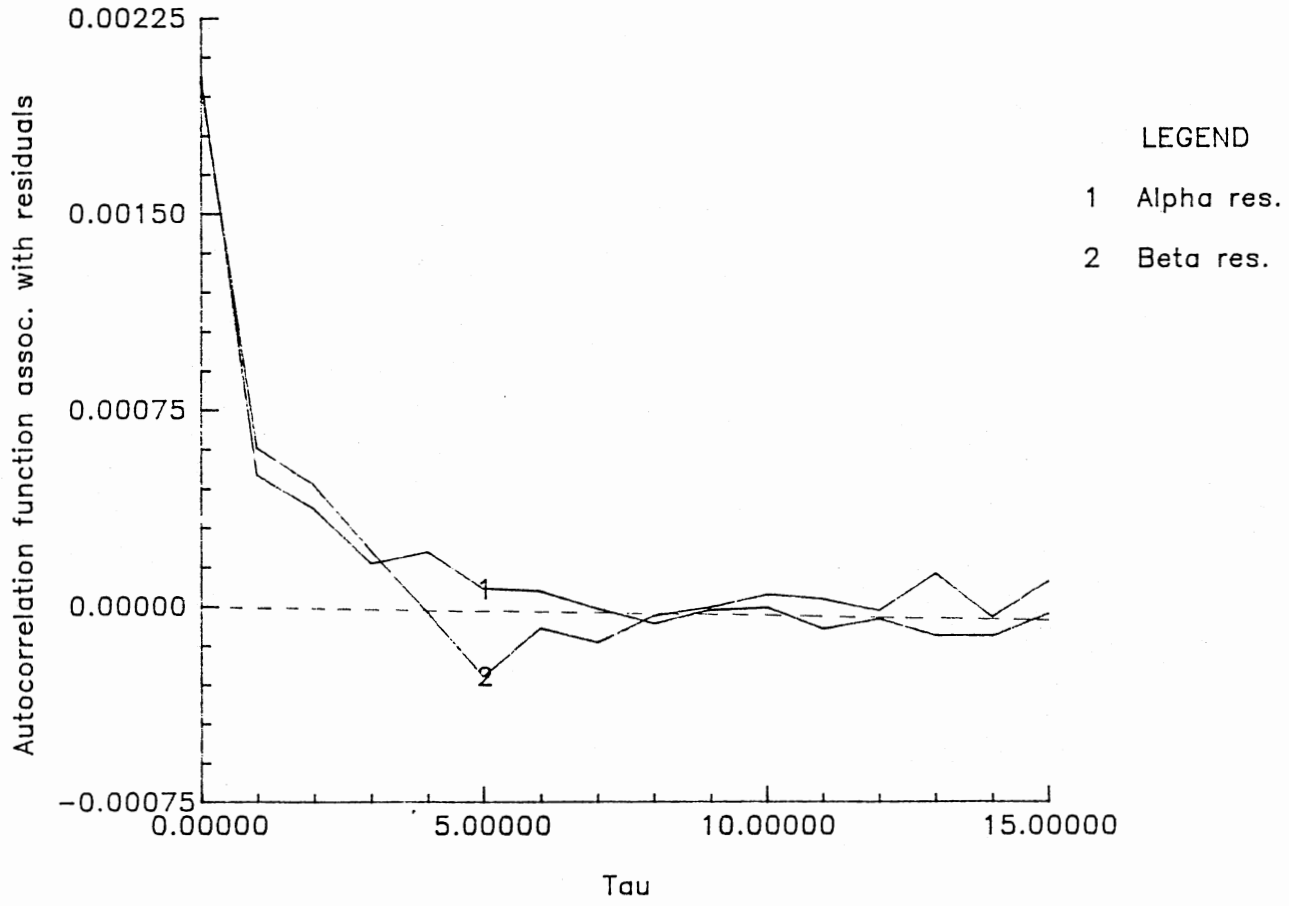

Figure 19. Autocorrelation function of the residuals for the states α and β associated with correlated atmospheric turbulence.

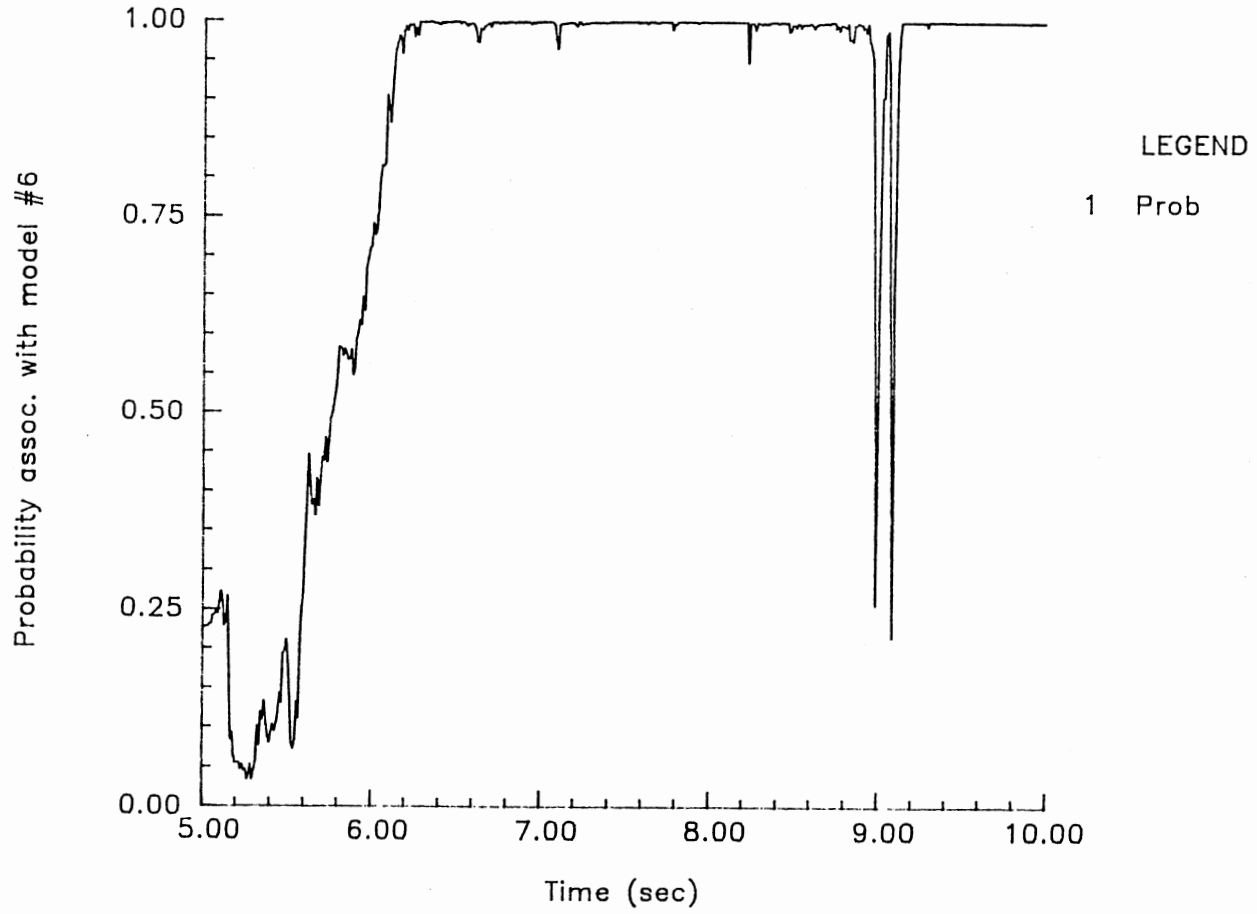

Figure 20. Probability convergence to the closest model (N = 12, $C_{L_{\delta EL}} = 0.121$) for correlated atmospheric turbulence.

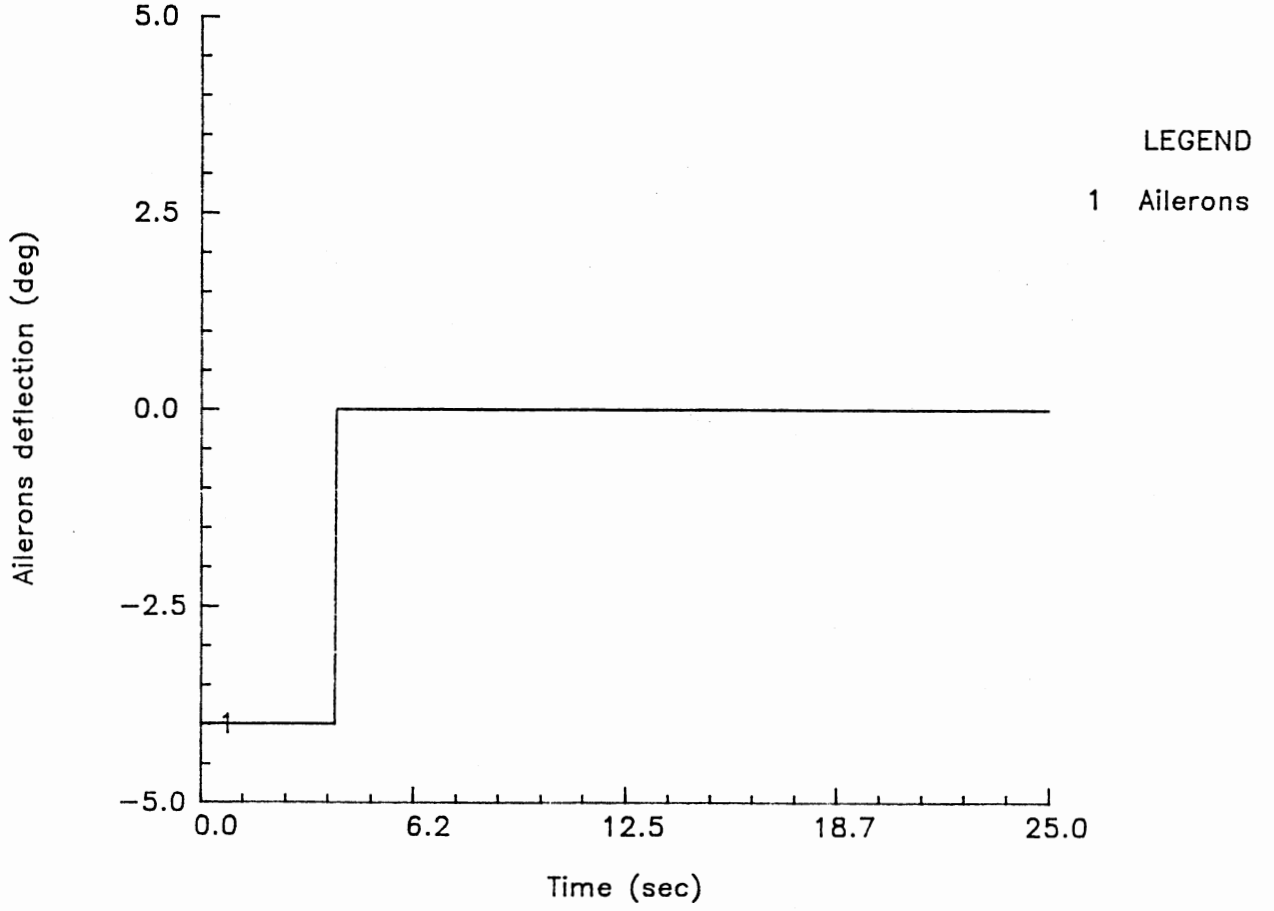

Figure 21. Ailerons inputs.

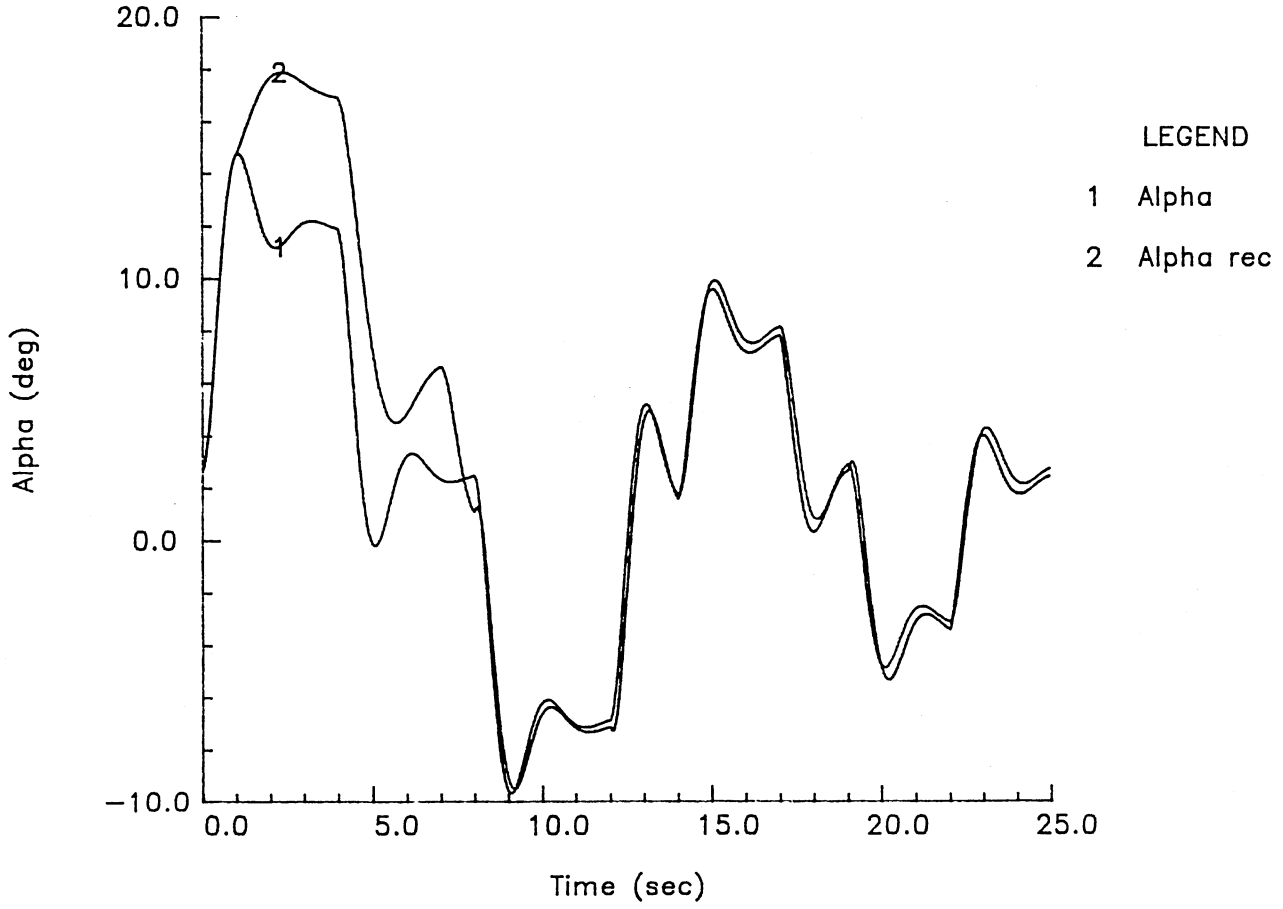

Figure 22. CASE 1 - Angle of attack vs. time (1-nominal; 2-reconfigured).

