A TOTAL ENERGY SENSOR FOR GLIDEPATH AND SPEED CONTROL OF A TACTICAL AIRLIFTER IN WIND SHEAR

A Thesis

by

THOMAS EDWARD ANDERSON

Submitted to the Graduate College of
Texas A&M University
in partial fulfillment of the requirement for the degree of

MASTER OF SCIENCE

August 1987

Major Subject: Aerospace Engineering

A TOTAL ENERGY SENSOR FOR GLIDEPATH AND SPEED CONTROL OF A TACTICAL AIRLIFTER IN WIND SHEAR

A Thesis

by

THOMAS EDWARD ANDERSON

Approved as to style and content by:

Donald T. Ward
(Chairman of Committee)

John L. Junkins
(Member)

Jo W. Howze
(Member)

Alton L. Highsmith
(Member)

Walter E. Haisler
(Head of Department)

August 1987

ABSTRACT

A Total Energy Sensor for Glidepath and Speed Control
of a Tactical Airlifter in Wind Shear. (August 1987)
Thomas Edward Anderson, B.S., The Ohio State University
Chairman of Advisory Committee: Dr. Donald T. Ward

The effectiveness of a total energy sensor in a closed-loop environment during wind shear is presented. Models of the Lockheed High-Technology Test Bed, Nicks' total energy sensor and wind shear are developed and combined to provide the computer simulation used for this evaluation. The aircraft was flown using short take-off and landing techniques down a 6° glidepath through wind shear layers of various size and strength. Better control of glidepath and airspeed was available when total energy rate was used through a feedback controller in the elevator and spoiler channels. With the TE rate controller engaged, the control deflections occured earlier and were smoother than with the basic controller. However, the effectiveness of total energy rate feedback was limited by angle of attack bounds imposed by the aircraft data base. Nevertheless, total energy rate feedback was effective in reducing glidepath, airspeed and angle of attack deviations.

To Mom and Dad

ACKNOWLEDGMENTS

This research was supported by the Lockheed-Georgia Company under TEES Contract 032525-21490. The author would like to thank Chet Payne, Pat Moore, Mike McCarty and Sandy Hoffman, all of Lockheed-Georgia, for their guidance and help in acquiring necessary data, Tom Gally for his programming expertise, Rosa Oseguera for her insight into the total energy sensor, and Bob Eller for his suggestions for the control law derivations. The author is also grateful to Dr. Don Ward for his support, guidance and patience when the research was slow. Last but by far not least, the author wishes to thank Jenny for her never ending support, love and understanding on those many long nights.

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

LIST OF TABLES

LIST OF FIGURES

LIST OF FIGURES (Continued)

NOMENCLATURE

AGL	above ground level
b	wing span
c	wing chord
C_D	drag coefficient
C_L	lift coefficient
C_M	pitching moment coefficient
C_N	yawing moment coefficient
C_P	pressure coefficient
C_R	rolling moment coefficient
C_X	longitudinal force coefficient
C_Y	side force coefficient
C_Z	vertical force coefficient
$CG\%$	percent center of gravity
DOF	degrees of freedom
DLC	direct lift control
$EO1$	number of engines not operating
fps	feet per second
F_x	longitudinal force - body axes
F_y	lateral force - body axes
F_z	vertical force - body axes
g	gravitational acceleration

NOMENCLATURE (Continued)

h	altitude
i	$\sqrt{-1}$
$\underline{i}$	unit vector in X-direction
I	moment of inertia
$\underline{k}$	unit vector in Z-direction
kts	knots
L	rolling moment about X_b
m	airframe mass
M	pitching moment about Y_b
N	yawing moment about Z_b
p	angular velocity about X_b - roll rate
$P_{ambient}$	ambient pressure
P_{local}	local pressure
P_{sensor}	pressure at TES orifice
psi	pounds per square inch
psid	pounds per square inch differential
q	angular velocity about Y_b - pitch rate
$\bar{q}$	dynamic pressure
r	angular velocity about Z_b - yaw rate
$\underline{R}$	distance from axes origin
ROC	rate of climb

NOMENCLATURE (Continued)

s	Laplace variable
S	wing area
STOL	short take-off and landing
t	time
t_2	time to double amplitude
T_c	thrust coefficient
T_{eng}	thrust per engine
T_T	total propulsive force
TE	total energy
TES	total energy sensor
u_{20}	wind speed at $h = 20$ feet
u_w	wind speed
u_w'	vertical variation of wind speed with altitude
U	linear velocity along X_b
V	linear velocity along Y_b
$\underline{V}$	linear velocity in body axes
V_T	true airspeed
W	linear velocity along Z_b
α	angle of attack
β	sideslip angle
δ_f	flap deflection

NOMENCLATURE (Continued)

ϕ	roll angle relative to flat earth
θ	pitch angle relative to flat earth
ψ	yaw angle relative to flat earth
ζ	damping ratio
ω_n	undamped natural frequency

Subscripts

$()_{a/c}$	aircraft
$()_{ail}$	aileron
$()_b$	body axes
$()_{dlc}$	direct lift control
$()_{dyn}$	dynamic term
$()_{elev}$	elevator
$()_{eo}$	engine out
$()_g$	gear
$()_{ge}$	ground effect
$()_o$	initial conditions
$()_{rud}$	rudder
$()_s$	stability axes
$()_{splr}$	spoiler
$()_w$	wind axes

NOMENCLATURE (Continued)

$()_{wbt}$ wing, body, tail combination

$()_{\beta}$ sideslip

CHAPTER I

INTRODUCTION

Major advancements have been made in air travel since the Wright brothers' first flight at Kitty Hawk in 1903. As aircraft have become more complex, the workload on the pilot has increased and the need for precise control has become essential. In terms of workload, the approach to landing is still by far the most demanding. The airline pilot is running final checks, making necessary radio calls, and configuring the aircraft for landing while the military pilot, in addition to the above, may be trying to make an assault landing in battle conditions. To complicate matters, atmospheric disturbances can adversely affect aircraft and pilot performance. The most violent of these disturbances, wind shear, can occur with no visual warning and have disastrous effects, as in the case of Eastern Flight 66 in New York (1975) and Delta Flight 191 in Dallas (1985).

Wind shear is defined as a change in wind magnitude and direction with altitude and is commonly associated with thunderstorms. This variation in wind velocity creates a change in the relative wind speed leading to a deviation in the flight path angle. Since short field take-off and landing (STOL) approaches are flown on the back side of the power curve where descent is controlled with throttle, returning to the initial flight path necessitates a change in throttle setting.

The conventional glidepath-hold autopilot usually performs well provided there are no severe wind changes and the ground-based glidepath transmitter is operational. However, if the landing is made in poor weather conditions (high wind shear probabilities) or if an assault approach is flown STOL or in battle conditions where no transmitter is available, some airborne measurement device must be available to

Journal model is *AIAA Journal of Aircraft*.

aid the pilot.

Energy methods have been used to analytically evaluate aircraft performance[1] The problem has been how to quickly and accurately measure the aircraft total energy in practice. In 1976, Robert Joppa at the University of Washington experimented with total energy rate to detect wind shear. Two methods were tested: an electronic system and a pneumatic system. The electronic system used signals from the inertial navigation system and air data computer that are available on most larger aircraft. This method provided reliable data with minimum time lag. Unfortunately, an air data computer, because of its size and cost, is not readily available for all aircraft. For less complex aircraft, a pneumatic system consisting of a venturi total energy probe and vertical speed indicator was tested. This subsystem also provided reliable data, but a significant time lag of 3 to 6 seconds was encountered[2].

Following Joppa, Joseph Gera at NASA Langley developed a piloted, nonlinear, real-time simulation of a modified Boeing 737 in wind shear. Using a small-disturbance model, a first-order divergence instability, 'tuck' mode, was discovered when head wind decreased with altitude. By combining attitude control with energy rate feedback to the throttle, the 'tuck' mode was stabilized. Approaches were flown with various combinations of pitch and throttle control. Gera concluded that approaches were smoother and pilot workload smaller when both pitch and throttle were controlled. However, the pilots disliked the throttle augmentation by energy rate feedback because manual throttle commands were overridden by the controller, leaving the impression that the throttle was out of control[3].

Also in 1976 at NASA Langley, Oran Nicks developed a total energy sensor (TES). Originally designed to optimize glider performance, the TES was a simple,

low cost sensor that proved to be effective at a wide range of angles of attack and sideslip[4].

Then in 1983, Ostroff, *et. al.*, tested Nicks' TES on a transport aircraft in wind shear. Two probes were installed, one on each side of the fuselage near the nose allowing both combined and separate system evaluation. As expected, the combined system showed superior results over the separate system configuration due to sideslip sensitivity. This system correlated well with a theoretical approach which modeled the TES and necessary electronics with a second order transfer function[5]. Ostroff agreed with other researchers by concluding that the TES system has potential for wind shear detection and integration into a flight control system.

However, the TES has more applications than just wind shear. With its response time and precision, total energy rate feedback can provide early warning of deviation from nominal flight path. Rosa Oseguera of Texas A&M University showed that pilots flew more accurate approaches with total energy rate feedback than without[6], and for this reason, the TES application can be extended to precision approaches, STOL approaches and assault landings.

Investigation Objectives

The objective of this research was to develop a simulation and evaluate the usefulness of the TES in a STOL approach through wind shear. The work was divided into three phases. In the first phase, a mathematical model of the subject aircraft, the Lockheed High-Technology Test Bed (HTTB), was developed and validated. The simulation was completed on a DEC VAX-8600 computer using a Lockheed-supplied shell program containing trim, time history and integration algorithms. The aircraft model, control algorithms and supporting software were

generated at Texas A&M during this phase. Next, a suitable wind shear model was developed using derivations by Gera[7] and Sherman[8]. In the final phase, an analytical model of the TES was derived based on experimental data from a series of bench tests. By integrating these three tasks and developing the control laws for total energy rate feedback, the TES was evaluated in a STOL approach through wind shear.

The Subject Aircraft

The Lockheed HTTB (Fig. 1) is a modified 382E model stretch C-130 Commercial Hercules Airfreighter (N130X)[9]. Originally a cargo transport, the aircraft is an all-metal, high-wing monoplane with a distinctive, glossy black finish. The HTTB is powered by four 4368 SHP Allison 501-D22A engines and is 100 inches longer than a standard C-130.

Acquired by Lockheed in February 1984, the aircraft was modified for flight test with the installation of flight test instrumentation and advanced avionics. STOL capabilities were improved with the modification of the double slotted trailing edge flaps and the addition of "horsals" and "dorsals" (extended horizontal and vertical tail fairings). The preliminary changes included two composite wing tip airspeed booms, four Rosemount airspeed probes, three angle of attack vanes, and a high sink rate landing gear.

The initial flight tests were run at the Lockheed-Georgia plant in Marietta, Georgia (Dobbins AFB) from June to October 1984. The flight test data collected by the Lockheed Airborne Data System provided the data base for the HTTB simulation portion of this research.

The purpose of the HTTB is to provide an aircraft dedicated to the technical

community that is capable of enhancing development and evaluation of future assault transport and tactical airlifter technologies. The aircraft is available to industry and universities for the advancement of assault STOL capabilities, aircraft survivability, flight controls, electronics and avionics. The HTTB will give advanced aircraft technologies the chance to mature past the conceptual stage and be developed, integrated and evaluated in a realistic flight environment.

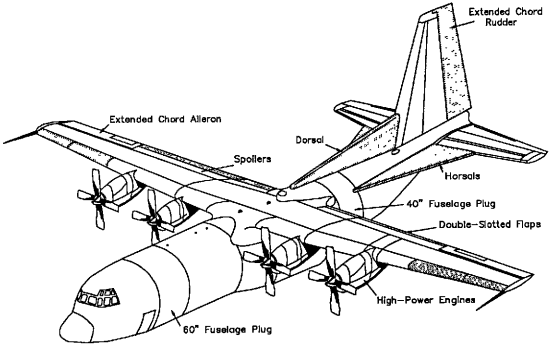

Fig. 1 The Lockheed High-Technology Test Bed

CHAPTER II

HTTB SIMULATION DEVELOPMENT

Equations of Motion

The equations of motion used in this research represent a second generation model. The first generation model was for an unmodified C-130 in a powered, 3° glidepath approach. However, because this research was concerned with a STOL approach on a 6° glidepath, the weight, moments of inertia, constant stability derivatives and absence of spoiler and control surface time lag effects did not truly represent the HTTB. Consequently, a second set of data was acquired[10]. As discussed earlier, these data were obtained through HTTB flight tests.

The equations of airframe motion are based on an axis system which is fixed in the body with its origin at the aircraft center of mass. This system was chosen because it is the natural frame of reference for pilot observations and aircraft motion measurements. However, standard aerodynamic forces and moments are defined in stability-axes where X_s is along the initial aircraft velocity vector (Fig. 2). During the simulation, the velocity vector $\underline{V}$ and angle α vary and hence X_s will no longer remain along $\underline{V}$. Therefore, Lockheed resolved the data into the wind-axes system which is similar to the stability-axes, but rotates to keep X aligned with $\underline{V}$. A simple transformation between wind-axes and body-axes can then be used:

$$\begin{pmatrix} X_b \\ Y_b \\ Z_b \end{pmatrix} = \begin{pmatrix} \cos\alpha & 0 & -\sin\alpha \\ 0 & 1 & 0 \\ \sin\alpha & 0 & \cos\alpha \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} \tag{1}$$

if β is assumed zero. The body-fixed axes are shown in Fig. 3 with the symbols, notation, and sign convention used throughout this research.

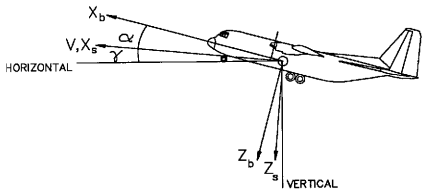

Fig. 2 Aircraft Stability Axes

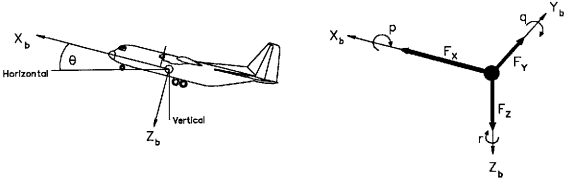

Fig. 3 Aircraft Body Axes

The body-fixed equations of motion (derived in Appendix A) are:

$$\Sigma F_{X_b} = m(\dot{U} + Wq - Vr + g\sin\theta) \tag{2}$$

$$\Sigma F_{Y_b} = m(\dot{V} + Ur - Wp - g\sin\phi\cos\theta) \tag{3}$$

$$\Sigma F_{Z_b} = m(\dot{W} + Vp - Uq - g\cos\phi\cos\theta) \tag{4}$$

$$\Sigma L_b = \dot{p}I_x + qr(I_z - I_y) - I_{xz}(\dot{r} + pq) \tag{5}$$

$$\Sigma M_b = \dot{q}I_y + pr(I_x - I_z) + I_{xz}(p^2 - r^2) \tag{6}$$

$$\Sigma N_b = \dot{r}I_z + pq(I_y - I_x) + I_{xz}(qr - \dot{p}). \tag{7}$$

Three assumptions are made: 1) the aircraft is a rigid body, 2) the aircraft is symmetric in the X-Z plane, and 3) the aircraft mass is constant.

The aerodynamic forces and moments used in Eqs. 2-7 are first calculated in the wind-axes and transformed via Eq. 1 into the body-axes. The forces and moments are calculated in non-dimensional form from 41 data tables constructed from flight test data. When given α, β, T_c, δ_f and the control surface deflections, the tables provide the non-dimensional forces and moments by linear interpolation. These are summed to give the non-dimensional forces and moments[10]:

$$C_{X_s} = -C_{D_{wbt}} - \Delta C_{D_{ge}} - \Delta C_{D_{elev}} - \Delta C_{D_{dlc}} - \Delta C_{D_g} - \Delta C_{D_{rud}} - \Delta C_{D_{ail}} +$$
$$-\Delta C_{D_\beta} + T_T\cos\alpha \tag{8}$$

$$C_{Y_s} = C_{Y_\beta} + \Delta C_{Y_{rud}} + \Delta C_{Y_{ail}} + \Delta C_{Y_{splr}} + \Delta C_{Y_{eo}} \tag{9}$$

$$C_{Z_s} = -C_{L_{wbt}} - \Delta C_{L_{ge}} - \Delta C_{L_{elev}} - \Delta C_{L_{dlc}} - \Delta C_{L_{rud}} - \Delta C_{L_{ail}} +$$
$$-\Delta C_{L_\beta} + \Delta C_{L_{eo}} \tag{10}$$

$$C_{M_s} = C_{M_{wbt}} + \Delta C_{M_{ge}} + \Delta C_{M_{elev}} + \Delta C_{M_{dlc}} + \Delta C_{M_{rud}} + \Delta C_{M_{ail}} + \Delta C_{M_\beta} +$$

$$+ C_{M_{dyn}} + \Delta C_{M_{eo}} \tag{11}$$

$$C_{N_s} = C_{N_\beta} + \Delta C_{N_{rud}} + \Delta C_{N_{ail}} + \Delta C_{N_{splr}} + C_{Y_s} c(CG\% - 25)/(100b) +$$

$$+ C_{N_{dyn}} + \Delta C_{N_{eo}} \tag{12}$$

$$C_{R_s} = C_{R_\beta} + \Delta C_{R_{rud}} + \Delta C_{R_{ail}} + \Delta C_{R_{splr}} + C_{R_{dyn}} + \Delta C_{R_{eo}}. \tag{13}$$

Using Eq. 1, the non-dimensional forces and moments become:

$$C_{X_b} = C_{X_s} \cos\alpha - C_{Z_s} \sin\alpha \tag{14}$$

$$C_{Y_b} = C_{Y_s} \tag{15}$$

$$C_{Z_b} = C_{Z_s} \cos\alpha + C_{X_s} \sin\alpha \tag{16}$$

$$C_{M_b} = C_{M_s} - C_{Z_b} c(CG\% - 25)/(100b) \tag{17}$$

$$C_{N_b} = C_{N_s} \cos\alpha + C_{R_s} \sin\alpha \tag{18}$$

$$C_{R_b} = C_{R_s} \cos\alpha - C_{N_s} \sin\alpha. \tag{19}$$

Dimensionalizing:

$$F_{X_b} = C_{X_b} S\bar{q} \tag{20}$$

$$F_{Y_b} = C_{Y_b} S\bar{q} \tag{21}$$

$$F_{Z_b} = C_{Z_b} S\bar{q} \tag{22}$$

$$L_b = C_{R_b} S\bar{q} b \tag{23}$$

$$M_b = C_{M_b} S\bar{q} c \tag{24}$$

$$N_b = C_{N_b} S\bar{q} b. \tag{25}$$

The aircraft orientation is given by the standard 3-2-1 Euler angles[11]:

$$\dot{\theta} = q\cos\phi - r\sin\phi \tag{26}$$

$$\dot{\psi} = \frac{r\cos\phi + q\sin\phi}{\cos\theta} \tag{27}$$

$$\dot{\phi} = p + (q\sin\phi + r\cos\phi)\tan\theta. \tag{28}$$

Taking Eqs. 2-7 and 26-28, solving for the time derivatives and substituting the known forces and moments of Eqs. 20-25 gives:

$$\dot{U} = \frac{\Sigma F_{X_b}}{m} - Wq + Vr - g\sin\theta \tag{29}$$

$$\dot{V} = \frac{\Sigma F_{Y_b}}{m} - Ur + Wp + g\sin\phi\cos\theta \tag{30}$$

$$\dot{W} = \frac{\Sigma F_{Z_b}}{m} - Vp + Uq + g\cos\phi\cos\theta \tag{31}$$

$$\dot{\theta} = q\cos\phi - r\sin\phi \tag{32}$$

$$\dot{\psi} = \frac{r\cos\phi + q\sin\phi}{\cos\theta} \tag{33}$$

$$\dot{\phi} = p + (q\sin\phi + r\cos\phi)\tan\theta \tag{34}$$

$$\dot{p} = q(pC_2 + rC_1) + L_bC_3 + N_bC_4 \tag{35}$$

$$\dot{q} = prC_5 - (p^2 - r^2)C_6 + M_bC_7 \tag{36}$$

$$\dot{r} = q(pC_8 - rC_2) + L_bC_4 + N_bC_{10} \tag{37}$$

where $C_{1,10}$ are ratios of moments of inertia.

These nine equations define the states, U, V, W, ϕ, θ, ψ, p, q and r, of the large-disturbance, nonlinear model. (In this discussion, all the variables are large disturbance values not to be confused with the small perturbation variables that are

used in related texts.) However, three more variables must be defined to completely describe the location of the aircraft. These variables are combinations of the nine states above:

$$\dot{h} = U \sin\theta - V \cos\theta \sin\phi - W \cos\theta \cos\phi \tag{38}$$

$$h = h_o + \int_0^t \dot{h} dt \tag{39}$$

$$\dot{X}_{a/c} = U \cos\theta \cos\psi + V(\sin\phi \sin\theta \cos\psi - \cos\phi \sin\psi) +$$

$$+ W(\cos\phi \sin\theta \cos\psi + \sin\phi \sin\psi). \tag{40}$$

Note that an implicit assumption is made in the $X_{a/c}$ equation. The earth-axes that are used in determining the aircraft position are taken to be fixed in space.

Included in the HTTB model are control surface rate and position limits and time delays[10]. Shaping functions are used to express the rate and authority limits of the actuators as functions of dynamic pressure and true airspeed, respectively:

$$f(\bar{q}) = (20 - \bar{q}) * 0.4/180 + 1.0 \qquad 0.6 \leq f(\bar{q}) \leq 1.0$$

for $V_T < 170$,

$$f(V_T) = (50 - V_T) * 0.1/120 + 1.0 \qquad 0.9 \leq f(V_T) \leq 1.0$$

and for $V_T \geq 170$,

$$f(V_T) = (170 - V_T) * 0.3/80 + 0.9 \qquad 0.6 \leq f(V_T) \leq 0.9$$

Table 1 shows the actuator transfer functions and rate and authority limits for the elevator, rudder, ailerons, spoilers, throttle and flaps. The FORTRAN code of the HTTB model is given in SUBROUTINE LD of the simulation code in Appendix B.

TABLE 1 Actuator Dynamics and Limits

Control	Transfer Function	Authority Limits	Rate Limits
Elevator	$\frac{15.3}{s+15.3}$	$15f(V_T), -40f(V_T)$	$\pm 32f(\bar{q})$
Throttle	$\frac{2.04}{(0.15s+1)(s^2+1.73s+2.04)}$	1,0	± 0.6354
Spoiler	$\frac{25}{s+25}$	0,−40	$\pm 30f(\bar{q})$
Aileron	$\frac{17}{s+17}$	$\pm 40f(V_T)$	$\pm 43f(\bar{q})$
Rudder	$\frac{15}{s+15}$	30.8,−40	$\pm 25f(\bar{q})$
Flap	$\frac{4}{s+4}$	54,0	± 4
Gear	$\frac{30}{s+30}$	1,0	± 0.333

Control Laws

Because this research simulates an automatic approach, a set of closed-loop control laws are necessary. To completely control the aircraft in the approach phase, both speed and glidepath must be maintained. However, in the STOL configuration, the HTTB has an unstable phugoid mode which requires stabilization before control laws can be implemented.

Stabilizing State Feedback

When the HTTB has its trailing edge flaps extended to the full 30°, as in the STOL configuration, the aircraft becomes unstable with a pole at $s = 0.25$ ($t_2 = 2.77$ seconds). By using state feedback, the poles can be shifted into the left half of the s-plane thus stabilizing the system. However, the stable system must have reasonably small feedback gains which do not cause control surface transients.

To stabilize the HTTB, state feedback is used to modify the elevator command. The first set of gains used resulted in poles well left of the imaginary axis, but caused an initial 6° transient in the elevator. This controller behavior is unacceptable. Because the aircraft is initially trimmed, this transient causes a deviation from trim followed by a slow return to the trim condition. Therefore, the poles are allowed nearer the imaginary axis keeping the elevator, and hence the aircraft, in trim.

Varying the feedback gains to eliminate the elevator transient yields the state feedback configuration shown in Fig. 4. This configuration was chosen because it provided a stable system with no elevator transient and the characteristics:

$$s = -1.2837 \pm 0.6091i \quad \zeta = 0.9035 \quad \omega_n = 1.4209$$

$$s = -0.0550 \pm 0.2230i \quad \zeta = 0.2395 \quad \omega_n = 0.2297.$$

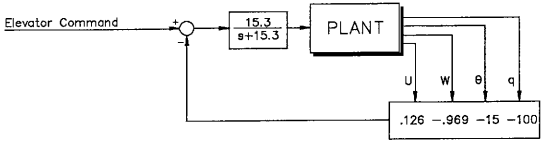

Fig. 4 Stabilizing State Feedback Configuration

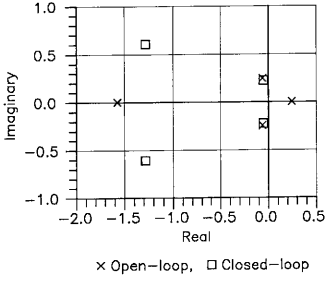

Fig. 5 Open and Closed Loop Aircraft Poles

This final configuration uses a non-unique set of gains because poles of the stable system were arbitrarily chosen. Fig. 5 shows the open-loop and closed-loop poles of the HTTB.

It must be noted that these stabilizing gains were designed for the no-wind condition. It quickly became obvious that these values caused unnecessary elevator deflections when there was an initial head wind or tail wind. By altering the state feedback gains, this transient was eliminated. On an aircraft, though, stabilizing controllers controllers are not designed for different wind conditions. Therefore, the gain values from the no wind condition are used.

Pitch Stability and Control Augmentation System (PSCAS)

Some means of pitch control is necessary to control the altitude and/or speed of the aircraft. The PSCAS discussed here is a modified form of the PSCAS used on the Lockheed man-in-the-loop HTTB simulation. Changes necessary to convert from such a manned system to a closed loop system included increasing the error gains. The commanded pitch was input directly to the controller instead of indirectly from the control column position.

The PSCAS gains have been set using System Identification in Matrix$_X$[12], a linear analysis package. Using a maximum likelihood algorithm, a desired trajectory for pitch can be tracked by varying the controller gains. The Matrix$_X$ routine MAXLIKE estimates the gains, simulates the system, compares the output with the desired trajectory and corrects the estimated gains. Given the form of the model of the system, MAXLIKE modifies the system gains to minimize the sum of the squares of the error between the actual and desired output. The PSCAS was put into Matrix$_X$ using the System Build capability. A command file, a program that executes Matrix$_X$ commands, was written to enter System Build, modify the

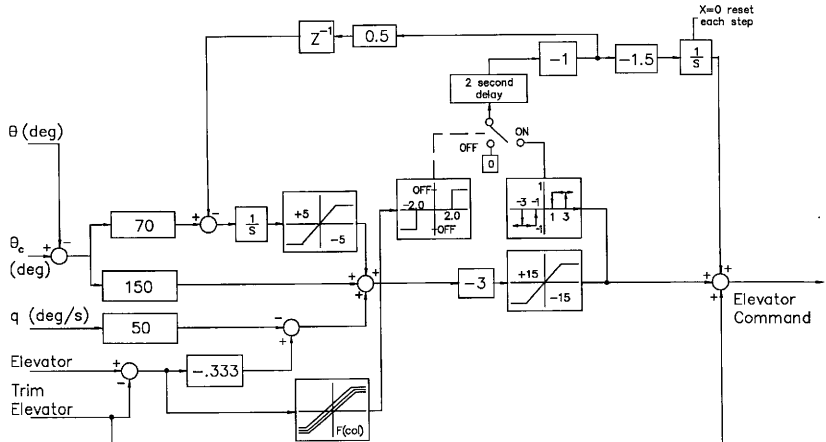

Fig. 6 Pitch Stability and Control Augmentation System

gain blocks and simulate the system. This command file is called and executed from within MAXLIKE. The final values of the gains and configuration of the controller are shown in Fig. 6.

The desired trajectory discussed earlier is simply a prescribed path that the controlled state should follow. In the development of each of these basic controllers, a specified, or desired, trajectory is proposed by specifying the rise and settling times. This response is chosen to reflect a physically reasonable and desirable transient response with zero steady state error. Obviously, this approach yields controllers that are non-optimal; however, future work could improve the PSCAS performance by optimizing the gains to eliminate unnecessary elevator movement. For PSCAS tuning, the desired trajectory specifies that a step in pitch be tracked in 1.5 seconds (Fig. 7).

Speed Control

In level flight, airspeed is controlled with the throttle. However, the area of interest for this research is not level flight; it is the approach to landing phase on a 6° glidepath in a high drag configuration with minimal power. This region of the flight envelope is known as the back side of the power curve where elevator primarily controls airspeed and throttle primarily controls glidepath.

