AN ADAPTIVE FLIGHT CONTROL SYSTEM FOR A DRONE AIRCRAFT

BY

LUIS EDUARDO ALVARADO

MASTER THESIS

AN ADAPTIVE FLIGHT CONTROL SYSTEM

FOR THE QF-106 DRONE AIRCRAFT

LUIS EDUARDO ALVARADO

Department of Electrical Engineering

APPROVED:

Dr. David Nemir, Chair

Dr. Michael Austin

Dr. Andrew Swift

Dean of the Graduate School

TO MY WIFE

MARISOL

AND MY SONS

LUIS AND RAFAEL

AN ADAPTIVE FLIGHT CONTROL SYSTEM FOR THE QF-106 DRONE AIRCRAFT

by

LUIS EDUARDO ALVARADO, B.S.

THESIS

Presented to the Faculty of the Graduate School of

The University of Texas at El Paso

in Partial Fulfillment

of the Requirements

for the degree of

MASTER OF SCIENCE

Department of Electrical Engineering

THE UNIVERSITY OF TEXAS AT EL PASO

December 1990

ACKNOWLEDGEMENTS

The author would like to express his deepest appreciation and gratitude to Dr. Nemir who, through his support and guidance made possible this paper. Appreciation is extended to Dr. Michael Austin and Dr. Andrew Swift, members of the graduate committee. Gratitude is also due to my employer, the IBM Corporation, for giving me time off to finish my graduate studies. Finally, I want to express my gratitude and appreciation to my Father and Mother, for their incentive and support throughout my education.

October 5, 1990.

ABSTRACT

The aerodynamic characteristics of an aircraft and hence the control characteristics vary widely with altitude, speed, and configuration of the aircraft. Ensuring the stability and good handling qualities of the aircraft in the whole flight envelope is a difficult and time consuming task because aircraft parameters are subject to drastic changes. There is, therefore, a need for a special class of a flight control system which can automatically compensate for these variations. Adaptive control is a very promising technique to solve these problems and to improve the performance deficiencies of conventional stability systems with preprogrammed control gains.

This thesis describes the design of an adaptive controller for the longitudinal axis of the QF-106 drone aircraft. This aircraft is presently used as a drone at White Sands Missile Range, New Mexico, for weapons testing. The approach used in this thesis is based on digital control and on a special stabilization concept which requires only four dynamic parameters of the aircraft to control its longitudinal axis. These four parameters can not be measured directly. Therefore, a recursive weighted least squares algorithm is used which delivers sufficiently fast and accurate on-line parameter estimates. The performance of the adaptive control

system has proven to be satisfactory using a six degree of freedom simulation of the QF-106 drone aircraft. Its performance was compared against the performance of a gain-scheduling flight controller for different flight conditions.

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

LIST OF FIGURES

LIST OF FIGURES (Continued)

LIST OF FIGURES (Continued)

CHAPTER 1

INTRODUCTION

1.1 Objective

The primary objective of this thesis is the conceptual design and performance evaluation of an adaptive flight control system for the longitudinal axis of the QF-106 drone aircraft. The control of the longitudinal motion was chosen because it is the main control axis of the aircraft, the parameter variations are strongly marked and the control problem is easy to understand. The design principle presented in this thesis can be used for the lateral motion in similar manner, but the problems are more involved as a result of the strong coupling of the roll and yaw axes of the aircraft.

1.2 Motivation

One of the first serious attempts to design a practical adaptive control system was for an autopilot in a high performance aircraft [1]. Unfortunately, the early enthusiasm over adaptive control did not make up for poor computing hardware and inadequate theory which led to a famous disaster in a flight test run. The recollection in the aerospace community of some of the initial failures of adaptive control to produce reliable results led to few new applications in this area in the sixties and early seventies. However, due to numerous significant advances over the past ten years in digital hardware and in the theory of adaptive control, today there is a renewed interest in the application of adaptive control to aerospace systems.

At White Sands Missile Range, New Mexico, the Drone Formation Control System (DFCS) uses the QF-106 aircraft as a remotely controlled drone for weapons testing. DFCS is a computer controlled microwave-multilateration system designed to automatically control multiple targets in close formation. The QF-106 drone is a supersonic aircraft capable of achieving speeds in excess of Mach 2.0 and altitudes of 50,000 ft above Mean Sea Level (MSL). The QF-106 drone carries an on-board digital flight control system and an aero-data computer to compute aero-data parameters such as Mach, airspeed and altitude. DFCS uses Distance Measuring Equipment (DME) to estimate with accuracy the position, velocity and altitude of the aircraft. The DFCS system controls the position and velocity of the aircraft by issuing the necessary autopilot commands (uplink) to the aircraft. For DFCS to successfully takeoff, land and fly up to six aircraft in close formation it is required that the on-board flight control system perform flawlessly. On many occasions, the present gain-scheduling controller will not perform as desired for many reasons; failures in the aero data sensors, (due to enemy fire), or changes in engine performance, or changes in drag, weight and mass center, (due to the dropping of external tanks or payloads).

The control of a highly maneuverable drone such as the QF-106 aircraft is complicated by the time variance in aerodynamic parameters such as dynamic pressure and roll rate. The present method of handling parameter variations is based on gain scheduling. In this scheme, sensors are used to give direct measurements of varying parameters, and then, these measurements are used to "schedule" the gains in the controller. The scheme is open loop in the sense that the resulting controller gains depend only on the sensor measurements of the varying parameters. In contrast, in an adaptive control scheme, the controller gains depend on measurements of the system inputs and outputs. Since the performance of the system is based on system inputs and outputs, there is a "closed loop" interconnecting system performance and controller gains. As a result, adaptive control has the potential of providing a much greater degree of robustness to uncertainties and unknown disturbances than gain scheduled control.

In particular, due to the open loop format, gain-scheduled control will be more sensitive to errors in sensor measurements or sensor failures than will adaptive control.

Another disadvantage of gain-scheduling controllers is that it takes considerable time and effort to derive the correct gain schedule for the whole flight envelope of the aircraft. Also, every time a new aircraft control system is developed, the gain scheduling controller has to be "tuned" for the new drone. This involves many man-hours of flying time and ,therefore, a lot of money. An adaptive controller will not require tuning and could be easily adapted to another aircraft of similar aerodynamic characteristics such as the F-100 and the F-4.

1.3 Methodology

The overall autopilot design effort consisted of four major tasks. The first task consisted of the implementation of a six degree of freedom (6-DOF) FORTRAN simulation program of the QF-106 aircraft on a personal computer for design and evaluation of the adaptive controller. This simulation algorithm provides aircraft rotational and translational motion with six degrees of freedom; an atmospheric representation, mathematical models of aerodynamic, gravity and engine forces and moments on the vehicle. It also simulates an adaptive and a conventional flight control system. Chapter 2 provides an overview of the simulator. Appendix A provides a detailed software description of the QF-106 simulation program.

The second task consisted of the design and evaluation of the system identification models for the longitudinal control axis of the aircraft (pitch and throttle axes). This task required the use of a linear analysis computer program named PC-MATLAB [15]. The name MATLAB stands for Matrix Laboratory. PC-MATLAB runs on IBM and other MS-DOS personal computers. It is an interactive program uniquely designed for numerical linear algebra and matrix computation that allow the user to develop complex linear models. Built-in system identification tools such as the ARX function were used extensively to define the correct model for the longitudinal axis of the aircraft. The ARX function uses the recursive least squares (RLS) approximation algorithm for parameter estimation. Once the correct model was identified, a FORTRAN version of the system identification model was integrated into the QF-106 simulation program. Chapter 2 provides an overview of the RLS system identification algorithm used. Chapter 3 describes the discrete model used to represent the dynamics of the longitudinal axis of the drone aircraft.