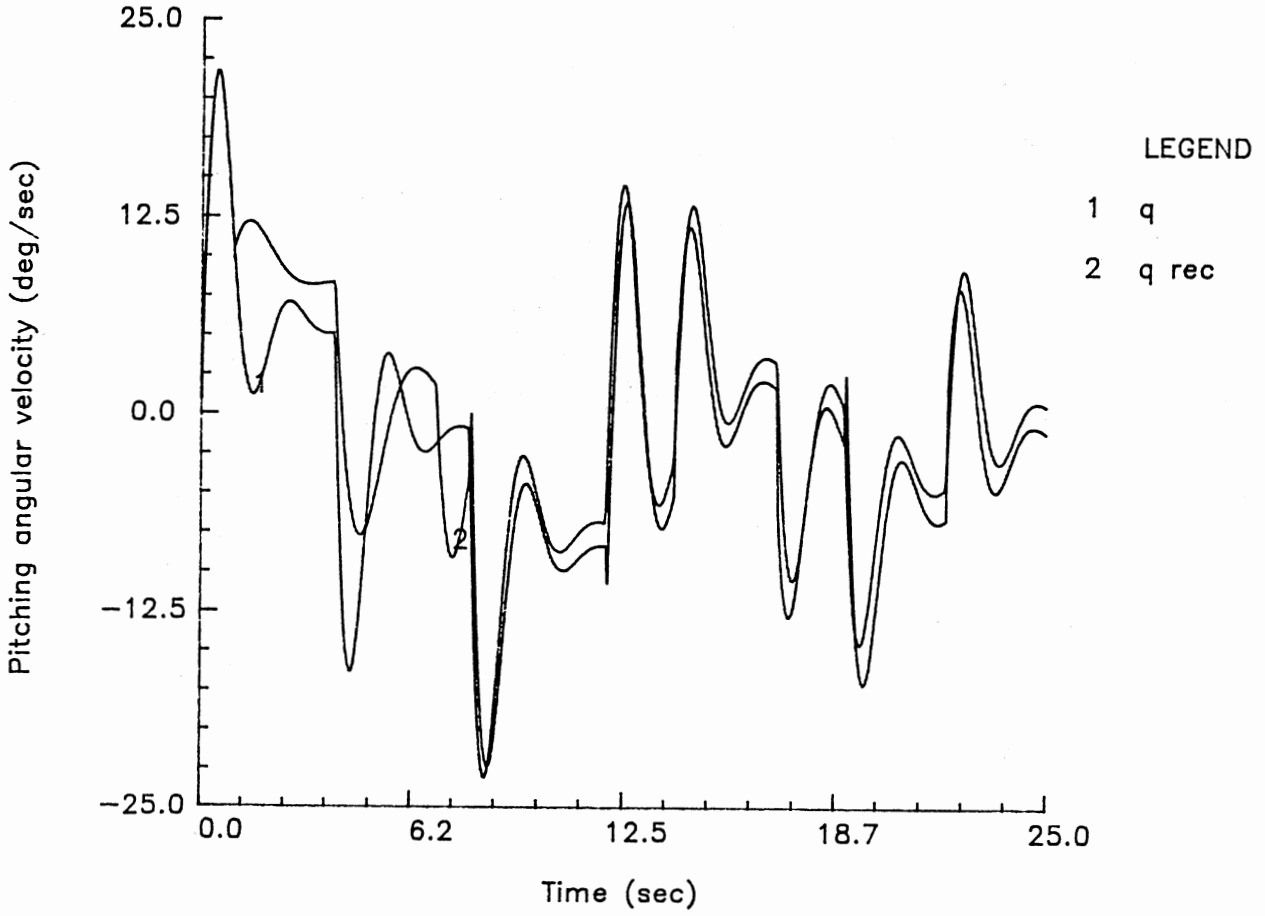

Figure 23. CASE 1 - Pitch angular velocity vs. time (1-nominal; 2-reconfigured).

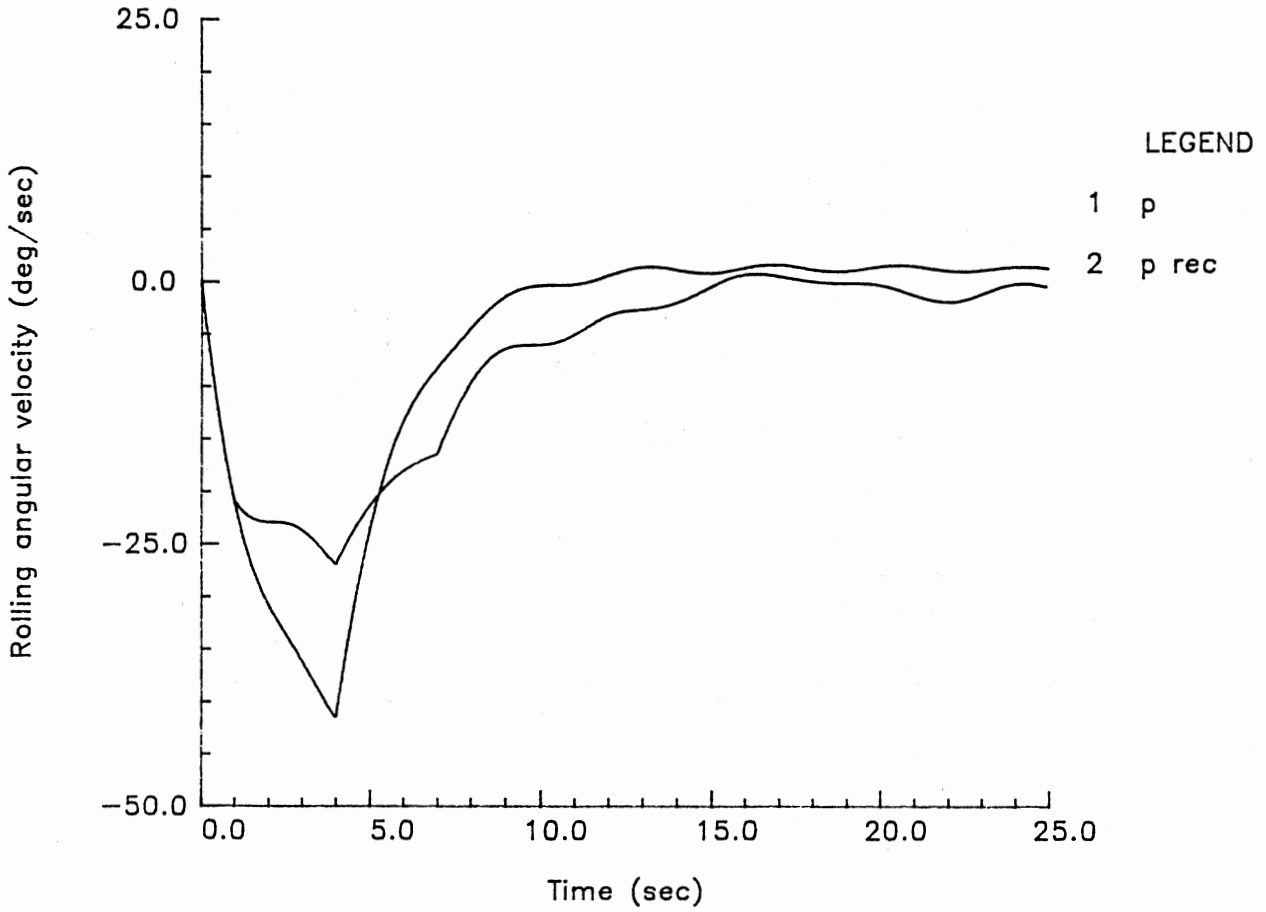

Figure 24. CASE 1 - Roll angular velocity vs. time (1-nominal; 2-reconfigured).

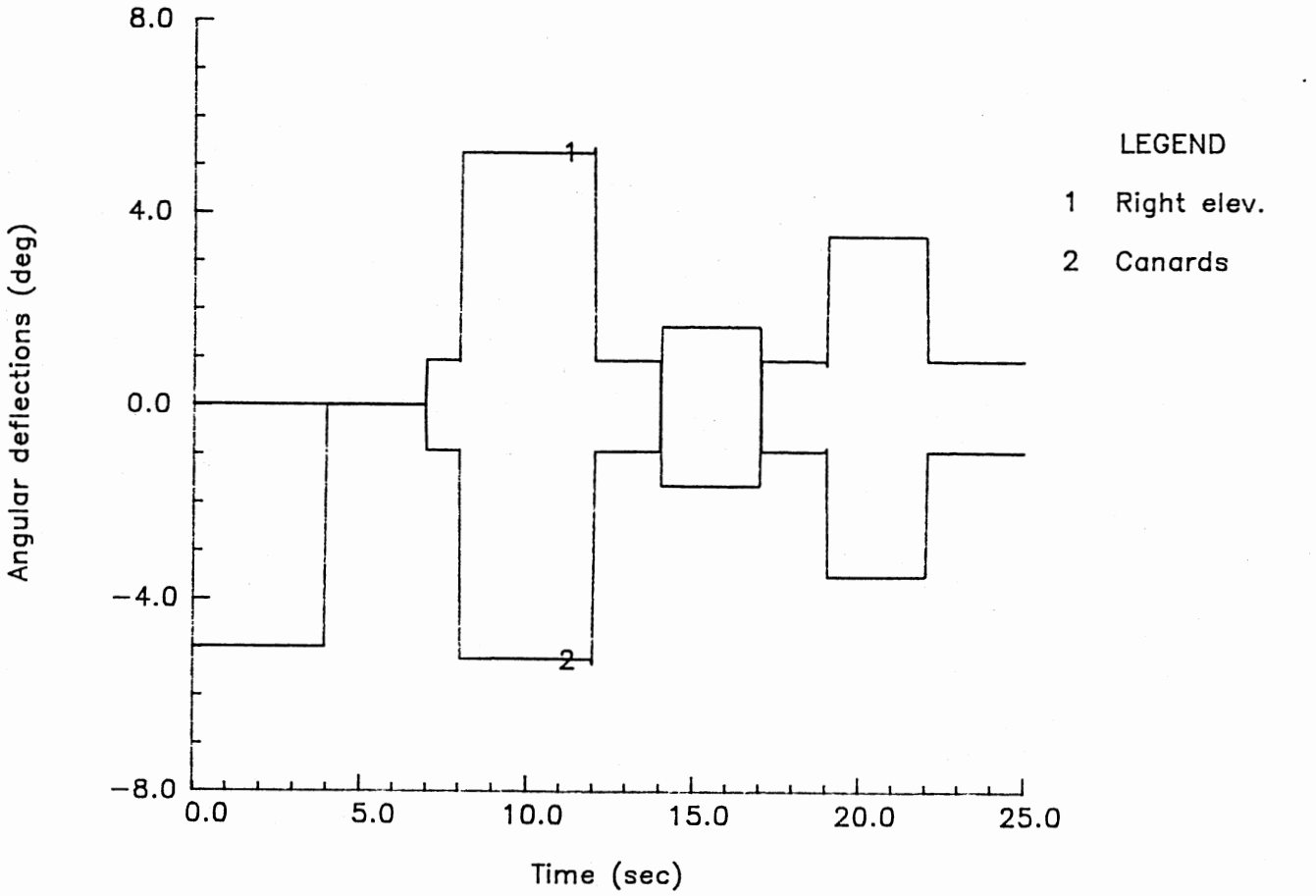

Figure 25. CASE 1 - Right elevator and canards deflections inputs.

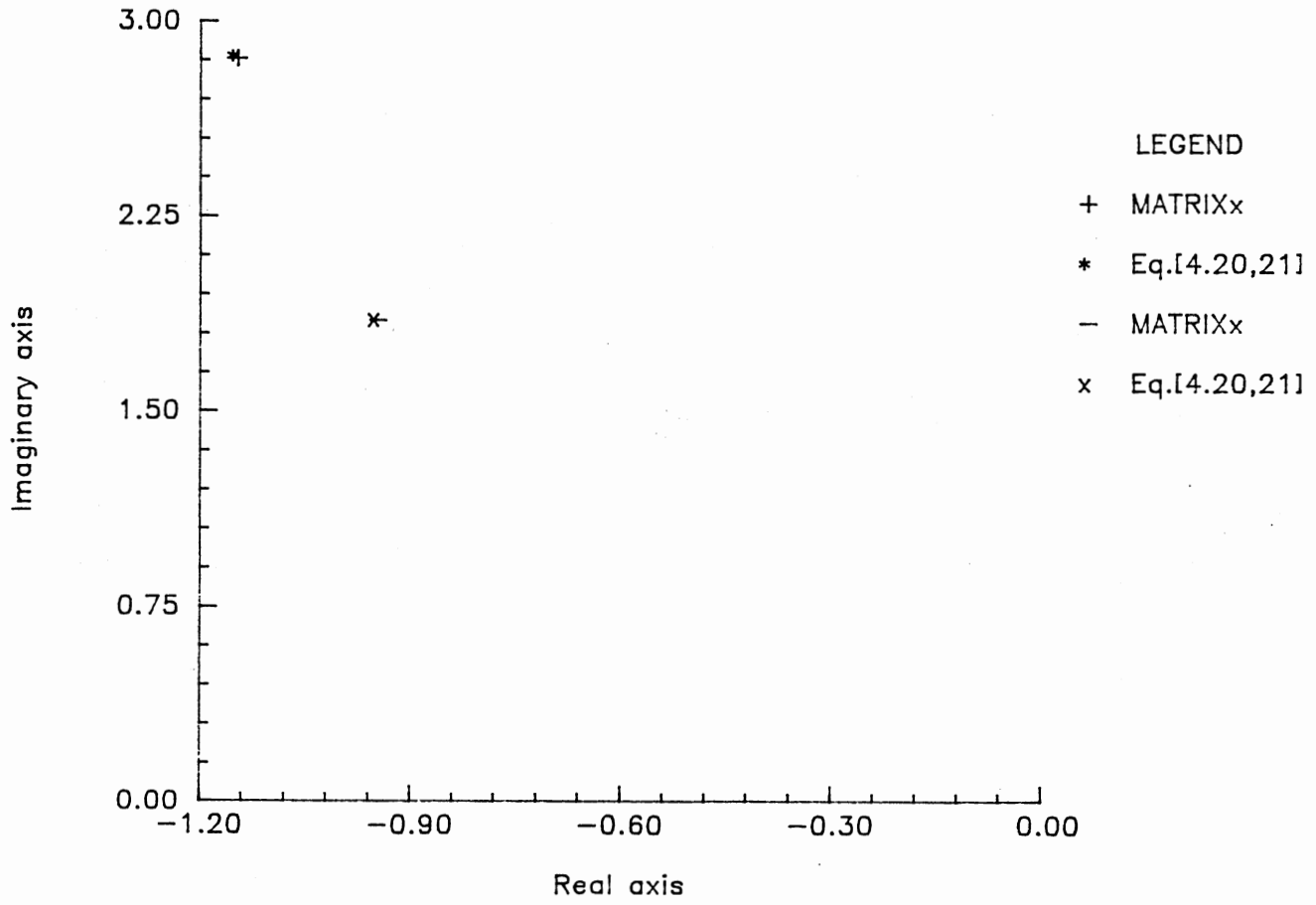

Figure 26. Short period poles location for nominal and damaged conditions.

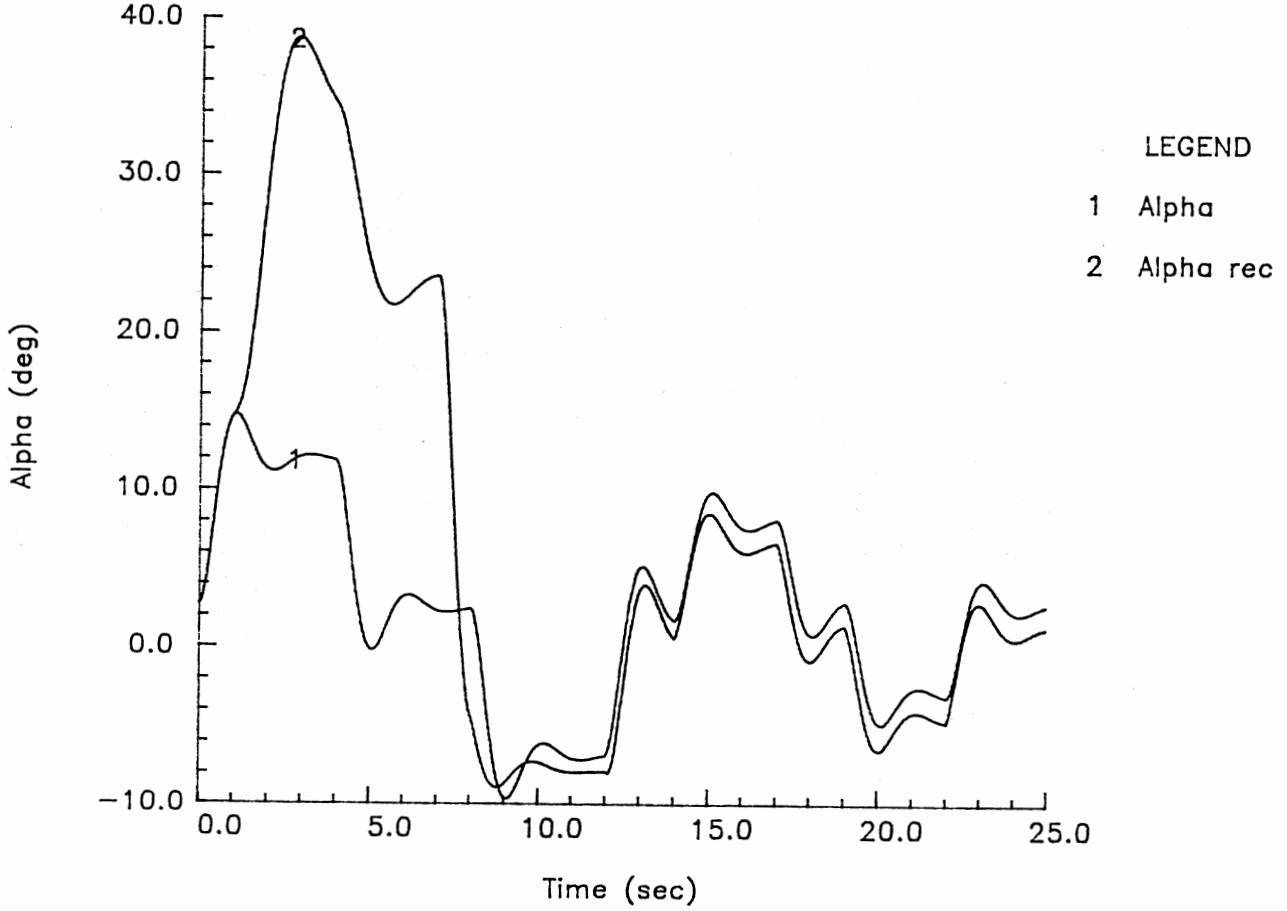

Figure 27. CASE 2 - Angle of attack vs. time (1-nominal; 2-reconfigured).