The overall speed and glidepath controllers are shown in Fig. 8. To maintain airspeed, the speed controller output is the PSCAS input. A simple proportional controller provides a satisfactory transient response, however, a non-zero steady state error results. Adding an integrator to the controller eliminates this error. As before, MAXLIKE is used to tune the controller given the trajectory in Fig. 9 yielding the configuration shown in Fig. 10.

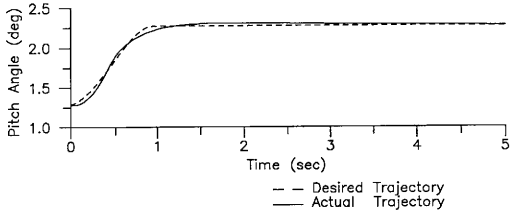

Fig. 7 PSCAS Tuning Trajectory

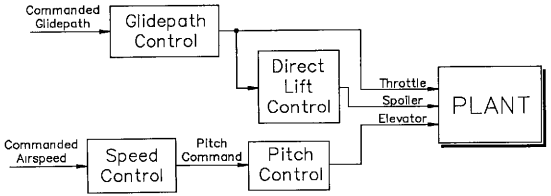

Fig. 8 General HTTB Glidepath and Speed Control

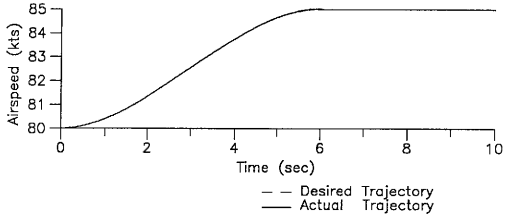

Fig. 9 Speed Tuning Trajectory

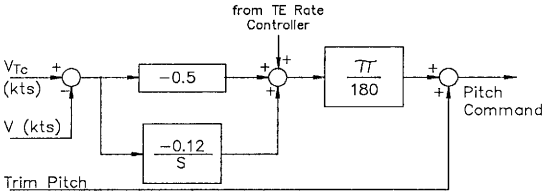

Fig. 10 Speed Controller

Glidepath Control

As discussed earlier, throttle is used to control glidepath. Concurrently, the throttle command drives direct lift control (DLC) by symmetrically activating the spoilers. The Lockheed DLC has been implemented with a modification to the gain. The original gain did not provide enough spoiler authority. Therefore, the gain, and hence spoiler authority, is increased by increasing the gain. However, spoiler authority is still limited. Because of the nature of the third-order shaping filter that is used to smooth the input to the DLC, the spoilers become ineffective and return to the initial uprig when the throttle becomes saturated. To remedy this situation, the altitude error signal from the glidepath controller is added to the throttle command to drive the DLC (Fig. 11).

The spoiler uprig mentioned here is the nominal spoiler deflection that accompanies flap deflection. On the HTTB, the flaps and spoilers are coupled so that as the flaps are extended, the spoilers are also extended. This linear coupling continues until the flaps are fully extended and the spoilers are deployed 20°. This coupling allows better lift control and hence, better altitude control.

The glidepath controller (Fig. 12) actually controls altitude as opposed to glidepath angle. If glidepath angle were to be controlled, deviation from the actual glidepath would lead to the tracking of some other 6° line. That is, glidepath angle control will track a family of glidepaths instead of one specific line in space. Therefore, if the distance to the glideslope transmitter is known from the DME equipment for example, the commanded altitude for that distance is also known.

The proportional error gain tracked the glidepath well, but was somewhat oscillatory. To increase the damping, the derivative of the altitude error and the

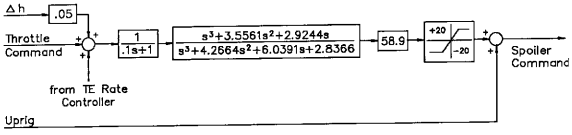

Fig. 11 Direct Lift Controller

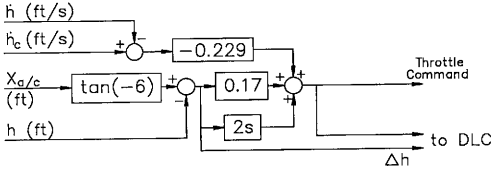

Fig. 12 Glidepath Controller

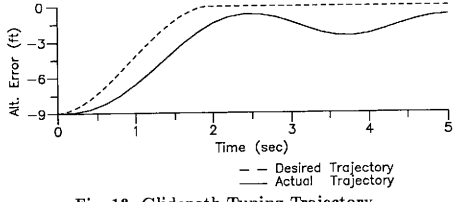

Fig. 13 Glidepath Tuning Trajectory

rate of descent error are included. Once again MAXLIKE is used to tune the controller. In this case, the HTTB is initially set 9 feet below the glidepath. The desired trajectory is such that the aircraft recaptures the glidepath after 2 seconds (Fig. 13).

This glidepath controller is designed for use with the speed controller. It enables the HTTB to maintain speed and glidepath to touchdown. Because the HTTB is executing a STOL approach, there is no flare before touchdown. However, as the HTTB approaches the ground, some flare is apparent due to ground effects that are included in this simulation (Fig. 14).

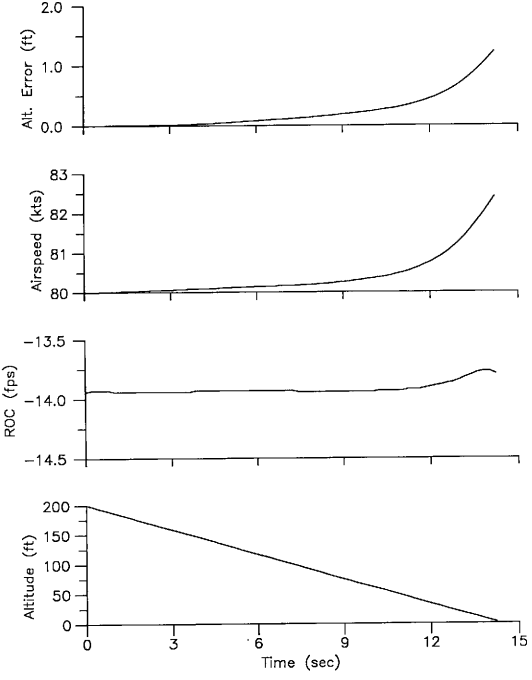

Fig. 14 HTTB Touchdown with Ground Effect

CHAPTER III

WIND SHEAR MODELING

Work in wind shear is somewhat complicated due to a lack of universal conventions in terminology. Vertical wind shear, more commonly referred to simply as wind shear, is the change in wind magnitude and direction with altitude. Head winds are negative and tail winds are positive. A positive wind shear gives a decrease in head wind or changes head wind into tail wind. Similarly, a negative wind shear produces a decreasing tail wind (Fig. 15).

The derivation shown here is based upon wind shear derivations by Gera[7] and Sherman[8]. However, three-dimensional wind velocity complicates the equations of motion. Because of the proximity to the ground, a simplification in the wind profile can be made. Near the ground, the vertical component of the wind velocity is typically small when compared with the horizontal component and can be neglected[8,13]. Therefore, the 6-DOF model previously developed can be reduced to a longitudinal 3-DOF system.

This wind shear model is developed with the assumption that the earth-axes are fixed in space and the wind is either a horizontal head wind or tail wind. The absolute linear acceleration in the aircraft body-axes is:

$$\underline{V} = \frac{d}{dt}(U + u_w \cos\theta)\underline{i} + \frac{d}{dt}(W + u_w \sin\theta)\underline{k} +$$

$$+ q\underline{j} \times [(U + u_w \cos\theta)\underline{i} + (W + u_w \sin\theta)\underline{k}] \tag{41}$$

where $\times$ is the vector cross product. Eq. 41 simplifies to:

$$\dot{\underline{V}} = (\dot{U} + Wq + \dot{u}_w \cos\theta)\underline{i} + (\dot{W} - Uq + \dot{u}_w \sin\theta)\underline{k} \tag{42}$$

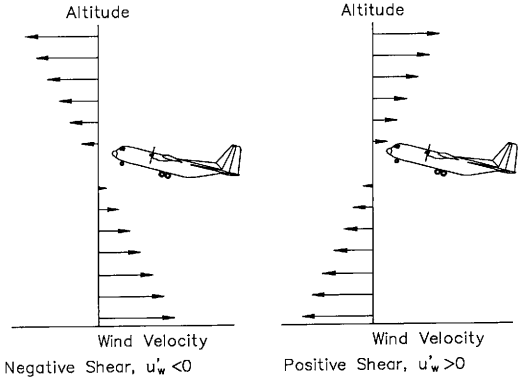

Fig. 15 Definition of Positive and Negative Wind Shears

Using the chain rule, the wind speed time derivative becomes:

$$\dot{u}_w = \frac{du_w}{dz}\frac{dz}{dt} = u_w' \frac{dz}{dt} \tag{43}$$

where u_w' is the vertical variation in wind speed with altitude and $\frac{dz}{dt}$ is the negative rate of climb of the aircraft:

$$\frac{dz}{dt} = -\frac{dh}{dt} = -U\sin\theta + W\cos\theta$$

and finally:

$$\dot{u}_w = u_w'(W\cos\theta - U\sin\theta).$$

The final form of the linear acceleration becomes:

$$\underline{\dot{V}} = [\dot{U} + Wq + u_w'(W\cos\theta - U\sin\theta)\cos\theta]\underline{i} +$$

$$+[\dot{W} - Uq + u_w'(W\cos\theta - U\sin\theta)\sin\theta]\underline{k} \tag{44}$$

Summing the forces on the aircraft and using Eq. 44, the modified longitudinal equations of motion become:

$$\dot{U} = \frac{\Sigma F_{X\,b}}{m} - Wq - g\sin\theta - u_w'(W\cos\theta - U\sin\theta)\cos\theta \tag{45}$$

$$\dot{W} = \frac{\Sigma F_{Z\,b}}{m} + Uq + g\cos\theta - u_w'(W\cos\theta - U\sin\theta)\sin\theta \tag{46}$$

$$\dot{\theta} = q \tag{47}$$

$$\dot{q} = \frac{\Sigma M_b}{I_{yy}}. \tag{48}$$

The ground track of the aircraft, $X_{a/c}$, must also be modified to reflect the horizontal wind. Because the wind in this derivation is either a head wind or

a tail wind, simply adding the magnitude of the wind to the $X_{a/c}$ equation will suffice. This equation has also been simplified for longitudinal motion:

$$\dot{X}_{a/c} = U\cos\theta + W\sin\theta + u_w. \tag{49}$$

Eqs. 45-49 represent the longitudinal equations of unsteady motion. In 1984, Frost and Bowles[14] developed 6-DOF equations of motion incorporating wind shear terms. The modified equations of motion developed here agree with those of Frost and Bowles when their equations are simplified for head winds and tail winds only.

An expression for u_w' must now be determined. Looking first at the gradient profile, u_w' can simply be specified as some constant gradient for all altitudes. The MIL-F-8785C profile provides a logarithmic variation of u_w'. The profile determines the wind speed at a given altitude by:

$$u_w = u_{20}\frac{ln(h/z_o)}{ln(20/z_o)} \tag{50}$$

where $z_o = .15$ for a Category C flight phase. From Eq. 50, u_w' is found by:

$$u_w' = \frac{u_w(h) - u_w(h+\Delta h)}{\Delta h}$$

where Δh is the height difference over which the gradient is calculated, normally 100 feet.

Both the gradient and logarithmic (MIL-F-8785C) wind shear profiles (Fig. 16) are implemented in the simulation and can be found in SUBROUTINE LD in Appendix B.

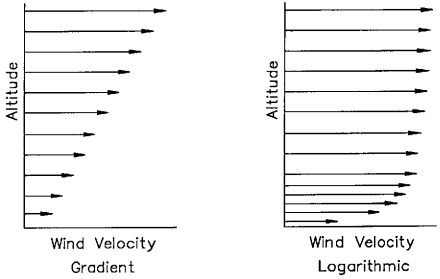

Fig. 16 Wind Shear Profiles

CHAPTER IV

TOTAL ENERGY SENSOR

Theory

A thorough understanding of the TES is necessary before it can be modeled. The sensing probe is rather simple: a 3/16" diameter steel tube with a 1/16" diameter orifice located two tube diameters from the slightly beveled tip. The probe is swept 20° into the flow with the orifice on the downstream side 5 to 6 inches from the aircraft surface (Fig. 17).

Because total energy is defined as the sum of potential and kinetic energies, Nicks' probe[4] must sense instantaneous pressure changes related to these energies if it is to function as a total energy instrument:

$$P_{sensor} = P_{ambient} - \bar{q}. \tag{51}$$

The non-dimensional form, pressure coefficient, is defined as:

$$C_p = \frac{P_{local} - P_{ambient}}{\bar{q}}. \tag{52}$$

Therefore, the pressure coefficient at the probe orifice ($P_{local} = P_{sensor}$) becomes:

$$C_p = \frac{(P_{ambient} - \bar{q}) - P_{ambient}}{\bar{q}} = -1.0 \tag{53}$$

Simply stated, the pressure coefficient at the orifice must be -1.0 to ensure total energy rate measurements by the probe. In his final design, Nicks showed that $C_p = -1.0$ for large sideslip angles of $\pm 80°$ and sweep angles of 10-25° (into the flow). Wind tunnel and flight tests verified that the TES was insensitive to

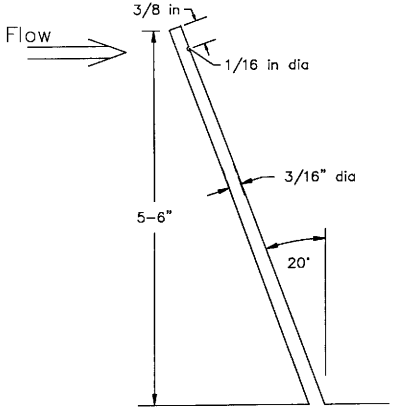

Fig. 17 Total Energy Probe

Reynolds number and altitude effects although Reynolds number does limit the low end of the airspeed envelope.

The probe output is displayed on a unit developed by Redwood Instrument Company, Incorporated (RICO). The probe is connected in series to a gust filter, a measurement tube and a reference chamber (Fig. 18). The measurement tube contains two 0.010" diameter thermistor beads (Fig. 19) which sense the flow of air within the measurement tube. These beads, made of a metallic oxide semiconductor material that is extremely temperature sensitive, are wired into an electrical bridge designed to maintain both thermistors at a constant temperature of 100°C above ambient. When a pressure difference exists between the probe orifice and the reference chamber, that is, a change in total energy is sensed, a very low Reynolds number flow will result in the measurement tube. This causes the upstream bead to be cooled and the downstream bead to be heated due to a hot air bubble from the upstream thermistor. The voltage required by the bridge circuit to equalize the bead temperatures is the output which drives the TES differential amplifier and feeds the cockpit display.

The display consists of two vertical "bar gauges": one displaying total energy rate (TE) from the TES in knots, and the other, vertical speed (VSI) from the aircraft static source in knots. These displays must be used concurrently to accurately determine the aircraft situation. If, for example, the aircraft is in level, unaccelerated flight, any change in altitude or airspeed will be displayed on the TE side while only altitude variations will be shown on the VSI side. To help the pilot recognize a variation in total energy rate, an audible click is activated when the TE is less than the VSI. This click varies in frequency and pitch with the amount of error between the displays.

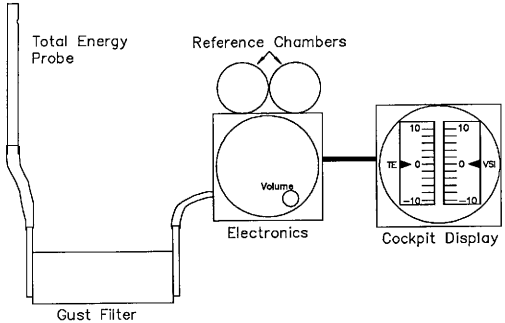

Fig. 18 TES Set Up

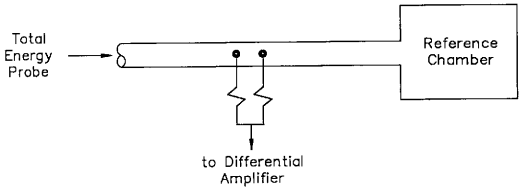

Fig. 19 Thermistors

Bench Tests

The TES bench tests were completed in the laboratories at TAMU with a test stand (Fig. 20) developed by Harp, Hart, and Rose[15]. Preliminary runs were made to determine which measurements would be necessary. The flow tube was connected to shop air and the flow regulated with the water separator valve. Initial and final velocities were computed from a pitot-static probe/water manometer combination. This combination was replaced with the total energy probe and RICO display package and connected to a Racal 4-channel data recorder. Channel 1 was the trigger signal marking the drop of the step-function gate; channel 2 was the TES output signal. The recorded signals were digitized using an IBM/PC compatible computer with a TECMAR analog/digital converter board and Labmaster[16] software.

In these runs, two major problems occurred: the signal was noisy and the response was not repeatable. By recording a constant signal and playing it back, it became apparent that the noise was due to the recorder. Therefore, the data recorder was omitted and the TES and trigger signals were fed directly into the computer A/D board run by the code in Appendix C.

The lack of repeatability was more difficult to remedy. Inaccuracies during visual inspection of the pitot-static probe/water manometer combination could account for small discrepancies; however, other errors were apparent. Because the shop air flow was not necessarily constant, the magnitude of the step change varied. This violated the assumption that once set, the step magnitude was constant. Between the time the step was measured and implemented, the flow velocities changed, giving an unknown input to the system.

The simplest remedy was to record the input pressure history for use in

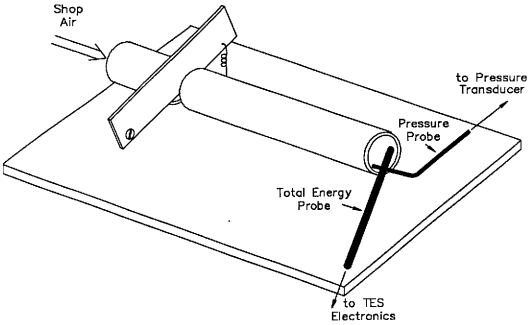

Fig. 20 Test Stand

modeling. The pitot-static probe was connected to a ± 0.125 PSID Validyne pressure transducer and fed to Channel 3 on the A/D board.

Modeling and Verification

Twenty-one runs were made recording time histories for both the TES and pressure signals. Matrix$_X$ was used to reduce the data and identify the model.

Initially, the TES and pressure signals were both voltages. Multiplying the TES signal by 20 kts/V and the pressure signal by 0.025 psi/V, the signals were transformed to physical quantities. However, the proper signal must be input to the system. For example, if the pressure at the orifice is constant (constant altitude and velocity), the system output is zero. Therefore, the system input must be the change in pressure instead of the measured pressure signal. The change in pressure, Δp, was found from:

$$\Delta p_i = p_i - p_{i-1}.$$

Once the input was found, the Matrix$_X$ MAXLIKE[12] algorithm was used to identify the model.

MAXLIKE is a maximum likelihood estimator that maximizes the likelihood of the parameter estimates given both the input and output histories. This algorithm operates on the entire time history at once instead of one point at a time. MAXLIKE requires that the model form be specified prior to identification. Based on Ostroff's results, a second order model was assumed:

$$\frac{\dot{TE}}{\Delta p} = \frac{A}{s^2 + Bs + C}.$$

Although the probe was the same, the electronics differed between Ostroff's tests and these tests, making duplication of Ostroff's results improbable.

The coefficients A, B and C were determined with 8 iterations of the MAXLIKE algorithm. These values with statistical results are given in Table 2. The means of these coefficients identify the TES model as:

$$\frac{\dot{TE}}{\Delta p} = \frac{19464}{s^2 + 2.1448s + 2.4116}. \tag{54}$$

Comparing Eq. 54 with Ostroff's model (Table 3) shows the dissimilarity of the two representations. A significant difference exists in the poles of the two models; one model has complex poles and the other has real poles. In addition, Ostroff's model has a zero at $s = 0$ that is not present in Eq. 54. These differences are not surprising however. Although the same probe was used in both modeling procedures, the display units were markedly different causing the variation in the models. Four of the experimental runs and associated model fits are shown in Fig. 21.

TABLE 2 TES Modeling Parameters

Run	Error	A	B	C
1	7.62	12251	1.7308	2.0214
2	9.18	19205	2.3124	2.0138
3	6.76	14446	1.9636	1.7384
4	11.23	14154	2.4973	1.7959
5	6.20	19615	1.8494	2.3400
6	13.24	27443	2.2986	2.4163
7	13.15	36843	3.4122	3.3151
8	10.41	22731	2.4178	2.4826
9	10.40	15541	1.9658	1.7001
10	11.36	22196	2.5383	2.0880
11	6.17	24083	2.2832	1.6981
12	15.42	33011	2.7697	2.6568
13	4.10	15828	2.7131	2.2483
14	3.78	9567	2.0925	1.5369
15	5.29	17443	3.0218	1.7564
16	10.31	23721	2.4853	2.7342
17	5.60	17005	2.9469	2.2395
18	6.93	15154	1.9883	3.4450
19	6.27	15683	2.6899	3.3033
20	5.31	17983	3.1197	3.2999
21	7.86	14842	2.9576	3.8136
Standard Deviation			0.4577	0.6753

TABLE 3 Comparison of Anderson and Ostroff TES Models

Researcher	Transfer Function	Poles	ζ	w_n
Anderson	$\frac{19464}{s^2+2.4788s+2.4116}$	$-1.2394 \pm 0.9357i$	0.7981	1.5529
Ostroff	$\frac{1.344s}{s^2+2.96s+1.344}$	$-0.56, -2.4$	–	–

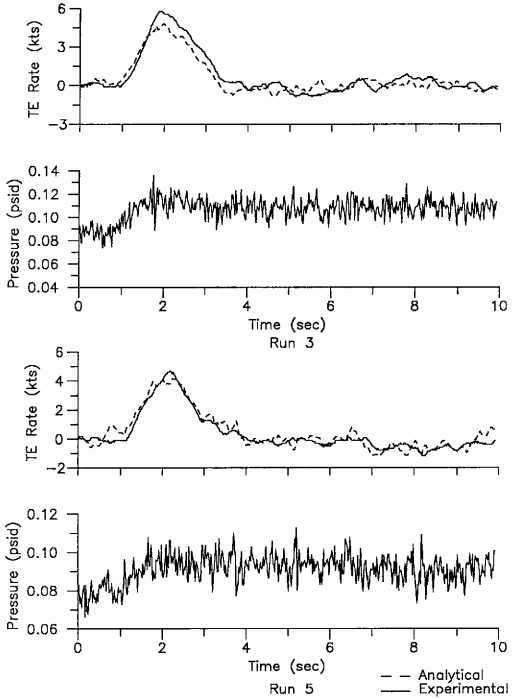

Fig. 21 TES Modeling Samples

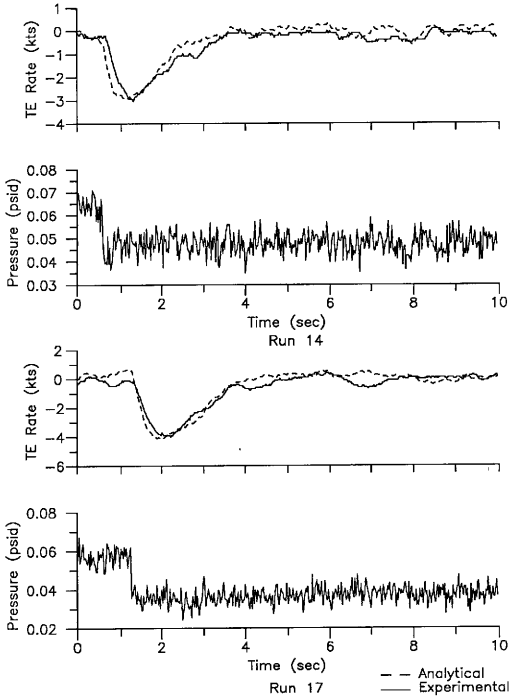

Fig. 21 Continued

CHAPTER V

TOTAL ENERGY FEEDBACK RESULTS

The modified equations of motion (Eqs. 45-49), wind shear model (Eq. 50) and TES model (Eq. 54) have been combined to produce the simulation given in Appendix B. A Digital Equipment Corporation VAX-8600 with FORTRAN-77 version 4.5 was used to run these simulations using 4^{th}-Order Runge Kutta integration and a 0.05 second time step.

Six runs have been made to evaluate the usefulness of the TES in a closed-loop environment. In each case, the simulation begins trimmed at 600 feet with either no wind, a 20 knot tail wind, or a 20 knot head wind. With both the glidepath and speed controllers engaged, the HTTB descends to 400 feet at which point the wind shear begins. The shear is encountered until the HTTB passes through 64-288 feet depending on the severity of the shear. The HTTB continues its descent until it touches down without flare. The ± 20 knot wind speed was agreed upon by this researcher and Lockheed engineers as a representative wind envelope. The wind shear gradients used in the gradient wind profiles vary between ± 0.3/sec. These represent severe wind shears that are slightly larger than those found in actual data[3,7,8].

Even though the HTTB is to be equipped with a high sink rate landing gear, care must be taken to stay below its 14 fps sink rate limit. Because of this limitation and the fact that the HTTB is executing a STOL approach, certain wind constraints must be observed. In the no wind case, the aircraft sink rate is 13.94 fps at touchdown. If a steady tailwind is introduced, the sink rate must increase due to the increased ground speed. Landing with a tail wind will cause the sink rate at touchdown to exceed structural limits. Therefore, no surface tail winds were

simulated.

The six simulation profiles are shown in Table 4. Each approach was flown twice, both with and without TE rate feedback, with the exception of the first approach which encountered the logarithmic wind profile of MIL-F-8785C. When total energy feedback is engaged, four different controller designs are used. It must be stressed that the controllers used are in no way optimal. They have been designed to illustrate the effect of TE rate feedback and have been tuned just enough to provide an improved response over the no-feedback case. Future work could apply optimal control theory to these compensators to reduce control deflections by heavily weighting the control matrix in the cost function.

Profile A: 20 knot Tail Wind, MIL-F-8785C, No Wind

Fig. 22 shows the response of the HTTB simulation to a light wind shear. This logarithmic shear (u_{20}=15 knots) does not appreciably affect the STOL approach. However, some small oscillations are noticeable in α, θ, altitude error and the controls. These oscillations are due to the steady 20 knot tail wind in the initial conditions. As discussed in the controller design, the state feedback gains were developed for the no wind case. As the initial conditions depart from this, an initial transient is introduced in the elevator. For the 20 knot tail wind, an elevator deviation of 4 degrees is encountered which causes a 1.5° variation in angle of attack and pitch. However, this elevator-induced oscillation does not preclude glidepath tracking.