The third task was the design of the actual adaptive controller. Two different adaptive control algorithms were considered during the regulator design; a minimum variance control algorithm and a pole-placement controller. The objective of the minimum variance controller is to bring the predicted system output to a desired value in finite time so the control errors are minimum. The basic objective of the pole-placement controller is to control the position of the closed loop poles of the system, and by doing this, control the stability and transient response of the system. Early simulator results showed that the minimum variance control algorithm was too "active" to be used for aircraft control. That is, the amount of control effort required to minimize the control error was excessive and totally unacceptable for drone control. On the other hand, the pole-placement controller, although slightly noisier than a conventional controller, has an improved performance. Chapter 3 provides a detailed description of the adaptive pole-placement control algorithm.

The fourth and final task involved the performance evaluation of the adaptive flight control system. This was accomplished by comparing the simulator results of the adaptive flight control system against those of a conventional gain-scheduling controller. Our 6-DOF simulation capability was an excellent tool for control system design and verification. Chapter 4 summarizes the performance evaluation of the adaptive flight control system for different flight conditions.

CHAPTER 2

PRELIMINARIES

2.1 Adaptive Control

This section provides an overview of adaptive control with an emphasis on self tuning control. Its objective is to provide an overview of some of the basic control concepts that were used in the design of the adaptive flight control system for the Qf-106 drone aircraft. Section 2.1 is organized as follows: Subsection 2.1.1 lists several definitions of adaptive control collected from different authors. Subsection 2.1.2 considers different classifications of adaptive control schemes and introduces two main approaches: the model reference adaptive control systems and the self-tuning regulators. Subsection 2.1.3 discusses the most important concept in adaptive control, system identification. Finally, Subsection 2.1.4 discusses the pole-placement self tuning controllers. Emphasis is placed in the description of the pole-placement controller since this control scheme was the one selected for the design of the adaptive flight control system.

2.1.1 Definitions

Several definitions have been proposed for adaptive control. None of the definitions are general enough to include all of the work done by researchers in this area. There is no universally accepted definition at present. However, the following two definitions attempt to delineate the most important aspects of adaptive control.

(1) An adaptive control system must provide continuous information about the present state of the plant, that is, it must identify the process; it must compare present system performance with the desired or optimum performance and make a decision to adapt the system so as to tend toward optimum performance; and finally, it must initiate a proper modification so as to drive the system toward the optimum. These three functions are inherent in an adaptive system [3].

(2) An adaptive control system "learns" as time evolves, adapting its behavior in response to changes in the dynamics of the process under control, changes in the environment, or to reflect new knowledge of the system [5].

Although it is realized that there is, at present, no universally acceptable definition, in this thesis, it is presumed that adaptive control is a special type of nonlinear feedback control where the regulator parameters are continually being updated to reflect changes in system performance. Therefore, gain scheduling is not regarded as an adaptive controller since the regulator parameters are determined by a schedule without any feedback from the mmeasured performance.

2.1.2 Classification

There are two principal approaches to adaptive control, namely, model reference adaptive control (MRAC) and self tuning regulators (STR). MRAC was originally developed by Wittaker and his co-workers at MIT in 1958 [6] for designing adaptive autopilots. In MRAC the aim is to make the output of an unknown plant approach asymptotically the output of a given reference model, which is part of the control system, as shown in Figure 2.1. The control strategy can be thought of as consisting of two loops. The inner loop is an ordinary control loop consisting of the plant and regulator. The parameters of the regulator are adjusted by the outer loop in such a way that the error between the ideal (model) output and the actual output is minimized. The main problem of this control system is to determine the adaptation mechanism such that not only the error is brought to zero but also the resulting system is stable. If stability is not guaranteed, the adaptive system is not of practical utility.

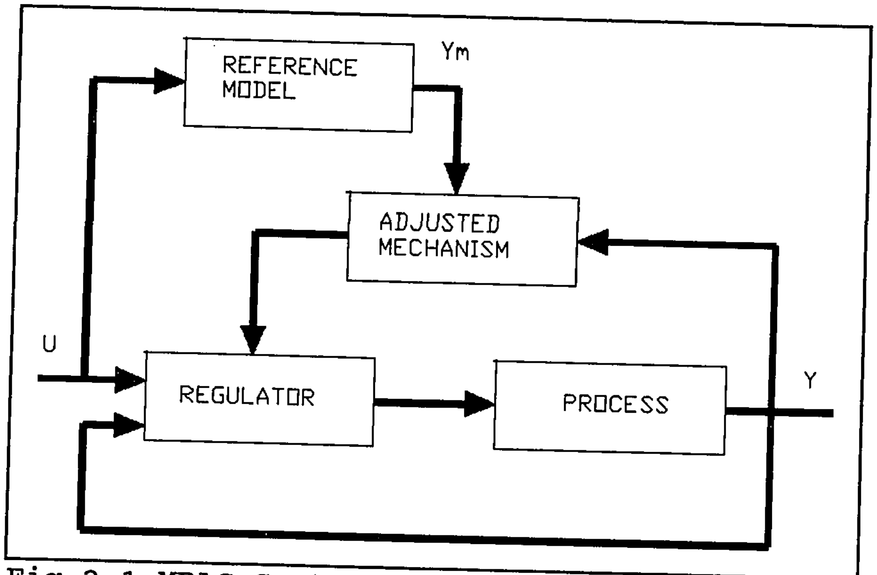


Fig 2.1 MRAC System

The self-tuning regulator is another important form of an adaptive system. It was originally introduced by Astrom and Wittenmark in 1973 [7]. The block diagram of such a system is shown in Figure 2.2. The adaptive regulator can be thought of as composed of two loops. The inner loop consists of the plant and an ordinary linear feedback regulator. The parameters of the regulator are adjusted by the outer loop, which is composed of a recursive parameter estimator and a design calculation. To obtain good estimates, it may also be necessary to introduce perturbation signals. This function is not shown in Figure 2.2 to keep the figure simple.

STR's have become very popular in recent years because of the versatility and the ease with which they can be implemented with microprocessors, In addition, the self-tuning adaptive controllers have the advantage of being able to adapt for any disturbance if designed well [3]. This is the main reason why this scheme was preferred over the MRAC technique when the adaptive flight control system for the QF-106 drone aircraft was designed.

The design of a self tuning control is conceptually simple. A natural approach is to combine a particular parameter estimation technique with any control law. In this thesis, parameter estimation and control laws are treated separately. Parameter estimation (system identification) is treated in Section 2.1.3. Section 2.1.4 describes, in more detail, self tuning regulators with special reference to the pole-placement approach.

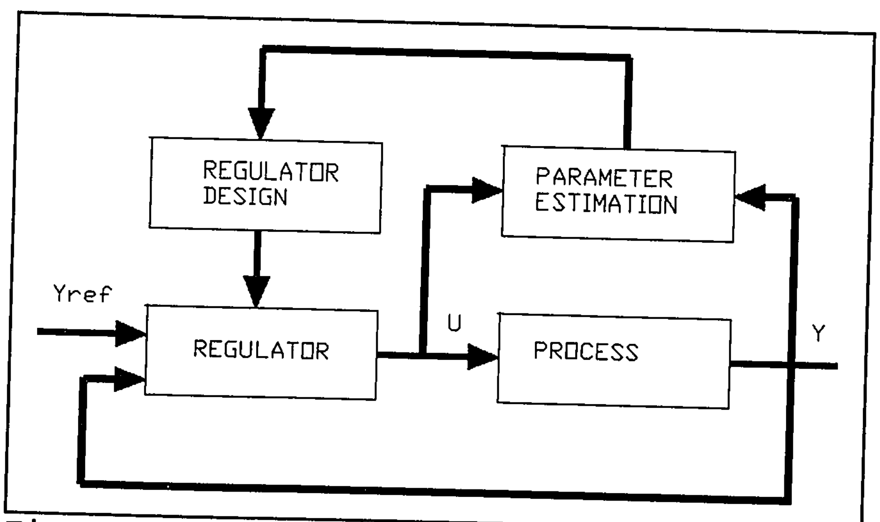


Fig 2.2 Self tuning regulator

2.1.3 System Identification

System identification is one of the most important concepts in adaptive control. It deals with the problem of building mathematical models based on the observed data from the system. Subsection 2.1.3.1 discusses what it is involved in the construction of a mathematical model for a given system based on the observed data. Subsection 2.1.3.2 provides a quick overview of the different system identification methods. It also discusses in detail, the recursive least squares (RLS) system identification method. It should be pointed out that RLS was the system identification scheme utilized in the design of the self-tuning flight control system for the QF-106 drone aircraft.