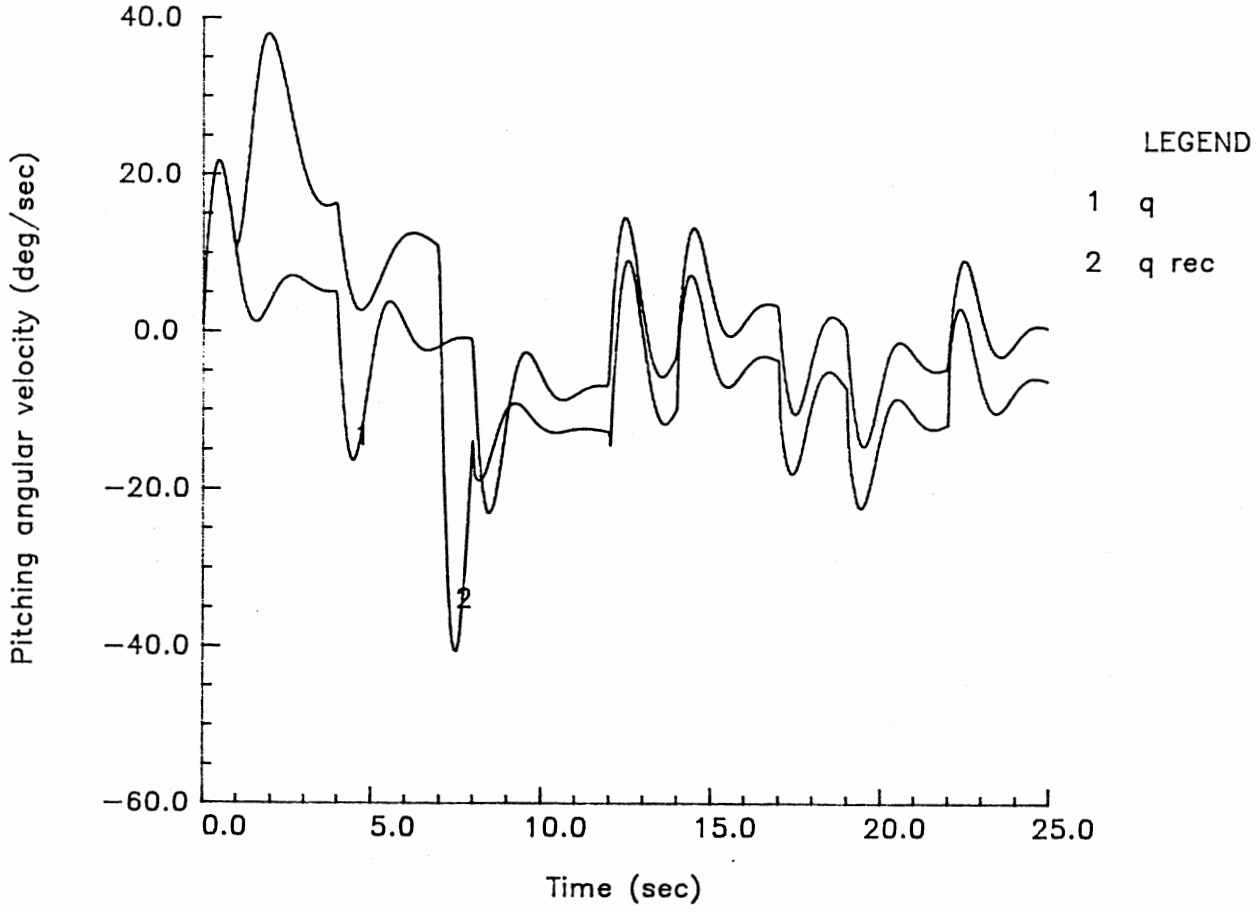

Figure 28. CASE 2 - Pitch angular velocity vs. time (1-nominal; 2-reconfigured).

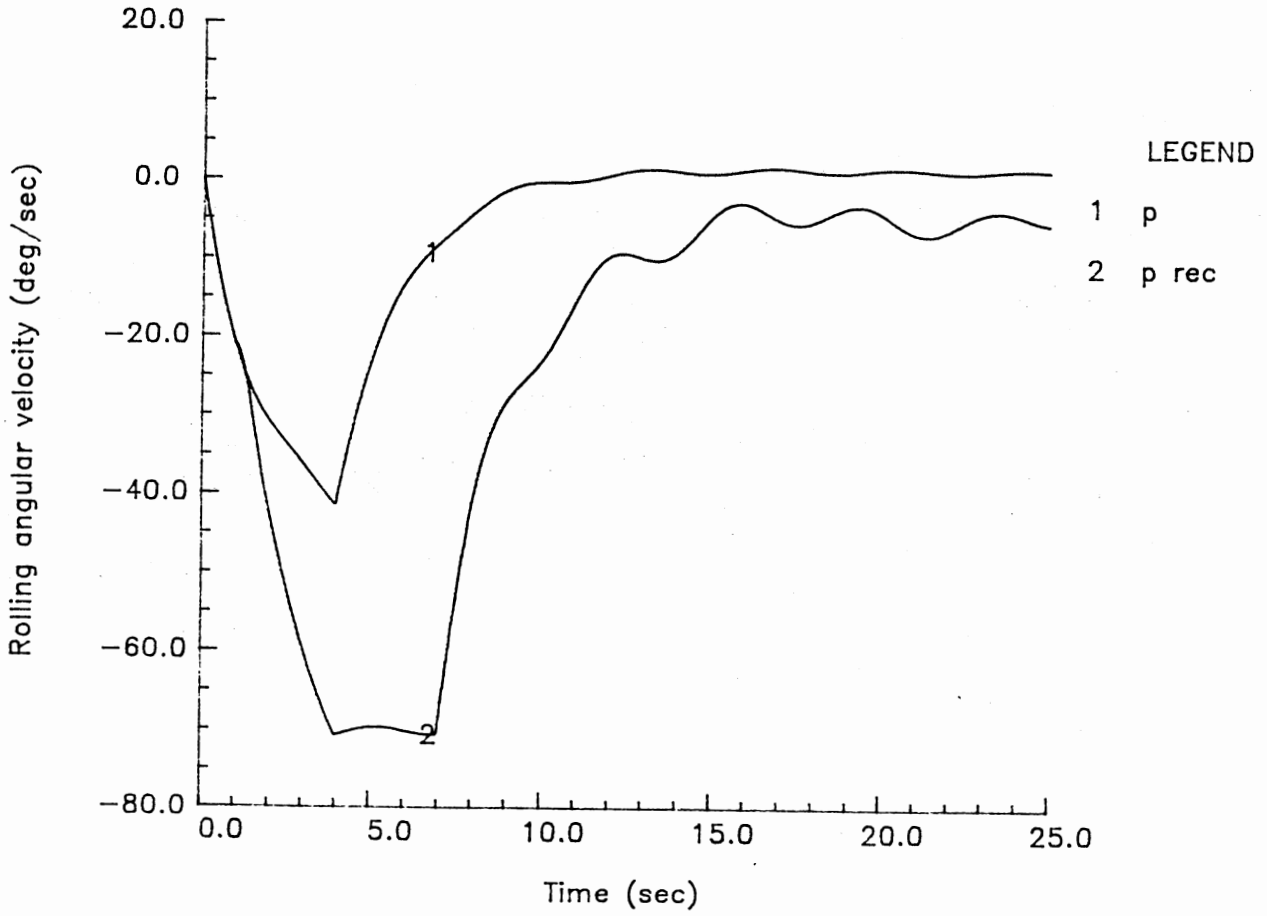

Figure 29. CASE 2 - Roll angular velocity vs. time (1-nominal; 2-reconfigured).

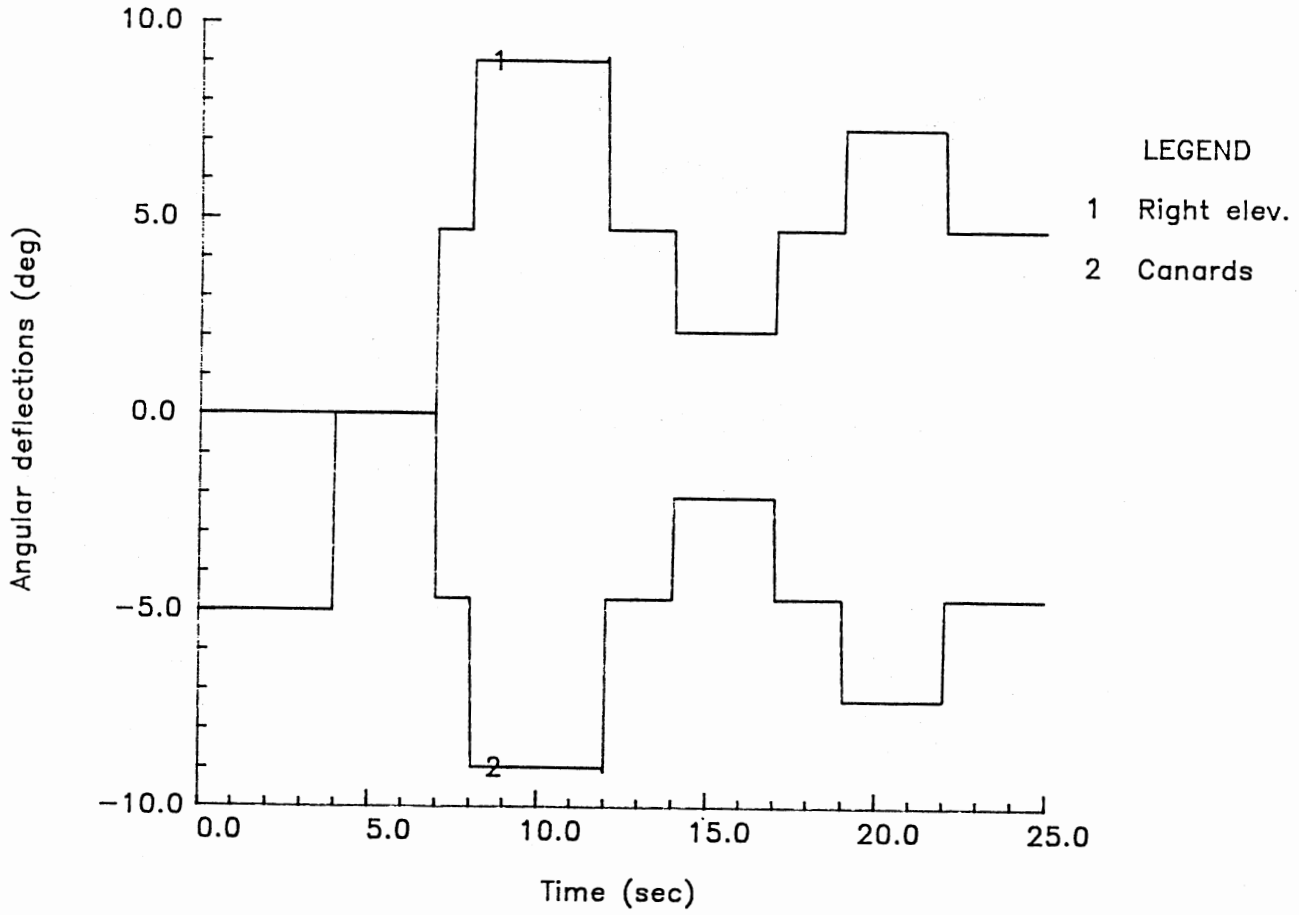

Figure 30. CASE 2 - Right elevator and canards deflections inputs.

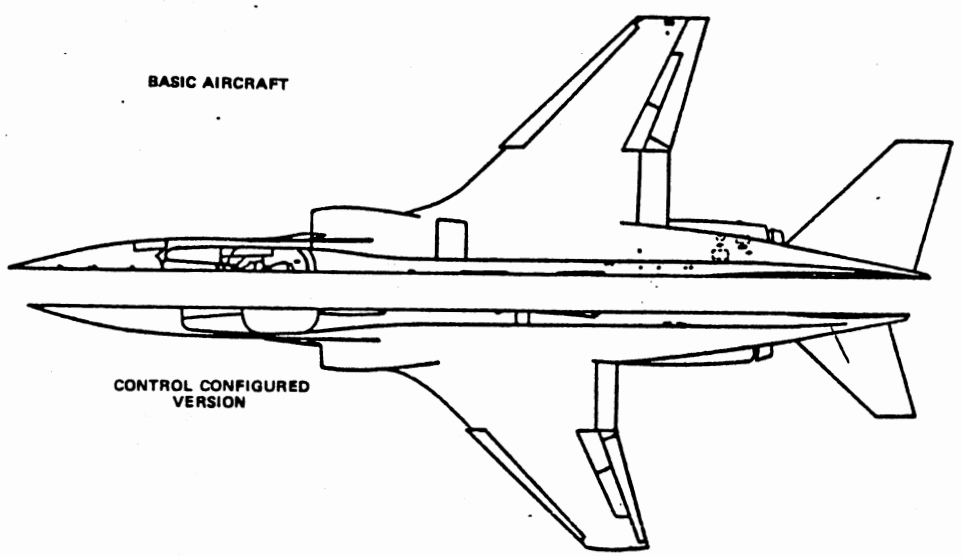

Figure 31. Effect of 'artificial' stability on a fighter aircraft size.

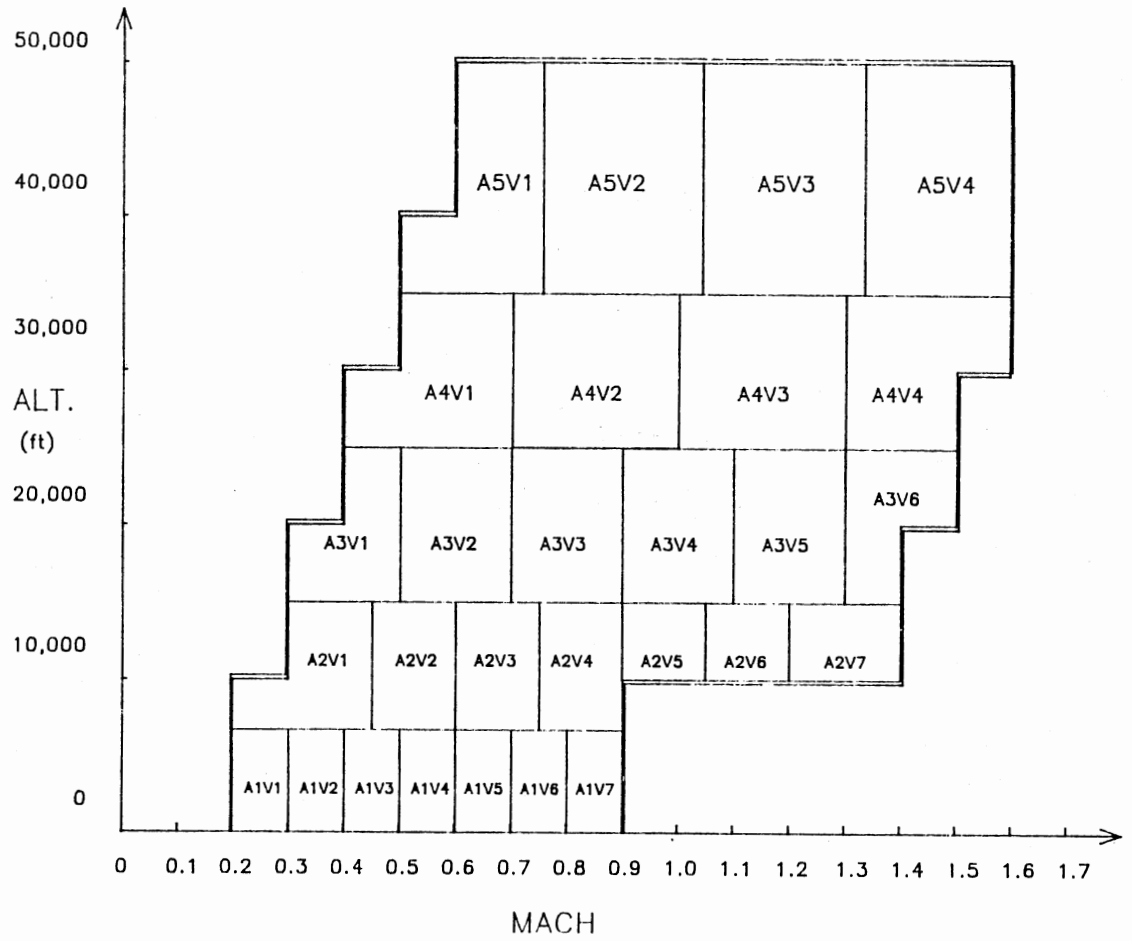

Figure 32. Typical Flight Envelope for a fighter.