Because of the negligible effect of this wind shear profile, the remainder of the simulation runs use gradient wind shears. This is not to imply that a logarithmic wind profile is unimportant or unlikely to occur. However, the military specification

TABLE 4 Simulation Wind Profiles

Profile	Starting Wind (kts)	Starting Altitude (ft)	Shear	Ending Wind (kts)	Ending Altitude (ft)
A	20	103	MIL–F–8785C	0	0
B	20	400	−0.3	0	288
C	20	400	−0.3	−20	176
D	0	400	−0.3	−20	288
E	−20	400	+0.3	0	288
F	−20	400	+0.1	0	64

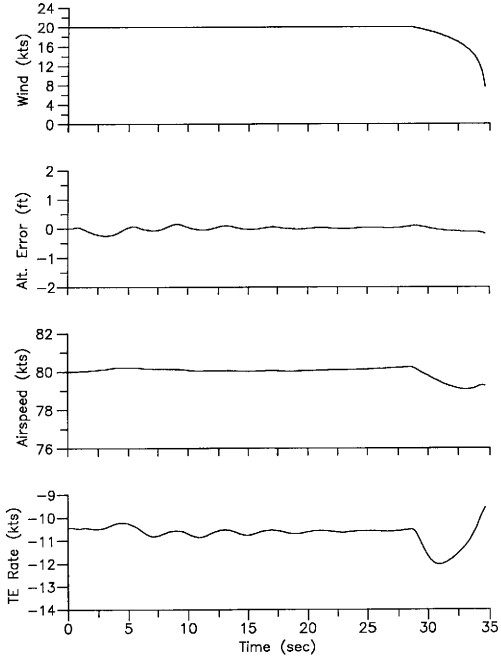

Fig. 22 Profile A: 20 knot Tail Wind, MIL-F-8785C, No Wind

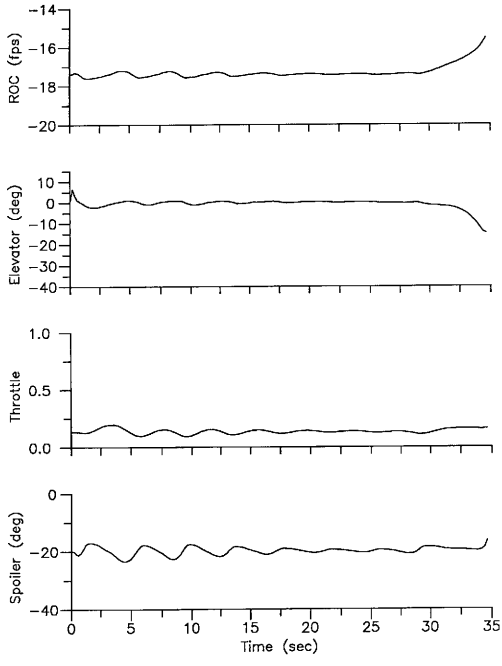

Fig. 22 Continued

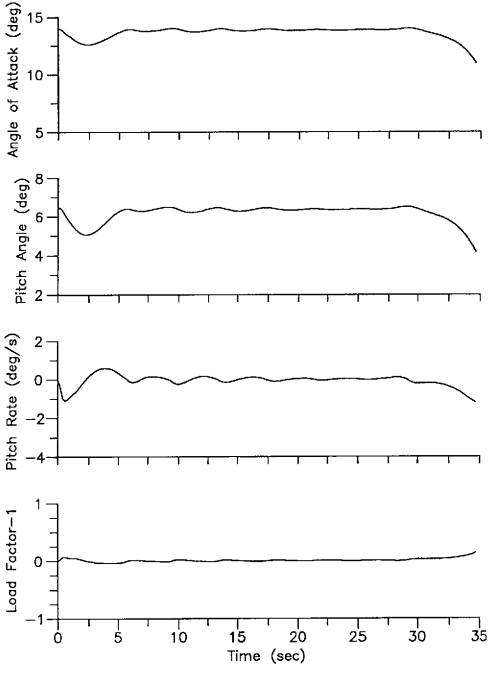

Fig. 22 Continued

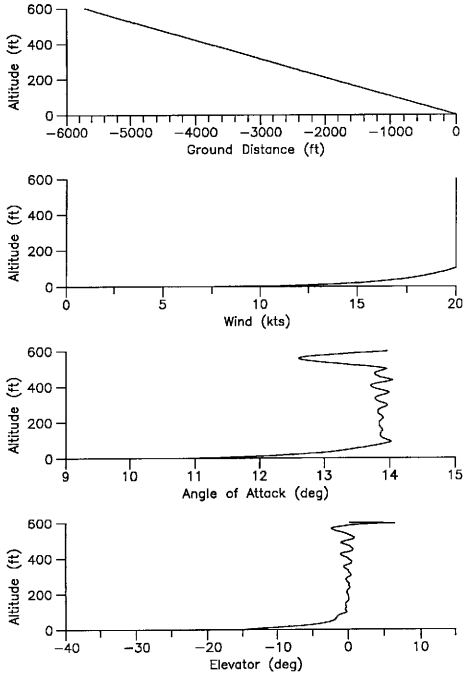

Fig. 22 Continued

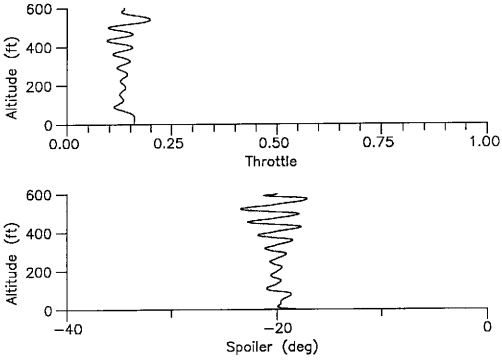

Fig. 22 Continued

for this wind profile has the largest wind gradients occurring below 20 feet AGL providing little altitude for recovery.

Profile B: 20 knot Tail Wind, −0.3 Gradient, No Wind

Unlike the logarithmic shear profile of MIL-F-8785C, the gradient shear profile causes the HTTB to significantly deviate from its desired airspeed and glidepath. As the aircraft enters the shear layer at 400 feet (Fig. 23), the airspeed increases to 89.6 kts and climbs 30 feet above the glidepath. The HTTB passes through the bottom of the shear layer 15 seconds later and begins recapturing the glidepath and commanded airspeed. However, a major problem is apparent from the angle of attack plot.

Miele, *et. al.*[17], showed that at the end of a shear layer, the angle of attack is near stall. The α history of Fig. 23 supports this. However, in this simulation, the data tables used for the non-dimensional forces and moments[10] are limited to $-6° < \alpha \leq 18°$. At 24 seconds into the approach, the angle of attack is 22°. Any time $\alpha > 18°$ in this simulation, the actual response of the HTTB is unknown. Incidentally, when angle of attack did exceed its limits, the force and moment coefficients were held constant at their 18° value eliminating any chance of a stall. Therefore, α must be kept within the limits of the data.

Because of time constraints on this research, a very primitive angle of attack limiting controller was developed and incorporated with the stabilizing state feedback controller (Fig. 24). In the normal OFF position, the α-limiter has no effect on the elevator, but as α exceeds the 16.5° threshold, the α-limiter commands a decrease in angle of attack. This threshold was selected so that when a large $\dot{\alpha}$ is present, the HTTB still remains below its α limit. The proportional and derivative

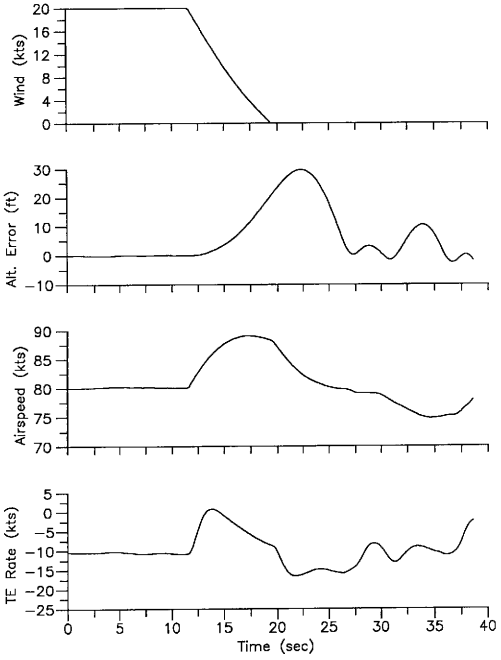

Fig. 23 Profile B: 20 knot Tail Wind, −0.3 Gradient, No Wind, First Run

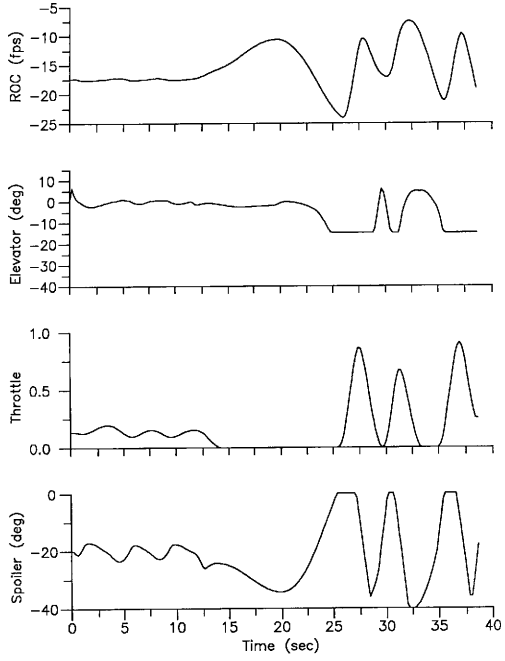

Fig. 23 Continued

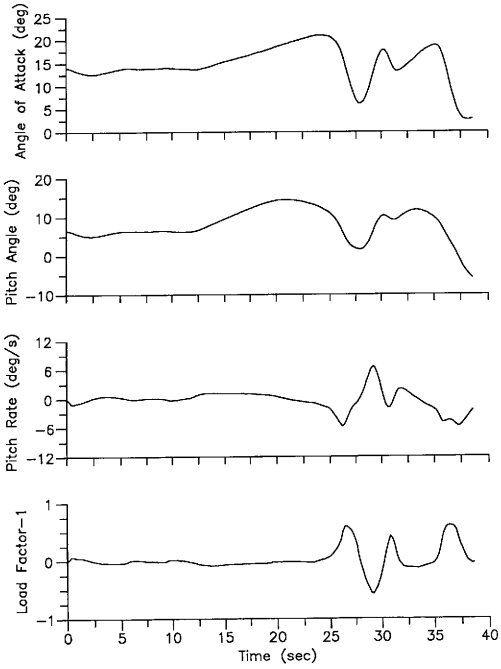

Fig. 23 Continued

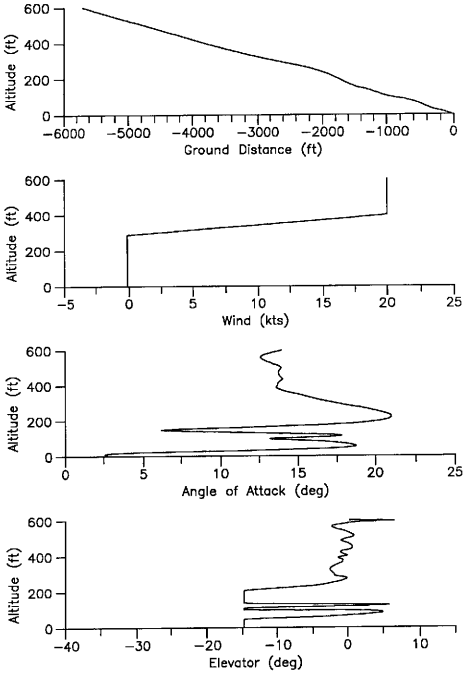

Fig. 23 Continued

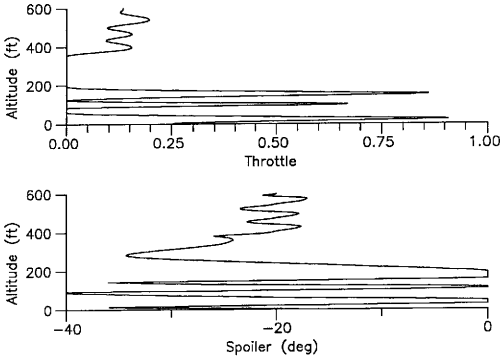

Fig. 23 Continued

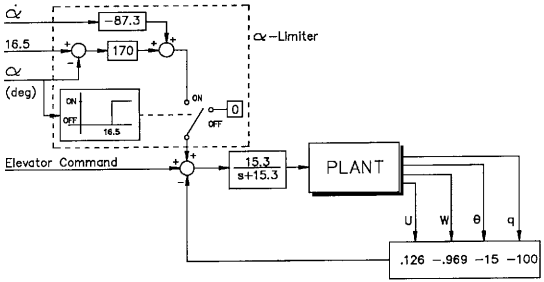

Fig. 24 Stabilizing State Feedback with α-Limiter

controller gains were found by a rather rudimentary trial and error process. Consequently, this controller, as with all of the others, is not optimal.

Profile B was rerun with the α-limiter engaged. A comparison of Figs. 23 and 25 shows that in this case, the α-limiter has little effect on the altitude and airspeed errors between the two runs. Nonetheless, angle of attack has been limited to 18°.

Once out of the shear layer, both airspeed and altitude error return to their commanded values. At touchdown, the HTTB is pitched up 8° with a rate of descent of 14.5 fps.

Including TE rate feedback in the control laws improves the glidepath and altitude tracking of the HTTB (Fig. 25). The TE rate controller (Fig. 26) was designed to improve the wind shear response without violating the angle of attack limits. Initially, the proportional error was sent to the elevator as a pitch command. This reduced the glidepath deviation, but adversely affected α; that is, the modified pitch command would sometimes override the α-limiter causing short duration α peaks above 18°. By using the derivative and integral of the error signal to modify the DLC command, the angle of attack peaks are eliminated. The TE rate feedback controller was tuned by manually varying the gains until an improved response was achieved.

A reduction in the rate of descent variation is apparent providing a softer touchdown at 13.2 fps. However, because the angles of attack in both the basic controller and TE rate feedback controller runs are at the limit, only a small improvement in altitude error can be expected (Fig. 25). This difference is apparent in the wind plot. With TE rate feedback, the HTTB climbs 26 feet above the glidepath, 4 feet less than with the basic controller. This means that the HTTB with TE rate feedback passes through the lower boundary of the shear layer before

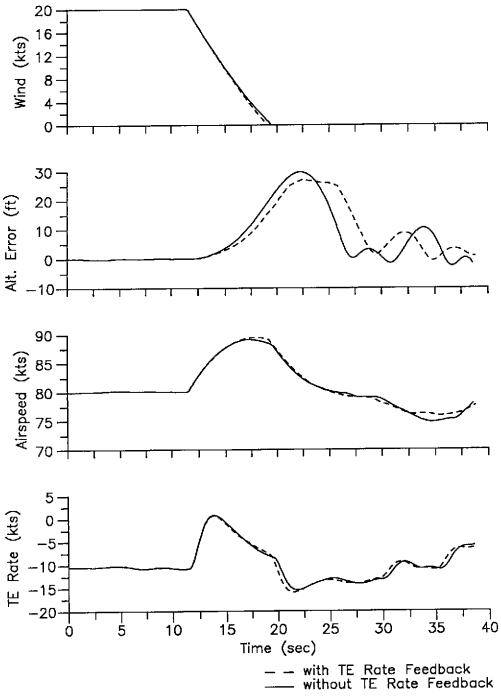

Fig. 25 Profile B: 20 knot Tail Wind, −0.3 Gradient, No Wind, Second Run

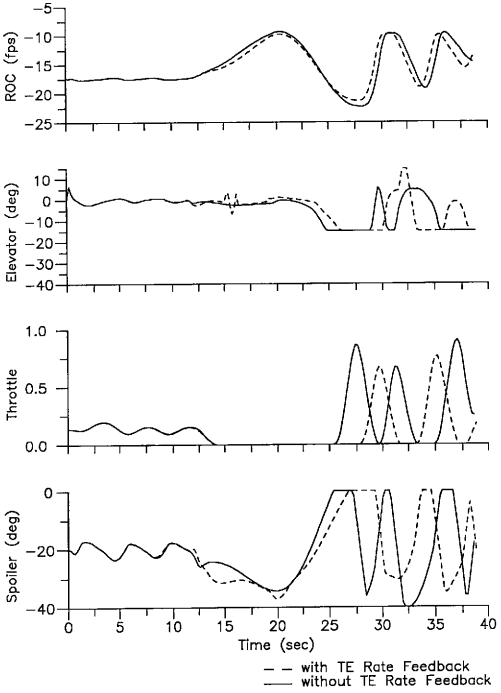

Fig. 25 Continued

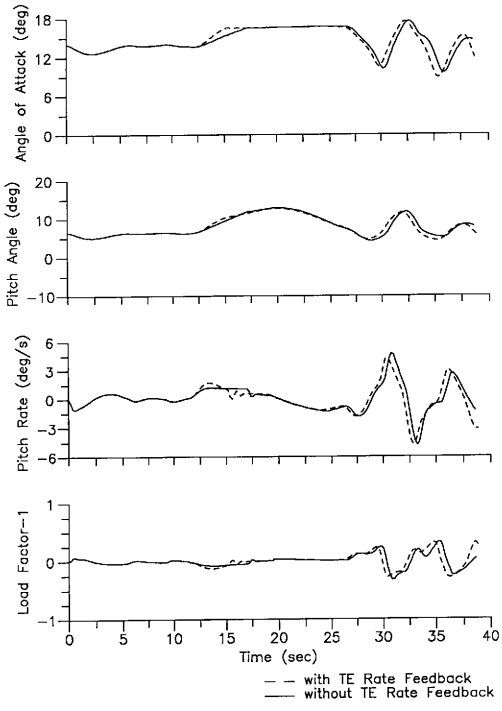

Fig. 25 Continued

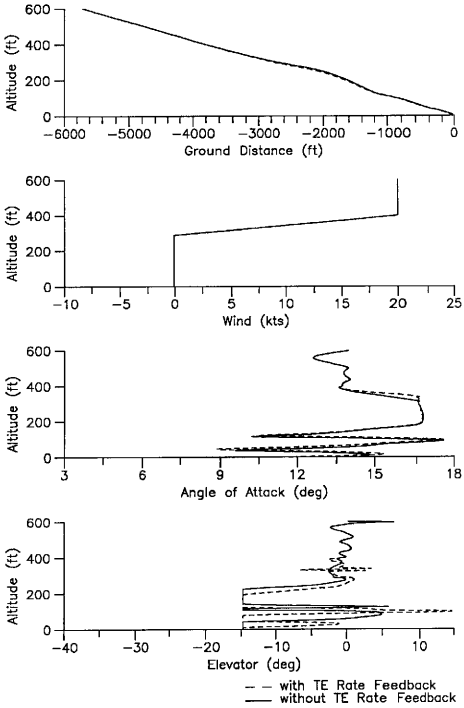

Fig. 25 Continued

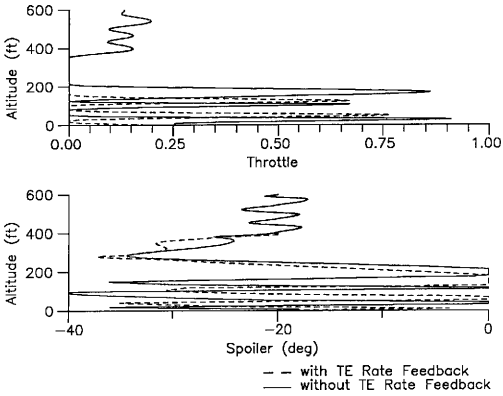

Fig. 25 Continued

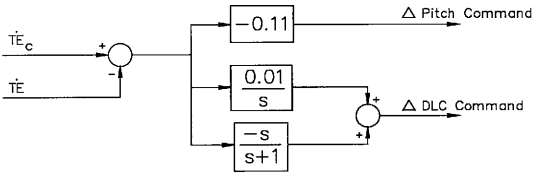

Fig. 26 TE Rate Feedback, Profiles B and C

it does with the basic controller. This explains the difference in the two traces of the wind plot.

A significant result of using TE rate feedback is the reduction in the control movements. Although the difference is small, the elevator, throttle and spoiler plots of Fig. 25 show that the control deflections are both smoother and smaller than those of the basic controller. In both cases however, rather large, quick throttle excursions are present. Even though these movements are physically possible, they are undesirable due to the increased engine wear and reduction of engine life. Tuning the controller with optimal gains, that is, gains set to minimize control deflection yet track the command, may reduce this motion.

Profile C: 20 knot Tail Wind, −0.3 Gradient, 20 knot Head Wind

By doubling the width of the shear layer, the HTTB experiences a tail wind changing to a head wind that causes a 54 foot altitude error and a 9 knot airspeed error (Fig. 27). After exiting the shear layer, the aircraft begins recovering from these deviations. However, just before touchdown, there is a noticeable pitch down due to an elevator pulse. This control deflection is necessary to maintain airspeed which has dropped below the commanded 80 kts. However, this elevator pulse causes the HTTB to overshoot its touchdown point by 74 feet.

Using the same TE rate feedback configuration as in Profile B (Fig. 26), the altitude deviation is reduced by 6 feet (Fig. 27). Once again, it seems that α is limiting the effectiveness of TE rate feedback. Close inspection of the time histories reveals that the only noticeable differences between the two approaches are the points at which α reaches its limiting value and the controls are activated. This implies that a large angle of attack is necessary to counteract the shear. However,

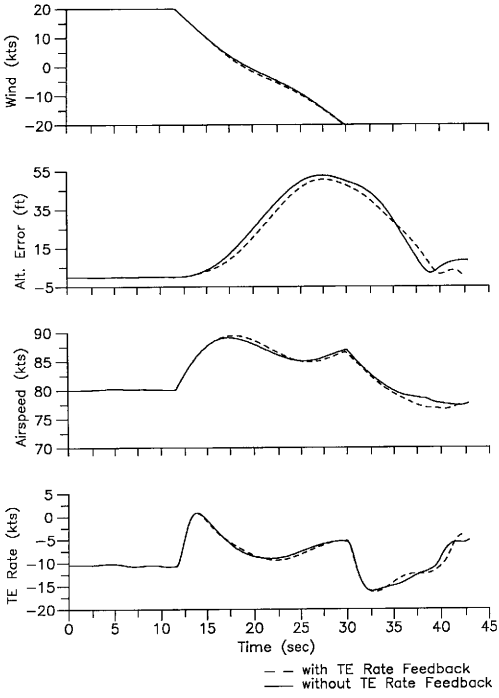

Fig. 27 Profile C: 20 knot Tail Wind, −0.3 Gradient, 20 knot Head Wind

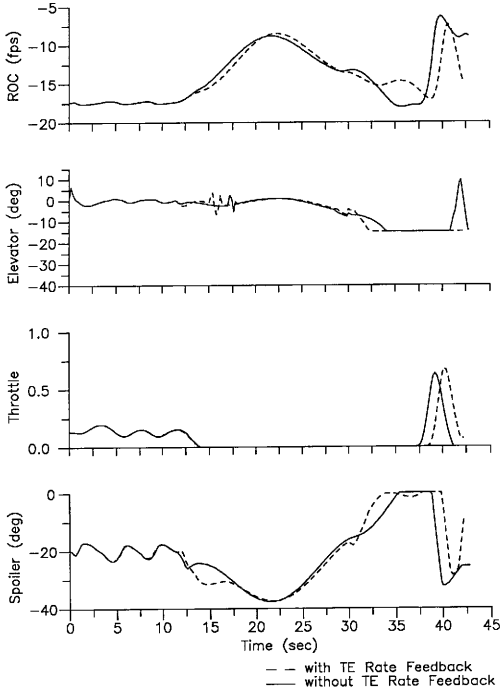

Fig. 27 Continued

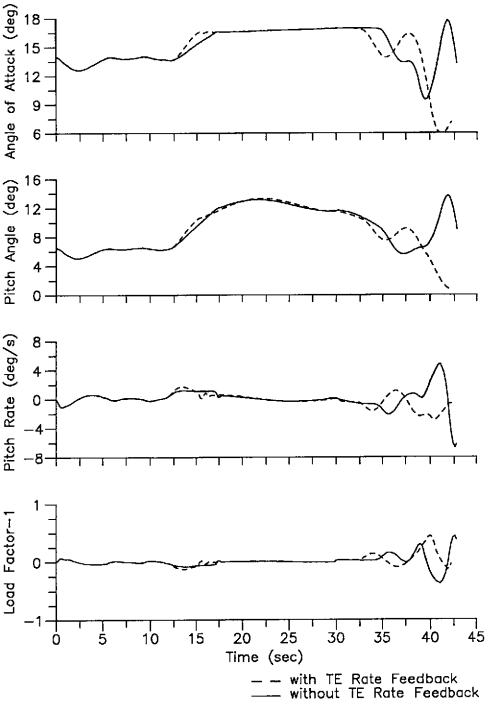

Fig. 27 Continued

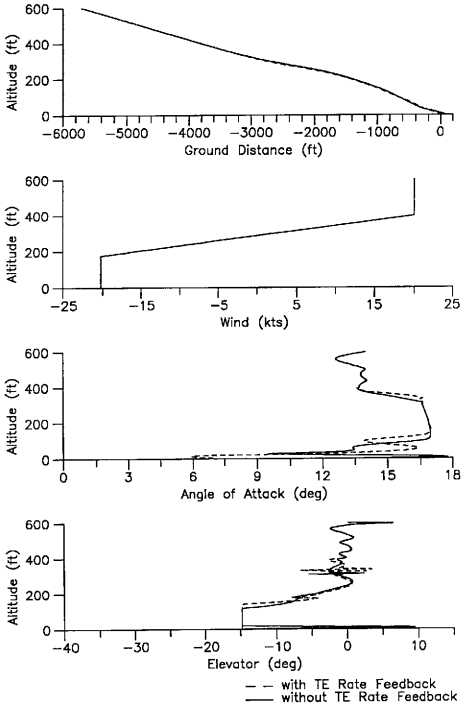

Fig. 27 Continued

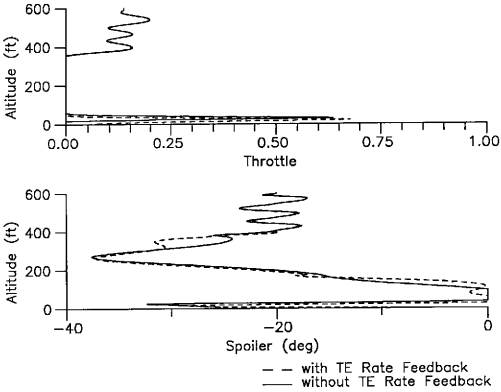

Fig. 27 Continued

this is not entirely accurate. The increase in α is necessary not to increase lift, but to increase drag for a larger sink rate. Consequently, because throttle is already idle and spoilers are nearly saturated, the only remaining way to increase drag is to increase induced drag. The leading nature of TE rate feedback is also apparent in the elevator and spoiler traces. These controls are activated 3 seconds before they were with the basic controller.

At 37 seconds into the TE rate feedback approach, the HTTB again pitches down. This time however, the change in θ, ROC and α is due to a throttle burst. As the HTTB approaches the glidepath, the throttle, and hence spoilers, increases (deploy) to avoid passing below the glidepath. The throttle causes the pitch down and the spoilers cause the increased sink rate such that the aircraft lands within 5 feet of the desired touchdown point with a descent rate of 15 fps in a nearly level attitude.

This increase in sink rate is undesirable because it exceeds the sink rate limit imposed by the landing gear. By decreasing the derivative gain of the glidepath controller, this large throttle deviation can be reduced. Unfortunately, this relaxation of the damping may degrade the transient response of the glidepath controller.

Profile D: No Wind, -0.3 Gradient, 20 knot Head Wind

This profile starts with no wind and a much lower angle of attack, 7.2°, than the tail wind case (Profile C) did previously, 14°. Assuming that α is limiting the TE rate effectiveness, a large improvement in tracking should result from TE rate feedback because of the lower initial angle of attack.

Looking first at the basic controller approach (Fig. 28) the HTTB climbs 37

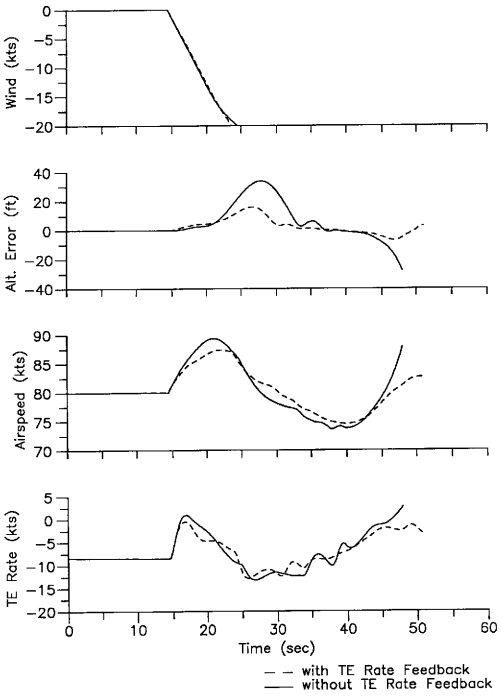

Fig. 28 Profile D: No Wind, −0.3 Gradient, 20 knot Head Wind

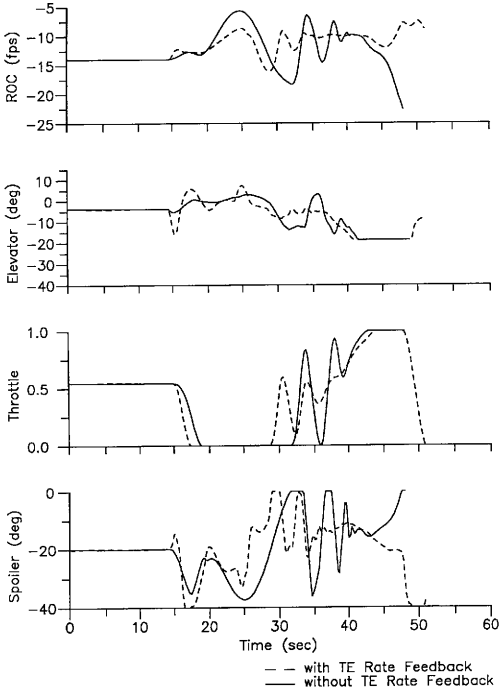

Fig. 28 Continued

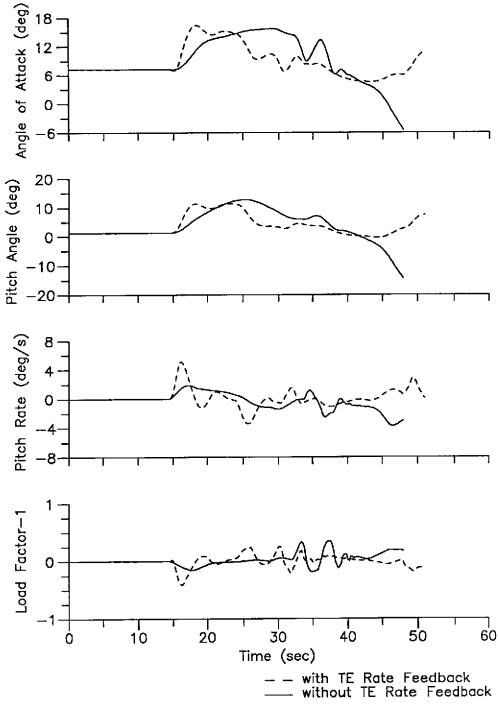

Fig. 28 Continued

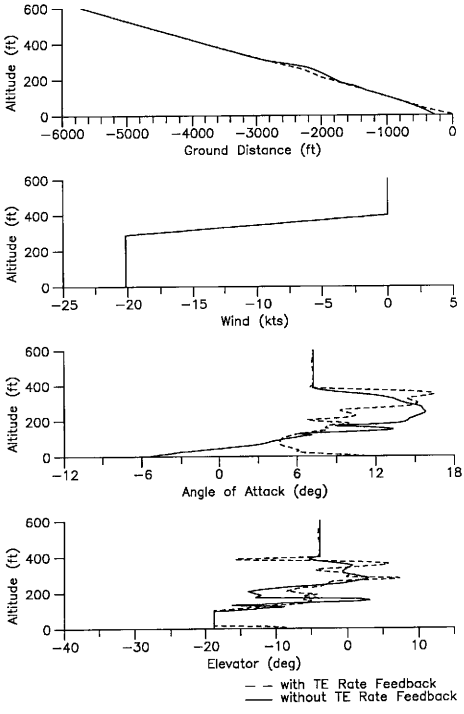

Fig. 28 Continued

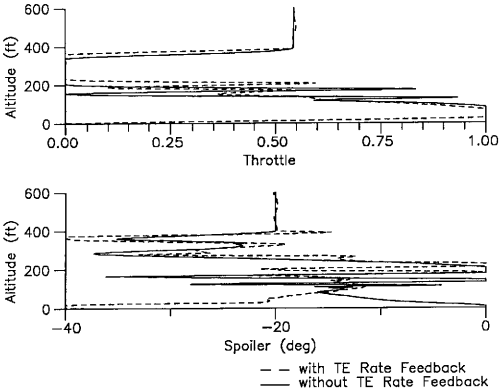

Fig. 28 Continued

feet above the glidepath to an airspeed of 89.3 kts. At 36 seconds and 190 feet AGL, the aircraft stalls and does not recover, crashing 290 feet short of the touchdown point in a 15° nose-down attitude. This stall occurs at an angle of attack of 13° and airspeed of 74 kts. The α history shows that the angle of attack had been exceeding 13° for over 12 seconds before the HTTB finally stalled. However, it was not until the end of that 12 second interval that the thrust coefficient dropped low enough to cause the stall. Reviewing the aircraft data table set up[10] clarifies this.