2.1.3.1 System Identification Procedure

The construction of a model from observed data involves three basic entities [8]:

(1) The data

(2) A set of candidate models

(3) A rule by which candidate models can be assessed using the data.

Let us comment briefly on each of these.

(1) The data record

The input-output data are sometimes recorded during a specifically designed identification experiment where the user may determine which signals to measure and when to measure them so that the data becomes maximally informative

(2) The set of models.

A set of candidate models is obtained by specifying within which collection of models we are going to look for a suitable one. This is the most important and the most difficult task of the system identification procedure. It is here that a priori knowledge and engineering intuition have to be combined with formal properties of the models. Sometimes a model with some unknown physical parameters is constructed from basic physical laws. In other cases standard linear models may be employed, without reference to the physical background.

(3) Determining the "best" model in the set

This is called model validation. The assessment of model quality is typically based on how the models perform when they attempt to reproduce the measured data.

The system identification procedure has a natural logic flow. First collect the data, then choose a model set, then pick the best model in this set. If the model first obtained does not pass the model validation tests, we must then go back and revise the various steps of the procedure. The System identification procedure is portrayed in Figure 2.3

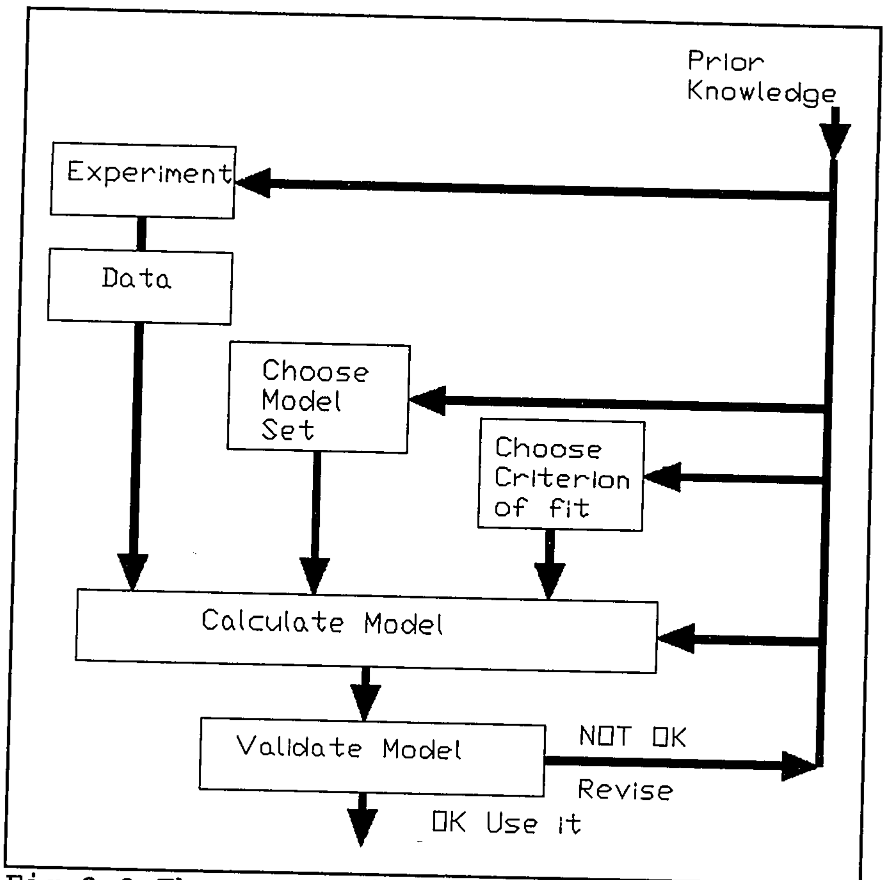


Fig 2.3 The system identification procedure

The system identification procedure picks the best model within the chosen model structured. The crucial question is then whether this "best" model is "good enough". This is the problem of model validation. The question has several aspects:

(1) Does the model agree sufficiently with the observed data?

(2) Is the model good enough for my purpose?

(3) Does the model describe the "true system"?

The following list describes some of the tools that are useful for discarding models, as well as for developing confidence in them [8].

(1) Model validation with respect to the purpose of modeling

There is always a certain purpose with the modeling. It might be that the model is required for regulator design (self tuning regulator), prediction or simulation. The ultimate validation is then to test whether the problem that motivated the modeling exercise can be solved using the obtained model. If a regulator based on the model gives satisfactory control, then the model is a "valid" one.

(2) Feasibility of Physical Parameters

For a model structure that is parameterized in terms of physical parameters, a natural and important validation is to confront the estimated values with what is reasonable from a priori knowledge. It is also a good practice to evaluate the sensitivity of the input-output behavior with respect to these parameters to check their practical identifiability. This should also be reflected by the estimated variances of the input, output and the estimated parameters.

(3) Model Reduction

If the model order can be reduced without affecting the input-output properties very much, then the original model was "unnecessarily complex".

(4) Parameter Confidence

Another procedure that checks whether the current model contains too many parameters is to compare the estimate with the corresponding estimated standard deviation. Using this information it is possible to develop confidence intervals for each parameter. That is, we can tell with a certain degree of confidence that the true value is to be found in a certain interval around the estimate. If the confidence interval is close to zero the parameter should be removed.

(5) Simulation

A commonly applied procedure that can be regarded as a test of the model's validity is to simulate the system with the actual input (not using measured output values) and compare the measured output with the simulated output. This is normally done by computing the residual, $\epsilon(t)$.

$$\epsilon(t) = y(t) - \hat{y}(t|\theta) \qquad (2.1)$$

where

$y(t)$ is the measured output and $\hat{y}(t|\theta)$ is the simulated output using the estimated parameters θ.

The resulting residual is then plotted and the validation of the model can be done, most of the time, by inspection. However, there are various statistical techniques, not covered in this overview, that can be used not only to evaluate the accuracy of the model but to check the properties of the resulting prediction errors such as independence of each other and of past data.

2.1.3.2 System Identification Methods

System identification methods can be classified as either non-parametric or parametric.

2.1.3.2.1 Non-Parametric

The non-parametric methods do not employ a finite dimensional parameter vector in the search for a best description of a model. Through direct techniques and without selecting a confined set of possible models, the non-parametric methods aim to determine the transfer function or the impulse response of an unknown system. Any linear time-invariant model can be fully described by its transfer function or by the corresponding impulse response [8]. Among the non-parametric methods we can list the following three:

(1) Transient Response Analysis

Information of the system is obtained by analyzing the system response to impulse and step functions. By definition, if a system is subject to an impulse input, its domain response (output) is the transfer function of the system. This simple idea is called the impulse response analysis. The step response analysis is normally used to determine some basic control related characteristics of the system such as: delay time, time constant, overshoot, and settling time.