APPENDIX C

KALMAN FILTER

KALMAN FILTER

In order to review the Kalman Filter (Ref.[23]-[26]), recall the state variable equations introduced in Chapter III:

$$X(k+1) = A\,X(k) + B\,U(k) + L\,W(k) \tag{C.1}$$

$$Z(k) = C\,X(k) + V(k) \tag{C.2}$$

We assume that:

$$E\,[\,W(i)\,W^T(j)\,] = Q\,\delta_{ij} \tag{C.3}$$

$$E\,[\,V(i)\,V^T(j)\,] = R\,\delta_{ij} \tag{C.4}$$

$$E\,[\,W(i)\,V^T(j)\,] = 0 \tag{C.5}$$

with $\delta_{ij} = 1$ for $i = j$

$\delta_{ij} = 0$ for $i <> j$

$X(0)$ is independent of $W(i)$'s and $V(i)$'s, with:

$$X(0) = N\,(m_X(0), P_X(0))$$

We are assuming a stationary process.

Our objective is to find an estimate X, given measurements Z. Particularly, we are looking for:

$\hat{X}(k/k)$ = best estimate of X at time 'k', given the measurements at time 'k'.

We can write $\hat{X}(k/k)$ as:

$$\hat{X}(k/k) = E\,[X(k)/z(1), z(2), \ldots., z(k)] \tag{C.6}$$

The term on the right hand side of Eq. (C.6) is a Conditional Mean.

This estimate will minimize:

$$P_x(k/k) = P(k/k) = E\left[(X(k)-\hat{X}(k/k))\ (X\ k)-\hat{X}(k/k))^T\right] \qquad (C.7)$$

The overall Kalman Filter process can be broken down into two separate parts:

PREDICTION: predict X(k+1), given data up to instant 'k';

FILTERING: update the estimation with data at instant 'k+1'.

The equations for these two parts are given next:

PREDICTION

$$\hat{X}(k+1/k) = A\ \hat{X}(k/k) + B\ U(k) \qquad (C.8)$$

$$P(k+1/k) = A\ P(k/k)\ A^T + LQL^T \qquad (C.9)$$

FILTERING

$$\hat{X}(k+1/k+1) = X(k+1/k) + K(k+1)\ \tilde{Z}(k+1/k) \qquad (C.10)$$

where $$\tilde{Z}(k+1/k) = Z(k+1) - C\ \hat{X}(k+1/k) \qquad (C.11)$$

is called 'Innovation Sequence' and its elements are called Filter Residuals.

$$K(k+1/k) = P(k+1/k)\ C^T\ [\ C\ P(k+1/k)\ C^T + R\ ]^{-1} \qquad (C.12)$$

$$P(k+1/k+1) = [\ I - K(k+1)\ C\ ]\ P(k+1/k) \qquad (C.13)$$

with the INITIAL CONDITIONS

$$\hat{X}(0/0) = m_x(0) \qquad (C.14)$$

$$P(0/0) = E\ [\ (X(0) - m_x(0))\ (X(0) - m_x(0))^T\ ] \qquad (C.15)$$

Note that if the disturbance or the measurement noises are not white, the innovation sequence is also not white.

As previously stated, our system is stationary; that is, the matrices of Eqs.(C.1) and (C.2) are time-invariant, also the system is assumed to be asymptotically stable; that is, all the eigenvalues of A lie inside the unit circle. Therefore, we can introduce:

$$\lim_{k \to \infty} P(k+1/k) = \bar{P} \qquad (C.16)$$

which satisfies the following steady-state version of Eq. (C.13) combined with Eq. (C.12):

$$\bar{P} = A \bar{P} [I - C^T (C \bar{P} C^T + R)^{-1} C \bar{P}] A^T + L Q L^T \qquad (C.17)$$

Eq.(C.17) is often referred as steady-state or algebraic Riccati equation. The steady-state version of Eq.(C.12) is :

$$\bar{K} = \bar{P} C^T [C \bar{P} C + R]^{-1} \qquad (C.18)$$

The eigenvalues of the steady-state Kalman Filter all lie within the unit circle, so that the filter is asymptotically stable; this means that:

$$| \text{eig}_i [A - \bar{K} C A] | < 1 \qquad (C.19)$$

If the dynamical model described in the Eqs. (C.1) and (C.2) is time-invariant and stationary, but it is not asymptotically stable, it can be shown that Eqs. (C.17),(C.18) and (C.19) still hold as long as the system is at least stabilizable (which means that all the unstable states are controllable) and detectable (which means that all the unobservable states are stable).

APPENDIX D

MODELING OF THE ATMOSPHERIC TURBULENCE

MODELING OF ATMOSPHERIC TURBULENCE

Atmospheric turbulence can be considered in aircraft dynamcs as a two- dimensional random vector $W(k)$, whose components are $\alpha_g(k)$ and $\beta_g(k)$. These are considered to be mutually independent, Gaussian, white noise random vectors with zero mean and known variances $\sigma^2_\alpha = \sigma^2_\beta = 0.0005$, which are the diagonal elements of the Q matrix. The off-diagonal elements of such matrix are zeros because of the mutual independence assumption.

The modeling of atmospheric turbulence has been and still is a topic of massive investigation and research (Ref. [27]-[30]). Several kinds of Gaussian and Non-Gaussian atmospheric turbulence models have been introduced in order to describe life turbulence and to reproduce it in flight simulators. A detailed description of atmospheric turbulence modeling procedures is beyond the scope of this study. However, it is worth mentioning that atmospheric turbulence can be described as components of the velocity ($U_g(k)$, $V_g(k)$, $W_g(k)$) acting on the airplane along the X,Y and Z body axes.

These components of the velocity behave essentially as random processes. For such a process the following definitions apply: If we collect an infinite number of time histories, such set is called an ensemble; a random process is stationary if its statistical properties are not dependent on the time of their measurement. In other

words, if we take an ensemble average across the ensemble at time t_1 and time t_2 and if these ensemble averages are not functions of the time, the process is stationary. A random process is homogenous if its statistical properties are independent of position. Thus, if atmospheric turbulence were homogenous (which is not), its statistical properties would be independent of altitude and geographical location. The lack of homogenoity is an important factor for modeling atmospheric turbulence for STOL flight simulators. Since, in turbulence measurements, it is virtually impossible to obtain ensembles of turbulence data from atmospheric measurements, it is necessary to use time averages to get statistical informations. If such a time average gives the same statistical properties as the ensemble average the process is called ergodic. Note that ergodicity imples stationarity. Realistically, turbulence is not an ergodic process; however, since only relatively short time histories are generally available, we can assume ergodicity. For the generic component U_g and W_g we can define the following quantities:

$$\underline{\text{MEAN VALUE}} = \bar{U} = 1/T \int_0^T U(t)dt = E[U] \qquad (D.1)$$

$$\underline{\text{VARIANCE}} = \sigma^2_u = 1/T \int_0^T [U(t) - \bar{U}]^2 dt = E[(U - E[U])^2] \qquad (D.2)$$

$$\underline{\text{AUTO-CORRELATION FUNCTION}} = R_u(\tau) = E[U(t)U(t+\tau)] \qquad (D.3)$$

$$\underline{\text{CROSS-CORRELATION FUNCTION}} = R_{uw}(\tau) = E[U(t)W(t+\tau)] \qquad (D.4)$$

$$\underline{\text{POWER SPECTRAL DENSITY}} =$$

$$\Phi_u(f) = \int_{-\infty}^{\infty} R_u(\tau)\, e^{-i\,(2\pi f)\tau} d\tau \qquad (D.5)$$

The white noise is a particular random process for which the power spectral density is a constant, independent of frequency. We also can define:

CROSS POWER SPECTRAL DENSITY =

$$\Phi_{uw}(f) = \int_{-\infty}^{\infty} R_{uw}(t)\, e^{-1(2\pi f)\tau}\, d\tau \qquad (D.6)$$

As long as the variables $U_g(t)$, $V_g(t)$ and $W_g(t)$ can be translated into an electronic signal, a power spectrum can actually be measured by means of filters or an harmonic analyzer. This is the manner in which turbulence data are usually represented and this is the reason why the turbulence power spectral density is a most important statistical parameter. The most accurate power spectral densities forms were proposed by Theodore Von Karman (Ref. [27]). However, these forms are not very convenient for turbulence model work because they cannot be exactly matched using linear filters.

This problem is usually overcome by assuming the power spectral density suggested by H.L. Dryden (Ref. [27]-[29]):

$$\Phi_{u_g}(\Omega) = \sigma^2_u \frac{2\, L_u}{\pi} \frac{1}{[1 + (L_u\, \Omega)^2]} \qquad (D.7)$$

$$\Phi_{v_g}(\Omega) = \sigma^2_v \frac{2\, L_v}{\pi} \frac{[1 + 3(L_v\, \Omega^2)]}{[1 + (L_v \Omega)^2]} \qquad (D.8)$$

$$\Phi_{w_g}(\Omega) = \sigma^2_w \frac{2\, L_w}{\pi} \frac{[1 + 3(L_w\, \Omega)^2]}{[1 + (L_w\, \Omega)^2]} \qquad (D.9)$$

where L_u, L_v, L_w are scale lengths;

Ω is a spatial frequency ($\Omega = 2\pi f/Vel$), where Vel is

the aircraft true airspeed.

In general the scale lengths and the variances of each gust component are expected to be functions of the heading angle, altitude, atmospheric stability and mean wind speed relative to the surface. Explicit relationships, results of a massive data investigation, are available (Ref. [30]) and can be implemented on a flight computer. On the other hand, very little is known about the cross spectral densities of the U_g, V_g and W_g gust components, particularly with variation of altitude, heading angle and other motion variables. However, such cross effects between the gust components are neglectable at high altitudes since they are mostly due to ground effects. The importance of the Eqs. (D.7), (D.8) and (D.9) is the fact that we can calculate σ^2_u, σ^2_v, σ^2_w as:

$$\sigma^2_i = {}_o\int^{\infty} \Phi_i (\Omega)\, d\Omega \quad \text{with } i = u,v,w \tag{D.10}$$

Once we calculated these variances we can then calculate σ^2_α and σ^2_β by using:

$$\alpha_g = W_g/U_1 \tag{D.11}$$

$$\beta_g = V_g/U_1 \tag{D.12}$$

where U_1 is the aircraft forward speed. Therefore, we have:

$$\Phi_{\alpha_g} (\Omega) = 1/U_1^2\, \Phi_{w_g} (\Omega) \tag{D.13}$$

$$\Phi_{\beta_g} (\Omega) = 1/U_1^2\, \Phi_{v_g} (\Omega) \tag{D.14}$$

$$\sigma^2_{\alpha_g} = 1/U_1^2\, \sigma^2_{w_g} \tag{D.15}$$

$$\sigma^2_{\beta_g} = 1/U_1^2\, \sigma^2_{v_g} \tag{D.16}$$

APPENDIX E

COMPUTER PROGRAM LISTINGS

COMPUTER PROGRAM LISTINGS

Damaged Model Estimation

A certain number of computer codes have been written for the purpose of estimating the mathematical model of the aircraft following damage. All the non-MATRIX$_x$ codes have been written in Pascal and implemented on the HP-9000.

First the continuous-time state variable models of the aircraft for damaged and undamaged conditions are calculated by using the program MODELING.P, with the aircraft data of Table I contained in the input file DATA1.DAT. Such a program has been used iteratively for calculating the set of N A_c and B_c matrices. These matrices, discretized by using MATRIXx, are then given as input to the program MODAUX.P. This is an auxiliary program for creating time-histories reflecting the occurrence of the damage in discrete-time with randomly generated noise with desired statistics. The main output files of this program are the data files KALMAN1.DAT and KALMAN2.DAT which contains the state variables data and the control data for Kalman Filter analysis. On the other hand, using MATRIX$_x$, with the user-defined function FILTER.FNC two sets of N=12 and N=23 steady-state Kalman Filter gain matrices (KE), inverse of the covariance of the residual matrices (S^{-1}) and β constants are calculated. The β constants, the KE, S^{-1}

matrices along with the matrices A, B, Q, L and R matrices are stored in the data files MODMM1.DAT and MODMM2.DAT, for N=12 and N=23, respectively.

The files KALMAN1.DAT, KALMAN2.DAT along with the files MODMM1.DAT (for N=12) or MODMM2.DAT (for N=23) are the input files of the program MODMMKF.P which performs the model estimation using the Multiple Model Kalman Filtering. The output of the MODMMKF.P program is the file PROB.DAT, which contains the probabilities associated with each A and B configuration.

The probability convergence sensitivity analysis for different values of the Q matrix is performed by changing the values of the Q matrices in the program MODAUX.P. A modification of the program MODMMKF.P is the program FILMODMMKF.P, where the probabilities filtering is performed.

The probability convergence sensitivity analysis for correlated atmospheric turbulence is instead performed using the program CORMODAUX.P where the discrete-time dynamics is simulated with a 10-th order system rather than a 8-th order system. The corresponding data files CORKALMAN1.DAT and CORKALMAN2.DAT are then used in the program MODMMKF.P which generates the file CORPROB.DAT. The analysis of the differences of the S^{-1} matrices for correlated and correlated atmospheric turbulence is instead performed with the $MATRIX_X$ user defined functions CORSTEP2.FNC and STEP2P2.FNC. We had to break down the analysis in two user-defined functions because of the large amount of memory required by the manipulation of the large size matrices involved in the analysis.

In the next pages the listings of the following programs are

enclosed:

MODELING.P

MODAUX.P

MODMMKF.P

along with the MATRIX$_X$ user-defined functions:

FILTER.FNC

CORSTEP2.FNC

STEP2P2.FNC .

```
program modeling (input,output);
type    data1 = array[1..8,1..8] of real;
        data2 = array[1..8] of real;
        data3 = array[1..8,1..3] of real;
        data4 = array[1..8,1..6] of real;

var     w,h,ro,mach:real;                (* flight conditions *)
  betald,thetald,alphald,phild:real; (* angles with respect stability a
  beta1,theta1,alpha1,phi1:real;       (* angles with respect stability a
        cseg,s,xcg,bs:real;            (* geometric parameters *)
        u1,v1,w1:real;                 (* initial linear velocity *)
        p1,q1,r1:real;                 (* initial angular velocity *)
        iyy,ixxb,izzb,ixzb:real;       (* weight distribution in body axes
        ixxs,izzs,ixzs:real;           (* weight distribution in stab axes

        (*longitudinal aerodynamic dimensionless and dimensional derivat
var     cdu,cdl,ctxu,ctxl,cda,cdde :real;
        cmu,cml,cmtu,cmtl,cma,cmta,cmadot,cmq,cmde :real;
        cll,clu,cla,cladot,clq,clde :real;
        xu,xtu,xa,xde :real;
        zu,za,zadot,zq,zde :real;
        mu,mtu,ma,mta,madot,mq,mde: real;

var     dcdu,dcdl,dctxu,dctxl,dcda,dcdde :real;
        dcmu,dcml,dcmtu,dcmtl,dcma,dcmta,dcmadot,dcmq,dcmde: real;
        dcll,dclu,dcla,dcladot,dclq,dclde: real;
        dxu,dxtu,dxa,dxde:real;
        dzu,dza,dzadot,dzq,dzde:real;
        dmu,dmtu,dma,dmta,dmadot,dmq,dmde :real;

        (*lat-direz. aerodynamic dimensionless and dimensional derivatiu
var     cyb,cyp,cyr,cyda,cydr: real;
        clb,clp,clr,clda,cldr: real;
        cnb,cntb,cnp,cnr,cnda,cndr :real;
        yb,yp,yr,yda,ydr:real;
        lb,lp,lr,lda,ldr:real;
        nb,ntb,np,nr,nda,ndr:real;
        lder,ldel,ldsr,ldsl,ldcr,ldcl : real;

 var    dcyb,dcyp,dcyr,dcyda,dcydr:real;
        dclb,dclp,dclr,dclda,dcldr:real;
        dcnb,dcntb,dcnp,dcnr,dcnda,dcndr:real;
        dyb,dyp,dyr,dyda,dydr:real;
        dlb,dlp,dlr,dlda,dldr:real;
        dnb,dntb,dnp,dnr,dnda,dndr:real;

        (* control derivatives for the reconfiguration task distribution
var     cldel,clder,cldsl,cldsr,cldcr,cldcl : real;
        cmdel,cmder,cmdsl,cmdsr,cmdcr,cmdcl : real;
        croldcr,croldcl,croldsr,croldsl,crolder,croldel : real;
```

```
var     recdzdel,recdzder : real;
        recdzdsl,recdzdsr : real;
        recdzdcl,recdzdcr : real;
        recdmdel,recdmder : real;
        recdmdsl,recdmdsr : real;
        recdmdcl,recdmdcr : real;

 var    fildat1,fildat2,fildat3: string[20];
        f1,f2,f3: text;
 var    l,n,i,j,k : integer;
        pos,side :integer;
 var    a1,b1,c1,d1 : real;
 var    aa,daa : data1;
        bb,dbb : data3;
        recbb : data4;
 var    run,ans : char;

 procedure intro;
begin
writeln('*************************************************************************
writeln;
writeln('          W E L C O M E                  T O      ');
writeln;
writeln('                  M O D E L I N G .  P  ');
writeln;
writeln('*************************************************************************
writeln;
writeln('    This program calculates the continuous-time A and B matrice
writeln('of the aircraft for damaged and undamged conditions.');
writeln;
writeln(' Hit <Return> to continue.');
readln;
end;

procedure input_data;
var  i : integer;
begin
writeln('Enter the name of the file where you saved the data : ');
readln(fildat1);
reset(f1,fildat1);
while not eof(f1) do
begin
readln(f1,w,h,ro,mach);
readln(f1,theta1d,alpha1d,phi1d,beta1d);
readln(f1,oseg,s,xcg,bs);
readln(f1,ul,vl,wl);
readln(f1,pl,ql,rl);
readln(f1,iyy,ixxb,izzb,ixzb);
readln(f1,cdu,cdl,ctxu,ctxl,cda,cdde);
readln(f1,cmu,cml,cmtu,cmtl,cma);
readln(f1,cmta,cmadot,cmq);
```

```
readln(f1,cndsl,cndsr,cndcr,cndcl);
readln(f1,cll,clu,cla,cladot,clq);
readln(f1,cldsl,cldsr,cldcr,cldcl);
readln(f1,clde,cnde);
readln(f1,cyb,cyp,cyr,cyda,cydr);
readln(f1,clb,clp,clr,clda,cldr);
readln(f1,cnb,cntb,cnp,cnr,cnda,cndr);
readln(f1,dcdu,dcdl,dctxu,dctxl,dcda,dcdde);
readln(f1,dcmu,dcml,dcmtu,dcmtl,dcmta);
readln(f1,dcll,dclu);
readln(f1,dclde,dcnde);
readln(f1,dcyb,dcyp,dcyr,dcyda,dcydr);
readln(f1,dclb,dclp,dclr,dclda,dcldr);
readln(f1,dcnb,dcntb,dcnp,dcnr,dcnda,dcndr);
readln(f1,pos,side);
end;
close(f1);
end;


procedure set_init_cond;
var  i : integer;
begin
p1 := p1/57.3;
q1 := q1/57.3;
r1 := r1/57.3;
alpha1 := alpha1d/57.3;
theta1 := theta1d/57.3;
phi1 := phi1d/57.3;
beta1 := beta1d/57.3;
end;


procedure stab_derivatives;
const  g=32.1752;
var    q,qs,qsc,qsb:real;
       m           :real;
begin
m := w/g;
q := 0.5 * ro * sqr(u1);
qs := q * s;
qsc := qs * cseg;
qsb := qs * bs;
xu := -(qs*(cdu+2*cd1))/(m*u1);
xtu := (qs*(ctxu+2*ctx1))/(m*u1);
xa := -(qs*(cda-cl1))/m;
xde := -(qs*cdde)/m;
zu := -(qs*(clu+2*cl1))/(m*u1);
za := -(qs*(cla+cd1))/m;
zadot := -(qsc*cladot)/(2*m*u1);
zq := -qsc*clq/(2*m*u1);
```

```
zde := -(qs*clde)/(m);
mu := qsc * (cmu + 2*cml)/(iyy*ul);
mtu:=qsc*(cmtu+2*cmtl)/(iyy*ul);
ma :=qsc*cma/iyy;
mta:=qsc*cmta/iyy;
madot:=(qsc*cseg*cmadot)/(2*iyy*ul);
mq := (qsc*cseg*cmq)/(2*iyy*ul);
mde := (qsc*cmde)/(iyy);
ixxs := ixxb*sqr(cos(alpha1))+izzb*sqr(sin(alpha1))-ixzb*sin(2*alpha1);
izzs := ixxb*sqr(sin(alpha1))+izzb*sqr(cos(alpha1))+ixzb*sin(2*alpha1);
ixzs := 0.5*ixxb*sin(2*alpha1)-0.5*izzb*sin(2*alpha1)+ixzb*cos(2*alpha1
yb := qs*cyb/m;
yp := qsb*cyp/(2*m*ul);
yr := qsb*cyr/(2*m*ul);
yda := qs*cyda/m;
ydr := qs*cydr/m;
lb := qsb*clb/ixxs;
lp := (qsb*bs*clp)/(2*ixxs*ul);
lr := (qsb*bs*clr)/(2*ixxs*ul);
lda := (qsb*clda)/ixxs;
ldr := (qsb*cldr)/ixxs;
nb := (qsb*cnb)/izzs;
ntb := (qsb*cntb)/izzs;
np := (qsb*bs*cnp)/(2*izzs*ul);
nr := (qsb*bs*cnr)/(2*izzs*ul);
nda := (qsb*cnda)/izzs;
ndr := (qsb*cndr)/izzs;
end;

procedure d_stab_derivatives;
const    g = 32.1752;
var     q,qs,qsc,qsb:real;
        m  : real;
        temp1,temp2,temp3,temp4,temp5,temp6 : real;
begin
m:= w/g;
q := 0.5 * ro * sqr(ul);
qs := q * s;
qsc := qs * cseg;
qsb := qs * bs;
temp1 := 3.957;
temp2 := -12.051;
temp3 := -2.635;
temp4 := 2.105;
temp5 := 9.977;
temp6 := -4.984;
dcla := cla - (clde-dclde);
dcma := cma - (cmde-dcmde);
dcladot := temp1*dclde;
dcmadot := temp2*dclde;
dclq := (temp3*dcmde)+temp4;
```

```
dcmq := (temp5*dcmde)+temp6;
dxu := -(qs*(dcdu+2*dcdl))/(m*ul);
dxtu := (qs*(dctxu+2*dctxl))/(m*ul);
dxa  :=-(qs*(dcda-dcll))/m;
dxde :=-(qs*dcdde)/m;
dzu := -(qs*(dclu+2*dcll))/(m*ul);
dza := -(qs*(dcla+dcdl))/m;
dzadot := -qsc*dcladot/(2*m*ul);
dzq := -qsc*dclq/(2*m*ul);
dzde := -(qs*dclde)/(m);
dmu := qsc*(dcmu+2*dcml)/(iyy*ul);
dmtu := qsc*(dcmtu+2*dcmtl)/(iyy*ul);
dma := qsc*dcma/iyy;
dmta := qsc*dcmta/iyy;
dmadot := (qsc*cseg*dcmadot)/(2*iyy*ul);
dmq := (qsc*cseg*dcmq)/(2*iyy*ul);
dmde := (qsc*dcmde)/(iyy);
dyb := (qs*dcyb)/m;
dyp := (qsb*dcyp)/(2*m*ul);
dyr := (qsb*dcyr)/(2*m*ul);
dyda := (qs*dcyda)/m;
dydr := (qs*dcydr)/m;
dlb := (qsb*dclb)/ixxs;
dlp := (qsb*bs*dclp)/(2*ixxs*ul);
dlr := (qsb*bs*dclr)/(2*ixxs*ul);
dlda := (qsb*dclda)/ixxs;
dldr := (qsb*dcldr)/ixxs;
dnb := (qsb*dcnb)/izzs;
dntb := (qsb*dcntb)/izzs;
dnp := (qsb*bs*dcnp)/(2*izzs*ul);
dnr := (qsb*bs*dcnr)/(2*izzs*ul);
dnda := (qsb*dcnda)/izzs;
dndr := (qsb*dcndr)/izzs;
end;

procedure  a_matrix;
const   g = 32.1752;
begin
a1:= 1-(zadot/ul);
b1:= ul * (1-(zadot/ul));
c1 := ixxs - (sqr(ixzs)/izzs);
d1 := izzs - (sqr(ixzs)/ixxs);
aa[1,1]:=za/b1;
aa[1,2]:=(((zq/ul)+1)/a1) ;
aa[1,3]:=((zu+q1)/b1) ;
aa[1,4]:=((-g*cos(phi1)*sin(theta1))/b1) ;
aa[1,5] := (-p1/a1) ;
aa[1,6] := (-v1/b1) ;
aa[1,7] := 0.0;
aa[1,8] := (-g*sin(phi1)*cos(theta1)/b1) ;
aa[2,1] := ((ma+mta) + (madot*aa[1,1]));
```

```
aa[2,2] := (mq + (madot*aa[1,2]));
aa[2,3] := ((mu*mtu) + (madot*aa[1,3]));
aa[2,4] := (madot*aa[1,4]);
aa[2,5] := (madot*aa[1,5]);
aa[2,6] := ((izzs*r1/iyy)+(-ixxs*r1/iyy)+(-2*ixzs*p1/iyy)+(madot*aa[1,6
aa[2,7] := ((izzs*p1/iyy)+(-ixxs*p1/iyy)+(2*ixzs*r1/iyy)+(madot*aa[1,7]
aa[2,8] := madot*aa[1,8];
aa[3,1] := xa-(q1*u1) ;
aa[3,2] := -w1 ;
aa[3,3] := (xu+xtu);
aa[3,4] := (-g*cos(theta1));
aa[3,5] := (r1*u1) ;
aa[3,6] := 0.0;
aa[3,7] := v1 ;
aa[3,8] := 0.0;
aa[4,1] := 0.0;
aa[4,2] := cos(phi1);
aa[4,3] := 0.0;
aa[4,4] := 0.0;
aa[4,5] := 0.0;
aa[4,6] := 0.0;
aa[4,7] := -sin(phi1);
aa[4,8] := 0.0;
aa[5,1] := p1 ;
aa[5,2] := 0.0;
aa[5,3] := (-r1/u1) ;
aa[5,4] := (-g*sin(phi1)*sin(theta1)/u1);
aa[5,5] := (yb/u1);
aa[5,6] := ((yp+w1)/u1) ;
aa[5,7] := ((yr-u1)/u1) ;
aa[5,8] := (g*cos(phi1)*cos(theta1)/u1);
aa[6,1] := 0.0;
aa[6,2] := ((-ixzs*iyy*p1/(izzs*c1)) + (ixzs*ixxs*p1/(izzs*c1)) +
    (-sqr(ixzs)*r1/(c1*izzs)) + (ixzs*p1/c1) + (iyy*r1/c1) + (-izzs*r1/
aa[6,3] := 0.0;
aa[6,4] := 0.0;
aa[6,5] := ((ixzs*nb/c1) + (lb*ixxs/c1));
aa[6,6] := ((-ixzs*iyy*q1/(c1*izzs))+(ixzs*ixxs*q1/(c1*izzs))+(ixzs*np/
        (ixzs*q1/c1)+(lp*ixxs/c1)) ;
aa[6,7] := ((-sqr(ixzs)*q1/(c1*izzs))+(ixzs*nr/c1)+(iyy*q1/c1)+(-ixzs*q
(lr*ixxs/c1)) ;
aa[6,8] := 0.0;
aa[7,1] := 0.0;
aa[7,2] := ((sqr(ixzs)*p1/(ixxs*d1)) + (ixzs*iyy*r1/(ixxs*d1)) + (-ixzs
r1/(ixxs*d1)) + (-iyy*p1/d1) + (ixxs*p1/d1) + (-ixzs*r1/d1)) ;
aa[7,3] := 0.0;
aa[7,4] := 0.0;
aa[7,5] := ((ixzs*lb/d1)+(nb*izzs/d1));
aa[7,6] := ((sqr(ixzs)*q1/(ixxs*d1)) + (ixzs*lp/d1) + (-iyy*q1/d1) + (i
d1) + (np*izzs/d1)) ;
aa[7,7] := ((ixzs*iyy*q1/(ixxs*d1)) + (-ixzs*izzs*q1/(ixxs*d1)) + (ixzs
(-ixzs*q1/d1) + (nr*izzs/d1)) ;
```

```
aa[7,8] := 0.0;
aa[8,1] := 0.0;
aa[8,2] := sin(phi1)*sin(theta1)/cos(theta1);
aa[8,3] := 0.0;
aa[8,4] := 0.0;
aa[8,5] := 0.0;
aa[8,6] := 1.0;
aa[8,7] := cos(phi1)*sin(theta1)/cos(theta1);
aa[8,8] := 0.0;
end;

procedure b_matrix;
begin
b1 := u1 * (1-(zadot/u1));
c1 := ixxs - (sqr(ixzs)/izzs);
d1 := izzs - (sqr(ixzs)/ixxs);
bb[1,1] := zde/b1;
bb[2,1] := mde + (madot*bb[1,1]);
bb[3,1] := xde;
bb[4,1] := 0.0;
bb[5,1] := 0.0;
bb[6,1] := 0.0;
bb[7,1] := 0.0;
bb[8,1] := 0.0;
bb[1,2] := 0.0;
bb[2,2] := 0.0;
bb[3,2] := 0.0;
bb[4,2] := 0.0;
bb[5,2] := yda/u1;
bb[6,2] := (ixzs*nda/c1)+(lda*ixxs/c1);
bb[7,2] := (ixzs*lda/d1)+(nda*izzs/d1);
bb[8,2] := 0.0;
bb[1,3] := 0.0;
bb[2,3] := 0.0;
bb[3,3] := 0.0;
bb[4,3] := 0.0;
bb[5,3] := ydr/u1;
bb[6,3] := (ixzs*ndr/c1)+(ldr*ixxs/c1);
bb[7,3] := (ixzs*ldr/d1)+(ndr*izzs/d1);
bb[8,3] := 0.0;
end;

procedure d_a_matrix;
const   g = 32.1752;
begin
a1:= 1-(dzadot/u1);
b1:= u1 * (1-(dzadot/u1));
c1 := ixxs - (sqr(ixzs)/izzs);
d1 := izzs - (sqr(ixzs)/ixxs);
daa[1,1]:=dza/b1;
daa[1,2]:=(((dzq/u1)+1)/a1) ;
```

```
daa[1,3]:=((dzu+ql)/bl) ;
daa[1,4]:=((-g*cos(phi1)*sin(theta1))/bl);
daa[1,5] := (-pl/al) ;
daa[1,6] := (-vl/bl) ;
daa[1,7] := 0.0;
daa[1,8] := (-g*sin(phi1)*cos(theta1)/bl) ;
daa[2,1] := ((dma+dmta) + (dmadot*daa[1,1]));
daa[2,2] := (dmq + (dmadot*daa[1,2]));
daa[2,3] := (dmu+dmtu) + (dmadot*daa[1,3]);
daa[2,4] := (dmadot*daa[1,4]);
daa[2,5] := (dmadot*daa[1,5]);
daa[2,6] := ((izzs*rl/iyy)+(-ixxs*rl/iyy)+(-2*ixzs*pl/iyy)+(dmadot*daa[
daa[2,7] := ((izzs*pl/iyy)+(-ixxs*pl/iyy)+(2*ixzs*rl/iyy)+(dmadot*daa[1
daa[2,8] := dmadot*daa[1,8];
daa[3,1] := dxa-(ql*ul) ;
daa[3,2] := -wl ;
daa[3,3] := (dxu+dxtu);
daa[3,4] := (-g*cos(theta1));
daa[3,5] := (rl*ul) ;
daa[3,6] := 0.0;
daa[3,7] := ul ;
daa[3,8] := 0.0;
daa[4,1] := 0.0;
daa[4,2] := cos(phi1);
daa[4,3] := 0.0;
daa[4,4] := 0.0;
daa[4,5] := 0.0;
daa[4,6] := 0.0;
daa[4,7] := -sin(phi1);
daa[4,8] := 0.0;
daa[5,1] := pl ;
daa[5,2] := 0.0;
daa[5,3] := (-rl/ul) ;
daa[5,4] := (-g*sin(phi1)*sin(theta1)/ul);
daa[5,5] := (dyb/ul);
daa[5,6] := ((dyp+wl)/ul) ;
daa[5,7] := ((dyr-ul)/ul) ;
daa[5,8] := (g*cos(phi1)*cos(theta1)/ul) ;
daa[6,1] := 0.0;
daa[6,2] := ((-ixzs*iyy*pl/(izzs*cl)) + (ixzs*ixxs*pl/(izzs*cl))
    + (-sqr(ixzs)*rl/(cl*izzs)) + (ixzs*pl/cl) + (iyy*rl/cl) + (-izzs*rl
daa[6,3] := 0.0;
daa[6,4] := 0.0;
daa[6,5] := ((ixzs*dnb/cl) + (dlb*ixxs/cl));
daa[6,6] := ((-ixzs*iyy*ql/(cl*izzs))+(ixzs*ixxs*ql/(cl*izzs))+(ixzs*dn
(ixzs*ql/cl)+(dlp*ixxs/cl)) ;
daa[6,7] := ((-sqr(ixzs)*ql/(cl*izzs))+(ixzs*dnr/cl)+(iyy*ql/cl)+(-ixzs
(dlr*ixxs/cl)) ;
daa[6,8] := 0.0;
daa[7,1] := 0.0;
```

```
daa[7,2] := ((sqr(ixzs)*p1/(ixxs*d1)) + (ixzs*iyy*r1/(ixxs*d1)) + (-ixz
r1/(ixxs*d1)) + (-iyy*p1/d1) + (ixxs*p1/d1) + (-ixzs*r1/d1));
daa[7,3] := 0.0;
daa[7,4] := 0.0;
daa[7,5] := ((ixzs*d1b/d1)+(dnb*izzs/d1));
daa[7,6] := ((sqr(ixzs)*q1/(ixxs*d1)) + (ixzs*d1p/d1) + (-iyy*q1/d1) +
d1) + (dnp*izzs/d1)) ;
daa[7,7] := ((ixzs*iyy*q1/(ixxs*d1))+(-ixzs*izzs*q1/(ixxs*d1)) + (ixzs*
+(-ixzs*q1/d1) + (dnr*izzs/d1));
daa[7,8] := 0.0;
daa[8,1] := 0.0;
daa[8,2] := sin(phi1)*sin(theta1)/cos(theta1);
daa[8,3] := 0.0;
daa[8,4] := 0.0;
daa[8,5] := 0.0;
daa[8,6] := 1.0;
daa[8,7] := cos(phi1)*sin(theta1)/cos(theta1);
daa[8,8] := 0.0;
end;

procedure rec_b_matrix;
var i,j : integer;
const   g = 32.1752;
var     q,qs,qsc,qsb:real;
        m  : real;
        bcan,bspoil,belev,bs2 : real;
begin
m:= w/g;
q := 0.5 * ro * sqr(u1);
qs := q * s;
qsc := qs * cseg;
qsb := qs * bs;
b1 := u1 * (1-(dzadot/u1));
c1 := ixxs - (sqr(ixzs)/izzs);
d1 := izzs - (sqr(ixzs)/ixxs);
bcan := 1.9;
bspoil := 2.7;
belev := 1.6;
bs2 := 5.0;
if (side = 1) then
   begin
      clder := clde/2.0;
      cldel := dclde - clder;
      cnder := cnde/2.0;
      cndel := dcnde - cnder;
      recdzder := -(qs*clder)/m;
      recdzdel := -(qs*cldel)/m;
      recdnder := (qsc * cnder)/iyy;
      recdndel := (qsc * cndel)/iyy;
      recbb[1,1] := recdzdel/b1;
```

```
      recbb[2,1] := recdmdel + (dmadot*recbb[1,1]);
      recbb[1,2] := recdzder/b1;
      recbb[2,2] := recdmder + (dmadot*recbb[1,2]);
      croldsr := -cldsr*(bspoil/bs2);
      croldsl := -croldsr;
      croldcr := -cldcr*(bcan/bs2);
      croldcl := -croldcr;