Simply stated, stall occurs when the wings stop producing lift, and because lift is dependent on angle of attack,

$$Lift = f(\alpha),$$

stall is normally associated with some stall angle of attack, the point where lift begins decreasing with α. This relationship is not constant over all flight regimes though. Lift is not only dependent on α, but also thrust coefficient, T_c. That is, the larger the thrust coefficient, the larger the angle of attack before stall begins:

$$Lift = f(\alpha, T_c)$$

. Therefore, if T_c had remained large (with the same α history), the HTTB would not have stalled and crashed (Table 5).

Including the TE rate feedback controller of Fig. 29 greatly improves the HTTB response to wind shear (Fig. 28). The altitude error is reduced to 16 feet while the airspeed peaks at 87 kts. The most important outcome, however, is the elimination of the stall and ensuing crash; the HTTB lands with a sink rate of 8 fps within 20 feet of the target. This is accomplished by using smoother and smaller control deflections and keeping tighter control on α. When the aircraft enters the shear

Table 5 Basic Lift Coefficient of HTTB with Full Flaps

Thrust Coefficient	Alpha									
	−8	−4	0	4	8	10	12	14	16	18
0.0	1.0773	1.5862	2.0553	2.5729	3.0573	3.2607	3.4407	3.5134	3.3122	3.1248
0.4	1.5931	2.3137	3.0276	3.6628	4.1114	4.2510	4.3740	4.4674	4.4668	4.0936
0.8	2.2184	2.8660	3.6307	4.2983	4.8031	5.0332	5.1491	5.2648	5.3313	5.1963
1.2	2.2733	3.1042	3.9292	4.7016	5.3454	5.6072	5.8255	5.9643	6.0242	5.9361
1.6	2.4358	3.2791	4.1227	4.9197	5.6103	5.8879	6.0811	6.2863	6.3764	6.3389
2.0	2.4980	3.3521	4.2106	5.0336	5.7374	6.0337	6.2730	6.4436	6.5401	6.5200

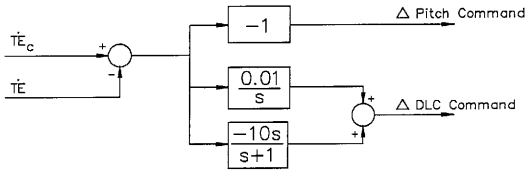

Fig. 29 TE Rate Feedback, Profile D

layer, however, a larger increase in α is noticeable. As with the other profiles, drag is controlled by angle of attack. When out of the shear layer, drag can be reduced so α returns to its original value. Once again, the controls, more specifically elevator and throttle, are actuated before they were by the basic controller.

Profile E: 20 knot Head Wind, +0.3 Gradient, No Wind

In this profile, an increasing tail wind is experienced by the HTTB. The difference in sign of the gradient will appreciably change the aircraft response from Profiles B-D. Fig. 30 shows the catastrophic result of the shear. As the aircraft passes through the shear layer, the basic controller can compensate well enough to control the HTTB until the 23 second mark when the controls become saturated and can no longer maintain altitude. When the HTTB encounters a positive wind shear, the increasing tail wind reduces the relative wind speed and hence the amount of lift produced. To compensate for this, the controller initiates a pitch up and a speed increase. Unfortunately, the controls become saturated with these conflicting commands and the aircraft crashes before the full recovery can be executed.

Fig. 31 shows the configuration of the TE rate feedback controller used in Profile E. The traces with TE rate feedback in Fig. 30 emphasize the effect of using TE rate feedback. Although the aircraft still varies from the glidepath by more than 100 feet, the controller remains effective enough to recapture the glidepath by increasing the airspeed and angle of attack with earlier control deflections. The touchdown with TE rate feedback is somewhat rough however; the HTTB has a sink rate of 17 fps and a 9° nose down pitch attitude. Most importantly, though, TE rate feedback gives the pilot 14 extra seconds to realize the situation and attempt to abort the approach.

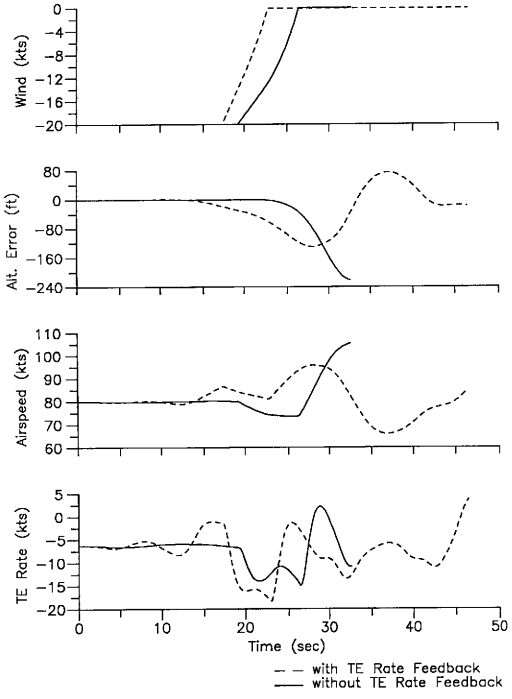

Fig. 30 Profile E: 20 knot Head Wind, +0.3 Gradient, No Wind

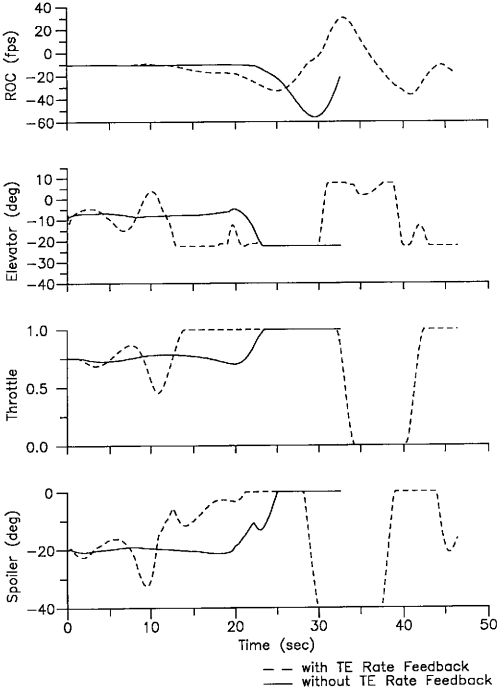

Fig. 30 Continued

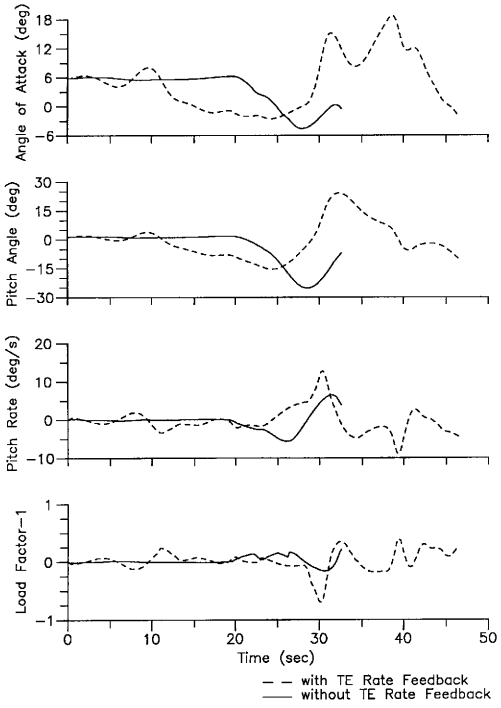

Fig. 30 Continued

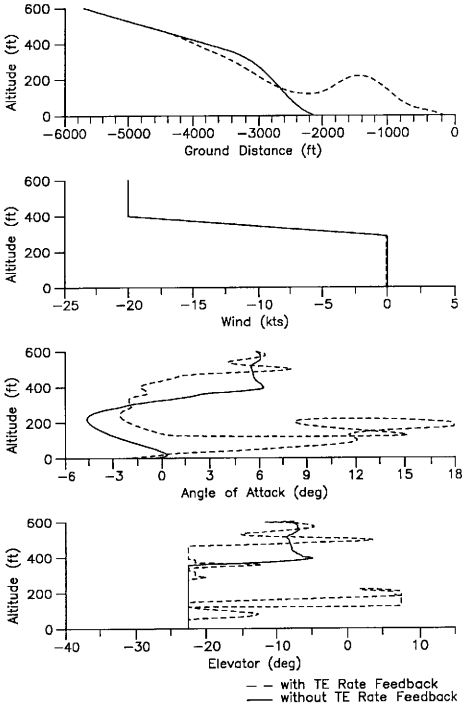

Fig. 30 Continued

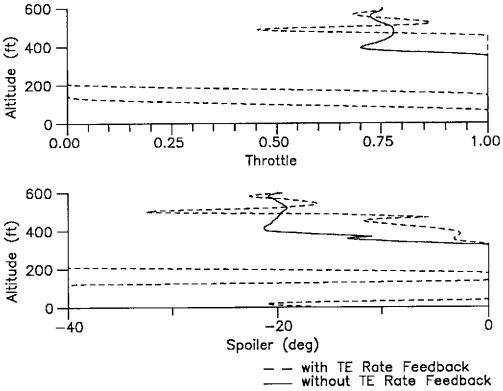

Fig. 30 Continued

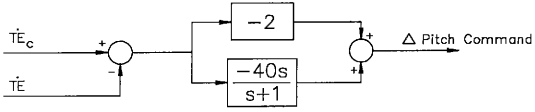

Fig. 31 TE Rate Feedback, Profile E

Profile F: 20 knot Head Wind, +0.1 Gradient, No Wind

The previous wind profiles have dealt with extremely severe wind shears that are somewhat unlikely to occur in nature. In some of these cases, the controller response included large control deflections that are unacceptable in normal flight. To evaluate the effectiveness of TE rate feedback on a physically reasonable wind shear profile, a gradient of +0.1/sec is used for this approach.

There is a great difference between the HTTB responses of Profiles E and F. As it enters the shear layer, the HTTB drops 17 feet below the glidepath before it recaptures the glidepath 5 seconds later (Fig. 32). Unlike earlier shear profiles, the aircraft is capable of recapturing the glidepath before it leaves the shear layer. This deviation and the variations in airspeed and ROC do not prevent the aircraft from landing on the target.

Incorporating a TE rate feedback controller (Fig. 33) almost negates the effects of shear penetration and better controls the airspeed by using much smoother control movements. This is most apparent in the spoiler where the high-frequency oscillations have been eliminated. Low frequency oscillations are evident in the controls which cause similar responses in α and θ. Modification of the control laws, both the basic and TE rate controllers, would eliminate these oscillations.

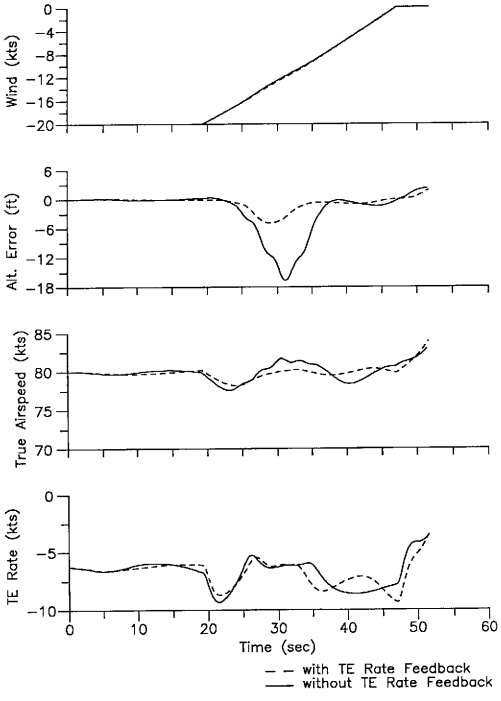

Fig. 32 Profile F: 20 knot Head Wind, +0.1 Gradient, No Wind

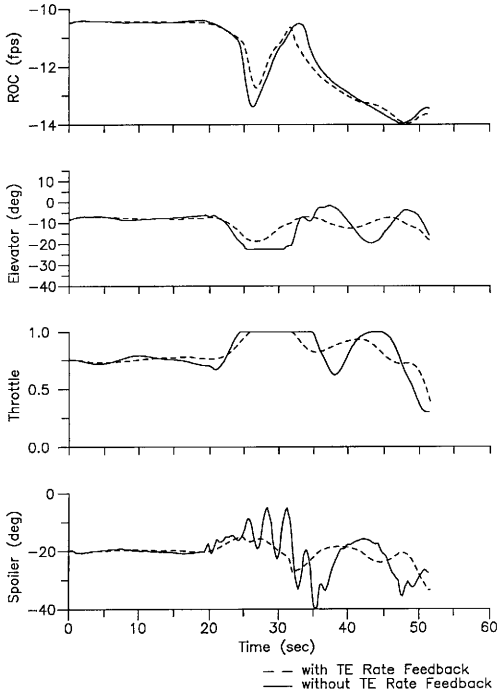

Fig. 32 Continued

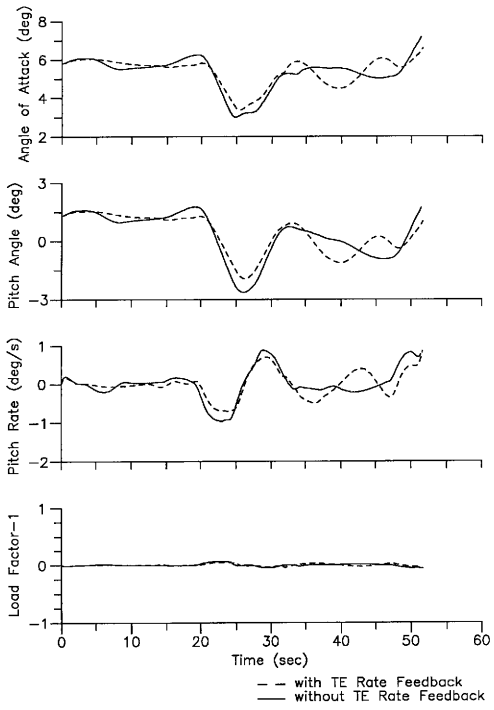

Fig. 32 Continued

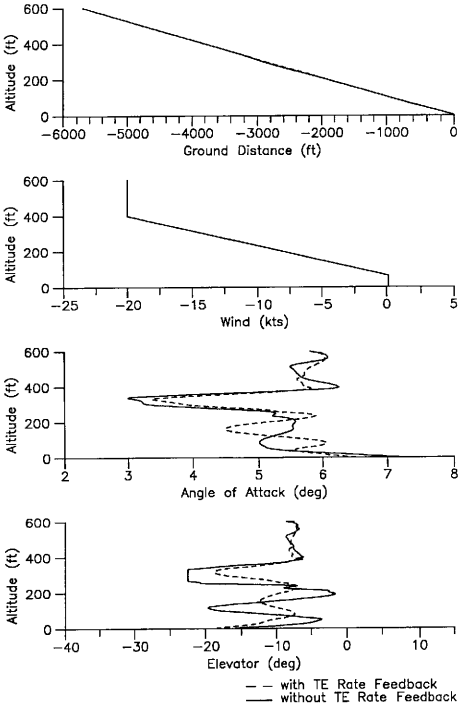

Fig. 32 Continued

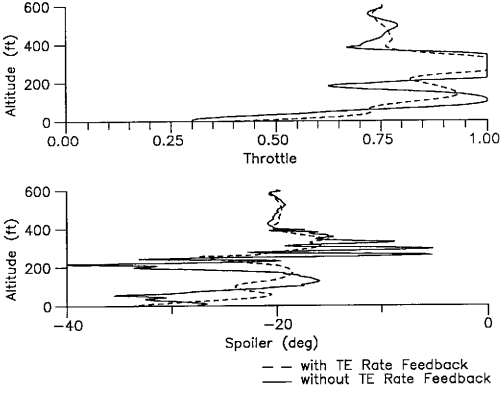

Fig. 32 Continued

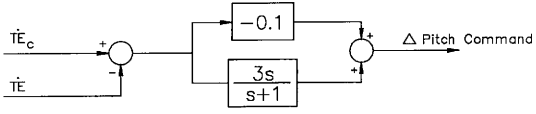

Fig. 33 TE Rate Feedback, Profile F

CHAPTER VI

CONCLUSIONS AND RECOMMENDATIONS

Conclusions

The TES performed well with no indications of modeling error. Nevertheless, the standard deviations of the individual model coefficients are rather large. It may prove beneficial to run more bench tests in an effort to enlarge the data base and reduce the scatter of the individual models' coefficients. Using a steady flow source might help reduce the oscillations present in the pressure histories. In addition, any flow tube boundary effects that are present can be minimized by increasing the diameter of the flow tube upstream of the TES and pitot probe. Irregardless, the second-order model developed here resembles Ostroff's representation enough to indicate that this model is reasonable even though the two are different as a result of differences in electronics.

Profiles B-F depict the usefulness of TE rate feedback in moderate and severe wind shears. By using this type of controller in a closed-loop environment, glidepath and airspeed can be better controlled and angle of attack more tightly maintained. However, work is necessary to refine the controllers, both the basic and TE rate feedback, before any other implementations are considered. Profile F (Fig. 32) illustrates the smoothing characteristic of TE rate feedback. Unfortunately, in the more severe shear profiles, the basic controller causes large, rapid control deflections as the HTTB passes through the shear layers. These can be reduced, if not eliminated, by employing optimal control theory to tune the controllers. A cost function with heavily weighted controls will reduce the deflections, but may also inhibit tracking.

Total energy rate feedback leads the basic controller. That is, the controls are activated 1-7 seconds before they are by the basic controller, allowing quicker response and less tracking error (Profiles C-E, Figs. 27, 28 and 30). The basic controller must wait until the aircraft actually deviates from the glidepath and commanded airspeed which may take several seconds due to the inertia of the HTTB. The TE rate feedback controller, on the other hand, begins corrective action as soon as a change in the total energy of the aircraft and/or surrounding air mass is sensed.

Although the simulation is an accurate representation of the aircraft in the data tables, the actual HTTB is not quite as limited as the data suggest. The data tables include angles of attack to 18°. However, profiles B-D fly near this limit but do not stall, implying that there is a region of the flight envelope above $\alpha = 18°$ at which the HTTB can safely operate without stalling.

The severity of wind shear that the simulation can endure is bounded by controller limits. The throttle and spoiler channels of the glidepath controller are full authority controllers. However, the PSCAS is not. The $\pm 15°$ from trim elevator limit is included as a safety factor. Relaxation of this feature would allow control through stronger wind shears, but could compromise safety if a hard-over failure were to occur.

Recommendations

Further work on TE rate feedback is necessary to more fully understand its impact on wind shear penetration. There are at least five areas that require further attention:

1. Expand the simulation to 6 degrees of freedom with 3 dimensional winds

and wind shears and evaluate the effect of the lateral/directional modes on TE rate feedback.

2. Revise basic controller and TE rate feedback logic. Because the effort of this research was to determine the usefulness of the TE rate signal, the actual control laws could be refined to provide improved responses. Lambregts of Boeing has developed an integrated total energy control system[18], a generic controller with few aircraft dependent features. This controller uses throttle to maintain total energy and elevator to distribute the energy between airspeed and altitude. Implementation of this controller could reduce the effect of the limitations imposed by angle of attack and more precisely maintain glidepath and airspeed.
3. Expand the aircraft data base to higher angles of attack enlarging the operating envelope of this simulation. However, this may lead to prolonged flight at or near the stall which is undesirable.
4. Develop go-around logic to automatically retract flaps and landing gear and abort the approach.
5. Expand the data base for the TES model to reduce the scatter of the coefficients from the individual runs.

REFERENCES

[1] Rutowski, E. S., "Energy Approach to the General Aircraft Performance Problem," *Journal of the Aeronautical Sciences*, Vol. 21, Mar. 1954, pp. 187-195.

[2] Joppa, R. G., "Wind Shear Detection Using Measurement of Aircraft Total Energy Change," University of Washington, NASA CR-137839, Aug. 1976.

[3] Gera, J., "Longitudinal Stability and Control in Wind Shear with Energy Height Rate Feedback," NASA Langley Research Center, NASA TM-81828, November 1980.

[4] Nicks, O. W., "A Simple Total Energy Sensor," NASA Langley Research Center, NASA TMX-73928, Mar. 1976.

[5] Ostroff, A. J., Hueschen, R. M., Hellbaum, R. F., Belcastro, C. H., and Creedon, J. F., "Evaluation of a Total Energy-Rate Sensor on a Transport Airplane," NASA Langley Research Center, NASA TP-2212, Nov. 1983.

[6] Oseguera, R. M., "Evaluation of a Total Energy Sensor for Glide Path Control," Aerospace Engineering, Texas A&M University, Master's Thesis, May 1987.

[7] Gera, J., "The Influence of Vertical Wind Gradients on the Longitudinal Motion of Airplanes," NASA Langley Research Center, NASA TND-6430, Sept. 1971.

[8] Sherman, W. L., "A Theoretical Analysis of Airplane Longitudinal Stability and Control as Affected by Wind Shear," NASA Langley Research Center, NASA TND-8496, July 1977.

[9] "High Technology Test Bed Flight Test Results - Baseline," HTTB Test Team, LG85ER0020, Lockheed-Georgia Company, Marietta, Georgia, Feb. 1985.

[10] "HTTB Dynamic Simulation Data for Sperry," Flight Controls Group, LG86WP7242-001, Lockheed-Georgia Company, Marietta, Georgia, March 1986.

[11] Junkins, J. L., *Optimal Spacecraft Rotational Maneuvers*, 1st ed., Elsevier Scientific Publishing Company, New York, 1985, pp. 13-25.

[12] "Matrix$_X$ Users Guide", Integrated Systems, Inc., Palo Alto, California, ver. 6.0.

[13] Anonymous, N.N.,"Flight Path Control in Windshear," Boeing Airliner, Seattle, Washington, Jan.-Mar. 1985, pp. 1-12.

[14] Frost, W., and Bowles, R. L., "Wind Shear Terms in the Equations of Motion," *Journal of Aircraft*, Vol. 21, No. 11, 1984, pp. 866-872.

[15] Harp, D. B., Hart, T. H., and Rose, J. A., "Dynamic Response Characteristics of a Total Energy System," Department of Aerospace Engineering, Texas A&M University, AERO 489C, Dec. 1985.

[16] "Lab Master User's Guide," TECMAR, Inc., Solon, Ohio, 1984.

[17] Miele, A., Wang, T., and Melvin, W.W., "Guidance Strategies for Near-Optimum Take-Off Performance in a Windshear," AIAA Paper, 86-0181, Jan. 1986.

[18] Lambregts, A. A., "Integrated System Design for Flight and Propulsion Control Using Total Energy Principles." Aircraft Design, Systems and Technology Meeting, AIAA, Fort Worth, Texas, 83-2561, Oct. 1983.

[19] Etkin, B., *Dynamics of Flight*, 2nd ed., John Wiley & Sons, Toronto, Canada, 1985. pp. 84-93.

[20] Roskam, J., *Airplane Flight Dynamics and Automatic Flight Controls*, 1st ed., Roskam Aviation and Engineering Corp., Ottawa, Kansas, 1982, pp. 11-33.

APPENDIX A

AIRCRAFT EQUATIONS OF MOTION

In the interest of completeness, the general, rigid-body equations of motion are derived from basic principles[19,20].

Assume that δm, an element of mass of the aircraft, has velocity $\underline{v}$ relative to the inertial axes and that $\delta \underline{F}$ is the force acting on it.

From Newton's second law, the equation of motion of δm is:

$$\delta \underline{F} = \delta m \frac{d\underline{v}}{dt}. \tag{A1}$$

If all of the elements of the aircraft are summed, Eq. $A1$ becomes

$$\Sigma \delta \underline{F} = \Sigma \delta m \frac{d\underline{v}}{dt} = \frac{d}{dt} \Sigma \underline{v} \delta m. \tag{A2}$$

$\Sigma \delta \underline{F}$ is the sum of all forces acting upon all the elements of the aircraft both internal and external. By Newton's third law, all of the internal forces cancel out because they occur in equal and opposite pairs. Therefore, $\Sigma \delta \underline{F}$ is simply the resultant external force $\underline{F}$ acting on the aircraft.

The velocity of δm is

$$\underline{v} = \underline{v}_c + \frac{d\underline{R}}{dt}. \tag{A3}$$

Substituting,

$$\Sigma \underline{v} \delta m = \Sigma (\underline{v}_c + \frac{d\underline{R}}{dt} \delta m) = m \underline{v}_c + \frac{d}{dt} \Sigma \underline{R} \delta m. \tag{A4}$$

However, C is the center of mass of the aircraft, so $\Sigma \underline{R} \delta m = 0$ leaving

$$\Sigma \underline{v} \delta m = m \underline{v}_c. \tag{A5}$$

Substituting into Eq. A2,

$$\underline{F} = m\frac{d\underline{v}_c}{dt}. \tag{A6}$$

Notice that the mass of the aircraft is assumed constant. Otherwise, an additional $\underline{v}_c \frac{dm}{dt}$ term would have to be included.

The relations between force and mass center motion have now been established. Similarly, the relation between moment and rotation can be developed by considering the angular momentum. The angular momentum of an element is defined as.

$$\delta\underline{h} = \underline{R} \times \underline{v}\delta m. \tag{A7}$$

Consider the rate of change of angular momentum:

$$\frac{d}{dt}(\delta\underline{h}) = \frac{d}{dt}(\underline{R} \times \underline{v})\delta m = \frac{d\underline{R}}{dt} \times \underline{v}\delta m + \underline{R} \times \frac{d\underline{v}}{dt}\delta m. \tag{A8}$$

From Eqs. A3 and A1,

$$\frac{d\underline{R}}{dt} = \underline{v} - \underline{v}_c$$

and

$$\underline{R} \times \frac{d\underline{v}}{dt}\delta m = \underline{R} \times \delta\underline{F}.$$

By defining $\delta\underline{G}$ as the moment of the elemental force about the center of mass:

$$\delta\underline{G} = \underline{R} \times \delta\underline{F}. \tag{A9}$$

Equation A8 becomes

$$\delta\underline{G} = \frac{d}{dt}(\delta\underline{h}) - (\underline{v} - \underline{v}_c) \times \underline{v}\delta m. \tag{A10}$$

However, $\underline{v} \times \underline{v} = 0$ so Eq. A10 becomes

$$\delta\underline{G} = \frac{d}{dt}(\delta\underline{h}) + \underline{v}_c \times \underline{v}\delta m. \tag{A11}$$

Summing for all elements yields:

$$\Sigma\delta\underline{G} = \frac{d}{dt}\Sigma(\delta\underline{h}) + \underline{v}_c \times \Sigma\underline{v}\delta m. \qquad (A12)$$

Using a similar argument for the force, $\Sigma\delta\underline{G}$ is the external moment $\underline{G}$ about the mass center. $\Sigma(\delta\underline{h})$ is the angular momentum $\underline{h}$ of the aircraft. Equation $A12$ simplifies to:

$$\underline{G} = \frac{d\underline{h}}{dt} + \underline{v}_c \times m\underline{v}_c$$

or

$$\underline{G} = \frac{d\underline{h}}{dt}. \qquad (A13)$$

This equation is valid when $\underline{G}$ and $\underline{h}$ are referred to the center of mass only. In general, other reference points do not apply. It should also be noted that Eqs. $A6$ and $A13$ allow for the relative motion of parts of the aircraft.