(2) Frequency Response Analysis

A sinusoidal wave is input to the system to observe its response to different frequencies. The amplitude and phase shift of the output signal for a number of frequencies in the frequency band of interest are used to estimate the transfer function of the system.

(3) Spectral Analysis

This method involves the use of statistical techniques in determining the transfer function of linear systems.

2.1.3.2.2 Parametric

The parametric methods assume that the model of the system is parameterized. The are several well known parametric system identification methods such as the least squares method

(LS), the maximum-likelihood (MAL) method and the Bayesian maximum a posteriori (MAP) estimation method. In this thesis, only the least squares method is described. A variation of the LS method, the recursive LS identification method, was the algorithm selected for the design of the adaptive controller for the QF-106 drone aircraft. A detailed description of the other two methods can be found in reference [8]. The following subsections describe briefly two different presentations of the least-squares identification algorithm.

(1) The Batch Least-Squares system identification method

The least squares technique is a mathematical procedure by which a model can achieve a best fit to data using the minimum error squares criterion. The following derivation of the LS algorithm is similar to the LS derivation in reference [5].

Suppose that a given linear system can be represented by the following linear regression model:

$$y(t) = \phi(t)\,\theta \qquad t=1,2,3..... \qquad (2.2)$$

where the input vector is

$$\phi(t) = [\phi_1(t), \phi_2(t), \ldots, \phi_m(t)]$$

and the unknown parameter vector is represented by

$$\theta = [\theta_1, \theta_2, \ldots \theta_m]^T$$

and y(t) is the observation vector. Assume now that r measurements are processed, where r > m. Then we have a system of equations described by

$$Y = \phi^T \theta \tag{2.3}$$

or more explicitly

$$\begin{vmatrix} y(t_1) \\ \cdot \\ \cdot \\ y(t_r) \end{vmatrix} = \begin{vmatrix} \phi1(t_1) & \phi1(t_2) & \cdot\ \cdot & \phi1(t_r) \\ \cdot & & & \\ \cdot & & & \\ \phi m(t_1) & \phi m(t_2) & \cdot\ \cdot & \phi m(t_r) \end{vmatrix}^T \begin{vmatrix} \theta 1 \\ \cdot \\ \cdot \\ \theta m \end{vmatrix} . \tag{2.4}$$

The least squares principle requires that the estimate of Θ minimize the performance index

$$J = (Y - \phi^T \theta)^T (Y - \phi^T \theta) . \tag{2.5}$$

Taking the partial of J with respect to θ and equating it to 0, we find that

$$\frac{\partial J}{\partial \theta} = 2\,\theta\,(Y - \phi^T\,\theta) = 0 \qquad (2.6)$$

which yields the least squares parameter estimate

$$\theta = (\phi\,\phi^T)^{-1}\,\phi\,Y. \qquad (2.7)$$

(2) The recursive Least-Squares system identification method

The procedure presented above is commonly known as "batch processing least squares" because all the data is processed simultaneously. In many cases it is necessary, or useful, to have a model of the system available on-line while the system is in operation. The model should then be based on observations up to the current time. The identification techniques that comply with this requirement are called "recursive identification methods" since the measured input-output data are processed recursively (sequentially) as they become available [8]. The recursive LS algorithm is described below.

Assume again that the system is described by Equation (2.3) which is repeated next for clarity.

$$Y = \phi^T \theta \tag{2.8}$$

Over r measurement sets we wish to minimize

$$J_r = \sum_{t=1}^{r} [y(t) - \phi^T(t)\theta]^2 \tag{2.9}$$

Differentiating J_r with respect to θ and equating it to 0 we find that the least squares parameter estimate is given by

$$\theta_r = \left[\sum_{t=1}^{r} \phi(t)\phi^T(t)\right]^{-1} \sum_{t=1}^{r} \phi(t)y(t). \tag{2.10}$$

In order to cast the algorithm in sequential order it is required to define

$$Q_r = \sum_{t=1}^{r} \phi(t)\phi^T(t) = Q_{r-1} + \phi(r)\phi^T(r) \tag{2.11}$$

Using Equations (2.9) through (2.11) we determine the estimated parameters vector

$$\theta_r = \theta_{r-1} + Q_r^{-1}(r)\phi(r)\left[y(r) - \phi^T(r)\theta_{r-1}\right]. \qquad (2.12)$$

To avoid inverting Q_r^{-1} at each step, it is convenient to introduce:

$$P_r = Q_r^{-1}. \qquad (2.13)$$

Then using equation 2.11 we find that

$$P_r = \left[P_{r-1}^{-1} + \phi(r)\phi^T(r)\right]^{-1}. \qquad (2.14)$$

Applying the Matrix Inversion Lemma

$$[A+BCD]^{-1} = A^{-1} - A^{-1}B\,[DA^{-1}B + C^{-1}]^{-1}DA^{-1} \qquad (2.15)$$

to Equation (2.14) yields

$$P_r = P_{r-1} - P_{r-1}\phi(r)\left[\phi^T(r)P_{r-1}\phi(r) + 1\right]^{-1}\phi^T(r)P_{r-1}. \qquad (2.16)$$

The following table summarizes the logical sequence of the Recursive Least Squares algorithm.

STEP	PROCESS
0	Initialize r = 0
1	Initialize covariance matrix Pr = C * I where C is very large
2	Pickup next measurement set, and increment r by 1
3	Define the measurement vectors y(r) and ϕ(r).
4	Compute covariance matrix Pr using equation 2.16
5	Calculate estimated parameters vector θ using equations (2.12) and (2.13)
6	Are all measurements processed? IF YES - EXIT IF NO go back to step 2

The standard least squares algorithm gives the same weight to a new measurement as it does to old measurements when computing a parameter estimate. When system parameters are time variable, recent observations contain better information about the present parameter values than the old measurements. In such cases, a discounted performance index can be used which down-weights old data.

$$J_r = \sum_{t=1}^{r} \lambda^{r-t} [y(t) - \phi^T(t)\theta]^2 \qquad (2.17)$$

where λ is the forgetting factor and satisfies

$$0 < \lambda < 1.$$

It should be noted that when $\lambda = 1$, Equation (2.17) becomes the classical Least Squares performance index (2.9).

Note that Equation (2.17) is also called the weighted LS criterion. The weighted sequential LS algorithm is represented by Equations (2.18) and (2.19). Equation 2.18 is the same as Equation (2.12). When the forgetting factor $\lambda = 1$ Equation (2.19) becomes identical to the classical LS algorithm Equation (2.16).

$$\theta_r = \theta_{r-1} + Q_r^{-1}(r)\phi(r)\left[y(r) - \phi^T(r)\theta_{r-1}\right] \quad (2.18)$$

$$P_r = \frac{1}{\lambda}\left[P_{r-1} - P_{r-1}\phi(r)\left[\phi^T(r)P_{r-1}\phi(r) + \lambda\right]^{-1}\phi^T(r)P_{r-1}\right] \quad (2.19)$$

2.1.4 Self Tuning Regulators

Self tuning regulators represent an important class of adaptive controllers; they are easy to implement and are applicable to complex processes with a variety of characteristics involving unknown parameters, time-varying process dynamics, and stochastic disturbances. Whereas in MRAC the basic idea is to asymptotically drive the output of an unknown plant to that of a reference model, in self-tuning the basic procedure is to select a design of known plant structure and apply it to the unknown plant using recursively estimated values of plant parameters.

2.1.4.1 Digital Pole Placement Controller

The pole placement method involves the placement of the closed-loop poles at prescribed locations which define a required transient response, while leaving their zeros in their open-loop positions. Robustness is the essential advantage of this method. The pole-placement controller can be applied to non-minimum-phase plants, and this is a distinct advantage, as non-minimum-phase behavior is encountered in many practical applications [3].