      crolder := -clder*(beleu/bs2);
      croldel := cldel*(beleu/bs2);
      ldcr := qsb*croldcr/ixxs;
      ldcl := qsb*croldcl/ixxs;
      ldsr := qsb*croldsr/ixxs;
      ldsl := qsb*croldsl/ixxs;
      lder := qsb*crolder/ixxs;
      ldel := qsb*croldel/ixxs;
      recbb[6,1] := ldel*ixxs/cl;
      recbb[7,1] := ldel*ixzs/dl;
      recbb[6,2] := lder*ixxs/cl;
      recbb[7,2] := lder*ixzs/dl;
    end;
if (side = 2) then
    begin
      cldel := clde/2.0;
      clder := dclde - cldel;
      cmdel := cmde/2.0;
      cmder := dcmde - cmdel;
      recdzder := -(qs*clder)/m;
      recdzdel := -(qs*cldel)/m;
      recdmder := (qsc * cmder)/iyy;
      recdmdel := (qsc * cmdel)/iyy;
      recbb[1,1] := recdzder/b1;
      recbb[2,1] := recdmder + (dmadot*recbb[1,1]);
      recbb[1,2] := recdzdel/b1;
      recbb[2,2] := recdmdel + (dmadot*recbb[1,2]);
      croldsr := -cldsr*(bspoil/bs2);
      croldsl := -croldsr;
      croldcr := -cldcr*(bcan/bs2);
      croldcl := -croldcr;
      crolder := -clder*(beleu/bs2);
      croldel := cldel*(beleu/bs2);
      ldcr := qsb*croldcr/ixxs;
      ldcl := qsb*croldcl/ixxs;
      ldsr := qsb*croldsr/ixxs;
      ldsl := qsb*croldsl/ixxs;
      lder := qsb*crolder/ixxs;
      ldel := qsb*croldel/ixxs;
      recbb[6,1] := lder*ixxs/cl;
      recbb[7,1] := lder*ixzs/dl;
      recbb[6,2] := ldel*ixxs/cl;
      recbb[7,2] := ldel*ixzs/dl;
     end;
   for i:=3 to 5 do
        begin
```

```
            for j := 1 to 2 do
                begin
                recbb[i,j] := 0.0;
                end;
            end;
        for j := 1 to 2 do
            begin
            recbb[8,j] := 0.0;
            end;
    recdzdsl := -(qs*cldsl)/m;
    recdzdsr := -(qs*cldsr)/m;
    recdzdcl := -(qs*cldcl)/m;
    recdzdcr := -(qs*cldcr)/m;
    recdmdsl := (qsc*cmdsl)/iyy;
    recdmdsr := (qsc*cmdsr)/iyy;
    recdmdcl := (qsc*cmdcl)/iyy;
    recdmdcr := (qsc*cmdcr)/iyy;
    recbb[1,3] := recdzdsl/b1;
    recbb[1,4] := recdzdsr/b1;
    recbb[1,5] := recdzdcl/b1;
    recbb[1,6] := recdzdcr/b1;
    recbb[2,3] := recdmdsl + (dmadot*recbb[1,3]);
    recbb[2,4] := recdmdsr + (dmadot*recbb[1,4]);
    recbb[2,5] := recdmdcl + (dmadot*recbb[1,5]);
    recbb[2,6] := recdmdcr + (dmadot*recbb[1,6]);
    for i := 3 to 8 do
        begin
          for j := 3 to 6 do
          begin
            recbb[i,j] := 0.0;
          end;
        end;
     recbb[6,3] := ldsl*ixxs/c1;
     recbb[6,4] := ldsr*ixxs/c1;
     recbb[6,5] := ldcl*ixxs/c1;
     recbb[6,6] := ldcr*ixxs/c1;
     recbb[7,3] := ldsl*ixzs/d1;
     recbb[7,4] := ldsr*ixzs/d1;
     recbb[7,5] := ldcl*ixzs/d1;
     recbb[7,6] := ldcr*ixzs/d1;
end;

procedure d_b_matrix;
begin
b1 := u1 * (1-(dzadot/u1));
c1 := ixxs - (sqr(ixzs)/izzs);
d1 := izzs - (sqr(ixzs)/ixxs);
dbb[1,1] := dzde/b1;
dbb[2,1] := dmde + (dmadot*dbb[1,1]);
dbb[3,1] := dxde;
dbb[4,1] := 0.0;
```

```
dbb[5,1] := 0.0;
dbb[6,1] := recbb[6,1]+recbb[6,2];
dbb[7,1] := recbb[7,1]+recbb[7,2];
dbb[8,1] := 0.0;
dbb[1,2] := 0.0;
dbb[2,2] := 0.0;
dbb[3,2] := 0.0;
dbb[4,2] := 0.0;
dbb[5,2] := dyda/u1;
dbb[6,2] := (ixzs*dnda/c1)+(dlda*ixxs/c1);
dbb[7,2] := (ixzs*dlda/d1)+(dnda*izzs/d1);
dbb[8,2] := 0.0;
dbb[1,3] := 0.0;
dbb[2,3] := 0.0;
dbb[3,3] := 0.0;
dbb[4,3] := 0.0;
dbb[5,3] := dydr/u1;
dbb[6,3] := (ixzs*dndr/c1)+(dldr*ixxs/c1);
dbb[7,3] := (ixzs*dldr/d1)+(dndr*izzs/d1);
dbb[8,3] := 0.0;
end;

procedure input_display;
var  i,j : integer;
begin
writeln('Enter the name of the file where you want to save the coeffici
writeln('of the A and B matrices : ');
readln(fildat3);
rewrite(f3,fildat3);
writeln(f3,'           A matrix ');
for i:=1 to 8 do
begin
   writeln(f3,aa[i,1],aa[i,2],aa[i,3],aa[i,4],aa[i,5],aa[i,6],aa[i,7],a
end;
writeln(f3,'  ');
writeln(f3,'           B matrix ');
for j :=1 to 8 do
begin
   writeln(f3,bb[j,1],bb[j,2],bb[j,3]);
end;
writeln(f3,' ');
writeln(f3,'           A matrix of the damaged aircraft ');
for i:=1 to 8 do
begin
   writeln(f3,daa[i,1],daa[i,2],daa[i,3],daa[i,4],
               daa[i,5],daa[i,6],daa[i,7],daa[i,8]);
end;
writeln(f3,'  ');
writeln(f3,'           B matrix of the damaged aircraft ');
for j :=1 to 8 do
begin
```

```
   writeln(f3,dbbE j,1],dbbE j,2],dbbE j,3]);
end;
writeln(f3,'  ');
writeln(f3,'           8 matrix of the reconfigured aircraft ');
for j := 1 to 8 do
begin
 writeln(f3,recbbE j,1],recbbE j,2],recbbE j,3],recbbE j,4],recbbE j,5],recb
end;
close(f3);
end;

begin
   intro;
   input_data;
   set_init_cond;
   stab_derivatives;
   d_stab_derivatives;
   a_matrix;
   b_matrix;
   d_a_matrix;
   rec_b_matrix;
   d_b_matrix;
   input_display;
end.
```

```
program aux (input,output);

type data = array[1..2501] of real;
     data1 = array[1..8,1..8] of real;
     data2 = array[1..8,1..9] of real;
     data3 = array[1..8] of real;
     data4 = array[1..8,1..2] of real;

var  a,na,da : data1;
     b,nb,db : data2;
     l,nl,dl : data4;
     damcol : data3;
     thalpha,thq,thu,ththeta : data;
     thbeta,thp,thr,thphi : data;
     uturb,vturb : data;
     u1,u2,u3,u4,u5,u6,u7,u8,u9,time : data;
     i,j,k,nlim : integer;
     jdam,jdet : integer;
     seed1,seed2,seed3 : integer;
     sed1,sed2,sed3 : integer;
     valvar,convfac : real;
     x,dx : real;
     beta1d,theta1d,alpha1d,phi1d : real;
     uu1,v1,w1 : real;
     p1,q1,r1 : real;
     fildat1,fildat2,fildat3,fildat4,fildat5,fildat6,fildat7 : string[2
     f1,f2,f3,f4,f5,f6,f7 : text;
     ans : char;

procedure intro;
begin
writeln('*****************************************************************');
writeln;
writeln('        W E L C O M E           T O ');
writeln;
writeln('                  M O D A U X . P');
writeln;
writeln('*****************************************************************');
writeln;
writeln('    This program simulates the aircraft dynamics in ');
writeln('discrete-time with the occurrence of the damage at time ');
writeln(' = 1 sec. The dynamics is affected by atmospheric turbu-');
writeln('lence simulated with the desired statistics. ');
writeln;
writeln('Hit <Return> to continue.');
readln;
end;

function random1: real;
var temp : real;
 begin
seed1 := 171 * (seed1 mod 177) - 2 * (seed1 div 177);
if seed1 < 0 then
```

```
seed1 := seed1 + 30269;
seed2 := 172 * (seed2 mod 176) - 35 * (seed2 div 176);
if seed2 < 0 then
seed2 := seed2 + 30307;
seed3 := 170 * (seed3 mod 178) - 63 * (seed3 div 178);
if seed3 < 0 then
seed3 := seed3 + 30323;
temp := seed1/30269.0 + seed2/30307.0 + seed3/30323.0;
random1 := temp - trunc(temp);
end;

function random2: real;
var temp : real;
 begin
sed1 := 171 * (sed1 mod 177) - 2 * (sed1 div 177);
if sed1 < 0 then
sed1 := sed1 + 30269;
sed2 := 172 * (sed2 mod 176) - 35 * (sed2 div 176);
if sed2 < 0 then
sed2 := sed2 + 30307;
sed3 := 170 * (sed3 mod 178) - 63 * (sed3 div 178);
if sed3 < 0 then
sed3 := sed3 + 30323;
temp := sed1/30269.0 + sed2/30307.0 + sed3/30323.0;
random2 := temp - trunc(temp);
end;

function normal:real;
var u,d,z,sig2:real;
begin
u := random1;
d := random2;
sig2 := sqrt(valvar);
z := sqrt(-2*ln(d))*sig2;
normal := convfac*z*cos(6.28318*u);
end;

procedure input_data;
var i : integer;
begin
writeln('Enter the name of the file where you saved your A and B matric
writeln('and your initial conditions : ');
readln(fildat1);
reset(f1,fildat1);
for i := 1 to 8 do
begin
readln(f1,na[i,1],na[i,2],na[i,3],na[i,4],na[i,5],na[i,6],na[i,7],na[i,
end;
for i := 1 to 8 do
begin
readln(f1,da[i,1],da[i,2],da[i,3],da[i,4],da[i,5],da[i,6],da[i,7],da[i,
end;
```

```
for i := 1 to 8 do
begin
readln(f1,nbE[i,1],nbE[i,2],nbE[i,3],nbE[i,4],nbE[i,5],nbE[i,6],nbE[i,7],nbE[i,
          nbE[i,9]);
end;
for i := 1 to 8 do
begin
readln(f1,dbE[i,1],dbE[i,2],dbE[i,3],dbE[i,4],dbE[i,5],dbE[i,6],dbE[i,7],dbE[i,
          dbE[i,9]);
end;
readln(f1,theta1d,alpha1d,phi1d,beta1d);
readln(f1,uu1,v1,w1);
readln(f1,p1,q1,r1);
readln(f1,dx,nlim);
readln(f1,valvar,convfac);
close(f1);
for i := 1 to 8 do
  begin
  n1E[i,1] := naE[i,1];
  n1E[i,2] := naE[i,5];
  d1E[i,1] := daE[i,1];
  d1E[i,2] := daE[i,5];
  end;
seed1 := 10;
seed2 := 100;
seed3 := 1000;
sed1 := 1001;
sed2 := 101;
sed3 := 11;
writeln(' ');
writeln('Enter the time when damages occurs (as multiple of dx) : ');
readln(jdam);
writeln('Enter the time when damages have been detected and identified
writeln('(as multiple of dx) : ');
readln(jdet);
end;

procedure manoeuver;
var i,j : integer;
begin
j := 1;
x := 0.0;
repeat
  u1[j] := -5.0/57.3;
  u2[j] := -5.0/57.3;
  time[j] := x;
  x := x + dx;
  j := j + 1;
until j = 401;
```

```
repeat
  u1[j] := 0.0/57.3;
  u2[j] := 0.0/57.3;
  time[j] := x;
  x := x + dx;
  j := j + 1;
until j = 801;
repeat
  u2[j] := 5.0/57.3;
  u1[j] := 5.0/57.3;
  time[j] := x;
  x := x + dx;
  j := j + 1;
until j = 1201;
repeat
  u1[j] := 0.0;
  u2[j] := 0.0;
  time[j] := x;
  x := x + dx;
  j := j + 1;
until j = 1401;
repeat
  u1[j] := -3.0/57.3;
  u2[j] := -3.0/57.3;
  time[j] := x;
  x := x + dx;
  j := j + 1;
until j = 1701;
repeat
  u1[j] := 0.0;
  u2[j] := 0.0;
  time[j] := x;
  x := x + dx;
  j := j + 1;
until j = 1901;
repeat
  u1[j] := 3.0/57.3;
  u2[j] := 3.0/57.3;
  time[j] := x;
  x := x + dx;
  j := j + 1;
until j = 2201;
repeat
  u1[j] := 0.0;
  u2[j] := 0.0;
  time[j] := x;
  x := x + dx;
  j := j + 1;
until j > nlim;

j := 1;
```

```
repeat
  u7[j] := -4.0/57.3;
  u8[j] := -4.0/57.3;
  j := j + 1;
until j = 401;
repeat
  u7[j] := 0.0;
  u8[j] := 0.0;
  j := j + 1;
until j > nlim;
for i := 1 to nlim do
begin
 u3[i] := 0.0;
 u4[i] := 0.0;
 u5[i] := 0.0;
 u6[i] := 0.0;
 u9[i] := 0.0;
end;
end;

procedure simulation;
type data5 = array[0..50] of real;
var acf : data5;
var lag,tau : integer;
var i,j,k,kp1 : integer;
begin
thalpha[1] := 0.0;
thq[1] := 0.0;
thu[1] := 0.0;
ththeta[1] := 0.0;
thbeta[1] := 0.0;
thp[1] := 0.0;
thr[1] := 0.0;
thphi[1] := 0.0;
for i :=1 to 8 do
begin
  for j := 1 to 8 do
  begin
    a[i,j] := na[i,j];
  end;
end;
for i := 1 to 8 do
begin
  for j := 1 to 9 do
  begin
    b[i,j] := nb[i,j];
  end;
end;
for i := 1 to 8 do
begin
  l[i,1] := nl[i,1];
  l[i,2] := nl[i,2];
end;
```

```
k := 1;
repeat
 if (k >= jdam) then
   begin
     for i := 1 to 8 do
     begin
       for j := 1 to 8 do
       begin
         a[i,j] := da[i,j];
       end;
     end;
     for i := 1 to 8 do
     begin
       for j := 1 to 9 do
       begin
         b[i,j] := db[i,j];
       end;
     end;
     for i := 1 to 8 do
     begin
       l[i,1] := dl[i,1];
       l[i,2] := dl[i,2];
     end;
     u1[k] := -5.0/57.3;
   end;
  kp1 := k + 1;
 thalpha[kp1] := (a[1,1]*thalpha[k] + a[1,2]*thq[k] + a[1,3]*thu[k] +
    a[1,4]*ththeta[k] + a[1,5]*thbeta[k] + a[1,6]*thp[k] + a[1,7]*thr[k
    a[1,8]*thphi[k]) + (b[1,1]*u1[k] + b[1,2]*u2[k] + b[1,3]*u3[k] +
    b[1,4]*u4[k] + b[1,5]*u5[k] + b[1,6]*u6[k] + b[1,7]*u7[k] +
    b[1,8]*u8[k] + b[1,9]*u9[k]) + (l[1,1]*normal) + (l[1,2]*normal);
  thq[kp1] := (a[2,1]*thalpha[k] + a[2,2]*thq[k] + a[2,3]*thu[k] +
    a[2,4]*ththeta[k] + a[2,5]*thbeta[k] + a[2,6]*thp[k] + a[2,7]*thr[k
    a[2,8]*thphi[k]) + (b[2,1]*u1[k] + b[2,2]*u2[k] + b[2,3]*u3[k] +
    b[2,4]*u4[k] + b[2,5]*u5[k] + b[2,6]*u6[k] + b[2,7]*u7[k] +
    b[2,8]*u8[k] + b[2,9]*u9[k]) + (l[2,1]*normal) + (l[2,2]*normal);
  thu[kp1] := (a[3,1]*thalpha[k] + a[3,2]*thq[k] + a[3,3]*thu[k] +
    a[3,4]*ththeta[k] + a[3,5]*thbeta[k] + a[3,6]*thp[k] + a[3,7]*thr[k
    a[3,8]*thphi[k]) + (b[3,1]*u1[k] + b[3,2]*u2[k] + b[3,3]*u3[k] +
    b[3,4]*u4[k] + b[3,5]*u5[k] + b[3,6]*u6[k] + b[3,7]*u7[k] +
    b[3,8]*u8[k] + b[3,9]*u9[k]) + (l[3,1]*normal) + (l[3,2]*normal);
  ththeta[kp1] := (a[4,1]*thalpha[k] + a[4,2]*thq[k] + a[4,3]*thu[k] +
    a[4,4]*ththeta[k] + a[4,5]*thbeta[k] + a[4,6]*thp[k] + a[4,7]*thr[k
    a[4,8]*thphi[k]) + (b[4,1]*u1[k] + b[4,2]*u2[k] + b[4,3]*u3[k] +
    b[4,4]*u4[k] + b[4,5]*u5[k] + b[4,6]*u6[k] + b[4,7]*u7[k] +
    b[4,8]*u8[k] + b[4,9]*u9[k]) + (l[4,1]*normal) + (l[4,2]*normal);
  thbeta[kp1] := (a[5,1]*thalpha[k] + a[5,2]*thq[k] + a[5,3]*thu[k] +
    a[5,4]*ththeta[k] + a[5,5]*thbeta[k] + a[5,6]*thp[k] + a[5,7]*thr[k
    a[5,8]*thphi[k]) + (b[5,1]*u1[k] + b[5,2]*u2[k] + b[5,3]*u3[k] +
    b[5,4]*u4[k] + b[5,5]*u5[k] + b[5,6]*u6[k] + b[5,7]*u7[k] +
    b[5,8]*u8[k] + b[5,9]*u9[k]) + (l[5,1]*normal) + (l[5,2]*normal);
```

```
  thp[kp1] := (a[6,1]*thalpha[k] + a[6,2]*thq[k] + a[6,3]*thu[k] +
    a[6,4]*ththeta[k] + a[6,5]*thbeta[k] + a[6,6]*thp[k] + a[6,7]*thr[k
    a[6,8]*thphi[k]) + (b[6,1]*u1[k] + b[6,2]*u2[k] + b[6,3]*u3[k] +
    b[6,4]*u4[k] + b[6,5]*u5[k] + b[6,6]*u6[k] + b[6,7]*u7[k] +
    b[6,8]*u8[k] + b[6,9]*u9[k]) + (1[6,1]*normal) + (1[6,2]*normal);
  thr[kp1] := (a[7,1]*thalpha[k] + a[7,2]*thq[k] + a[7,3]*thu[k] +
    a[7,4]*ththeta[k] + a[7,5]*thbeta[k] + a[7,6]*thp[k] + a[7,7]*thr[k
    a[7,8]*thphi[k]) + (b[7,1]*u1[k] + b[7,2]*u2[k] + b[7,3]*u3[k] +
    b[7,4]*u4[k] + b[7,5]*u5[k] + b[7,6]*u6[k] + b[7,7]*u7[k] +
    b[7,8]*u8[k] + b[7,9]*u9[k]) + (1[7,1]*normal) + (1[7,2]*normal);
  thphi[kp1] := (a[8,1]*thalpha[k] + a[8,2]*thq[k] + a[8,3]*thu[k] +
    a[8,4]*ththeta[k] + a[8,5]*thbeta[k] + a[8,6]*thp[k] + a[8,7]*thr[k
    a[8,8]*thphi[k]) + (b[8,1]*u1[k] + b[8,2]*u2[k] + b[8,3]*u3[k] +
    b[8,4]*u4[k] + b[8,5]*u5[k] + b[8,6]*u6[k] + b[8,7]*u7[k] +
    b[8,8]*u8[k] + b[8,9]*u9[k]) + (1[8,1]*normal) + (1[8,2]*normal);
  k := k + 1;
until k = nlim;
for k:= 1 to nlim do
begin
  thalpha[k] := (thalpha[k]*57.3) + alpha1d;
  thq[k] := (thq[k]*57.3) + q1;
  thu[k] := thu[k] + uu1;
  ththeta[k] := (ththeta[k]*57.3) + theta1d;