To determine $\underline{h}$, the aircraft angular velocity is defined as:

$$\underline{\omega} = p\underline{i} + q\underline{j} + r\underline{k}. \qquad (A14)$$

From mechanics, the velocity of an element in a translating and rotating body is:

$$\underline{v} = \underline{v}_c + \underline{\omega} \times \underline{R}. \qquad (A15)$$

Applying this to the definition of angular momentum,

$$\underline{h} = \Sigma(\delta\underline{h}) = \Sigma(\underline{R} \times \underline{v})\delta m$$

which yields

$$\underline{h} = \Sigma\underline{R} \times (\underline{v}_c + \underline{\omega} \times \underline{R})\delta m = \Sigma\underline{R} \times \underline{v}_c\delta m + \Sigma\underline{R} \times (\underline{\omega} \times \underline{R})\delta m. \qquad (A16)$$

Because $\Sigma \underline{R}\delta m = 0$ from before, the first term in Eq. A16 is eliminated.

Using the mathematical rule for a vector triple,

$$\underline{R} \times (\underline{\omega} \times \underline{R}) = \underline{\omega}(\underline{R} \cdot \underline{R}) - \underline{R}(\underline{\omega} \cdot \underline{R}) = \underline{\omega}R^2 - \underline{R}(\underline{\omega} \cdot \underline{R}),$$

Equation A16 becomes

$$\underline{h} = \Sigma \underline{\omega} R^2 - \underline{R}(\underline{\omega} \cdot \underline{R})\delta m. \tag{A17}$$

By definition,

$$\underline{R} = x\underline{i} + y\underline{j} + z\underline{k}.$$

Substituting,

$$\underline{h} = \underline{\omega}\Sigma(x^2 + y^2 + z^2)\delta m - \Sigma \underline{R}(px + qy + rz)\delta m. \tag{A18}$$

If each element is infinitesimally small, the summations become integrals and Eq. A18 becomes

$$\underline{h} = (p\underline{i} + q\underline{j} + r\underline{k}) \int (x^2 + y^2 + z^2)dm - \int (x\underline{i} + y\underline{j} + z\underline{k})(px + qy + rz)dm. \tag{A19}$$

The components of $\underline{h}$ are

$$h_x = p \int (y^2 + z^2)dm - q \int xydm - r \int xzdm$$

$$h_y = -p \int xydm + q \int (x^2 + z^2)dm - r \int yzdm$$

$$h_z = -p \int xzdm - q \int yzdm + r \int (x^2 + y^2)dm.$$

Each of the integrals in the previous equation represents the moments and products of inertia about a specified axis.

Finally, the angular momentum becomes

$$h_x = p(I_y^2 + I_z^2) - qI_{xy} - rI_{zz}$$

$$h_y = -pI_{xy} + q(I_x^2 + I_z^2) - rI_{yz} \qquad (A20)$$

$$h_z = -pI_{xz} - qI_{yz} + r(I_x^2 + I_y^2).$$

Equation $A13$ requires the time derivative of $\underline{h}$. If the reference axes are non-rotating, earth-fixed for example, the moments and products of inertia vary with the aircraft orientation requiring time derivatives of the inertia terms and the angular rates. This is avoided by fixing the reference frame to the aircraft allowing the reference axes to rotate with the aircraft. Now the moments and products of inertia are constants.

Because most aircraft are very nearly symmetric, the usual assumption of symmetry is made. By locating the reference axes at the mass center and orienting the axes as shown in Fig. 3, the X-Z plane becomes the plane of symmetry making both I_{xy} and I_{yz} zero.

Therefore, Eq. $A20$ simplifies to

$$h_x = p(I_y^2 + I_z^2) - rI_{xz}$$

$$h_y = q(I_x^2 + I_z^2) \qquad (A21)$$

$$h_z = -pI_{xz} + r(I_x^2 + I_y^2).$$

In general, the time derivative of a vector transformed into a rotating reference frame is given by

$$\frac{d\underline{A}}{dt} = \frac{\delta \underline{A}}{\delta t} + \underline{\omega} \times \underline{A} \qquad (A22)$$

with

$$\frac{\delta A}{\delta t} = \frac{dA_x}{dt}\underline{i} + \frac{dA_y}{dt}\underline{j} + \frac{dA_z}{dt}\underline{k}.$$

Applying this to Eqs. $A6$ and $A13$ yields

$$\underline{F} = m\frac{\delta \underline{v}_c}{\delta t} + m\underline{\omega} \times \underline{v}_c$$

$$\underline{G} = \frac{\delta \underline{h}}{\delta t} + \underline{\omega} \times \underline{h}$$

or

$$\underline{F} = m\left(\frac{dv_{cx}}{dt}\underline{i} + \frac{dv_{cy}}{dt}\underline{j} + \frac{dv_{cz}}{dt}\underline{k}\right) + m(p\underline{i} + q\underline{j} + r\underline{k}) \times \underline{v}_c \qquad (A23a)$$

$$\underline{G} = \frac{dh_x}{dt}\underline{i} + \frac{dh_y}{dt}\underline{j} + \frac{dh_z}{dt}\underline{k} + (p\underline{i} + q\underline{j} + r\underline{k}) \times \underline{h}. \qquad (A23b)$$

Remembering that $\underline{F}$ has components (F_x, F_y, F_z), $\underline{G}$ has components (L, M, N) and $\underline{v}_c$ has components (U, V, W). Eqs. $A23a$ and $A23b$ become

$$\begin{aligned} F_x &= m(\dot{U} + Wq - Vr) \\ F_y &= m(\dot{V} + Ur - Wp) \\ F_z &= m(\dot{W} + Vp - Uq) \end{aligned} \qquad (A24)$$

and

$$\begin{aligned} L &= \dot{h}_x + qh_z - rh_y \\ M &= \dot{h}_y + rh_x - ph_z \\ N &= \dot{h}_z + ph_y - qh_x \end{aligned} \qquad (A25)$$

and finally substituting the angular momentum yields:

$$L = \dot{p}I_x + qr(I_z - I_y) - I_{xz}(\dot{r} + pq)$$

$$M = \dot{q}I_y + pr(I_x - I_z) + I_{xz}(p^2 - r^2) \qquad (A26)$$

$$N = \dot{r}I_z + pq(I_y - I_x) + I_{xz}(qr - \dot{p}).$$

Equations $A24$ and $A26$ represent the Euler equations of motion of the aircraft.

The motion of the aircraft has now been defined; however, since a body-fixed frame of reference is used, the position and orientation of the aircraft cannot be described relative to this reference frame. For this reason, an earth-fixed reference frame (x', y', z') is introduced (Fig. 34). Let z' be vertical down and x' horizontal in the vertical plane containing the initial aircraft velocity vector.

The aircraft orientation is defined by Euler angles, a series of three consecutive rotations about each of the three earth-fixed axes. These three rotations (ψ, θ, ϕ) must be performed in the proper order to ensure the correct orientation[11]:

1. A rotation ψ about z_1, carrying the axes to (x_2, y_2, z_2) - yaw.
2. A rotation θ about y_2, carrying the axes to (x_3, y_3, z_3) - pitch.
3. A rotation ϕ about x_3, carrying the axes to their final position - roll.

These Euler angles can be determined by the angular velocities (p, q, r). Let the aircraft rotate by some infinitesimal angle in some time Δt from ψ, θ, ϕ to $\psi + \Delta\psi, \theta + \Delta\theta, \phi + \Delta\phi$. Vectorially, this rotation would be

$$\Delta\underline{n} = \Delta\phi\underline{i} + \Delta\theta\underline{j}_3 + \Delta\psi\underline{k}_2$$

with angular velocity

$$\underline{\omega} = \lim_{\Delta t \to 0} \frac{\Delta\underline{n}}{\Delta t} = \dot{\phi}\underline{i} + \dot{\theta}\underline{j}_3 + \dot{\psi}\underline{k}_2. \qquad (A27)$$

Projecting the components of $\underline{\omega}$ onto the body-fixed reference frame,

$$p = \dot{\phi} - \dot{\psi}\sin\theta$$

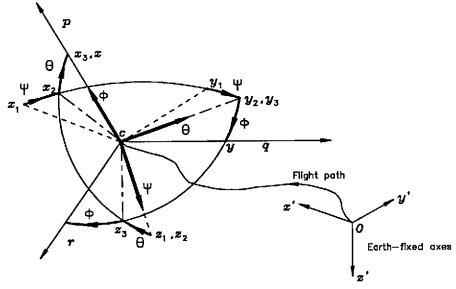

Fig. 2 Airplane Orientation

$$q = \dot{\theta}\cos\theta + \dot{\psi}\cos\theta\sin\phi$$

$$r = \dot{\psi}\cos\theta\cos\phi - \dot{\theta}\sin\phi.$$

Rearranging these equations into first order differential equations yields:

$$\dot{\phi} = p + q\sin\phi\tan\theta - r\cos\phi\tan\theta$$

$$\dot{\theta} = q\cos\phi - r\sin\phi \tag{A28}$$

$$\dot{\psi} = (q\sin\phi + r\cos\phi)\sec\theta.$$

From a tracking standpoint, the aircraft's position must also be known relative to the earth-fixed reference frame. This gives

$$\frac{dx'}{dt} = U_1 = \dot{X}_{a/c}$$

$$\frac{dy'}{dt} = V_1 = \dot{Y}_{a/c} \tag{A29}$$

$$\frac{dz'}{dt} = W_1 = \dot{Z}_{a/c}$$

where these velocities are found from the Euler rotations

$$\begin{pmatrix} U_1 \\ V_1 \\ W_1 \end{pmatrix} = \begin{pmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} U_2 \\ V_2 \\ W_2 \end{pmatrix} \tag{A30}$$

$$\begin{pmatrix} U_2 \\ V_2 \\ W_2 \end{pmatrix} = \begin{pmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{pmatrix} \begin{pmatrix} U_3 \\ V_3 \\ W_3 \end{pmatrix} \tag{A31}$$

$$\begin{pmatrix} U_3 \\ V_3 \\ W_3 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi & \cos\phi \end{pmatrix} \begin{pmatrix} U \\ V \\ W \end{pmatrix}. \tag{A32}$$

Multiplying Eqs. A30, A31 and A32 yields:

$$\begin{pmatrix} \dot{X}_{a/c} \\ \dot{Y}_{a/c} \\ \dot{Z}_{a/c} \end{pmatrix} = \begin{pmatrix} U_1 \\ V_1 \\ W_1 \end{pmatrix} = \begin{pmatrix} c\theta c\psi & (s\phi s\theta c\psi - c\phi s\psi) & (c\phi s\theta c\psi + s\phi s\psi) \\ c\theta s\psi & (s\psi s\theta s\psi + c\phi c\psi) & (c\phi s\theta s\psi - s\phi c\psi) \\ -s\theta & s\phi c\theta & c\phi c\theta \end{pmatrix} \begin{pmatrix} U \\ V \\ W \end{pmatrix} \tag{A33}$$

where c and s are cosine and sine, respectively. A simple integration of Eq. A33 with specified initial conditions defines the position of the aircraft. Note $\dot{Z}_{a/c}$ is the negative rate of climb and $Z_{a/c}$ is the altitude corrected for initial conditions.

Equations A24, A26, A28 and A33 completely define linear and angular accelerations, orientation and position of the aircraft. However, to simplify the force terms in Eq. A24, let

$$\begin{aligned} F_x &= F_{x_{aero}} - mg\sin\theta \\ F_y &= F_{y_{aero}} + mg\sin\phi\cos\theta \\ F_z &= F_{z_{aero}} + mg\cos\phi\cos\theta \end{aligned} \tag{A34}$$

from geometry which eliminates the gravity effects from the forces.