The pole-placement controller has a transfer function of $P(q^{-1})/L(q^{-1})$ and its general structure is portrayed in Figure 2.4.

In Figure 2.4, the system is described by a single-input-single output auto-regressive exogenous (ARX) type time series of the form

$$A(q^{-1})y(t)=B(q^{-1})\,u(t) + w(t) \tag{2.20}$$

where

$$A(q^{-1}) = 1 - a_1q^{-1} - \ldots - a_nq^{-1} \tag{2.21}$$

$$B(q^{-1}) = b_0\,q^{-d} + b_1q^{-d-1} + \ldots + b_mq^{-d-m}. \tag{2.22}$$

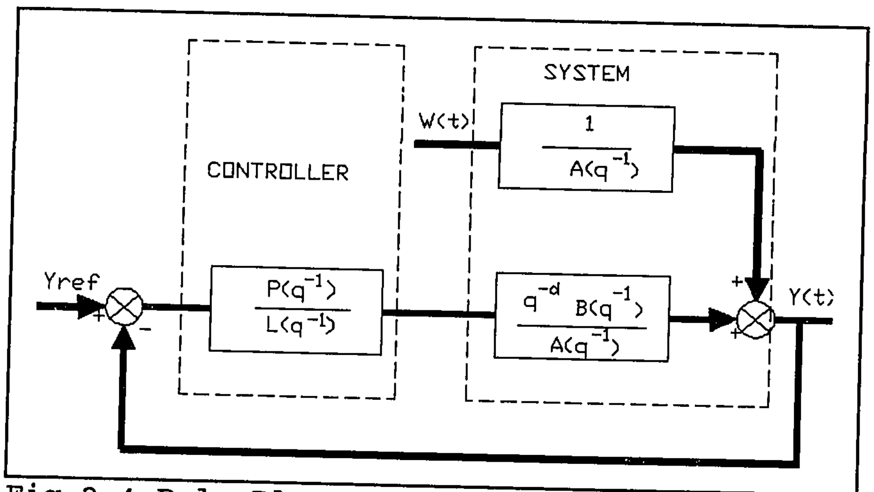


Fig 2.4 Pole-Placement STR

This type of model we denote as an ARX(n,m,d) model since it has n auto-regressive terms, m exogenous terms, and a delay d, where n, m, and d are positive integers. The terms y(t) and w(t) are respectively, the output and white noise time sequences. The desired output is yref(t). The coefficients of $A(q^{-1})$ and $B(q^{-1})$ represent the parameters estimated by the on-line system identification algorithm. The q^{-1} is the delay operator (i.e. $q^{-1}\, y(t) = y(t-1)$).

From Figure 2.4 it can also be seen that the closed loop transfer function is given by

$$y(t) = \frac{P(q^{-1})\; q^{-d}\, B(q^{-1})}{L(q^{-1})\; A(q^{-1}) + q^{-1}\, B(q^{-1})\; P(q^{-1})}\; y_{ref}(t). \qquad (2.23)$$

From Equation (2.23) it is inferred that the polynomials $P(q^{-1})$ and $L(q^{-1})$ may be chosen so as to give the closed-loop system a desired dynamical response (characteristic polynomial). This is done by choosing $L(q^{-1})$ and $P(q^{-1})$ so that

$$L(q^{-1})\,A(q^{-1}) + B(q^{-1})\,P(q^{-1}) = A^*(q^{-1}), \tag{2.24}$$

where $A^*(q^{-1})$ defines the desired dynamical response. Once the polynomials $L(q^{-1})$ and $P(q^{-1})$ have been determined, the control signal $u(t)$ can be calculated from the following feedback control law:

$$L(q^{-1})\,u(t) = -P(q^{-1})\,[y_{ref}(t) - y(t)]. \tag{2.25}$$

2.1.4.2 PI and PID Pole Placement Controllers

A drawback of direct digital design of the pole placement controller is that it is normally difficult to translate the controller to a PI or PID (Proportional, Integral and Derivative) structure. A way to overcome this problem is to design the controller in the continuous time. This requires that a continuous time model of the process is obtained first. This technique is recommended only for processes with very high sampling rates and without dead-time compensation problems [2]. The following example shows the design of a PI pole-placement controller.

Assume that the process is described by the following digital model:

$$y(k+1) = a_1\, y(k) + b_1\, u(k)\,. \tag{2.26}$$

Using the Euler approximation with sampling period T

$$y'(k) = \frac{y(k+1) - y(k)}{T}\,. \tag{2.27}$$

The representation of the model, Equation (2.26), in the continuous time is

$$y'(k)\, T + (1 - a_1)\, y(k) - b_1\, u(k) = 0\,. \tag{2.28}$$

By calculating the Laplace Transform of Equation (2.28) we determine the transfer function of the system.

$$Gr(s) = \frac{b_1}{T\, s + (1 - a_1)} \tag{2.29}$$

Provided that the transfer function of the PI controller is given by

$$Gp(s) = K_2 + \frac{K_1}{s}, \tag{2.30}$$

the transfer function of the close loop system is

$$G(s) = \frac{Gp(s)\ Gr(s)}{1 + Gp(s)\ Gr(s)} \tag{2.31}$$

Therefore, the characteristic equation of the close loop system is

$$T\ s^2 + (1 - a_1 + k_2\ b_1)\ s + K\ b_1 = 0. \tag{2.32}$$

Now suppose that the desired closed loop poles are characterized by their relative damping ζ and their natural frequency Wn. The desired characteristic equation then becomes

$$s^2 + 2\ \xi\ Wn\ s + Wn^2 = 0. \tag{2.33}$$

Making the coefficients of these two characteristic equations equal gives two equations for determining K_1 and K_2,

$$K_1 = \frac{T\,Wn}{b_1}, \qquad K_2 = \frac{a_1 - 1 + 2\,\zeta\,Wn\,T}{b_1}. \tag{2.34}$$

2.2 Aircraft Aerodynamics and Control

The purpose of this section is to provide the reader with an understanding of some of the basic concepts in aerodynamics and flight control systems that are referenced in later sections. Since the intent of this section is only to provide an overview in these areas, most of the equations are given without derivation. Also furnished, is a detailed description of a conventional flight control system that was used in this thesis as a BENCHMARK to compare and evaluate the performance of the proposed adaptive flight control system for the QF-106 drone aircraft.

2.2.1 Basic Concepts in Aerodynamics

2.2.1.1 Forces, Moments and Velocities

The aerodynamic forces and moments depend on the velocity of the aircraft relative to the air mass [9]. More accurately, they depend on the dynamic pressure term

$$Q_B = \rho \frac{V_r^2}{2}, \tag{2.35}$$

where ρ is the air density and V_r is the total speed of the aircraft. Since the dimension of dynamic pressure is force per unit area, each dynamic force can be expressed as a product of aerodynamic pressure times a dimensionless aerodynamic coefficient times a reference area, (usually the frontal area of the aircraft). The moments are the product of the aerodynamic forces acting on the aircraft times a reference length. Normally different reference lengths are used to compute the different aircraft moments (pitching, rolling and yawing). The aerodynamic coefficients in turn are functions of the vehicle velocity (linear and angular) components, and functions of the deflection of the control surfaces (i.e. elevators, ailerons, and rudder) from their position of reference.