  thbeta[k] := (thbeta[k]*57.3) + beta1d;
  thp[k] := (thp[k]*57.3) + p1;
  thr[k] := (thr[k]*57.3) + r1;
  thphi[k] := (thphi[k]*57.3) + phi1d;
  uturb[k] := normal*uu1;
  vturb[k] := normal*uu1;
  u1[k] := u1[k]*57.3;
  u2[k] := u2[k]*57.3;
  u7[k] := u7[k]*57.3;
  u8[k] := u8[k]*57.3;
end;
repeat
 writeln('Would you like to save your longitudinal data on a file (Y/N)
 readln(ans);
until ans in ['y','n'];
if ans = 'y' then
begin
  writeln('Enter the name of the file where you want to save the longit
  writeln('data of the simulation : ');
  readln(fildat2);
  rewrite(f2,fildat2);
  i := 1;
  repeat
    writeln(f2,time[i],thalpha[i],thq[i],ththeta[i],u1[i],u2[i]);
    i := i + 5;
  until i > 2497;
  close(f2);
end;
```

```
repeat
 writeln('Would you like to save your lat-directional data on a file (Y
 readln(ans);
until ans in ['y','n'];
if ans = 'y' then
begin
  writeln('Enter the name of the file where you want to save the direct
  writeln('data of the simulation : ');
  readln(fildat3);
  rewrite(f3,fildat3);
  i := 1;
  repeat
    writeln(f3,time[i],thbeta[i],thp[i],thr[i],thphi[i],u7[i]);
    i := i + 5;
  until i > 2497;
  close(f3);
end;
repeat
 writeln('Would you like to save the velocity components "w" and "v" ')
 writeln('of the turbulence on a data file (Y/N) : ');
 readln(ans);
until ans in ['y','n'];
if ans = 'y' then
begin
  writeln('Enter the name of the file where you want to save the veloci
  writeln('components of the turbulence : ');
  readln(fildat6);
  rewrite(f6,fildat6);
  i := 1;
  repeat
    writeln(f6,time[i],wturb[i],vturb[i]);
    i := i + 5;
  until i > 2497;
  close(f6);
end;
repeat
  writeln('Once you have the turbulence data, would you like to analyze
  writeln('their autocorrelation function ? (Y/N) ');
  readln(ans);
until ans in ['y','n'];
if ans = 'y' then
begin
  writeln('Enter the "lag" of your autocorrelation function (integer) :
  readln(lag);
  for j := 0 to lag do
  begin
    acf[j] := 0.0;
    tau := nlim - j;
    for i := 1 to (nlim-j) do
    begin
      acf[j] := acf[j] + (wturb[i]*wturb[i+j])/tau;
    end;
  end;
```

```
  writeln('Enter the name of the file where you want to save the');
  writeln('turbulence autocorrelation function :');
  readln(fildat7);
  rewrite(f7,fildat7);
  for i := 0 to lag do
  begin
    writeln(f7,i,acf[i]);
  end;
  close(f7)
end;
repeat
 writeln('Would you like to save your time histories on a file  ');
 writeln('for Kalman filter analysis ? (Y/N) : ');
 readln(ans);
until ans in ['y','n'];
if ans = 'y' then
begin
  writeln('Enter the name of the file for the state variables data : ')
  readln(fildat4);
  rewrite(f4,fildat4);
  writeln('Enter the name of the file for the controls data : ');
  readln(fildat5);
  rewrite(f5,fildat5);
  i := (jdet+1);
  repeat
    writeln(f4,time[i],' ',thalpha[i],' ',thq[i],' ',thu[i],' ',ththeta
    writeln(f4,thbeta[i],' ',thp[i],' ',thr[i],' ',thphi[i]);
    writeln(f5,time[i],' ',u1[i],' ',u2[i],' ',u3[i],' ',u4[i]);
    writeln(f5,u5[i],' ',u6[i],' ',u7[i],' ',u8[i],' ',u9[i]);
    i := i + 1;
  until i > (jdet+501);
  close(f4);
  close(f5);
end;
end;

begin
intro;
input_data;
manoeuver;
simulation;
end.
```

```
program modmmkf (input,output);

var  i,j,k,n,m : integer;
     fc,nfc,nov,iterations : integer;
     seed11,seed12,seed21,seed22,seed31,seed32 : integer;
     dt,tf,time : real;
     sig,norm : real;

type data = array[1..12,1..12] of real;
     data1 = array[1..8,1..9,1..12] of real;
     data2 = array[1..8,1..8,1..12] of real;
     data3 = array[1..505,1..12] of real;
     data4 = array[0..50,1..6] of real;
     vector = array[1..12] of real;

var  r,h,xhat,xpred,zr : data;
     b : data1;
     ke,a,sinv : data2;
     beta,x0,x,prob,u,z,v : vector;
     res : data3;
     acf : data4;

var  fildat1,fildat2,fildat3,fildat4,fildat5 : string[20];
     f1,f2,f3,f4,f5 : text;

var  ans : char;

procedure intro;
begin
writeln('*****************************************************************');
writeln;
writeln('            W E L C O M E         T O ');
writeln;
writeln('                  M O D M M K F . P ');
writeln;
writeln('*****************************************************************');
writeln;
writeln('    This program performs the damaged model estimation using')
writeln('a Multiple Model Kalman Filtering approach.');
writeln;
writeln('Hit <RETURN> to continue.');
readln;
end;

function random1 : real;
var temp : real;
begin
  seed11 := 171 * (seed11 mod 177) - 2 * (seed11 div 177);
  if seed11 < 0 then
  seed11 := seed11 + 30269;
  seed21 := 172 * (seed21 mod 176) - 35 * (seed21 div 176);
  if seed21 < 0 then
  seed21 := seed21 + 30307;
```

```
  seed31 := 170 * (seed31 mod 178) - 63 * (seed31 div 178);
  if seed31 < 0 then
  seed31 := seed31 + 30323;
  temp := seed11/30269.0 + seed21/30307.0 + seed31/30323.0;
  random1 := temp - trunc(temp);
end;

function random2 : real;
var temp : real;
begin
  seed12 := 171 * (seed12 mod 177) - 2 * (seed12 div 177);
  if seed12 < 0 then
  seed12 := seed12 + 30269;
  seed22 := 172 * (seed22 mod 176) - 35 * (seed22 div 176);
  if seed22 < 0 then
  seed22 := seed22 + 30307;
  seed32 := 170 * (seed32 mod 178) - 63 * (seed32 div 178);
  if seed32 < 0 then
  seed32 := seed32 + 30323;
  temp := seed12/30269.0 + seed22/30307.0 + seed32/30323.0;
  random2 := temp - trunc(temp);
end;

procedure normal (sig : real; var norm : real);
var u,d,z,sig2 : real;
begin
  u := random1;
  d := random2;
  sig2 := sqrt(sig);
  z := sqrt(-2.0*ln(d))*sig2;
  norm := z*cos(6.28318*u);
end;

procedure get_data_in;
var i,j,k : integer;
begin
  writeln('Enter the name of the file where the matrices data and the s
  writeln('data for the Multiple Model Kalman filter analysis :');
  readln(fildat1);
  reset(f1,fildat1);
  readln(f1,n);
  readln(f1,m);
  readln(f1,nfc);
  readln(f1,fc);
  readln(f1,nov);
  readln(f1,dt);
  readln(f1,tf);
  readln(f1,seed11,seed21,seed31);
  readln(f1,seed12,seed22,seed32);
    for k := 1 to nfc do
    begin
      for i := 1 to n do
      begin
```

```
      readln(f1,a[i,1,k],a[i,2,k],a[i,3,k],a[i,4,k],a[i,5,k],
                a[i,6,k],a[i,7,k],a[i,8,k]);
      end;
    end;
    for K := 1 to nfc do
    begin
      for i := 1 to n do
      begin
      readln(f1,b[i,1,k],b[i,2,k],b[i,3,k],b[i,4,k],b[i,5,k],
                b[i,6,k],b[i,7,k],b[i,8,k],b[i,9,k]);
      end;
    end;
    for i := 1 to nov do
    begin
    readln(f1,r[i,1],r[i,2],r[i,3],r[i,4],r[i,5],r[i,6]);
    end;
    for i := 1 to nov do
    begin
    readln(f1,h[i,1],h[i,2],h[i,3],h[i,4],h[i,5],h[i,6],h[i,7],h[i,8]);
    end;
    for k := 1 to nfc do
    begin
      for i := 1 to n do
      begin
      readln(f1,ke[i,1,k],ke[i,2,k],ke[i,3,k],ke[i,4,k],ke[i,5,k],ke[i,
      end;
    end;
    for k := 1 to nfc do
    begin
      for i := 1 to nov do
      begin
      readln(f1,sinv[i,1,k],sinv[i,2,k],sinv[i,3,k],sinv[i,4,k],sinv[i,
             sinv[i,6,k]);
      end;
    end;
    for k := 1 to nfc do
    begin
    readln(f1,beta[k]);
    end;
    for k := 1 to nfc do
    begin
    readln(f1,prob[k]);
    end;
    readln(f1,x0[1],x0[2],x0[3],x0[4],x0[5],x0[6],x0[7],x0[8]);
  close(f1);
end;

procedure simulation;
var     i,ii,j,k : integer;
        sum,sum1,sum2 : real;
        lag,tau : integer;
type    vector = array[1..12] of real;
var     temp,mr,e : vector;
```

```
begin
  iterations := round(tf/dt);
  writeln('Enter the name of the file with the state variables data : '
  readln(fildat2);
  reset(f2,fildat2);
  readln(f2,time,x[1],x[2],x[3],x[4]);
  readln(f2,x[5],x[6],x[7],x[8]);
  writeln('Enter the name of the file with the control data : ');
  readln(fildat3);
  reset(f3,fildat3);
  writeln('Enter the name of the file where you want to save the probab
  writeln('for each of the selected configurations of matrix B : ');
  readln(fildat4);
  rewrite(f4,fildat4);
    for i := 1 to n do
    begin
      if ( i <> 3 ) then
        begin
        x[i] := x[i]/57.3;
        end;
     x[i] := x[i] - x0[i];
     end;
     for i := 1 to nov do
     begin
     normal(r[i,i],v[i]);
     end;
     for i := 1 to nov do
     begin
     sum1 := 0.0;
       for j := 1 to n do
       begin
       sum1 := sum1 + h[i,j]*x[j];
       end;
     z[i] := sum1 + v[i];
     end;
     for k := 1 to nfc do
     begin
       for i := 1 to n do
       begin
       xpred[i,k] := x[i];
       end;
       for i := 1 to nov do
       begin
       sum := 0.0;
         for j := 1 to n do
         begin
         sum := sum + h[i,j]*xpred[j,k];
         end;
       zr[i,k] := z[i] - sum;
           if (k=fc) then res[1,i] := zr[i,k];
       end;
       for i := 1 to n do
       begin
```

```
      sum := 0.0;
        for j := 1 to nov do
        begin
        sum := sum + ke[i,j,k]*zr[j,k];
        end;
      xhat[i,k] := xpred[i,k] + sum;
      end;
    end;
writeln(f4,time,prob[fc-2],prob[fc-1],prob[fc],prob[fc+1],prob[fc+2])
for ii := 2 to (iterations+1) do
begin
  for i := 1 to nov do
  begin
  normal(r[i,i],v[i]);
  end;
  readln(f2,time,x[1],x[2],x[3],x[4]);
  readln(f2,x[5],x[6],x[7],x[8]);
  readln(f3,time,u[1],u[2],u[3],u[4]);
  readln(f3,u[5],u[6],u[7],u[8],u[9]);
  for i := 1 to m do
  begin
  u[i] := u[i]/57.3;
  end;
  for i := 1 to n do
  begin
    if (i <> 3 ) then
    begin
    x[i] := x[i]/57.3;
    end;
  x[i] := x[i] - x0[i];
  end;
  for i := 1 to nov do
  begin
  sum1 := 0.0;
    for j := 1 to n do
    begin
    sum1 := sum1 + h[i,j]*x[j];
    end;
  z[i] := sum1 + v[i];
  end;
  for k := 1 to nfc do
  begin
    for i := 1 to n do
    begin
    sum1 := 0.0;
      for j := 1 to n do
      begin
      sum1 := sum1 + a[i,j,k]*xhat[j,k];
      end;
    sum2 := 0.0;
      for j := 1 to m do
      begin
      sum2 := sum2 + b[i,j,k]*u[j];
      end;
    xpred[i,k] := sum1 + sum2;
    end;
```

```
  for i := 1 to nov do
  begin
  sum := 0.0;
    for j := 1 to n do
    begin
    sum := sum + h[i,j]*xpred[j,k];
    end;
  zr[i,k] := z[i] - sum;
        if (k=fc) then res[ii,i] := zr[i,k];
  end;
  for i := 1 to n do
  begin
  sum := 0.0;
    for j := 1 to nov do
    begin
    sum := sum + ke[i,j,k]*zr[j,k];
    end;
  xhat[i,k] := xpred[i,k] + sum;
  end;
end;
for k := 1 to nfc do
begin
  for i := 1 to nov do
  begin
  sum := 0.0;
    for j := 1 to nov do
    begin
    sum := sum + sinv[i,j,k]*zr[j,k];
    end;
  temp[i] := sum;
  end;
  sum := 0.0;
    for i := 1 to nov do
    begin
    sum := sum + zr[i,k]*temp[i];
    end;
  nr[k] := sum;
  sum := -sum/2.0;
  if (sum < -40.0 ) then
    begin
    sum := -40.0;
    end;
  e[k] := exp(sum);
end;
sum := 0.0;
  for k := 1 to nfc do
  begin
  sum := sum + prob[k]*beta[k]*e[k];
  end;
  for k := 1 to nfc do
  begin
  prob[k] := prob[k]*beta[k]*e[k]/sum;
    if (prob[k] < 0.0001 ) then
    begin
    prob[k] := 0.0001;
    end;
```

```
      end;
    sum := 0.0;
      for k := 1 to nfc do
      begin
      sum := sum + prob[k];
      end;
      for k := 1 to nfc do
      begin
      prob[k] := prob[k]/sum;
      end;
    writeln(f4,time,prob[fc-2],prob[fc-1],prob[fc],prob[fc+1],prob[fc+2
  end;
  close(f4);
  close(f3);
  close(f2);
  repeat
    writeln('Once you have the residual data, would you like to analyze
    writeln('their autocorrelation function ? (Y/N) ');
    readln(ans);
  until ans in ['y','n'];
    if ans = 'y' then
    begin
    writeln('Enter the "lag" of the autocorrelation functions (integer)
      readln(lag);
      for i := 1 to nov do
      begin
        for j := 0 to lag do
        begin
          acf[j,i] := 0.0;
          tau := iterations-j;
          for ii := 1 to (iterations-j) do
          begin
            acf[j,i] := acf[j,i] + (res[ii,i]*res[ii+j,i])/tau;
          end;
        end;
      end;
      writeln('Enter the name of the file where you want to save the');
      writeln('aircraft data autocorrelation functions : ');
      readln(fildat5);
      rewrite(f5,fildat5);
      for i := 0 to lag do
      begin
      writeln(f5,i,acf[i,1],acf[i,2],acf[i,3],acf[i,4],acf[i,5],acf[i,6
      end;
      close(f5);
    end;
end;
begin
intro;
get_data_in;
simulation;
end.
```

```
//[pp,ke,pzri,beta] = filter(a,b,h,d,r,q)
for i=1:8, g(i,1)=a(i,1);
for i=1:8, g(i,2)=a(i,5);
gqg = g*q*g';
[eval,k,p] = destimator(a,h,gqg,r);
ke = inv(a)*k;
pp = a*p*a' + gqg;
pzr = h*pp*h' + r;
beta = 1./sqrt(((6.2832)**6)*det(pzr));
pzri = inv(pzr);
retf
```

```
//[phib,hb,gamqgam] = corstep2(a,b,h,d,r,q)
for i=1:8, g(i,1)=a(i,1);
for i=1:8, g(i,2)=a(i,5);
gqg = g*q*g';
[eval,k,p] = destimator(a,h,gqg,r);
ke = inv(a)*k;
pp = a*p*a' + gqg;
pzr = h*pp*h' + r;
beta = 1./sqrt(((6.2832)**6)*det(pzr));
pzri = inv(pzr);
i8 = eye(8);
i2 = eye(2);
i6 = eye(6);
a11 = a*(i8-ke*h);
a12 = g;
a13 = -a*ke;
a21 = 0.0*eye(2,8);
a22 = [0.5,0.0;0.0,0.5];
a23 = 0.0*eye(2,6);
a31 = 0.0*eye(6,8);
a32 = 0.0*eye(6,2);
a33 = 0.0*eye(6,6);
phib = [ a11,a12,a13;a21,a22,a23;a31,a32,a33 ];
g11 = 0.0*eye(8,2);
g12 = 0.0*eye(8,6);
g21 = i2;
g22 = 0.0*eye(2,6);
g31 = 0.0*eye(6,2);
g32 = i6;
gammab = [g11,g12;g21,g22;g31,g32];
qb12 = 0.0*eye(2,6);
qb21 = 0.0*eye(6,2);
qb = [ q,qb12;qb21,r];
gamqgam = gammab*qb*gammab';
hb12 = 0.0*eye(6,8);
hb = [h,hb12];
retf
```

```
//[pp1,pzr1,pzri1,beta1] = step2p2(phib,hb,gamqgam,r)
[eval,k,p] = destimator(phib,hb,gamqgam,r);
pp1 = phib*p*phib' + gamqgam;
pzr1 = hb*pp1*hb' + r;
pzri1 = inv(pzr1);
beta1 = 1.0/sqrt(((6.2832)**6)*det(pzr1));
retf
```