APPENDIX B

HTTB SIMULATION SOURCE CODE

State Definitions

State	Definition	Units	State	Definition	Units
1	U	fps	31	DLC Shape	-
2	V	fps	32	DLC Shape	-
3	W	fps	33	DLC Shape	-
4	Phi	rads	34	PSCAS Int	-
5	Theta	rads	35	Tri-State	deg/s
6	Psi	rads	36	SPEED Int	-
7	p	rad/s	37	dELVCMD/dt	deg/s
8	q	rad/s	38	Alt Error	feet
9	r	rad/s	39	TE int	-
10	Alpha	deg	40	W dot	ft/s/s
11	Hdot	fps	41		
12	H	feet	42		
13	Xa/c	feet	43		
14	Ya/c	feet	44	-TES-	-
15	Za/c	feet	45	-TES-	-
16	Elevator	deg	46	TES Output	kts
17	Throttle	%	47		
18	Aileron	deg	48		
19	Rudder	deg	49		
20	Left Splr	deg	50		
21	Right Splr	deg	51		
22	Gear	%	52		
23	Flap	-	53		
24	Wind	fps	54		
25	Gamma	deg	55		
26	TAS	kts	56		
27	TAS dot	fps/s	57		
28	-THTL-	-	58		
29	-THTL-	-	59		
30	DLC Wash	-	60		

```
      IMPLICIT REAL*8(A-H,O-Z)
      COMMON/MODEL/IMODEL
      COMMON/AERODAT/ADATA(700,43)
```

```
      CALL CLEAR
      WRITE(*,*)
      WRITE(*,*)
      WRITE(*,*)
      WRITE(*,*)
      WRITE(*,*)
      WRITE(*,*)
      WRITE(*,*)
      WRITE(*,*)
      WRITE(*,*)
      WRITE(*,*)
      WRITE(*,*)'        loading database ...'
      CALL LOADEM
1     CALL CLEAR
      WRITE(*,*)
      WRITE(*,*)
      WRITE(*,*)
      WRITE(*,*)
      WRITE(*,*)
      WRITE(*,*)
      WRITE(*,*)
      WRITE(*,*)
      WRITE(*,*)
      WRITE(*,*)'            HTTB Simulation'
      WRITE(*,*)
      WRITE(*,*)'      ENTER:    1  for TRIM'
      WRITE(*,*)'                2  for TIME HISTORY'
      WRITE(*,*)'                3  for QUIT'
      READ(*,*,ERR=1)ICHOICE
      IF((ICHOICE.GT.3).OR.(ICHOICE.LT.1))GOTO 1
      IF(ICHOICE.EQ.3)GOTO 4
      IMODEL=1
      IF(ICHOICE.EQ.1)CALL TRIM
      IF(ICHOICE.EQ.2)CALL TIMEH
      GOTO 1
4     END

C-------------------------------------------------------------C
C       Routines LDD(T,Z,X,XD) and LD(T,X,XD) to be           C
C                  supplied by user                           C
C            Paul Vesty, Dept 72-42, 1983                     C
C         Modified 1986, Tom Anderson, TAMU                   C
C-------------------------------------------------------------C

      SUBROUTINE TIMEH
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION IX(16),IU(16),Y(0:1000,0:16),X(256),XD(256),
     &          YY(0:1000,0:16)
      COMMON/MODEL/IMODEL
      COMMON/SHEAR/U20,ISHEAR,GRAD,SHRALT,SHREND,ZT,WIND
      COMMON/AERODAT/ADATA(700,43)
```

```
      COMMON/U     /U(32)
      COMMON/AUTHOR/QBAR,VT
      COMMON/LDDID/ID
      CHARACTER NAME*20,NAME2*20

1     TO=0.0D0
      CALL CLEAR
      WRITE(*,*)
      WRITE(*,*)
      WRITE(*,*)'             Time History'
      WRITE(*,*)
      WRITE(*,*)'  ENTER:  0 for rectangular integration.'
      WRITE(*,*)'          1 for fourth-order Runge-Kutta.'
      WRITE(*,*)'          2 for variable-step '
      WRITE(*,*)'           Runge-Kutta-Fehlberg.'
      WRITE(*,*)
      READ(*,*) IR
      IF((IR.LT.0).OR.(IR.GT.2)) GOTO 1
      E=1.0D-5
      IF(IR.EQ.2) WRITE(*,*) 'Accuracy ? (default is 1E-5)'
      IF(IR.EQ.2) READ(*,*) E
2     IF(IR.NE.2) WRITE(*,*) 'dt ?'
      IF(IR.EQ.2) WRITE(*,*) 'Maximum dt ?'
      READ(*,*) ZT
      WRITE(*,*) 'Plotting interval ? (in dt''s)'
      READ(*,*) IL
      WRITE(*,*) 'Print interval ? (in plotting intervals)'
      READ(*,*) IP
      WRITE(*,*) 'How many print intervals ?'
      READ(*,*) NP
      WRITE(*,*) 'Run time = ',IL*IP*NP*SNGL(ZT),'     OK ?'
      CALL NO(&2)

99    ICOUNT=0
      CALL RXU(NX,NU,X)

      DO 3 I=1,NX
3     XD(I)=0.0D0
      WRITE(*,*) 'How many states to be plotted ?'
      READ(*,*) MX
      IF(MX.NE.0) THEN
        WRITE(*,*) 'Which ones ?'
        READ(*,*) (IX(I),I=1,MX)
      ENDIF
      WRITE(*,*)'How many supplemental states?'
      READ(*,*)MN
      IF(MN.NE.0)THEN
        WRITE(*,*)'Which ones ?'
        READ(*,*) (IU(I),I=1,MN)
```

```
        WRITE(*,*)'Input supplemental file name'
        READ(*,'(A)')NAME2
      ENDIF
      ID=0
      ISHEAR=0
      U20=0
      GRAD=0
      SHRALT=0
73    WRITE(*,*) 'Input WIND SHEAR profile: 0 - None'
      WRITE(*,*) '                          1 - MIL-F-8785C'
      WRITE(*,*) '                          2 - Gradient'
      READ(*,*) ISHEAR
      IF((ISHEAR.LT.0).OR.(ISHEAR.GT.2))GOTO 73
      IF(ISHEAR.NE.0)THEN
            WRITE(*,*)'Shear starting altitude '
            READ(*,*)SHRALT
            WRITE(*,75)SHRALT,X(24)*36/60
75          FORMAT(' Wind speed at 'F5.0' feet = 'F4.0
     &            ' kts')
            WIND=X(24)
            WRITE(*,*)'     HEADWINDS -       TAILWINDS +'
            IF(ISHEAR.EQ.2)THEN
                  WRITE(*,*)'Shear ending altitude '
                  READ(*,*)SHREND
            ENDIF
      ENDIF
      IF(ISHEAR.EQ.1)THEN
            U20=X(24)*LOG(20/.15)/LOG(SHRALT/.15)
            WRITE(*,76)U20*36/60
76          FORMAT(' Wind speed at 20 feet = 'F4.0' kts')
      ENDIF
      IF(ISHEAR.EQ.2)THEN
            WRITE(*,*) 'Wind Gradient'
            READ(*,*)GRAD
            WRITE(*,78)(X(24)+GRAD*(SHRALT-SHREND))*36/60
78          FORMAT(' Ending wind speed = 'F4.0' kts')
      ENDIF
      WRITE(*,*)
      WRITE(*,*) 'Include discrete time function ? (NO)'
      CALL NO(&4)
      ID=1
      CALL LDD(T0,ZT,X,XD,ICOUNT)
4     CALL LD(T0,X,XD)
      ICOUNT=1

      CALL CLEAR
5     FORMAT('   Time',16I12)
      WRITE(*,5) (IX(I),I=1,MX)
      Y(0,0)=T0
```

```
      IF(MX.NE.0) THEN
        DO 6 I=1,MX
6       Y(0,I)=X(IX(I))
      ENDIF
      IF(MN.NE.0)THEN
        OPEN(UNIT=13,FILE=NAME2,STATUS='NEW')
        WRITE(13,*)MN+1
        WRITE(13,7)T0,(X(IU(I)),I=1,MN)
      ENDIF
7     FORMAT(1P17E12.3)
      WRITE(*,7) (Y(0,I),I=0,MX)

      K=0
      M=0
      KFLAG=0
      DO 11 N=1,NP
      DO 10 J=1,IP
      K=K+1
      DO 8 L=1,IL
      T=T0+DFLOAT(M)*ZT
      IF(ID.EQ.1) CALL LDD(T,ZT,X,XD,ICOUNT)
      IF(IR.EQ.0) CALL RK0(T,ZT,X,XD,NX)
      IF(IR.EQ.1) CALL RK4(T,ZT,X,XD,NX)
      IF(IR.EQ.2) CALL RKF(T,ZT,X,XD,NX,E,M)
      IF(X(12).LT.0.0)X(12)=0.0
      IF((X(12).EQ.0).AND.(X(11).LE.0))THEN
       NNN=N
       KFLAG=1
       NNJ=J
       GOTO 77
      ENDIF
8     M=M+1
      Y(K,0)=T
      IF(X(26).LT.170.)THEN
              FVTKT = (50.-X(26))*.1/120.+1.0
              CALL LIMITS(FVTKT,1.D0,.9D0)
      ELSE
              FVTKT = (170.-X(26))*.3/80.+0.9
              CALL LIMITS(FVTKT,.9D0,.6D0)
      ENDIF
      CALL LIMITS(X(16),15.D0*FVTKT,-40.D0*FVTKT)
      CALL LIMITS(X(17),1.D0,0.D0)
      CALL LIMITS(X(18),40.D0*FVTKT,-40.D0*FVTKT)
      CALL LIMITS(X(19),30.8D0,-40.D0)
      CALL LIMITS(X(20),0.D0,-40.D0)
      CALL LIMITS(X(21),0.D0,-40.D0)
      CALL LIMITS(X(22),1.D0,0.D0)
      CALL LIMITS(X(23),54.D0,0.D0)
      IF(MX.NE.0) THEN
```

```
            DO 9 I=1,MX
   9        Y(K,I)=X(IX(I))
          ENDIF
          IF(MN.NE.0) WRITE(13,7)T,(X(IU(I)),I=1,MN)
  10      CONTINUE
  11      WRITE(*,7) (Y(K,I),I=0,MX)

  77      WRITE(*,*)
            WRITE(*,*) 'Save in file ? (default is NO)'
            CALL NO(&13)
            WRITE(*,*) 'File name ?'
            READ(*,'(A)') NAME
            OPEN(UNIT=20,FILE=NAME,STATUS='NEW',RECL=204)
            WRITE(20,*)MX+1
            MZ=IP*NP
            IF(KFLAG.NE.0) MZ=IP*(NNN-1)+NNJ-1
            DO 12 I=0,MZ
  12        WRITE(20,7) (Y(I,J),J=0,MX)
            CLOSE(UNIT=20)
  13      WRITE(*,*) 'Another run ? (default is NO)'
          CALL YES(&99)
          RETURN
          END

C-------------------------------------------------------------C
C               Numerical Integration Algorithms              C
C                  Rectangular Integration                    C
C-------------------------------------------------------------C

          SUBROUTINE RK0(TT,DT,XX,XD,NX)
          IMPLICIT REAL*8 (A-H,O-Z)
          REAL*8 XX(*),XD(*)
          COMMON/AERODAT/ADATA(700,43)
          COMMON/LDDID/ID
          IF(ID.EQ.1)CALL LDD(TT,DT,XX,XD,1)
          IF(IE.EQ.0)CALL LD(TT,XX,XD)
          DO 1 M=1,NX
   1      XX(M)=XX(M)+XD(M)*DT
          TT=TT+DT
          RETURN
          END



C-------------------------------------------------------------C
C       Runga-Kutta 4th Order Fixed Step Integration          C
C-------------------------------------------------------------C
```

```
      SUBROUTINE RK4(TT,DT,XX,XD,NX)
      IMPLICIT REAL*8 (A-H,O-Z)
      REAL*8 XX(*),XD(*)
      COMMON/AERODAT/ADATA(700,43)
      COMMON/LDDID/ID
      DIMENSION X(256),XA(256)
      IF(ID.EQ.1)CALL LDD(TT,DT,XX,XD,1)
      IF(ID.EQ.0)CALL LD(TT,XX,XD)
      DO 1 M=1,NX
      XA(M)=XD(M)*DT
    1 X(M)=XX(M)+0.5D0*XA(M)
      T=TT+0.5D0*DT
      IF(ID.EQ.1)CALL LDD(T,DT,X,XD,1)
      IF(ID.EQ.0)CALL LD(T,X,XD)
      DO 2 M=1,NX
      Q=XD(M)*DT
      X(M)=XX(M)+0.5D0*Q
    2 XA(M)=XA(M)+Q+Q
      IF(ID.EQ.1)CALL LDD(T,DT,X,XD,1)
      IF(ID.EQ.0)CALL LD(T,X,XD)
      DO 3 M=1,NX
      Q=XD(M)*DT
      X(M)=XX(M)+Q
    3 XA(M)=XA(M)+Q+Q
      TT=TT+DT
      IF(ID.EQ.1)CALL LDD(TT,DT,X,XD,1)
      IF(ID.EQ.0)CALL LD(TT,X,XD)
      DO 4 M=1,NX
    4 XX(M)=XX(M)+(XA(M)+XD(M)*DT)/6.0D0
      RETURN
      END

C-------------------------------------------------------------C
C          Runga-Kutta-Felberg Variable Step Integration      C
C-------------------------------------------------------------C

      SUBROUTINE RKF(TT,ZT,XX,XD,NX,E,INIT)
      IMPLICIT REAL*8 (A-H,O-Z)
      REAL*8 XX(*),XD(*)
      COMMON/AERODAT/ADATA(700,43)
      COMMON/LDDID/ID
      DIMENSION X(256),XA(256),XB(256),
     &                 XC(256),XE(256)
      TF=TT+ZT
      IF(INIT.EQ.0) DT=ZT
      K1=3
    1 DD=TF-TT
      IF(1.5D0*DT.GT.DD) DT=DD
      HD=0.5D0*DT
```

```
      IF((K1.GT.1).AND.(ID.EQ.1))CALL LDD(TT,ZT,XX,XA,1)
      IF(K1.GT.1)CALL LD(TT,XX,XA)
      Q=0.25D0*DT
      DO 2 M=1,NX
2     X(M)=XX(M)+XA(M)*Q
      T=TT+Q
      IF(ID.EQ.1)CALL LDD(TT,DT,X,XB,1)
      CALL LD(TT,X,XB)
      DO 3 M=1,NX
3     X(M)=XX(M)+(3.0D0*XA(M)+9.0D0*XB(M))*DT/32.0D0
      T=TT+0.375D0*DT
      IF(ID.EQ.1)CALL LDD(TT,DT,X,XC,1)
      CALL LD(TT,X,XC)
      DO 4 M=1,NX
4     X(M)=XX(M)+(1932.0D0*XA(M)+7296.0D0*XC(M)-7200.0D0*
     &      XB(M))*DT/2197.0D0
      T=TT+12.0D0*DT/13.0D0
      IF(ID.EQ.1)CALL LDD(TT,DT,X,XE,1)
      CALL LD(TT,X,XE)
      DO 5 M=1,NX
5     X(M)=XX(M)+(8341.0D0*XA(M)-845.0D0*XE(M)-32832.0D0*
     &      XB(M))+29440.0D0*XC(M))*DT/4104.0D0
      T=TT+DT
      IF(ID.EQ.1)CALL LDD(TT,DT,X,XD,1)
      CALL LD(TT,X,XD)
      Q=DT/20520.0D0
      DO 6 M=1,NX
6     X(M)=XX(M)+(-6080.0D0*XA(M)-5643.0D0*XD(M)+41040.0D0*
     &      XB(M)-28352.0D0*XC(M)+9295.0D0*XE(M))*Q
      T=TT+HD
      IF(ID.EQ.1)CALL LDD(TT,DT,X,XB,1)
      CALL LD(TT,X,XB)
      DO 7 M=1,NX
      X(M)=(902880.0D0*XA(M)+ 277020.0D0*XB(M)+3953664.0D0*
     &      XC(M)+3855735.0D0*XE(M)-1371249.0D0*XD(M))*DT/
     &      7618050.0D0
      E1=DABS((2375.0D0*XA(M)+ 11264.0D0*XC(M)+10985.0D0*
     &      XE(M)-4104.0D0*XD(M))*Q-X(M))
      AE=DABS(E)
      IF(E.GT.0.0D0.AND.XX(M).NE.0.0D0)
     &  AE=AE*MAX(DABS(X(M)),DABS(XX(M)))
      IF(                    E1.GT.AE) K1=0
      IF(K1.GT.1.AND. 5.0D0*E1.GT.AE) K1=1
      IF(K1.GT.2.AND.20.0D0*E1.GT.AE) K1=2
7     CONTINUE
      IF(K1.EQ.0) GOTO 9
      TT=TT+DT
      DO 8 M=1,NX
8     XX(M)=XX(M)+X(M)
```

```
      IF(DT.EQ.DD) RETURN
 9    DT=DT*(0.5D0+0.4D0*K1)
      K1=MIN(K1+1,3)
      GOTO 1
      END



C-----------------------------------------------C
C              General Input Algorithm          C
C-----------------------------------------------C

      SUBROUTINE RXU(NX,NU,X)
      IMPLICIT REAL*8 (A-H,O-Z)
      REAL*8 X(*)
      COMMON/U     /U(32)
      CHARACTER F*12
10    WRITE(*,*) 'Read states and inputs from file ? (NO)'
      CALL NO(&1)
      WRITE(*,*) 'File name ?'
      READ(*,'(A)',ERR=10) F
      OPEN(UNIT=20,FILE=F,STATUS='OLD')
      READ(20,*) NX,NU
      READ(20,*) (X(I),I=1,NX)
      IF(NU.NE.0) READ(20,*) (U(I),I=1,NU)
      CLOSE(UNIT=20)
      RETURN
 1    WRITE(*,*) 'How many states ?'
      READ(*,*) NX
      DO 2 I=1,NX
 2    X(I)=0.0D0
      WRITE(*,*) 'Enter values for states (ZERO) :'
      READ(*,*) (X(I),I=1,NX)
      WRITE(*,*) 'How many inputs ?'
      READ(*,*) NU
      IF(NU.EQ.0) RETURN
      DO 3 I=1,NU
 3    U(I)=0.0D0
      WRITE(*,*) 'Enter values for inputs (ZERO) :'
      READ(*,*) (U(I),I=1,NU)
      RETURN
      END

C-------------------------------------------------------------C
C                General aircraft trim routine                C
C         Brian Stevens & Robert Marshall, Dept 72-11,        C
C               & Paul Vesty, Dept 72-42, 1985                C
C-------------------------------------------------------------C
```

```
      SUBROUTINE TRIM
      IMPLICIT REAL*8 (A-H,O-Z)
      REAL*8 NEWGAM
      DIMENSION S(6),CMIN(6),CMAX(6),WORK(87),IWORK(6)
      CHARACTER F*12
      COMMON/WTRIM/X(60),VUP,FPS,SPHI,CPHI,PSID,NEWGAM
      COMMON/U     /ELEV,THTL,AIL,RDR,SPLL,SPLR,GEAR,FLAP,
     &       EO,ETAB
      COMMON/AERODAT/ADATA(700,43)
      EXTERNAL TRIMPF
      DATA G /32.17D0/

C*************************************************************C
C            Set Boundaries for Trim Variables              C
C*************************************************************C

      CMIN(1)=-40.D0                          ! Elevator (deg)
      CMAX(1)=15.D0
      CMIN(2)=0.D0                            ! Throttle (%)
      CMAX(2)=1.D0
      CMIN(3)=-100.D0                         ! W (fps)
      CMAX(3)=100.D0
      CMIN(4)=-40.D0                          ! Aileron (deg)
      CMAX(4)=40.D0
      CMIN(5)=-40.D0                          ! Rudder (deg)
      CMAX(5)=30.8D0
      CMIN(6)=-25.D0                          ! V (fps)
      CMAX(6)=25.D0
      S(1)=0.D0
      S(2)=0.D0
      S(3)=0.D0
      S(4)=0.D0
      S(5)=0.D0
      S(6)=0.D0

      PI=4.0D0*DATAN(1.0D0)
1     CALL CLEAR
      WRITE(*,*)
      WRITE(*,*) 'Altitude ?'
      READ(*,*) H
      WRITE(*,*) 'Climb angle (degrees) ?'
      READ(*,*) GAMMA
2     WRITE(*,*)
      WRITE(*,*) 'Speed (ft/s) ?'
      READ(*,*) FPS
      IF(FPS.LE.0.0D0) GOTO 2
      WRITE(*,*)'Wind velocity (kts) ?  ->  HEADWINDS (-)'
     &   'TAILWINDS (+)'
```

```
      READ(*,*)WIND
      X(24)=WIND*60/36
      DELGAM=DASIND((X(24)/FPS)*DSIND(GAMMA))
      NEWGAM=(GAMMA+DELGAM)*PI/180.D0
      VUP=FPS*DSIN(GAMMA*PI/180)
      WRITE(*,*) 'Turn rate (degrees/minute) ?'
      READ(*,*) DPM

C*********************************************************C
C            Calculate Phi from Turn Rate                 C
C*********************************************************C
      PSID=DPM*PI/10800.0D0
      A=PSID*FPS*DCOS(GAMMA*PI/180)
      SPHI=A/DSQRT(G*G+A*A)
      CPHI=G/DSQRT(G*G+A*A)
      PHI=DATAN(A/G)
      WRITE(*,3) PHI*180.0D0/PI
    3 FORMAT(' Bank angle is',1PE12.3,' degrees.   OK ?')
      CALL NO(&2)

      WRITE(*,*) 'Engines out:  0 - none out'
      WRITE(*,*) '              1 - left outboard out'
      WRITE(*,*) '              4 - right outboard out'
      WRITE(*,*) '    (NO data for inboard engines out)'
      READ(*,*) EO
      WRITE(*,*) 'Gear position:  0 - up'
      WRITE(*,*) '                1 - down'
      READ(*,*) X(22)
      WRITE(*,*) 'Flap position (fore/aft):  0 -  0/0  flap'
      WRITE(*,*) '                           1 - 18/0  flap'
      WRITE(*,*) '                           2 - 36/0  flap'
      WRITE(*,*) '                           3 - 36/30 flap'
      READ(*,*) IFLAP
      IF(IFLAP.EQ.0)X(23)=0.
      IF(IFLAP.EQ.1)X(23)=18.
      IF(IFLAP.EQ.2)X(23)=36.
      IF(IFLAP.EQ.3)X(23)=54.
      WRITE(*,*)'Spoiler uprig for DLC (degrees) ?'
      READ(*,*) X(20)
      X(21)=X(20)
4     WRITE(*,*)
      WRITE(*,*)'Enter number of starting points:'
      READ(*,*)NPNTS

C*********************************************************C
C          Initialize Non-Varying States                  C
C*********************************************************C

      X( 4)= PHI
```

```
      X( 6)= 0.0D0
      X(10)= 0.0D0
      X(12)= H

      N=6
      IF(DPM.EQ.0.0D0) THEN
            WRITE(*,*)
            WRITE(*,*) 'Choose only longitudinal trim ? (NO)'
            CALL NO(&5)
            N=3
      ENDIF
 5    WRITE(*,*)
      WRITE(*,*)
      WRITE(*,*)'    Busy.......'
C*************************************************************C
C              Call IMSL Minimization Routine                 C
C*************************************************************C

      CALL ZXMWD(TRIMPF,N,12,CMIN,CMAX,NPNTS,S,F0,WORK,
     &                  IWORK,IER)
      CALL CLEAR
      IF(IER.EQ.129)
     &      WRITE(*,*)'May have converged to saddlepoint.'
      IF(IER.EQ.130)WRITE(*,*)'Cannot converge to NSIG.'
      IF(IER.EQ.131)WRITE(*,*)'Number of iterations exceeded '
      IF(IER.EQ.132)WRITE(*,*)' A(i) > B(i).'
      WRITE(*,6) F0
 6    FORMAT(' Trimmed  performance  function  is  ',1PE12.3)
      WRITE(*,*)
      WRITE(*,8) S(1)
 8    FORMAT(' Trimmed elevator is',1PE12.3,' degrees.')
      WRITE(*,9) S(2)
 9    FORMAT(' Trimmed throttle is',1PE12.3)
      WRITE(*,10) S(4)
 10   FORMAT(' Trimmed aileron  is',1PE12.3,' degrees.')
      WRITE(*,11) S(5)
 11   FORMAT(' Trimmed  rudder  is',1PE12.3,' degrees.')
      WRITE(*,*)
      WRITE(*,16)X(24)*36/60
 16   FORMAT(' Wind at altitude is 'F5.0' kts')
      THETA=X(5)*180.0D0/PI
      WRITE(*,12) THETA
 12   FORMAT(' Theta is',1PE12.3,' degrees.')
      WRITE(*,13) DASIN(X(2)/FPS*PI/180)
 13   FORMAT(' Beta is ',1PE12.3,' degrees.')
      WRITE(*,17) X(11)
 17   FORMAT(' Hdot = ',F7.3,' fps')
      WRITE(*,*)
      WRITE(*,*) 'Save in file ?'
```

```
      CALL NO(&15)
      WRITE(*,*) 'File name ?'
      READ(*,'(A)') F
      OPEN(UNIT=20,FILE=F,STATUS='NEW')
      WRITE(20,*) 60,9
      WRITE(20,14) (X(I),I=1,60)
      WRITE(20,14) S(1),S(2),S(4),S(5),SPLL,SPLR,GEAR,FLAP,EO
14    FORMAT(1PE15.6)
      CLOSE(UNIT=20)
15    WRITE(*,*) 'Another go ?'
      CALL YES(&1)
      RETURN
      END

C-------------------------------------------------------C
C              Trim Performance (Cost) Function         C
C-------------------------------------------------------C

      SUBROUTINE TRIMPF(N,S,FO)
      IMPLICIT REAL*8 (A-H,O-Z)
      REAL*8 S(*),NEWGAM
      DIMENSION XD(60)
      COMMON/AERODAT/ADATA(700,43)
      COMMON/U  /ELEV,THTL,AIL,RDR,SPLL,SPLR,GEAR,FLAP,EO,ETAB
      COMMON/WTRIM/X(60),VUP,FPS,SPHI,CPHI,PSID,NEWGAM

C*******************************************************C
C              Set Trim Variable to Control Input       C
C*******************************************************C

      X(16) = S(1)
      X(17) = S(2)
      X(18) = S(4)
      X(19) = S(5)

C*****************************************************C
C              Set Trim Variable to States          C
C*****************************************************C

      X(3) = S(3)
      X(2) = S(6)
      X(1) = DSQRT(FPS*FPS-X(2)*X(2)-X(3)*X(3))

C*****************************************************C
C               Calculate Pitch Attitude            C
C*****************************************************C

      X(5)=DATAN(X(3)/X(1))+NEWGAM
```

```
      STHETA=DSIN(X(5))
      CTHETA=DCOS(X(5))

      X(7)=-PSID*STHETA
      X(8)= PSID*SPHI*CTHETA
      X(9)= PSID*CPHI*CTHETA

C*************************************C
C              Call Model             C
C*************************************C

      CALL LD(0.0D0,X,XD)

C*********************************************C
C              Find Cost Function             C
C*********************************************C

      D  =       XD(1)**2+XD(2)**2+XD(3)**2
      FO = D+10*(XD(7)**2+XD(8)**2+XD(9)**2)
      RETURN
      END


C-------------------------------------------------C
C              General Prompt Algorithms          C
C-------------------------------------------------C

      SUBROUTINE NO(*)
      CHARACTER A*3
      READ(*,'(A)') A
      IF((A.NE.'1'  ).AND.
     &   (A.NE.'Y'  ).AND.
     &   (A.NE.'YES').AND.
     &   (A.NE.'Yes').AND.
     &   (A.NE.'y'  ).AND.
     &   (A.NE.'yes'))   RETURN 1
      END

      SUBROUTINE YES(*)
      CHARACTER A*3
      READ(*,'(A)') A
      IF((A.EQ.'1'  ).OR.
     &   (A.EQ.'Y'  ).OR.
     &   (A.EQ.'YES').OR.
     &   (A.EQ.'Yes').OR.
     &   (A.EQ.'y'  ).OR.
     &   (A.EQ.'yes'))   RETURN 1
      END
```

```
      SUBROUTINE NNO(*)
      CHARACTER A*3
      READ(*,'(A)')A
      IF((A.NE.'0').AND.
     &     (A.NE.'N').AND.
     &     (A.NE.'n').AND.
     &     (A.NE.'NO').AND.
     &     (A.NE.'no').AND.
     &     (A.NE.'No'))    RETURN 1
      END

      SUBROUTINE NNOT(*)
      CHARACTER A*3
      READ(*,'(A)')A
      IF((A.EQ.'0').OR.
     &     (A.EQ.'N').OR.
     &     (A.EQ.'n').OR.
     &     (A.EQ.'NO').OR.
     &     (A.EQ.'no').OR.
     &     (A.EQ.'No'))   RETURN 1
      END

C-----------------------------------------C
C                Control Limits           C
C-----------------------------------------C

      SUBROUTINE LIMITS(X,A,B)
      IMPLICIT REAL*8(A-H,O-Z)
      IF(X.GT.A)X=A
      IF(X.LT.B)X=B
      RETURN
      END


C=========================================================C
C                                                         C
C                      Table Lookups                      C
C                                                         C
C=========================================================C
C
      SUBROUTINE TABLES(X,VT,QBAR,PDIA,XGE,CALF,SALF,PSTAB,
     &                  RSTAB,QS,B,CBAR)
      IMPLICIT REAL*8(A-H,O-Z)
      REAL*8 X(*)
      COMMON/AERODAT/ADATA(700,43)
      COMMON/DLONG/CMWBT,DCMGE,DCMSPL,DCMSPR,DEPDAL,DCMEL,
     &     CMQ,DCMEO
      COMMON/DERIV/THRUST,CXS,CYS,CZS,CNS,CRS
```

```
      COMMON/U    /ELEV,THTL,AIL,RUD,SPLL,SPLR,GEAR,FLAP,EO,ETAB
      DATA CGPCNT,FSVT,FSREF,ICOUNT/30.D0,1086.01D0,528.51D0,0/
      PI=4.D0*DATAN(1.0D0)
      ALPHA=180.D0*DATAN(X(3)/X(1))/PI
      BETA=180.D0*DASIN(X(2)/VT)/PI
      VTKT=VT*36.D0/60.D0
      X(10)=ALPHA
      X(25)=X(5)*180.D0/PI-ALPHA
      X(26)=VTKT
      CALL DERIV1(X,VTKT,QBAR,PDIA,ALPHA,BETA,TC,TPENG,NTC,TC1,
     &           TC2,NFLAP,FLAP1,FLAP2,DEPDAL,ALFK,NALF,ALF1,ALF2
     &           ,BETAK,NBET,BT1,BT2,CLWBT,CMWBT,CDTOT,CYB0,
     &           CNB0,CRB0,DCYB,DCNB,DCRB,DCDB)
      CALL DERIV2(X,NFLAP,FLAP1,FLAP2,ALPHA,TC,BETAK,NBET,BT1,
     &           BT2,DCLGE,DCDGE,DCMGE,CMQ,CNP,CNR,CRR,CRP,DCLEL,
     &           DCMEL,DCDEL,RUDFAC,TCK,NTC,TC1,TC2)
      CALL DERIV3(X,ALPHA,NFLAP,FLAP1,FLAP2,TCK,NTC,TC1,TC2,
     &           DCRA,DCNA,DCLSPL,DCLSPR,DCMSPL,DCMSPR,DCDSPL
     &           ,DCDSPR,DCYSPL,DCYSPR,DCNSPL,DCNSPR,DCRSPL,
     &              DCRSPR)
C
C               DCLEO(FLAP,EO,TC,ALPHA)
C
      IF((X(23).EQ.0.).OR.(EO.EQ.0.))THEN
               DCLEO=0.
               DCMEO=0.
               DCYEO=0.
               DCNEO=0.
               DCREO=0.
               GOTO 20
      ENDIF
      ALFK=ALPHA
      CALL LIMITS(ALFK,18.D0,-8.D0)
      CALL COARSE(77,X(23),EO,2,2,NFLAP,NEO,0,2,FLAP1,
     &       FLAP2,EO1,EO2)
      CALL COARSE(77,TCK,ALFK,6,10,NTC,NALF,4,10,TC1,TC2,
     &       ALF1,ALF2)
      CALL FINE4(77,X(23),EO,TCK,ALFK,NFLAP,NEO,NTC,NALF,
     &       2,2,6,10,FLAP1,FLAP2,EO1,EO2,TC1,TC2,ALF1,ALF2,
     &       DCLEO)
C
C               DCMEO(FLAP,EO,TC,ALPHA)
C
      CALL FINE4(78,X(23),EO,TCK,ALFK,NFLAP,NEO,NTC,NALF,2,
     &       2,6,10,FLAP1,FLAP2,EO1,EO2,TC1,TC2,ALF1,ALF2,DCMEO)
C
C               DCYEO(FLAP,EO,TC,ALPHA)
C
      CALL FINE4(79,X(23),EO,TCK,ALFK,NFLAP,NEO,NTC,NALF,2,
```

```
     &       2,6,10,FLAP1,FLAP2,EO1,EO2,TC1,TC2,ALF1,ALF2,DCYEO)
C
C               DCNEO(FLAP,EO,TC,ALPHA)
C
      CALL FINE4(80,X(23),EO,TCK,ALFK,NFLAP,NEO,NTC,NALF,2,
     &       2,6,10,FLAP1,FLAP2,EO1,EO2,TC1,TC2,ALF1,ALF2,DCNEO)
C
C               DCREO(FLAP,EO,TC,ALPHA)
C
      CALL FINE4(81,X(23),EO,TCK,ALFK,NFLAP,NEO,NTC,NALF,2,
     &       2,6,10,FLAP1,FLAP2,EO1,EO2,TC1,TC2,ALF1,ALF2,DCREO)
C
C               DCNRDR(RDR)
C

20    RDRD1=X(19)
      CALL LIMITS(RDRD1,35.D0,-35.D0)
      CALL COARSE(68,RDRD1,99999.D0,9,0,NRDR,0,0,0,RDR1,
     &       RDR2,0.D0,0.D0)
      DCNRDR=ADATA(9+NRDR,28)+(ADATA(10+NRDR,28)-ADATA
     &       (9+NRDR,28))*(RDRD1-RDR1)/(RDR2-RDR1)
C
C               DCDRUD(RDR)
C
      CALL COARSE(52,X(19),99999.D0,10,0,NRDR,0,0,0,RDR1,
     &       RDR2,0.D0,0.D0)
      DCDRUD=ADATA(10+NRDR,12)+(ADATA(11+NRDR,12)-ADATA
     &       (10+NRDR,12))*(X(19)-RDR1)/(RDR2-RDR1)
C
C               DCDGR
C
      DCDGR=0.0175
C
C-------------------------------------------------------------C
C                                                             C
C               Calculate Stability Derivatives               C
C                                                             C
C-------------------------------------------------------------C
C
      THRUST=TPENG*(4-EO)
        CXS =-GDTOT-DCDGE*XGE/(1+0.0024*X(12)*X(12))-DCDEL
     &       -DCDSPL-DCDSPR-DCDGR*X(22)-DCDRUD-DCDB+THRUST*
     &       CALF/QS
        CYS = CYB0+DCYB+DCYSPL-DCYSPR-(DCNRDR*RUDFAC*12.*B)
     &       /(FSVT-FSREF)+DCYEO
        CZS =-CLWBT-DCLGE*XGE/(1+0.0024*X(12)*X(12))-DCLEL-
     &       DCLSPL-DCLSPR-DCLEO-DCLETAB-THRUST*SALF/QS
        CNS = CNB0+DCNB+DCNRDR*RUDFAC+DCNA+DCNSPL-DCNSPR+B*
     &         (CNP*PSTAB+CNR*RSTAB)/(2.*VT)+DCNEO+CYS*CBAR*
```

```
     &          (CGPCNT-25)/(100.*B)
         CRS = CRB0+DCRB-DCNRDR*RUDFAC*12.*B*(0.1136*CALF-
     &       0.3523*SALF)/(FSVT-FSREF)+DCRA+DCRSPL-DCRSPR+
     &       B*(CRP*PSTAB+CRR*RSTAB)/(2.*VT)+DCREO
        CYS=0.D0
        CNS=0.D0
        CRS=0.D0
        RETURN
        END



        SUBROUTINE DERIV1(X,VTKT,QBAR,PDIA,ALPHA,BETA,TC,
     &       TPENG,NTC,TC1,TC2,NFLAP,FLAP1,FLAP2,DEPDAL,
     &       ALFK,NALF,ALF1,ALF2,BETAK,NBET,BT1,BT2,CLWBT,
     &       CMWBT,CDTOT,CYB0,CNB0,CRB0,DCYB,DCNB,DCRB,DCDB)
        IMPLICIT REAL*8(A-H,O-Z)
        REAL*8 X(*),ADATA(700,43)
        COMMON/AERODAT/ADATA
C
C               TPENG(VTKT,THTL)
C
        CALL COARSE(41,X(17),VTKT,2,9,NTHL,NV,0,2,TL1,TL2,
     &       V1,V2)
        CALL FINE2(41,X(17),VTKT,NTHL,NV,2,9,TL1,TL2,V1,V2,
     &       TPENG)
        TC=TPENG/(2*QBAR*PDIA*PDIA)
C
C               CLWBT(FLAP,TC,ALPHA)
C
        ALFK=ALPHA
        TCK=TC
        CALL LIMITS(ALFK,18.D0,-8.D0)
        CALL LIMITS(TCK,2.D0,0.D0)
        CALL COARSE(43,X(23),TCK,4,6,NFLAP,NTC,0,4,FLAP1,FLAP2,
     &       TC1,TC2)
        CALL COARSE(43,ALFK,99999.D0,10,0,NALF,0,10,0,ALF1,ALF2,
     &       0.D0,0.D0)
        CALL FINE3(43,X(23),TCK,ALFK,NFLAP,NTC,NALF,4,6,10,FLAP1,
     &       FLAP2,TC1,TC2,ALF1,ALF2,CLWBT)
C
C               CMWBT(FLAP,TC,ALPHA)
C
        CALL FINE3(44,X(23),TCK,ALFK,NFLAP,NTC,NALF,4,6,10,FLAP1,
     &       FLAP2,TC1,TC2,ALF1,ALF2,CMWBT)
C
C               CDTOT(FLAP,TC,ALPHA)
C
```

```
      EO1=0
      IF(EO.NE.0)EO1=.25
      TCK1=TCK*(1-EO1)
      CALL FINE3(45,X(23),TCK1,ALFK,NFLAP,NTC,NALF,4,6,10,
     &       FLAP1,FLAP2,TC1,TC2,ALF1,ALF2,CDTOT)
C
C               DEPDAL(EPSLN(FLAP,TC,ALPHA))
C
      CALL LIMITS(ALFK,17.998D0,-7.998D0)
      CALL COARSE(56,ALFK+.001,99999.D0,10,0,NAEP,0,10,0,
     &       AEP1,AEP2,0.D0,0.D0)
      CALL FINE3(56,X(23),TCK,ALFK+.001,NFLAP,NTC,NAEP,4,6,
     &       10,FLAP1,FLAP2,TC1,TC2,AEP1,AEP2,EPSLN1)
      CALL COARSE(56,ALFK-.001,99999.D0,10,0,NAEP,0,10,0,AEP1,
     &       AEP2,0.D0,0.D0)
      CALL FINE3(56,X(23),TCK,ALFK-.001,NFLAP,NTC,NAEP,4,6,10,
     &       FLAP1,FLAP2,TC1,TC2,AEP1,AEP2,EPSLN2)
      DEPDAL=(EPSLN1-EPSLN2)/0.002
C
C               CYB0(FLAP,TC)
C
      CALL FINE2(59,X(23),TCK,NFLAP,NTC,4,6,FLAP1,FLAP2,TC1,
     &       TC2,CYB0)
C
C               CNB0(FLAP,TC)
C
      CALL FINE2(60,X(23),TCK,NFLAP,NTC,4,6,FLAP1,FLAP2,TC1,TC2,
     &       CNB0)
C
C               CRB0(FLAP,TC)
C
      CALL FINE2(61,X(23),TCK,NFLAP,NTC,4,6,FLAP1,FLAP2,TC1,TC2,
     &       CRB0)
C
C               DCYB(FLAP,TC,BETA)
C
      BETAK=BETA
      CALL LIMITS(BETAK,25.D0,-25.D0)
      CALL COARSE(62,BETAK,99999.D0,9,0,NBET,0,10,0,BT1,BT2,
     &       0.D0,0.D0)
      CALL FINE3(62,X(23),TCK,BETAK,NFLAP,NTC,NBET,4,6,9,FLAP1,
     &       FLAP2,TC1,TC2,BT1,BT2,DCYB)
C
C               DCNB(FLAP,TC,BETA)
C
      CALL FINE3(63,X(23),TCK,BETAK,NFLAP,NTC,NBET,4,6,9,FLAP1,
     &       FLAP2,TC1,TC2,BT1,BT2,DCNB)
C
C               DCRB(FLAP,TC,BETA)
```

```
C
      CALL FINE3(64,X(23),TCK,BETAK,NFLAP,NTC,NBET,4,6,9,FLAP1,
     &       FLAP2,TC1,TC2,BT1,BT2,DCRB)
C
C               DCDB(BETA)
C
      BETAL=BETA
      CALL LIMITS(BETAL,50.D0,-50.D0)
      CALL COARSE(72,BETAL,99999.D0,17,0,NBET,0,0,0,BT1,BT2,
     &       0.D0,0.D0)
      DCDB=ADATA(17+NBET,32)+(ADATA(18+NBET,32)-ADATA(17+
     &       NBET,32))*(BETAL-BT1)/(BT2-BT1)
      RETURN
      END

      SUBROUTINE DERIV2(X,NFLAP,FLAP1,FLAP2,ALPHA,TC,BETAK,
     &       NBET,BT1,BT2,DCLGE,DCDGE,DCMGE,CMQ,CNP,MCNR,CRR,
     &       CRP,DCLEL,DCMEL,DCDEL,RUDFAC,TCK,NTC,TC1,TC2)
      IMPLICIT REAL*8(A-H,O-Z)
      REAL*8 X(*),ADATA(700,43)
      COMMON/AERODAT/ADATA
C
C               DCLGE(FLAP,TC,ALPHA)
C
      ALFK=ALPHA
      TCK=TC
      CALL LIMITS(ALFK,20.D0,-5.D0)
      CALL LIMITS(TCK,2.D0,-.2D0)
      CALL COARSE(46,TCK,ALFK,7,10,NTC,NALF,4,11,TC1,TC2,
     &       ALF1,ALF2)
      CALL FINE3(46,X(23),TCK,ALFK,NFLAP,NTC,NALF,4,7,10,
     &       FLAP1,FLAP2,TC1,TC2,ALF1,ALF2,DCLGE)
C
C               DCDGE(FLAP,TC,ALPHA)
C
      CALL FINE3(47,X(23),TCK,ALFK,NFLAP,NTC,NALF,4,7,10,
     &       FLAP1,FLAP2,TC1,TC2,ALF1,ALF2,DCDGE)
C
C               DCMGE(FLAP,TC,ALPHA)
C
      CALL FINE3(48,X(23),TCK,ALFK,NFLAP,NTC,NALF,4,7,10,
     &       FLAP1,FLAP2,TC1,TC2,ALF1,ALF2,DCMGE)
C
C               CMQ(FLAP,TC,ALPHA)
C
      ALFK=ALPHA
      TCK=TC
      CALL LIMITS(ALFK,14.D0,-8.D0)
      CALL LIMITS(TCK,2.D0,0.D0)
```

```
      CALL COARSE(57,TCK,ALFK,4,7,NTC,NALF,4,8,TC1,TC2,
     &       ALF1,ALF2)
      CALL FINE3(57,X(23),TCK,ALFK,NFLAP,NTC,NALF,4,4,7,
     &       FLAP1,FLAP2,TC1,TC2,ALF1,ALF2,CMQ)
C
C               CNP(FLAP,TC,ALPHA)
C
      ALFK=ALPHA
      CALL LIMITS(ALFK,18.D0,-6.D0)
      CALL COARSE(73,TCK,ALFK,5,5,NTC,NALF,4,9,TC1,TC2,
     &       ALF1,ALF2)
      CALL FINE3(73,X(23),TCK,ALFK,NFLAP,NTC,NALF,4,5,5,
     &       FLAP1,FLAP2,TC1,TC2,ALF1,ALF2,CNP)
C
C               CNR(FLAP,TC,ALPHA)
C
      CALL FINE3(74,X(23),TCK,ALFK,NFLAP,NTC,NALF,4,5,5,
     &       FLAP1,FLAP2,TC1,TC2,ALF1,ALF2,CNR)
C
C               CLR(FLAP,TC,ALPHA)
C
      CALL FINE3(76,X(23),TCK,ALFK,NFLAP,NTC,NALF,4,5,5,
     &       FLAP1,FLAP2,TC1,TC2,ALF1,ALF2,CRR)
C
C               CLP(FLAP,TC)
C
      CALL FINE2(75,X(23),TCK,NFLAP,NTC,4,5,FLAP1,FLAP2,TC1,
     &       TC2,CRP)
C
C               DCLEL(FLAP,ALPHA,TC,ELEV)
C
      CALL COARSE(49,TCK,X(16),6,5,NTC,NELD,9,15,TC1,TC2,
     &       ELD1,ELD2)
      CALL FINE4(49,X(23),ALFK,TCK,X(16),NFLAP,NALF,NTC,NELD,
     &       4,5,6,5,FLAP1,FLAP2,ALF1,ALF2,TC1,TC2,ELD1,ELD2,
     &       DCLEL)
C
C               DCMEL(FLAP,ALPHA,TC,ELEV)
C
      CALL FINE4(50,X(23),ALFK,TCK,X(16),NFLAP,NALF,NTC,NELD,
     &       4,5,6,5,FLAP1,FLAP2,ALF1,ALF2,TC1,TC2,ELD1,ELD2,
     &       DCMEL)
C
C               DCDEL(ELEV)
C
      CALL COARSE(51,X(16),99999.D0,5,0,NELD,0,0,0,ELD1,ELD2,
     &       0.D0,0.D0)
      DCDEL=ADATA(5+NELD,11)+(ADATA(6+NELD,11)-ADATA(5+NELD,11))
     &         *(X(16)-ELD1)/(ELD2-ELD1)
```

```
C
C               RUDFAC(FLAP,TC,RDR,BETA)
C
      IF(X(19).EQ.0)THEN
              RUDFAC=0.
              RETURN
      ENDIF
      SGN=-1
      IF(DABS(X(19)-DABS(X(19))).LE.1D-8)SGN=1
      CALL COARSE(69,TCK,SGN,4,2,NTC,NRDR,4,8,TC1,TC2,RDR1,
     &       RDR2)
      CALL COARSE(69,BETAK,99999.D0,11,0,NBET,0,10,0,BT1,BT2,
     &       0.D0,0.D0)
      CALL FINE4(69,X(23),TCK,SGN,BETAK,NFLAP,NTC,NRDR,NBET,
     &         4,4,2,11,FLAP1,FLAP2,TC1,TC2,RDR1,RDR2,BT1,BT2,
     &         RUDFAC)
      RETURN
      END


      SUBROUTINE DERIV3(X,ALPHA,NFLAP,FLAP1,FLAP2,TCK,NTC,TC1,
     &       TC2,DCRA,DCNA,DCLSPL,DCLSPR,DCMSPL,DCMSPR,DCDSPL,
     &       DCDSPR,DCYSPL,DCYSPR,DCNSPL,DCNSPR,DCRSPL,DCRSPR)
      IMPLICIT REAL*8(A-H,O-Z)
      REAL*8 X(*),ADATA(700,43)
      COMMON/AERODAT/ADATA
C
C               DCRA(FLAP,TC,ALPHA,AIL)
C
      ALFK=ALPHA
      CALL LIMITS(ALFK,16.D0,-4.D0)
      CALL COARSE(70,ALFK,X(18),8,5,NALF,NAIL,8,16,ALF1,ALF2,
     &       AIL1,AIL2)
      CALL FINE4(70,X(23),TCK,ALFK,X(18),NFLAP,NTC,NALF,NAIL,
     &       4,4,8,5,FLAP1,FLAP2,TC1,TC2,ALF1,ALF2,AIL1,AIL2,
     &       DCRA)
C
C               DCNA(FLAP,TC,ALPHA,AIL)
C
      CALL COARSE(71,ALFK,99999.D0,7,0,NALF,0,8,0,ALF1,ALF2,
     &       0.D0,0.D0)
      CALL FINE4(71,X(23),TCK,ALFK,X(18),NFLAP,NTC,NALF,NAIL,
     &       4,4,7,5,FLAP1,FLAP2,TC1,TC2,ALF1,ALF2,AIL1,AIL2,
     &       DCNA)
C
C               DCLSP(FLAP,ALPHA,TC,SPL) - LEFT & RIGHT
C
      IF(X(23).EQ.0.)THEN
              DCLSPL=0.
```

```
                DCLSPR=0.
                DCMSPL=0.
                DCMSPR=0.
                DCDSPL=0.
                DCDSPR=0.
                DCYSPL=0.
                DCYSPR=0.
                DCNSPL=0.
                DCNSPR=0.
                DCRSPL=0.
                DCRSPR=0.
                RETURN
              ENDIF
              ALFK=ALPHA
              CALL LIMITS(ALFK,18.0D0,-6.D0)
              CALL COARSE(53,X(23),ALFK,3,5,NFLAP,NALF,0,3,FLAP1,
     &             FLAP2,ALF1,ALF2)
              CALL COARSE(53,X(20),99999.D0,4,0,NSPL,0,12,0,SPL1,
     &             SPL2,0.D0,0.D0)
              CALL FINE4(53,X(23),ALFK,TCK,X(20),NFLAP,NALF,NTC,NSPL,
     &             3,5,4,4,FLAP1,FLAP2,ALF1,ALF2,TC1,TC2,SPL1,SPL2,
     &             DCLSPL)
              CALL COARSE(53,X(21),99999.D0,4,0,NSPR,0,12,0,SPR1,SPR2,
     &             0.D0,0.D0)
              CALL FINE4(53,X(23),ALFK,TCK,X(21),NFLAP,NALF,NTC,NSPR,
     &             3,5,4,4,FLAP1,FLAP2,ALF1,ALF2,TC1,TC2,SPR1,SPR2,
     &             DCLSPR)
C
C                 DCMSP(FLAP,ALPHA,TC,SPL) - LEFT & RIGHT
C
              CALL FINE4(54,X(23),ALFK,TCK,X(20),NFLAP,NALF,NTC,NSPL,
     &             3,5,4,4,FLAP1,FLAP2,ALF1,ALF2,TC1,TC2,SPL1,SPL2,
     &             DCMSPL)
              CALL FINE4(54,X(23),ALFK,TCK,X(21),NFLAP,NALF,NTC,NSPR,
     &             3,5,4,4,FLAP1,FLAP2,ALF1,ALF2,TC1,TC2,SPR1,SPR2,
     &             DCMSPR)
C
C                 DCDSP(FLAP,ALPHA,TC,SPL) - LEFT & RIGHT
C
              CALL FINE4(55,X(23),ALFK,TCK,X(20),NFLAP,NALF,NTC,NSPL,
     &             3,5,4,4,FLAP1,FLAP2,ALF1,ALF2,TC1,TC2,SPL1,SPL2,
     &             DCDSPL)
              CALL FINE4(55,X(23),ALFK,TCK,X(21),NFLAP,NALF,NTC,NSPR,
     &             3,5,4,4,FLAP1,FLAP2,ALF1,ALF2,TC1,TC2,SPR1,SPR2,
     &             DCDSPR)
C
C                 DCYSP(FLAP,ALPHA,TC,SPL) - LEFT & RIGHT
C
              CALL FINE4(65,X(23),ALFK,TCK,X(20),NFLAP,NALF,NTC,NSPL,
```

```
     &       3,5,4,4,FLAP1,FLAP2,ALF1,ALF2,TC1,TC2,SPL1,SPL2,
     &       DCYSPL)
        CALL FINE4(65,X(23),ALFK,TCK,X(21),NFLAP,NALF,NTC,NSPR,
     &       3,5,4,4,FLAP1,FLAP2,ALF1,ALF2,TC1,TC2,SPR1,SPR2,
     &       DCYSPR)
C