Since the natural axes for resolving the aerodynamic forces and moments are moving (rotating and accelerating), it is necessary to put the equations of motion into a moving coordinate system. It is customary to project all the acting forces and moments onto the body axis of the aircraft. Figure 2.5 illustrates the body fix (rotating) axis XYZ of the aircraft with respect to an Earth fixed, (inertial), coordinate system, X_i, Y_i, and Z_i. The Earth fixed (inertial) to body transformation matrix is provided below without derivation.

$$T_{IB} = \begin{vmatrix} T_{11} & T_{12} & T_{13} \\ T_{21} & T_{22} & T_{23} \\ T_{31} & T_{32} & T_{33} \end{vmatrix} \qquad (2.36)$$

where

$$T_{11} = \sin(\Psi)\ \cos(\Theta)$$

$$T_{12} = \cos(\Psi)\ \cos(\Theta)$$

$$T_{13} = \sin(\Theta)$$

$$T_{21} = \sin(\Theta)\ \sin(\Phi)\ \sin(\Psi) + \cos(\Theta)\ \cos(\Psi)$$

$$T_{22} = \sin(\Theta)\ \sin(\Phi)\ \cos(\Psi) - \cos(\Phi)\ \sin(\Psi)$$

$$T_{23} = -\cos(\Theta)\ \sin(\Phi)$$

$$T_{31} = \sin(\Theta)\ \cos(\Phi)\ \sin(\Psi) - \sin(\Phi)\ \cos(\Psi)$$

$$T_{32} = \sin(\Theta)\ \cos(\Phi)\ \cos(\Psi) + \sin(\Phi)\ \sin(\Psi)$$

$$T_{33} = -\cos(\Theta)\ \cos(\Phi).$$

The angles Θ, Ψ, Φ describe the orientation of the body axis relative to a translated Earth fixed coordinate system X_L, Y_L, Z_L. These angles are frequently referred to as the Euler angles. Θ is referred to as the pitch angle, Ψ is called the heading (or yaw) angle and, Φ is named the bank (or roll) angle. Figures 2.6, 2.7 and 2.8 are side, top and rear views of the QF-106 drone aircraft. These pictures clearly depict the Euler angles Θ, Ψ, and Φ.

Figures 2.6 and 2.7 also show the stability axis {Xs,Ys,Zs} that normally is used to show the lift (L_f), drag (D_f), and side (S_f) forces acting on the aircraft. The stability axis is obtained from the body axes {X,Y,Z} by rotating Y=Ys over an angle α until X coincides with the velocity vector V_r. The angle α is normally called the angle of attack. The angle ß is conventionally called the side-slip angle. As illustrated in Figures 2.6 and 2.7, α is the angle that the velocity vector makes with respect to the X axis in the pitch direction and ß is the angle it makes with respect to the X axis in the yaw direction.