Reconfiguration Algorithm

The reconfiguration algorithm introduced in Chapt IV has been implemented in the program RECONF4.P. The listing of this program is not enclosed because of its size; however, a list of the subroutines with their purposes is reported.

Procedure INTRO: the task of the program is introduced.

Procedure INPUT_DATA: the aerodynamic data for the damaged and undamaged conditions, the inertial and geometric characteristics and the flight conditions are given as input to the program.

Procedure SET_INIT_CONDIT: the angular conversions are performed and the deflection limits for the control surfaces are set.

Procedures STAB_DERIVATIVES, A_MATRIX, B_MATRIX: the state variable mathematical model of the aircraft for undamaged conditions is built using the dimensional stability and control derivatives. The modeling procedures are taken from Ref.[13].

Procedures D_STAB_DERIVATIVES, D_A_MATRIX, D_B_MATRIX, REC_B_MATRIX: the state variable mathematical model of the aircraft for damaged conditions is built using the dimensional stability and control derivatives. The 'RECB' matrix, calculated in the procedure REC_B_MATRIX, is relative to all the available control surfaces.

Procedure INPUT_DISPLAY: the resulting A, B matrices previosly calculated for nominal and damaged conditions are

saved on a file.

Procedure DE_FUN: a certain number of elevator inputs are built to be used for simulation purposes.

Procedure DA_FUN: a certain number of ailerons inputs are built to be used for simulation purposes.

Procedure DR_FUN: a certain number of rudder inputs are built to be used for simulation purposes.

Procedure RK4: a Rounge Koutta 4-th order integration is performed.

Procedure RK4_FUN: the state variable models for all the different flight conditions are implemented.

Procedure RECONF_INPUT: all the parameters for the dynamic simulation of the reconfiguration are given by the user, that is:

- Reconfiguration time step;
- Selected elevator manoeuver;
- Selected aileron manoeuver;
- Selected rudder manoeuver;
- Time instant when the damage occurs;
- Time instant when the reconfiguration algorithm is applied;
- Minimum value for [u(n)-u(n-1)].

Procedure WEIGHTS: the weight with which each available healthy control surface contributes to the reconfiguration is calculated. The used criterium is shown in Chapt. IV (Eqs. [4.15] and [4.16]).

Procedure RECONF_CONDIT: all the data and the conditions for

the reconfiguration process are saved on a file for future analysis and considerations.

Procedure WEIGHTED_RECONFIGURATION: the amounts of control needed for reconfiguring the selected states at that particular time instant (calculated in the procedure RECONFIGURATION) are spread among the available healthy control surfaces. A check is made for each control surface such that the resulting input desired for the reconfiguration doesn't exceed the deflection limits.

Procedure RECONFIGURATION: the overall dynamic simulation of the reconfiguration is performed, by using the procedures RK4 and RK4_FUN. First, from the initial time to the time when damage occurs, the dynamic responses of the damaged and undameged aircraft are coincident; then, we have a drastically changed dynamics from the time when the damage occurs to the time when the reconfiguration algorithm is applied; finally, the reconfiguration algorithm takes over and the amounts of control power needed in order to reconfigure the key states 'p','q' and 'r' for that particular time instant are calculated. Such amount is then sent to the WEIGHT_RECONFIGURATION procedure. All the results of the dynamic simulation can be saved, if so desired, on output files for future analysis.

Feedback Structure Redesign

A certain number of $MATRIX_X$ user defined functions have been implemented for redesigning the Feedback Structure with the approach introduced in Chapt.V. In the following pages the $MATRIX_X$ functions used in Step 1, Step 2 and Step 3 of Chapt. V are reported:

CONTPOLES.FNC

DAMDYNAMICS.FNC

REDESIGN.FNC

```
//[kl,kld,ac]=contpoles(along,blong,alat,blat);
polesl=[-1.1510+2.8598*jay,-0.0003+0.089*jay];
kl=poleplace(al,bl,polesl);
blk=bl*kl;
acl=al-blk;
polesld=[-0.0701+1.6858*jay,-0.5085,-0.008];
kld=poleplace(ald,bld,polesld);
bldk=bld*kld;
acld=ald-bldk;
z=0.0*eye(4);
ac=[acl,z;z,acld];
retf
```

```
//[adamcl,eivecdam,eivaldam]=damdyn(adam,bdam,k);
bdamk = bdam*k;
adamcl=adam-bdamk;
[eivecdam,eivaldam]=eig(adamcl);
retf
```

```
//[redk,redadamcl] = redesign(eivec,eival,ared,bred);
evec=eivec;
lam=eival;
a=ared;
b=bred;
n=8;
for i=1:n,...
eigvec1(i)=evec(i,1);
for i=1:n,...
eigvec2(i)=evec(i,2);
for i=1:n,...
eigvec3(i)=evec(i,3);
for i=1:n,...
eigvec4(i)=evec(i,4);
for i=1:n,...
eigvec5(i)=evec(i,5);
for i=1:n,...
eigvec6(i)=evec(i,6);
for i=1:n,...
eigvec7(i)=evec(i,7);
for i=1:n,...
eigvec8(i)=evec(i,8);
eigvec1t=eigvec1';
eigvec2t=eigvec2';
eigvec3t=eigvec3';
eigvec4t=eigvec4';
eigvec5t=eigvec5';
eigvec6t=eigvec6';
eigvec7t=eigvec7';
eigvec8t=eigvec8';
q=eye(8);
m=eye(8);
id=eye(8);
l1=inv(lam(1)*id - a)*b;
l2=inv(lam(2)*id - a)*b;
l3=inv(lam(3)*id - a)*b;
l4=inv(lam(4)*id - a)*b;
l5=inv(lam(5)*id - a)*b;
l6=inv(lam(6)*id - a)*b;
l7=inv(lam(7)*id - a)*b;
l8=inv(lam(8)*id - a)*b;
l1t=l1';
l2t=l2';
l3t=l3';
l4t=l4';
l5t=l5';
l6t=l6';
l7t=l7';
l8t=l8';
```

```
w1t=eigvec1t*q*l1*inv(l1t*q*l1);
w2t=eigvec2t*q*l2*inv(l2t*q*l2);
w3t=eigvec3t*q*l3*inv(l3t*q*l3);
w4t=eigvec4t*q*l4*inv(l4t*q*l4);
w5t=eigvec5t*q*l5*inv(l5t*q*l5);
w6t=eigvec6t*q*l6*inv(l6t*q*l6);
w7t=eigvec7t*q*l7*inv(l7t*q*l7);
w8t=eigvec8t*q*l8*inv(l8t*q*l8);
w1=w1t';
w2=w2t';
w3=w3t';
w4=w4t';
w5=w5t';
w6=w6t';
w7=w7t';
w8=w8t';
v1=inv(lam(1)*id - a)*b*w1;
v2=inv(lam(2)*id - a)*b*w2;
v3=inv(lam(3)*id - a)*b*w3;
v4=inv(lam(4)*id - a)*b*w4;
v5=inv(lam(5)*id - a)*b*w5;
v6=inv(lam(6)*id - a)*b*w6;
v7=inv(lam(7)*id - a)*b*w7;
v8=inv(lam(8)*id - a)*b*w8;
w=[w1,w2,w3,w4,w5,w6,w7,w8];
v=[v1,v2,v3,v4,v5,v6,v7,v8];
redk=w*inv(m*v);
redadamc1=a+b*redk*m;
retf
```

VITA

Marcello R. Napolitano

Candidate for the Degree of

Doctor of Philosophy

Thesis: A NEW APPROACH TO THE FLIGHT CONTROL SYSTEM RECONFIGURATION FOLLOWING A BATTLE DAMAGE AND/OR A GENERIC FAILURE ON A CONTROL SURFACE

Major Field: Mechanical Engineering

Biographical:

Personal Data: Born in Pomigliano D'Arco (Naples), Italy, the son of Giuseppe and Celestina Napolitano.

Education: Graduated from "G. Garibaldi" Classic High School, Pomigliano D'Arco (Naples), Italy, in July, 1979; received the Master of Science degree in Aeronautical Engineering from University of Naples, Faculty of Engineering, Aeronautical Department, Institute of Aircraft Design, Naples, Italy, in November, 1985; completed the requirements for the Doctor of Philosophy degree at Oklahoma State University in December, 1989.

Professional Experience: Teaching Assistant, School of Mechanical and Aerospace Engineering, Oklahoma State University, Fall 1986 to Fall 1989.

Professional Societies: American Institute of Aeronautics and Astronautics; Italian Engineers Board.