C                 DCNSP(FLAP,ALPHA,TC,SPL) - LEFT & RIGHT
C
        CALL FINE4(66,X(23),ALFK,TCK,X(20),NFLAP,NALF,NTC,NSPL,
     &       3,5,4,4,FLAP1,FLAP2,ALF1,ALF2,TC1,TC2,SPL1,SPL2,
     &       DCNSPL)
        CALL FINE4(66,X(23),ALFK,TCK,X(21),NFLAP,NALF,NTC,NSPR,
     &       3,5,4,4,FLAP1,FLAP2,ALF1,ALF2,TC1,TC2,SPR1,SPR2,
     &       DCNSPR)
C
C                 DCRSP(FLAP,ALPHA,TC,SPL) - LEFT & RIGHT
C
        CALL FINE4(67,X(23),ALFK,TCK,X(20),NFLAP,NALF,NTC,NSPL,
     &       3,5,4,4,FLAP1,FLAP2,ALF1,ALF2,TC1,TC2,SPL1,SPL2,
     &       DCRSPL)
        CALL FINE4(67,X(23),ALFK,TCK,X(21),NFLAP,NALF,NTC,NSPR,
     &       3,5,4,4,FLAP1,FLAP2,ALF1,ALF2,TC1,TC2,SPR1,SPR2,
     &       DCRSPR)
        RETURN
        END



C-------------------------------------------------------------C
C                                                             C
C            Search and Interpolation Routines                C
C                                                             C
C-------------------------------------------------------------C
C
C*****************************************************C
C            4 Independent Variables                  C
C*****************************************************C
C
        SUBROUTINE FINE4(IU,A,B,C,D,NA,NB,NC,ND,IA,IB,IC,ID,A1,
     &       A2,B1,B2,C1,C2,D1,D2,Y)
        IMPLICIT REAL*8 (A-H,O-Z)
        REAL*8 Z(14),ADATA(700,43)
        COMMON/AERODAT/ADATA
        IBASE=IA+IB+IC+ID
        BRAT=(B-B1)/(B2-B1)
        CRAT=(C-C1)/(C2-C1)
        DRAT=(D-D1)/(D2-D1)
        LOC=IBASE+(ID*IC*IB)*(NA-1)+(ID*IC)*(NB-1)+ID*(NC-1)+ND
        Z(1)=ADATA(LOC,IU-40)+(ADATA(LOC+1,IU-40)-ADATA(LOC,
```

```
     &       IU-40))*DRAT
       LOC=IBASE+(ID*IC*IB)*(NA-1)+(ID*IC)*(NB-1)+ID*NC+ND
       Z(2)=ADATA(LOC,IU-40)+(ADATA(LOC+1,IU-40)-ADATA(LOC,
     &       IU-40))*DRAT
       LOC=IBASE+(ID*IC*IB)*(NA-1)+(ID*IC)*NB+ID*(NC-1)+ND
       Z(3)=ADATA(LOC,IU-40)+(ADATA(LOC+1,IU-40)-ADATA(LOC,
     &       IU-40))*DRAT
       LOC=IBASE+(ID*IC*IB)*(NA-1)+(ID*IC)*NB+ID*NC+ND
       Z(4)=ADATA(LOC,IU-40)+(ADATA(LOC+1,IU-40)-ADATA(LOC,
     &       IU-40))*DRAT
       LOC=IBASE+(ID*IC*IB)*NA+(ID*IC)*(NB-1)+ID*(NC-1)+ND
       Z(5)=ADATA(LOC,IU-40)+(ADATA(LOC+1,IU-40)-ADATA(LOC,
     &       IU-40))*DRAT
       LOC=IBASE+(ID*IC*IB)*NA+(ID*IC)*(NB-1)+ID*NC+ND
       Z(6)=ADATA(LOC,IU-40)+(ADATA(LOC+1,IU-40)-ADATA(LOC,
     &       IU-40))*DRAT
       LOC=IBASE+(ID*IC*IB)*NA+(ID*IC)*NB+ID*(NC-1)+ND
       Z(7)=ADATA(LOC,IU-40)+(ADATA(LOC+1,IU-40)-ADATA(LOC,
     &       IU-40))*DRAT
       LOC=IBASE+(ID*IC*IB)*NA+(ID*IC)*NB+ID*NC+ND
       Z(8)=ADATA(LOC,IU-40)+(ADATA(LOC+1,IU-40)-ADATA(LOC,
     &       IU-40))*DRAT
       Z(9)=Z(1)+(Z(2)-Z(1))*CRAT
       Z(10)=Z(3)+(Z(4)-Z(3))*CRAT
       Z(11)=Z(5)+(Z(6)-Z(5))*CRAT
       Z(12)=Z(7)+(Z(8)-Z(7))*CRAT
       Z(13)=Z(9)+(Z(10)-Z(9))*BRAT
       Z(14)=Z(11)+(Z(12)-Z(11))*BRAT
       Y=Z(13)+(Z(14)-Z(13))*(A-A1)/(A2-A1)
       RETURN
       END

C*************************************************C
C              3 Independent Variables            C
C*************************************************C
C
       SUBROUTINE FINE3(IU,A,B,C,NA,NB,NC,IA,IB,IC,A1,A2,
     &         B1,B2,C1,C2,Y)
       IMPLICIT REAL*8 (A-H,O-Z)
       REAL*8 Z(6),ADATA(700,43)
       COMMON/AERODAT/ADATA
       IBASE=IA+IB+IC
       BRAT=(B-B1)/(B2-B1)
       CRAT=(C-C1)/(C2-C1)
       LOC=IBASE+(IC*IB)*(NA-1)+IC*(NB-1)+NC
       Z(1)=ADATA(LOC,IU-40)+(ADATA(LOC+1,IU-40)-ADATA(LOC,
     &       IU-40))*CRAT
       LOC=IBASE+(IC*IB)*(NA-1)+IC*NB+NC
       Z(2)=ADATA(LOC,IU-40)+(ADATA(LOC+1,IU-40)-ADATA(LOC,
```

```
     &       IU-40))*CRAT
        LOC=IBASE+(IC*IB)*NA+IC*(NB-1)+NC
        Z(3)=ADATA(LOC,IU-40)+(ADATA(LOC+1,IU-40)-ADATA(LOC,
     &       IU-40))*CRAT
        LOC=IBASE+(IC*IB)*NA+IC*NB+NC
        Z(4)=ADATA(LOC,IU-40)+(ADATA(LOC+1,IU-40)-ADATA(LOC,
     &       IU-40))*CRAT
        Z(5)=Z(1)+(Z(2)-Z(1))*BRAT
        Z(6)=Z(3)+(Z(4)-Z(3))*BRAT
        Y=Z(5)+(Z(6)-Z(5))*(A-A1)/(A2-A1)
        RETURN
        END

C************************************************C
C            2 Independent Variables             C
C************************************************C
C
        SUBROUTINE FINE2(IU,A,B,NA,NB,IA,IB,A1,A2,B1,B2,Y)
        IMPLICIT REAL*8 (A-H,O-Z)
        REAL*8 Z(2),ADATA(700,43)
        COMMON/AERODAT/ADATA
        IBASE=IA+IB
        BRAT=(B-B1)/(B2-B1)
        LOC=IBASE+IB*(NA-1)+NB
        Z(1)=ADATA(LOC,IU-40)+(ADATA(LOC+1,IU-40)-ADATA(LOC,
     &       IU-40))*BRAT
        LOC=IBASE+IB*NA+NB
        Z(2)=ADATA(LOC,IU-40)+(ADATA(LOC+1,IU-40)-ADATA(LOC,
     &       IU-40))*BRAT
        Y=Z(1)+(Z(2)-Z(1))*(A-A1)/(A2-A1)
        RETURN
        END

C*********************************************************C
C              Find General Location of Interest          C
C*********************************************************C
C
        SUBROUTINE COARSE(IU,A,B,JA,JB,KA,K,IA,IB,A1,A2,B1,B2)
        IMPLICIT REAL*8(A-H,O-Z)
        REAL*8 ADATA(700,43)
        COMMON/AERODAT/ADATA
        IF(A.GE.9999.D0)GOTO 41
        CALL READEM(IU,A,JA,KA,IA,A1,A2,IFLAG)
        IF(IFLAG.EQ.0)GOTO 41
        WRITE(*,50)1,IU,A
        STOP
41      IF(B.GE.9999.D0)GOTO 60
        CALL READEM(IU,B,JB,K,IB,B1,B2,IFLAG)
        IF(IFLAG.EQ.0)GOTO 60
```

```
      WRITE(*,50)2,IU,B
50    FORMAT(' NO MATCH FOUND IN PASS ',I1,' OF FILE ',I2,
     &       '   X=',1PD12.3)
      STOP
60    RETURN
      END

C*****************************************C
C              Read from Tables           C
C*****************************************C
C
      SUBROUTINE READEM(IU,X,J,K,I,X1,X2,IFLAG)
      IMPLICIT REAL*8(A-H,O-Z)
      REAL*8 ADATA(700,43)
      COMMON/AERODAT/ADATA
      IFLAG=0
      DO 10 K=1,J-1
      X1=ADATA(I+K,IU-40)
      X2=ADATA(I+K+1,IU-40)
      IF(((X.GE.X1).AND.(X.LE.X2)).OR.
     &       ((X.LE.X1).AND.(X.GE.X2)))RETURN
10    CONTINUE
      IFLAG=1
      RETURN
      END

C*****************************************************C
C            Load Data Tables into Memory             C
C*****************************************************C
      SUBROUTINE LOADEM
      IMPLICIT REAL*8(A-H,O-Z)
      COMMON/AERODAT/ADATA(700,43)
            DO 5 II=1,700
            DO 5 JJ=1,43
5           ADATA(II,JJ)=0.D0
            DO 10 II=41,83
            IF((II.EQ.42).OR.(II.EQ.58))GOTO 10
            OPEN(UNIT=II,STATUS='UNKNOWN')
            READ(II,*,END=10)(ADATA(JJ,II-40),JJ=1,700)
10          CLOSE(UNIT=II)
      RETURN
      END

C*****************************************C
C             Clear the Screen            C
C*****************************************C
C
      SUBROUTINE CLEAR
      CHARACTER CLEAN(10)
```

```
      DATA CLEAN/27,91,50,74,27,91,48,59,48,72/
      WRITE(*,1)CLEAN
1     FORMAT(' ',10A)
      RETURN
      END

C=============================================================C
C                                                             C
C                        Autopilot Set-up                     C
C                                                             C
C=============================================================C
C
      SUBROUTINE CONTROL(R,X,T,RC,SC,CC,PAP,AH,AT,GS,FC,PA,
     &       RA,INRG)
      IMPLICIT REAL*8(A-H,O-Z)
      REAL*8 RC(3,10),T(19,20),SC(7,10),CC(9,10),R(*),X(*)
      INTEGER PAP,AH,AT,GS,FC,PA,RA,YD
      COMMON/U     /ELEV,THTL,AIL,RDR,SPLL,SPLR,GEAR,FLAP,
     &       EO,ETAB
      DATA IAL,IAI,IP,I1,I2,I3,I4/0,0,0,0,0,0,0/
      DATA I5,I6,I7,J1,J2,J3,J4,J5,J6,J7,J8,J9/0,0,0,0,0,
     &       0,0,0,0,0,0,0/
      PA=0
      RA=0
      INRG=0
      PAP=0
      AH=0
      AT=0
      GS=0
      FC=0
      PI=4.0D0*DATAN(1.0D0)

C*****************************************************C
C              Initialize the TIME Array              C
C*****************************************************C
C
      DO 99 I=1,19
      DO 99 J=1,20
99    T(I,J)=99999999.
      CALL CLEAR
      WRITE(*,*)'Initial aircraft location (glideslope '
      WRITE(*,*)'transmitter at (0,0) with approach from '
      WRITE(*,*)'(X<0,Y=0) at 2.5 degrees, feet):'
      READ(*,*)X(13),X(14)
      WRITE(*,*)
      WRITE(*,*)'   Autopilot Configuration:  (YES)'
199   FORMAT(/,'                                        On')
      WRITE(*,100)
100   FORMAT('                  Pitch A/P on? ',$)
```

```
1       FORMAT('1')
        CALL NNOT(&2)
        PAP=1
        WRITE(*,199)
2       WRITE(*,102)
102     FORMAT('              Speed Control on? ',$)
        CALL NNOT(&4)
        AT=1
        WRITE(*,199)
4       WRITE(*,141)
141     FORMAT('             Glidepath Hold on? ',$)
        CALL NNOT(&41)
        GS=1
        WRITE(*,199)
41      WRITE(*,142)
142     FORMAT(' Total Energy Rate Feedback on? ',$)
        CALL NNOT(&42)
        INRG=1
        WRITE(*,199)
42      IF((PAP.EQ.0).AND.(AH.EQ.0).AND.(AT.EQ.0))GOTO 9
        WRITE(*,*)
        WRITE(*,*)'Initial Commands:   (YES)'
        IF(PAP.EQ.0)GOTO 8
        WRITE(*,110)R(2)
110     FORMAT('  Commanded pitch   ',1pe12.3,' degrees')
        CALL NNO(&7)
        WRITE(*,6)
6       FORMAT('  Enter new command:  ')
        READ(*,*)R(2)
        R(2)=R(2)
7       IF(AH.EQ.0)GOTO 8
        WRITE(*,111)R(1)
111     FORMAT('  Commanded altitude',1pe12.3,' feet')
        CALL NNO(&8)
        WRITE(*,6)
        READ(*,*)R(1)
8       IF(AT.EQ.0)GOTO 9
        WRITE(*,112)R(3)
112     FORMAT('  Commanded airspeed',1pe12.3,' ft/s')
        CALL NNO(&9)
        WRITE(*,6)
        READ(*,*)R(3)
9       WRITE(*,*)
        WRITE(*,*)'Disturbances:'
10      WRITE(*,*)'          Changes in  1) Commands'
        WRITE(*,*)'                      2) States'
        WRITE(*,*)'                      3) Control Inputs'
        WRITE(*,*)'                      4) Done'
        WRITE(*,103)
```

```
103     FORMAT('  Your Choice?:  ')
        READ(*,*)NDC
        IF((NDC.GT.4).OR.(NDC.LT.1))GOTO 10
        CALL CLEAR
        GOTO (11,16,26,36), NDC
11      WRITE(*,*)
        WRITE(*,*)'Command Disturbances:'
        WRITE(*,*)'      Available-  1) Altitude  2) Pitch'
        WRITE(*,*)'                  3) Airspeed  4) Done'
        READ(*,*)NCCM
        IF((NCCM.GT.4).OR.(NCCM.LT.1))GOTO 11
        GOTO (12,14,15,9), NCCM
12      IAL=IAL+1
        WRITE(*,111)R(1)
        WRITE(*,104)
104     FORMAT('  Enter new altitude:  ')
        READ(*,*)RC(1,IAL)
        WRITE(*,13)
13      FORMAT('  Enter starting time for disturbance:  ')
        READ(*,*)T(1,IAL)
        GOTO 11
14      IP=IP+1
        WRITE(*,110)R(2)
        WRITE(*,105)
105     FORMAT(' Enter new pitch angle:  ')
        READ(*,*)RC(2,IP)
        RC(2,IP)=RC(2,IP)
        WRITE(*,13)
        READ(*,*)T(2,IP)
        GOTO 11
15      IAI=IAI+1
        WRITE(*,112)R(3)
        WRITE(*,106)
106     FORMAT(' Enter new airspeed:  ')
        READ(*,*)RC(3,IAI)
        WRITE(*,13)
        READ(*,*)T(3,IAI)
        GOTO 11
16      WRITE(*,*)
        WRITE(*,*)'State Disturbances:'
        WRITE(*,*)'Available- 1) U         2) V         3) W'
        WRITE(*,*)'           4) Phi       5) Theta     6) Psi'
        WRITE(*,*)'           7) Altitude  8) Done'
        READ(*,*)NCS
        IF((NCS.GT.8).OR.(NCS.LT.1))GOTO 16
        GOTO (17,20,21,22,23,24,25,9), NCS
17      I1=I1+1
        WRITE(*,107)X(1)
107     FORMAT(' Initial U velocity',1pe12.3,' fps')
```

```
      WRITE(*,18)
18    FORMAT(' Enter disturbed value:  ')
      READ(*,*)SC(1,I1)
      WRITE(*,13)
      READ(*,*)T(4,I1)
      WRITE(*,19)
19    FORMAT(' Duration:  ')
      READ(*,*)T(4,I1+10)
      GOTO 16
20    I2=I2+1
      WRITE(*,108)X(2)
108   FORMAT(' Initial V velocity',1pe12.3,' fps')
      WRITE(*,18)
      READ(*,*)SC(2,I2)
      WRITE(*,13)
      READ(*,*)T(5,I2)
      WRITE(*,19)
      READ(*,*)T(5,I2+10)
      GOTO 16
21    I3=I3+1
      WRITE(*,109)X(3)
109   FORMAT(' Initial W velocity',1pe12.3,' fps')
      WRITE(*,18)
      READ(*,*)SC(3,I3)
      WRITE(*,13)
      READ(*,*)T(6,I3)
      WRITE(*,19)
      READ(*,*)T(6,I3+10)
      GOTO 16
22    I4=I4+1
      WRITE(*,120)X(4)*180./PI
120   FORMAT(' Initial bank angle',1pe12.3,' degrees')
      WRITE(*,18)
      READ(*,*)SC(4,I4)
      SC(4,I4)=SC(4,I4)*PI/180.
      WRITE(*,13)
      READ(*,*)T(7,I4)
      WRITE(*,19)
      READ(*,*)T(7,I4+10)
      GOTO 16
23    I5=I5+1
      WRITE(*,121)X(5)*180.D0/PI
121   FORMAT(' Initial pitch angle',1pe12.3,' degrees')
      WRITE(*,18)
      READ(*,*)SC(5,I5)
      SC(5,I5)=SC(5,I5)*PI/180.
      WRITE(*,13)
      READ(*,*)T(8,I5)
      WRITE(*,19)
```

```
      READ(*,*)T(8,I5+10)
      GOTO 16
24    I6=I6+1
      WRITE(*,122)X(6)*180./PI
122   FORMAT(' Initial yaw angle',1pe12.3,' degrees')
      WRITE(*,18)
      READ(*,*)SC(6,I6)
      SC(6,I6)=SC(6,I6)*PI/180.
      WRITE(*,13)
      READ(*,*)T(9,I6)
      WRITE(*,19)
      READ(*,*)T(9,I6+10)
      GOTO 16
25    I7=I7+1
      WRITE(*,123)X(12)
123   FORMAT(' Initial altitude',1pe12.3,' feet')
      WRITE(*,18)
      READ(*,*)SC(7,I7)
      WRITE(*,13)
      READ(*,*)T(10,I7)
      WRITE(*,19)
      READ(*,*)T(10,I7+10)
      GOTO 16
26    WRITE(*,*)
      WRITE(*,*)'Changing Control Inputs:'
      WRITE(*,*)'Available-1) Elevator  2) Throttle  3) Aileron'
      WRITE(*,*)'           4) Rudder    5) Gear      6) Flaps'
      WRITE(*,*)'           7) Left Spoiler  8) Right Spoiler'
      WRITE(*,*)'           9) Engine Failure        10) Done'
      READ(*,*)NCCI
      IF((NCCI.GT.10).OR.(NCCI.LT.1))GOTO 26
      GOTO (27,28,30,31,32,34,50,51,52,9), NCCI
27    J1=J1+1
      WRITE(*,130)X(16)
130   FORMAT(' Initial elevator angle',1pe12.3,' degrees')
      WRITE(*,18)
      READ(*,*)CC(1,J1)
      WRITE(*,13)
      READ(*,*)T(11,J1)
      GOTO 26
28    J2=J2+1
      WRITE(*,131)X(17)
131   FORMAT(' Initial throttle setting',1pe12.3)
29    WRITE(*,18)
      READ(*,*)CC(2,J2)
      WRITE(*,13)
      READ(*,*)T(12,J2)
      GOTO 26
30    J3=J3+1
```

```
      WRITE(*,132)X(18)
132   FORMAT(' Initial aileron angle',1pe12.3,' degrees')
      WRITE(*,18)
      READ(*,*)CC(3,J3)
      WRITE(*,13)
      READ(*,*)T(13,J3)
      GOTO 26
31    J4=J4+1
      WRITE(*,133)X(19)
133   FORMAT(' Initial rudder angle',1pe12.3,' degrees')
      WRITE(*,18)
      READ(*,*)CC(4,J4)
      WRITE(*,13)
      READ(*,*)T(14,J4)
      GOTO 26
32    J5=J5+1
      IF(INT(X(22)).EQ.0)WRITE(*,*)'Gear is UP'
      IF(INT(X(22)).EQ.1)WRITE(*,*)'Gear is DOWN'
33    WRITE(*,*)'Change gear position-  0) UP  1) DOWN'
      READ(*,*)CC(5,J5)
      IF((INT(CC(5,J5)).NE.0).AND.(INT(CC(5,J5)).NE.1))GOTO 33
      WRITE(*,13)
      READ(*,*)T(15,J5)
      GOTO 26
34    J6=J6+1
      IF(INT(X(23)).EQ.0)WRITE(*,*)'Currently   0/0  flaps'
      IF(INT(X(23)).EQ.18)WRITE(*,*)'Currently 18/0  flaps'
      IF(INT(X(23)).EQ.36)WRITE(*,*)'Currently 36/0  flaps'
      IF(INT(X(23)).EQ.54)WRITE(*,*)'Currently 36/30 flaps'
35    WRITE(*,*)'Change flap setting-  0) 0/0   18) 18/0   '
      WRITE(*,*)'                          36) 36/0  54) 36/30'
      READ(*,*)CC(6,J6)
      IF((INT(CC(6,J6)).LT.0).OR.(INT(CC(6,J6)).GT.54))GOTO 35
      WRITE(*,13)
      READ(*,*)T(16,J6)
      GOTO 26
50    J7=J7+1
      WRITE(*,*)'Left Spoiler'
      WRITE(*,140)X(20)
140   FORMAT(' Initial spoiler uprig',1PE12.3,' degrees')
      WRITE(*,18)
      READ(*,*)CC(7,J7)
      WRITE(*,13)
      READ(*,*)T(17,J7)
      GOTO 26
51    J8=J8+1
      WRITE(*,*)' Right Spoiler'
      WRITE(*,140)X(21)
      WRITE(*,18)
```

```
          READ(*,*)CC(8,J8)
          WRITE(*,13)
          READ(*,*)T(18,J8)
          GOTO 26
52        IF(J9.EQ.1)THEN
                  WRITE(*,*)' Only one engine failure per run'
                  GOTO 26
          ENDIF
          J9=1
          WRITE(*,13)
          READ(*,*)T(19,1)
          GOTO 26
36        RETURN
          END

C=============================================================C
C                                                             C
C                    Discrete Time Function                   C
C                                                             C
C=============================================================C
C
          SUBROUTINE LDD(TIME,DT,X,XD,ICOUNT)
          IMPLICIT REAL*8(A-H,O-Z)
          REAL*8 X(*),XD(*),R(4),T(19,20),RC(3,10),SC(7,10),
         &       CC(9,10),ZZ(17)
          COMMON/U     /ELEV,THTL,AIL,RDR,SPLL,SPLR,GEAR,FLAP,
         &       EO,ETAB
          COMMON/AERODAT/ADATA(700,43)
          INTEGER PAP,AH,AT,GS,FC,PA,RA,YD
          IF(ICOUNT.EQ.0)THEN
                  R(1)=X(12)
                  R(2)=X(5)*180.D0/(4.D0*DATAN(1.0D0))
                  R(3)=DSQRT(X(1)**2+X(3)**2)*36/60
                  R(4)=X(4)
                  X(15)=X(12)
                  GEARC=X(22)
                  FLAPC=X(23)
                  CALL CONTROL(R,X,T,RC,SC,CC,PAP,AH,AT,GS,FC,PA,
         &                RA,INRG)
          ENDIF
          DO 3 I=1,10

C                         Commands

          DO 1 J=1,3
1         IF(TIME.GE.T(J,I))R(J)=RC(J,I)
          VCMD=R(3)
          PITCMD=R(2)*(4.D0*DATAN(1.0D0))/180.D0
          ALTCMD=R(1)
```

```
C                          States

          DO 2, J=1,6
2         IF((TIME.GE.T(J+3,I)).AND.(TIME.LE.(T(J+3,I)+.005)))
         &                              X( J)=X(J)+SC(J,I)
          IF((TIME.GE.T( 10,I)).AND.(TIME.LE.(T( 10,I)+.005)))
         &                               X(12)=X(12)+SC(7,I)

C                        Control Inputs

          IF((TIME.GE.T(11,I)).AND.(TIME.LE.(T(11,I)+.005)))
         &                         ELEVCMD=ELEVCMD+CC(1,I)
          IF((TIME.GE.T(12,I)).AND.(TIME.LE.(T(12,I)+.005)))
         &                         THLCMD=THLCMD+CC(2,I)
          IF((TIME.GE.T(13,I)).AND.(TIME.LE.(T(13,I)+.005)))
         &                         AILCMD=AILCMD+CC(3,I)
          IF((TIME.GE.T(14,I)).AND.(TIME.LE.(T(14,I)+.005)))
         &                         RUDCMD=RUDCMD+CC(4,I)
          IF((TIME.GE.T(15,I)).AND.(TIME.LE.T(15,I)+.005))
         &                         GEARC=CC(5,I)
          IF((TIME.GE.T(16,I)).AND.(TIME.LE.T(16,I)+.005))
         &                         FLAPC=CC(6,I)
          IF((TIME.GE.T(17,I)).AND.(TIME.LE.(T(17,I)+.005)))
         &                         DLCCMD=DLCCMD+CC(7,I)
          IF((TIME.GE.T(18,I)).AND.(TIME.LE.(T(18,I)+.005)))
         &                         DLCCMD=DLCCMD+CC(8,I)
          IF(TIME.GE.T(19,I))EO=CC(9,1)
3         CONTINUE
          CALL APSAS(X,XD,DT,TIME,PAP,AT,GS,FC,ICOUNT,PITCMD,
         &       VCMD,DLCCMD,THLCMD,ELEVCMD,ALTCMD,GEARC,FLAPC,INRG)
          RETURN
          END

C=============================================================C
C                                                             C
C              C130/HTTB Large Disturbance Model              C
C                                                             C
C=============================================================C

          SUBROUTINE LD(TIME,X,XD)
          IMPLICIT REAL*8(A-H,O-Z)
          REAL*8 X(*),XD(*),LB,MB,NB,MASS,IXX,IYY,IZZ,IXZ,KPRES,
         &       KPOLD,KDELP
          COMMON/U      /ELEV,THTL,AIL,RDR,SPLL,SPLR,GEAR,FLAP,
         &       EO,ETAB
          COMMON/AERODAT/ADATA(700,43)
          COMMON/DLONG/CMWBT,DCMGE,DCMSPL,DCMSPR,DEPDAL,DCMEL,
         &       CMQ,DCMEO
```

```
      COMMON/DERIV/THRUST,CXS,CYS,CZS,CNS,CRS
      COMMON/AUTHOR/QBAR,VT
      COMMON/SHEAR/U20,ISHEAR,GRAD,SHRALT,SHREND,DT,WIND
      DATA CGPCNT,RHO,SIGMA,S,PDIA/30,2.3768E-3,1.,1745.5,
     &       13.5/
      DATA WEIGHT,B,CBAR,G/140000,131.8667,13.71,32.174/
      DATA WLCG,WLREF/234.1,234.1/
      DATA IXX,IYY,IZZ,IXZ/2100000,1375000,3250000,80000/
      IF((TIME.NE.0.D0).OR.(ICOUNT.EQ.1))GOTO 1
        MASS = WEIGHT/G
      ICOUNT = 0

C*******************************************************C
C              Moments of Inertia Constants            C
C*******************************************************C

        DEN = IXX*IZZ-IXZ*IXZ
         C1 = -((IZZ-IYY)*IZZ+IXZ*IXZ)/DEN
         C2 = IXZ*(IZZ-IYY+IXX)/DEN
         C3 = IZZ/DEN
         C4 = IXZ/DEN
         C5 = -(IXX-IZZ)/IYY
         C6 = IXZ/IYY
         C7 = 1/IYY
         C8 = (IXZ*C2+IXX-IYY)/IZZ
        C10 = (1+IXZ*C4)/IZZ

C***********************************************C
C             Shut off LAT/DIR terms            C
C***********************************************C

1       X(2) = 0.D0
        X(4) = 0.D0
        X(6) = 0.D0
        X(7) = 0.D0
        X(9) = 0.D0

C*******************************************************C
C                 Flight Condition Set-Up               C
C*******************************************************C

         XGE = 1.                    ! Ground Effect Flag
        VFSQ = X(1)*X(1)+X(3)*X(3)
          VF = DSQRT(VFSQ)
        VTSQ = VFSQ+X(2)*X(2)
          VT = DSQRT(VTSQ)           ! True Airspeed (kts)
        VTKT = VT*36./60.
        QBAR = .5*RHO*VTSQ           ! Dynamic Pressure
          QS = QBAR*S
```

```
            QSB = QS*B
            QSC = QS*CBAR
           SALF = X(3)/VF
           CALF = X(1)/VF
           VWND = 0.D0

C*****************************************C
C            Wind Shear Gradient          C
C*****************************************C

          IF((SHRALT.GE.X(12)).AND.(SHREND.LE.X(12)))THEN
                 CALL LIMITS(X(12),100000.D0,0.15D0)
                 IF(ISHEAR.EQ.1)THEN
                     VW = U20*LOG(X(12)/.15)/LOG(20/.15)
                   VW100 = U20*LOG((X(12)+100)/.15)/LOG(20/.15)
                    VWND = (VW100-VW)/100.D0
                    X(24) = VW
                 ENDIF
                 IF(ISHEAR.EQ.2)THEN
                    VWND = GRAD
                    X(24) = GRAD*(SHRALT-X(12))+WIND
                 ENDIF
          ENDIF

C*************************************************************C
C               Convert P and R to stability axis             C
C*************************************************************C

           PSTAB = X(7)*CALF+X(9)*SALF
           RSTAB = X(9)*CALF-X(7)*SALF

C*****************************************C
C                Control Limits           C
C*****************************************C

          IF(VTKT.LT.170.)THEN
                 FVTKT = (50.-VTKT)*.1/120.+1.0
                 CALL LIMITS(FVTKT,1.D0,.9D0)
          ELSE
                 FVTKT = (170.-VTKT)*.3/80.+0.9
                 CALL LIMITS(FVTKT,.9D0,.6D0)
          ENDIF
          CALL LIMITS(X(16),15.D0*FVTKT,-40.D0*FVTKT)
          CALL LIMITS(X(17),1.D0,0.D0)
          CALL LIMITS(X(18),40.D0*FVTKT,-40.D0*FVTKT)
          CALL LIMITS(X(19),30.8D0,-40.D0)
          CALL LIMITS(X(20),0.D0,-40.D0)
          CALL LIMITS(X(21),0.D0,-40.D0)
```

```
          CALL LIMITS(X(22),1.D0,0.D0)
          CALL LIMITS(X(23),54.D0,0.D0)

C***********************************************************C
C          Get aero-derivatives from look-up tables         C
C***********************************************************C

          CALL TABLES(X,VT,QBAR,PDIA,XGE,CALF,SALF,PSTAB,RSTAB,QS,
     &                B,CBAR)

C***********************************************************C
C             Dimensional Body Forces & Moments             C
C***********************************************************C

            FXB = QS*(CXS*CALF-CZS*SALF)
            FYB = QS*CYS
            FZB = QS*(CZS*CALF+CXS*SALF)
             LB = QSB*(CRS*CALF-CNS*SALF)
             NB = QSB*(CNS*CALF+CRS*SALF)

C***********************************************************C
C               Constants for State Equations               C
C***********************************************************C

           SPHI = DSIN(X(4))
           CPHI = DCOS(X(4))
            STH = DSIN(X(5))
            CTH = DCOS(X(5))
            TTH = DTAN(X(5))
           SPSI = DSIN(X(6))
           CPSI = DCOS(X(6))

C***************************************************C
C                    State Equations                C
C                                                   C
C                 Linear Acceleration               C
C***************************************************C

          XD(1) = FXB/MASS-X(3)*X(8)+X(2)*X(9)-G*STH-
     &       VWND*(X(3)*CTH-X(1)*STH)*CTH
          XD(2) = FYB/MASS-X(1)*X(9)+X(3)*X(7)+G*SPHI*CTH
          XD(3) = FZB/MASS-X(2)*X(7)+X(1)*X(8)+G*CPHI*CTH-
     &       VWND*(X(3)*CTH-X(1)*STH)*STH

C***************************************C
C               Orientation             C
C***************************************C
```

```
          XD(4) = X(7)+TTH*(X(8)*SPHI+X(9)*CPHI)
          XD(4) = 0.D0
          XD(5) = X(8)*CPHI-X(9)*SPHI
          XD(6) = (X(9)*CPHI+X(8)*SPHI)/CTH
          XD(6) = 0.D0

C**************************************C
C              Pitching Moment         C
C**************************************C
C
          ALDOT = (X(1)*XD(3)-X(3)*XD(1))/VFSQ
            CMS = CMWBT+DCMGE*XGE/(1+0.0024*X(12)*X(12))+DCMEL+
     &            DCMSPL+DCMSPR+CMQ*CBAR*(X(8)+DEPDAL*ALDOT/1.1)/
     &            (2.*VT)+DCMEO
            ZCG = (WLCG-WLREF)/(12.*CBAR)
            XCG = (CGPCNT-25)/100
           CMCG = CMS+(CXB*ZCG)-(CZB*XCG)
             MB = QSC*CMCG

C***********************************************C
C              Angular Accelerations            C
C***********************************************C

          XD(7) = X(8)*(X(7)*C2+X(9)*C1)+LB*C3+NB*C4
          XD(7) = 0.D0
          XD(8) = X(7)*X(9)*C5-(X(7)*X(7)-X(9)*X(9))*C6+MB*C7
          XD(9) = X(8)*(X(7)*C8-X(9)*C2)+LB*C4+NB*C10
          XD(9) = 0.D0

C*****************************************************C
C              Altitude and Rate of Climb             C
C*****************************************************C

         XD(12) = X(1)*STH-X(2)*CTH*SPHI-X(3)*CTH*CPHI
          X(11) = XD(12)

C************************************C
C              Position              C
C************************************C

          XD(13) =  X(1)*CTH*CPSI+X(2)*(SPHI*STH*CPSI-CPHI*SPSI)
     &             +X(3)*(CPHI*STH*CPSI+SPHI*SPSI)+X(24)
C         XD(14) =  X(1)*CTH*SPSI+X(2)*(SPHI*STH*SPSI+CPHI*CPSI)
C    &             +X(3)*(CPHI*STH*SPSI-SPHI*CPSI)
C         XD(15) = -X(1)*STH+X(2)*SPHI*CTH+X(3)*CPHI*CTH

C*********************************************C
C             Total Energy Sensor             C
C*********************************************C
```

```
      IF(TIME.EQ.0)THEN
               X(45)=X(11)*36/(60*57568.576)
               TIMOLD=-DT
               ICOUNT=1
      ENDIF
      IF(SNGL(TIME).EQ.SNGL(TIMOLD+DT))THEN
      TIMOLD = TIME
       KDELP = .000016506*VT*X(27)*DT
       PDELP = 77.26197128*(1-6.8756E-6*X(12))**
     &          (4.255916)*(-6.8756E-6*X(11))*DT
        DELP = KDELP-PDELP
      ENDIF
      XD(44) =  -2.4116D0*X(44)-2.4788D0*X(45)+DELP
      XD(45) = X(44)
       X(46) = 57568.57D0*X(45)
       X(27) = (X(1)*XD(1)+X(2)*XD(2)+X(3)*XD(3))/VT
      RETURN
      END

C=====================================================================C
C                                                                     C
C                            Autopilot                                C
C                                                                     C
C=====================================================================C
C
      SUBROUTINE APSAS(X,XD,DT,TIME,PAP,AT,GS,FC,ICOUNT,PITCMD,
     &     VCMD,DLCCMD,THLCMD,ELEVCMD,ALTCMD,GEARC,FLAPC,INRG)
      IMPLICIT REAL*8(A-H,K,O-Z)
      REAL*8 X(*),XD(*)
      COMMON/AERODAT/ADATA(700,43)
      DIMENSION ZZ(160),SPFLG(20)
      INTEGER PAP,AT,GS,FC,GSAT
      IF(ICOUNT.EQ.1)GOTO 1
      DO 2 II=1,160
2     ZZ(II)=0.D0
      PI=4.D0*DATAN(1.0D0)
      GLIDE=-DTAND(6.D0)
      TTHTL=X(17)
      THLCMD=TTHTL
      ELEVCMD=X(16)
      TELEV=X(16)
      THDOT=X(11)
      DLCCMD=X(20)
      UPRIG=X(20)
      TTES=X(11)*36/60
      SUM=0.D0
      ZIN=0.D0
      SAVE=X(5)
```

```
          IFLARE=0
          TRISTA=0.D0
          VOUT=0.D0
          TIMOLD=-DT

C*************************************************C
C            Get TE Rate Feedback Gains           C
C*************************************************C
C
          IF(INRG.EQ.1)THEN
                 WRITE(*,*)'EP,EI,ED,SP,SI,SD,TP,TI,TD:'
                 READ(*,*)EP,EI,ED,SP,SI,SD,TP,TI,TD
          ENDIF
          ICOUNT=1
1         VTSQ=X(26)**2
          VT=X(26)
          QBAR=.5*.0023768*VTSQ
          THDOT=-XD(13)*DTAND(6.D0)
          TTES=THDOT*36/60
          X(40)=XD(3)

C***************************************C
C               Rate Limits             C
C***************************************C

          FQBAR=(20.-QBAR)*.4/180.+1.0
          CALL LIMITS(FQBAR,1.D0,.6D0)

C*********************************************C
C              Authority Limits               C
C*********************************************C

          IF(X(26).LT.170.)THEN
                 FVTKT=(50.-X(26))*.1/120.+1.
                 CALL LIMITS(FVTKT,1.D0,.9D0)
          ELSE
                 FVTKT=(170.-X(26))*.3/180.+.9
                 CALL LIMITS(FVTKT,.9D0,.6D0)
          ENDIF

C*****************************************C
C               Aileron Actuator          C
C*****************************************C

C         CALL LIMITS(AILCMD,40.D0*FVTKT,-40.D0*FVTKT)
C         XD(18)=17.*(AILCMD-X(18))
C         CALL LIMITS(XD(18),43.D0*FQBAR,-43.D0*FQBAR)

C***************************************C
```

```
C               Rudder Actuator             C
C*******************************************C

C         IF(VTKT.LT.120.)CALL LIMITS(RUDCMD,7.D0,-10.D0)
C         IF((X(26).GE.120.).AND.(X(26).LT.138.))
C       &                 CALL LIMITS(RUDCMD,8.5D0,-12.D0)
C         CALL LIMITS(RUDCMD,30.8D0,-40.D0)
C         XD(19)=15.*(RUDCMD-X(19))
C         CALL LIMITS(XD(19),25.D0,-25.D0)

C*************************************C
C               Gear Actuator         C
C*************************************C

          XD(22)=30.*(GEARC-X(22))
          CALL LIMITS(XD(22),.333D0,-.333D0)

C*************************************C
C               Flap Actuator         C
C*************************************C

          XD(23)=4.*(FLAPC-X(23))
          CALL LIMITS(XD(23),4.D0,-4.D0)
          GSAT=0
          ALDOT=(X(1)*XD(3)-X(3)*XD(1))/(VT*60/36)**2
          IF((GS.EQ.0).OR.(X(13).LT.ALTCMD/(-.1051)).OR.
        &       (X(13).GE.0.))GOTO 10

C*************************************************************C
C            Glidepath Hold - Direct Lift Control             C
C*************************************************************C

          GSAT=1
          XD(39)=(TTES-X(46))
          DELTE=XD(39)-TEOLD
          TEOLD=XD(39)
          XD(30)=-10.D0*X(30)+(THLCMD-TTHTL)+.05D0*HERR+INRG*
        &         (SP*XD(39)+SI*X(39)+SD*DELTE)
          WASHOUT=10.D0*X(30)
          XD(31)=-4.2664D0*X(31)-6.0391D0*X(32)-2.8366D0*X(33)
        &       +WASHOUT
          XD(32)=X(31)
          XD(33)=X(32)
          SHAPOUT=-.7103D0*X(31)-3.1147D0*X(32)-2.8366D0*X(33)
        &       +WASHOUT
          DLCOUT=58.9D0*SHAPOUT
          CALL LIMITS(DLCOUT,20.D0,-20.D0)
          DLCCMD=DLCOUT+UPRIG
```

```
C*****************************************C
C               Spoiler Actuator          C
C*****************************************C

10        CALL LIMITS(DLCCMD,0.D0,-40.D0)
          CALL LIMITS(DLCCMD,0.D0,-40.D0)
          XD(20)=25.D0*(DLCCMD-X(20))
          XD(21)=25.D0*(DLCCMD-X(21))
          CALL LIMITS(XD(20),30.D0*FQBAR,-30.D0*FQBAR)
          CALL LIMITS(XD(21),30.D0*FQBAR,-30.D0*FQBAR)

          IF(PAP.EQ.0)GOTO 30

C*************************************C
C                Pitch SCAS           C
C*************************************C

          XD(34)=SUM
          IF(SUM.GT.0.D0)THEN
                IF(X(34).GE.5.0D0)XD(34)=0.D0
          ELSEIF(SUM.LT.0.D0)THEN
                 IF(X(34).LE.-5.D0)XD(34)=0.D0
          ENDIF
          XD(35)=TRISTA
          RATEOUT=DELE+X(35)
          X(35)=0.D0
          DELCOL=-.2618182D0*(X(16)-TELEV)
          DFORCE=-50.D0
          IF(DELCOL.GT.-5.0D0)DFORCE=5.64103*DELCOL-21.7949
          IF(DELCOL.GT.-1.1D0)DFORCE=26.66667*DELCOL+1.3333
          IF(DABS(DELCOL).LT.0.05D0)DFORCE=0.0D0
          IF(DELCOL.GT.0.05D0)DFORCE=26.66667*DELCOL-1.3333
          IF(DELCOL.GT.1.1D0)DFORCE=7.67857*DELCOL+19.55357
          IF(DELCOL.GE.6.7D0)DFORCE=71.0D0
          DEINV=.333D0*(X(16)-TELEV)
          THTD=-150.D0*X(8)
          IF(AT.EQ.0)GOTO 35

C*****************************************C
C               Speed Control             C
C*****************************************C

          VERR=VCMD-X(26)
          XD(36)=-0.12D0*VERR
          SPDP=X(36)-0.5D0*VERR+(ED*DELTE+EP*XD(39)+EI*X(39))*INRG
          VOUT=SPDP*PI/180.D0
          PITCMD=VOUT+SAVE
35        THERR=PITCMD-X(5)
          THTI=70.D0*THERR
```

```
      SUM=THTI-ZOUT
      THTP=150.D0*THERR
      PIOUT=THTP+X(34)
      DELEVIN=PIOUT+THTD+DEINV
      DELE=-3.0D0*DELEVIN
      ISWITCH=1
      IF((DFORCE.LT.-2.D0).OR.(DFORCE.GT.2.D0))ISWITCH=0
      XD(37)=(DELE-X(37))/.1D0
      DIFF=(DELE-X(37))/.1D0
      SIGN=DSQRT(DIFF**2)/(DIFF+1.D-10)
      HYSTIN=DELE-SIGN
      IHYST=-1
      IF((DELE.GE.-2.D0).AND.(DELE.LE.2.D0))IHYST=0
      IF(DELE.GT.2.D0)IHYST=1
      IHYST=IHYST*ISWITCH
      DO 55 JJ=160,2,-1
55    ZZ(JJ)=ZZ(JJ-1)
      ZZOUT=ZZ(160)
      ZZ(1)=DFLOAT(IHYST)
      DINVERT=-ZZOUT
      TRISTA=-1.5D0*DINVERT
      DELED=-1.5D0*DINVERT
      FBDEINV=-.333D0*DELED
      ZOUT=ZIN
      ZIN=FBDEINV

C*********************************************C
C              Elevator Actuator              C
C*********************************************C

30    ALPLIM=0.D0
      IF(X(10).GE.16.5D0)THEN
      ALPLIM=170*(16.5D0-X(10))-5000*ALDOT
      ENDIF
      STFDBK=.126D0*X(1)-.968D0*X(3)-15.D0*X(5)-100.D0*X(8)
      ELEVCMD=RATEOUT+TELEV-STFDBK-ALPLIM
      CALL LIMITS(ELEVCMD,15.D0+TELEV,-15.D0+TELEV)
      XD(16)=15.3*(ELEVCMD-X(16))
      CALL LIMITS(XD(16),32.D0*FQBAR,-32.D0*FQBAR)
      IF(GSAT.EQ.0)GOTO 90

C*****************************************************C
C            Glidepath Hold - Throttle              C
C*****************************************************C

      HDOTERR=X(11)-THDOT
      HDOTGN=-0.22916D0*HDOTERR
      HCMD=GLIDE*X(13)
      HERR=HCMD-X(12)
```

```
          DELH=HERR-HOLD
          HOLD=HERR
          X(38)=-HERR
          GSP=0.17D0*HERR+2.D0*DELH+INRG*(TP*XD(39)+TI*X(39)
         &          +TD*DELTE)
          THLCMD=GSP+HDOTGN+TTHTL

C*******************************************C
C               Throttle Actuator           C
C*******************************************C

90        CALL LIMITS(THLCMD,1.D0,0.D0)
          XD(28)=(2.04*(THLCMD-X(17))-1.2595*X(28)-2.036*
         &       X(29))/.15
          XD(29)=X(28)
          XD(17)=X(29)
          CALL LIMITS(XD(17),0.6354D0,-0.6354D0)

C*****************************************C
C               Evaluate Model            C
C*****************************************C

          CALL LD(TIME,X,XD)
          RETURN
          END
```