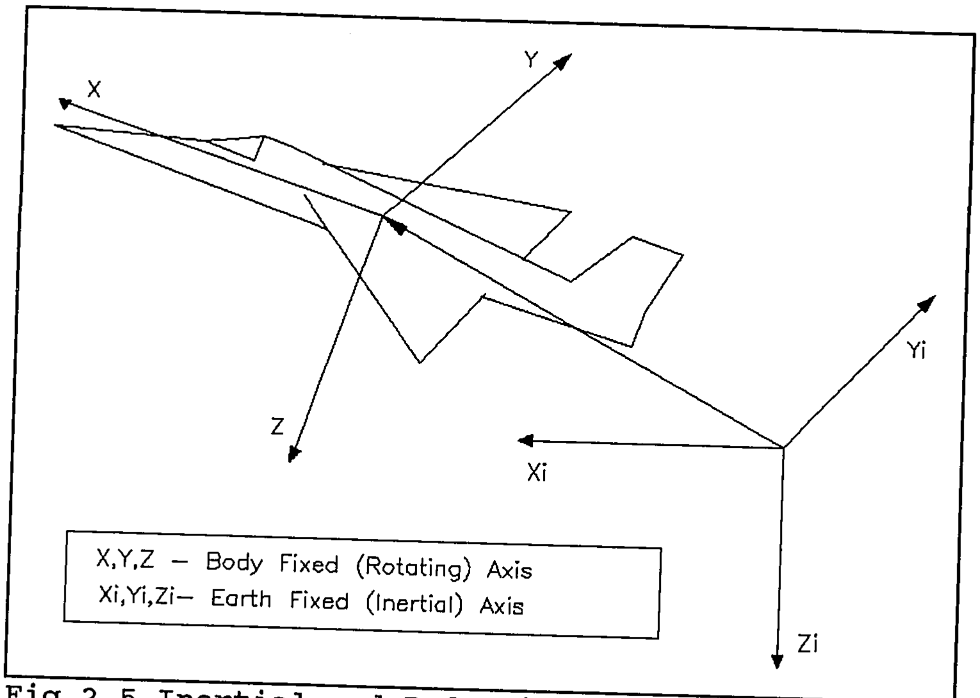


Fig 2.5 Inertial and Body Fixed coordinate system

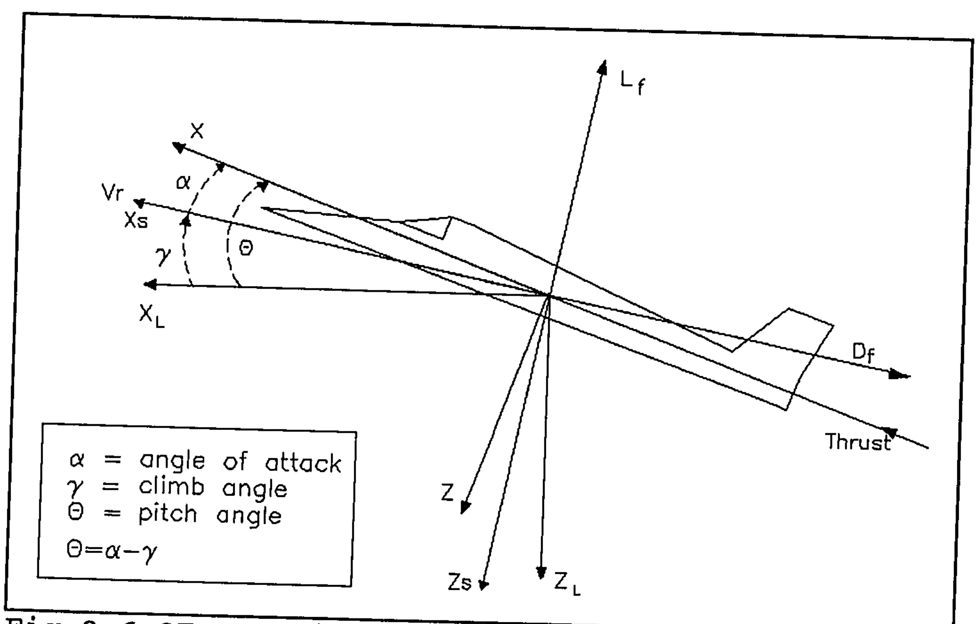


Fig 2.6 QF-106 side view

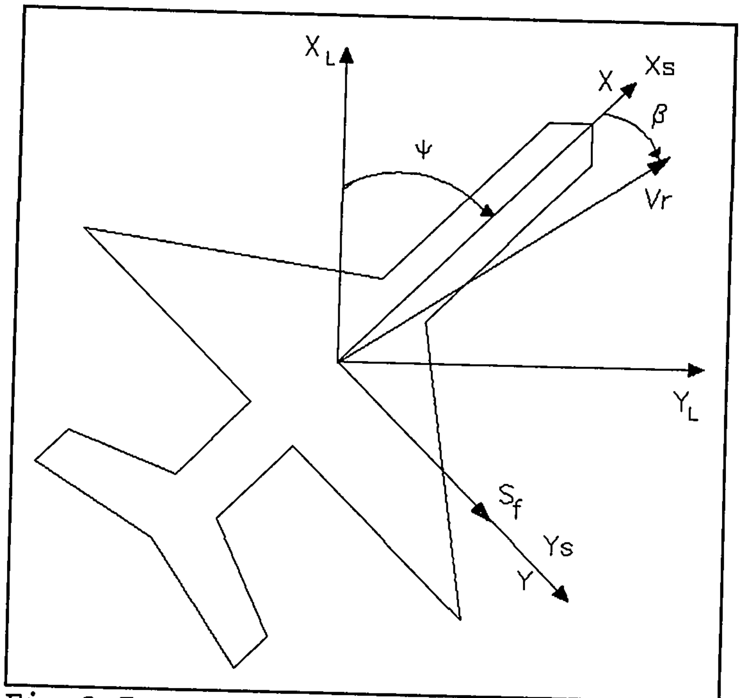


Fig 2.7 QF-106 top view

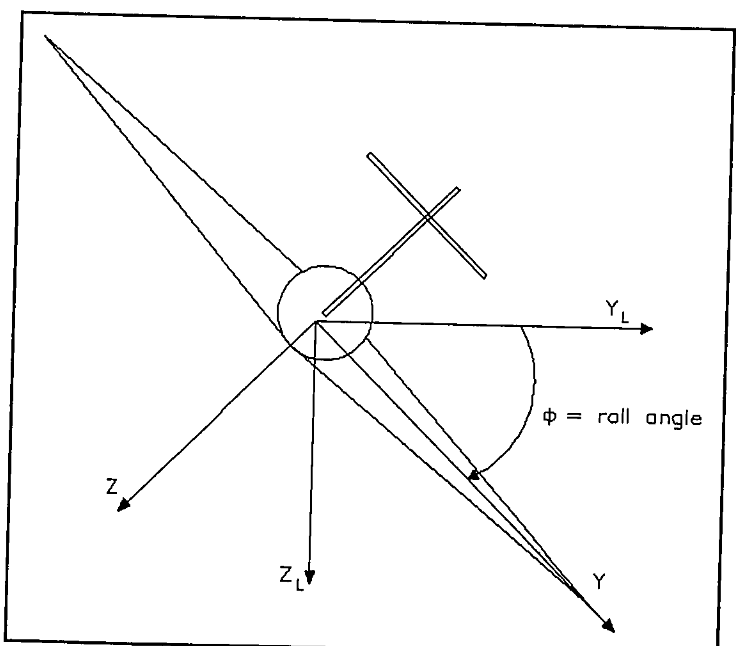


Fig 2.8 QF-106 rear view

The following summarizes ,in vector form, the dynamic forces and moments acting on the aircraft as well as the velocity components that describe the motion of the aircraft. These forces, moments, and velocity components are graphically depicted in Figure 2.9.

(1) Forces

$$F_a = iF_{ax} + jF_{ay} + kF_{az} \tag{2.37}$$

$$F_t = iF_{tx} + jF_{ty} + kF_{tz} \tag{2.38}$$

$$g = ig_x + jg_y + kg_z \tag{2.39}$$

where

F_a and F_t are the total aerodynamic and thrust force vectors. F_{ax}, F_{ay} and F_{az} are the projections of the drag, side-force and lift respectively. F_{tx}, F_{ty}, and F_{tz} are the thrust force components and g_x, g_y, and g_z are the components of gravitational acceleration.

(2) Moments

$$M_a = iL_a + jM_a + kN_a \tag{2.40}$$

$$M_t = iL_t + jM_t + kN_t \tag{2.41}$$

where

M_a - total aerodynamic moment vector

M_t - total thrust moment vector

L_a, L_t - aerodynamic and thrust rolling moments

M_a, M_t - aerodynamic and thrust pitching moments

N_a, N_t - aerodynamic and thrust yawing moments

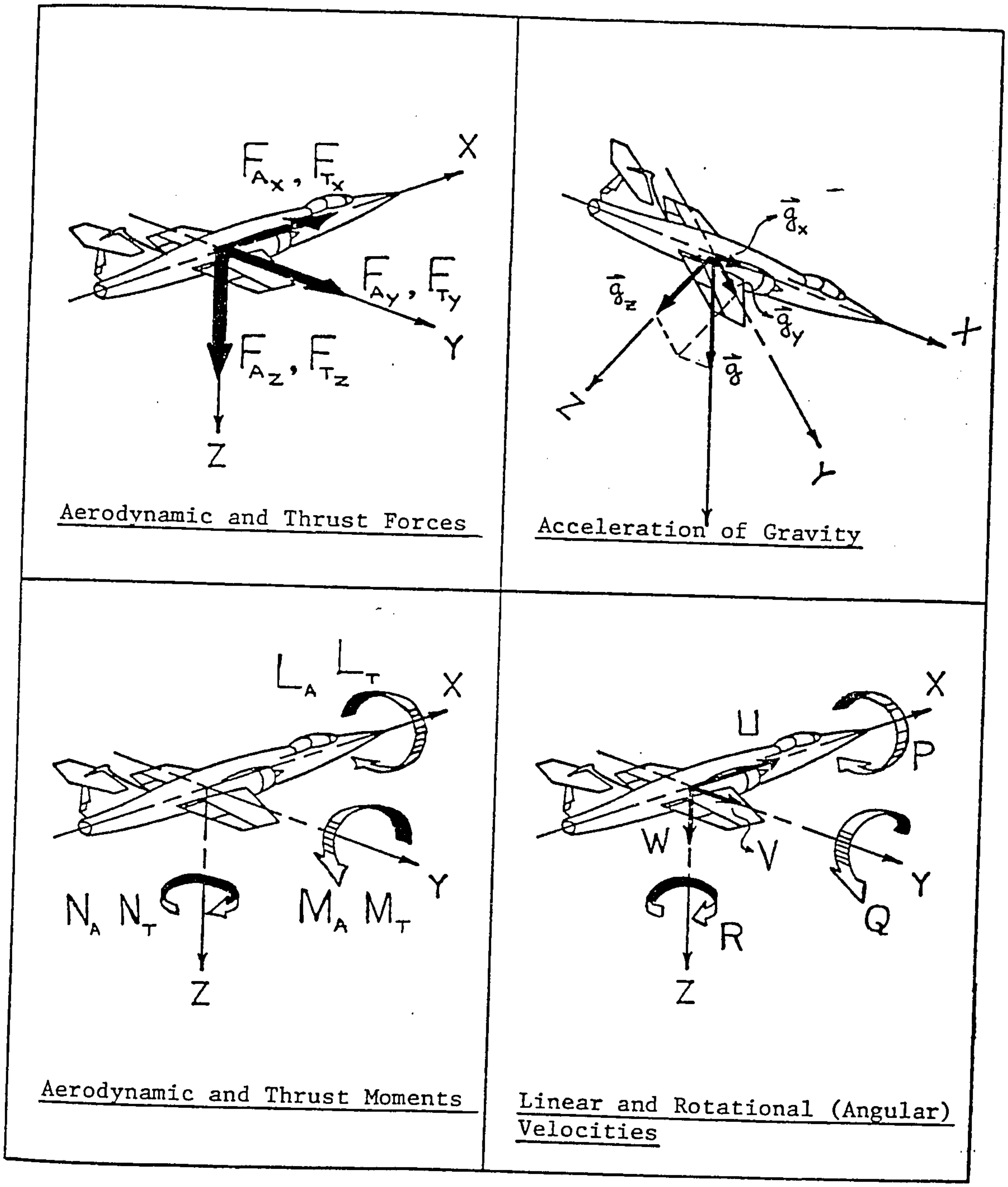
(3) Velocities

$$\omega = iP + jQ + kR \tag{2.42}$$

where P, Q and R are the angular velocity components: roll rate, pitch rate and yaw rate, respectively

$$V_r = iU + jV + kW \tag{2.43}$$

where U, V and W are the linear velocity components: forward velocity, side velocity and downward velocity, respectively.