APPENDIX C

DATA ACQUISITION SOURCE CODE

```
$STORAGE:2
        IMPLICIT INTEGER (A-Z)
        LOGICAL CHECK
        INTEGER DATA(10000,2)
        CHARACTER*14 FNAME,WAIT
        REAL TIME,TTIME,RDATA(10000,2),RTIME2,SF,GAIN(2),BIAS
10      WRITE(*,'(A\)')' ENTER FILENAME (D:FILENAME.EXT)'
        READ(*,'(A14)')FNAME
20      OPEN(3,FILE=FNAME,STATUS='NEW')
        WRITE(*, '(A\)')' SPEED FACTOR '
        READ(*,*)SF
        WRITE(*,'(A\)')' TOTAL TIME '
        READ(*,*)TTIME
        RTIME2=TTIME*SF*1000
        WRITE(*,'(A\)')' DELAY TIME '
        READ(*,*)DELAY
        WRITE(*,'(A)')' ENTER:  1 FOR LOW  -> HIGH'
        WRITE(*,'(A)')'         2 FOR HIGH -> LOW'
        READ(*,*)IRUN
        IF(IRUN.EQ.1)TOLD=0
        IF(IRUN.EQ.2)TOLD=201
        WRITE(*,'(A\)')' RECORDED PEAK VOLTAGE = '
        READ(*,*)GAIN(1)
        GAIN(2)=GAIN(1)
        CALL INIT
        WRITE(*,'(A)')' START RECORDER <RETURN>'
        READ(*,'(A14)')WAIT
30      CALL ADIN(2,TCHECK)
        DELTA=TCHECK-TOLD
        TOLD=TCHECK
        IF(ABS(DELTA).GT.10)GOTO 40
        GOTO 30
40      WRITE(*,'(A)')' .'
        K=-1
        CALL TIMST(0)
50      K=K+1
        IF(DELAY.EQ.0)GOTO 55
        CALL PAUSE(DELAY)
55      CALL ADIN(1,DATA(K,1))
        CALL ADIN(3,DATA(K,2))
        CALL TIMRD(0,TIMEC)
        IF(TIMEC.GE.32000)THEN
```

```
              RTIME2=RTIME2-TIMEC
              CALL TIMST(0)
              TIMEC=0
            ENDIF
            IF(TIMEC.LT.RTIME2)GOTO 50
            WRITE(*,*)' FINISHED READING...RECORDER OFF'
            TTIME=TTIME/K
            K66=INT((K+1)/6)*6.
            WRITE(3,'(A10,3I5,A10)')'Y       ',K66,3,0,'(1P6E12.3)'
            DO 61 JJ=1,2
            DO 60 I=0,K
60          RDATA(I,JJ)=0.004883*GAIN(JJ)*DATA(I,JJ)
61          CONTINUE
            K6=INT((K+1)/6)-1
            DO 65 I=0,K6
            I6=6*I
65          WRITE(3,'(1P6E12.3)')TTIME*I6,TTIME*(I6+1),TTIME*
           &         (I6+2),TTIME*(I6+3),TTIME*(I6+4),TTIME*(I6+5)
            DO 71 JJ=1,2
              DO 70 I=0,K6
            I6=6*I
70          WRITE(3,'(1P6E12.3)')RDATA(I6,JJ),RDATA(I6+1,JJ),
           &         RDATA(I6+2,JJ),RDATA(I6+3,JJ),RDATA(I6+4,JJ),
           &         RDATA(I6+5,JJ)
71          CONTINUE
            ENDFILE3
            CALL DISI
            STOP
            END
```

VITA

Thomas Edward Anderson was born November 9, 1963 in Cleveland, Ohio, to Mr. and Mrs. Lawson S. Anderson. He attended Charles F. Brush High School and graduated from there in 1981. In 1985, Mr. Anderson received a Bachelor of Science degree in Aeronautical and Astronautical Engineering from Ohio State University. Following graduation, he began to pursue his studies toward the Master of Science degree. Mr. Anderson is a member of the American Institute of Aeronautics and Astronautics. National Society of Professional Engineers, Sigma Gamma Tau and is certified as an Engineer In Training by the State of Ohio. Mr. Anderson can be reached through his parents at 949 Learidge Road. Lyndhurst. Ohio 44124.