Note: Positive sense is in the direction of the arrows

Figure 2.9 Aircraft Forces and Moments

2.2.1.2 Equations of Motion

2.2.1.2.1 General Equations of Motion

The derivation of the aircraft equations of motion is somewhat complicated and will not be covered in this thesis. A complete derivation of the equations of motion can be found in [9]. For the record, the aircraft equations of motion are provided below.

(1) Linear and angular acceleration equations

$$U' = V \cdot R - W \cdot Q - g \sin(\Theta) + [F_{ax} + F_{tx}]/m$$

$$V' = U \cdot R - W \cdot P + g \sin(\Phi) \cos(\Theta) + [F_{ay} + F_{ty}]/m \qquad (2.44)$$

$$W' = U \cdot R - W \cdot P + g \sin(\Phi) \cos(\Theta) + [F_{ay} + F_{ty}]/m$$

$$P' = \frac{(I_{zz}\, AA + I_{xz}\, CC)}{I_{xx}\, I_{zz} - I_{xz}^2}$$

$$Q' = \frac{BB}{I_{yy}} \qquad (2.45)$$

$$R' = \frac{(I_{xx}\, CC + I_{xz}\, AA)}{I_{xx}\, I_{zz} - I_{xz}^2}$$

where

$$AA = L_a + L_t - Q * R \,(\, Izz - Ixx \,) + P * Q * Ixz$$

$$BB = M_a + M_t - P * R \,(\, Ixx - Izz \,) + (\, R^2 + P^2 \,) * Ixz$$

$$CC = N_a + N_t - P * Q\ (\ Iyy - Ixx\ ) + Q * R * Ixz$$

$$DD = Ixx * Izz - Ixz * Ixz$$

(2) Euler angle rate equations

$$\Phi' = P + [Q \cdot \sin(\Phi) + R \cdot \cos(\Phi)] \tan(\Theta)$$

$$\Theta' = Q \cdot \cos(\Phi) - R \cdot \sin(\Phi) \qquad (2.46)$$

$$\Psi' = Q \cdot \sin(\Phi) + R \cdot \cos(\Phi) \cdot \sec(\Theta)$$

(3) Inertial velocity equations

$$\begin{vmatrix} X_i' \\ Y_i' \\ Z_i' \end{vmatrix} = [T_{IB}]^T \begin{vmatrix} U \\ V \\ W \end{vmatrix} \qquad (2.47)$$

where

X_i', Y_i', Z_i' -- inertial velocity components

Ixx, Iyy, Izz -- moments of inertia about the X,Y,Z axes

Ixz -- product of inertia about the y axis

Θ, Φ and Ψ -- Euler angles

U', V', W' -- forward, side and downward acceleration

U, V, W -- forward, side and downward velocities

P -- body roll rate

Q -- body pitch rate

R -- body yaw rate

P' -- body roll acceleration

Q' -- body pitch acceleration

R' -- body yaw acceleration

m -- mass of the aircraft

g -- Earth gravity constant

Equations (2.44), (2.45) and (2.46) form nine simultaneous nonlinear differential equations in terms of the nine motion variables U, V, W, P, Q, R, Ψ, Φ, and Θ. Equation (2.47) represents three simultaneous equations in terms of the variables U, V and W. Therefore the total dynamics of the aircraft are represented by 12 nonlinear differential equations.

Finally, to find the complete trajectory of the airplane, it is necessary to invoke Equation (2.47). This equation computes the inertial velocities from the projections of the vehicle velocity vector onto the body X,Y and Z axes. The transformation matrix T_{IB} is defined in Equation (2.36). The position of the drone aircraft in the inertial axis X_i, Y_i, and Z_i, is obtained by integrating the set of Equations (2.47)

2.2.1.2.2 Linearized Equations of Motion

For the purpose of control system design, the aircraft dynamics are frequently linearized about some operating condition or "flight regime", in which it is assumed that the aircraft velocity and attitude are constant. The control surfaces and engine thrust are set or "trimmed", to these conditions and the control system is designed to maintain them, i.e., to force any perturbations from these conditions to zero.

Since it is assumed that the forward speed is constant, then for small perturbations the angle of attack, α, and the angle of side slip, β, can be used as state variables instead of the vertical velocity, W, and the side velocity V, respectively.

In studying small perturbations from trim conditions it is customary to separate the longitudinal motion from the lateral motion. In many cases the lateral and longitudinal dynamics are only lightly coupled, and the control system can be designed for each channel without regard to the other.

The function of most control system designs is to regulate small motions rather than to control the absolute position of the aircraft. Thus the inertial position is not included in the state equations. This leaves nine equations, four in the longitudinal channel and five in the lateral channel. To differentiate from the general equations of motion, the state variables of the linearized (perturbed) equations of motion are represented by lower case symbols (i.e pitch rate Q is represented by q, roll rate P is represented by p...). The linearized equations of motion can be written in matrix form as follows:

(1) Longitudinal dynamics

$$\begin{vmatrix} u' \\ \alpha' \\ q' \\ \theta' \end{vmatrix} = \begin{vmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ 0 & 0 & 1 & 0 \end{vmatrix} \begin{vmatrix} u \\ \alpha \\ q \\ \theta \end{vmatrix} + \begin{vmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \\ b_{31} & b_{32} \\ 0 & 0 \end{vmatrix} \begin{vmatrix} \delta e \\ \delta t \end{vmatrix} \qquad (2.48)$$

(2) Lateral dynamics

$$\begin{vmatrix} v' \\ p' \\ \phi' \\ r' \\ \psi' \end{vmatrix} = \begin{vmatrix} c_{11} & c_{12} & c_{13} & c_{14} & c_{15} \\ c_{21} & c_{22} & c_{23} & c_{24} & c_{25} \\ c_{31} & c_{32} & c_{33} & c_{34} & c_{35} \\ c_{41} & c_{42} & c_{43} & c_{44} & c_{45} \end{vmatrix} \begin{vmatrix} v \\ p \\ \phi \\ r \\ \psi \end{vmatrix} + \begin{vmatrix} d_{11} & d_{12} \\ d_{21} & d_{22} \\ d_{31} & d_{32} \\ d_{41} & d_{42} \\ d_{51} & d_{52} \end{vmatrix} \begin{vmatrix} \delta a \\ \delta r \end{vmatrix} \qquad (2.49)$$

The coefficients a_{ij} and b_{ij} are called the stability and control derivatives, respectively, of the longitudinal axis. The coefficients c_{ij} and d_{ij} are called the stability and control derivatives of the lateral axis. These coefficients vary with the altitude and speed of the aircraft. Finally, the δe, δt, δa, δr are the elevator, throttle, aileron and rudder control surface deflections.

2.2.1.3 Aircraft Control Surfaces

Aircraft control is normally achieved using movable control surfaces and the throttle. As mentioned in the preceding section, the movement of the control surfaces affect directly the aerodynamic forces and moments acting on the aircraft. The throttle position determines the engine thrust. The aircraft longitudinal motion is typically controlled by the control surface called elevator and by the throttle. The elevator is used to control the pitch attitude of the aircraft and the throttle is used to control its speed. The lateral motion, roll and yaw, are controlled by the ailerons and the rudder respectively.

2.2.2 Automatic Flight Control Systems (AFCS)

The primary functions of an automatic flight control system (AFCS) are to improve air-frame stability and aircraft maneuvering, allow automatic navigation, including automatic takeoff and landing . Figure 2.10 illustrates the general structure of an AFCS.

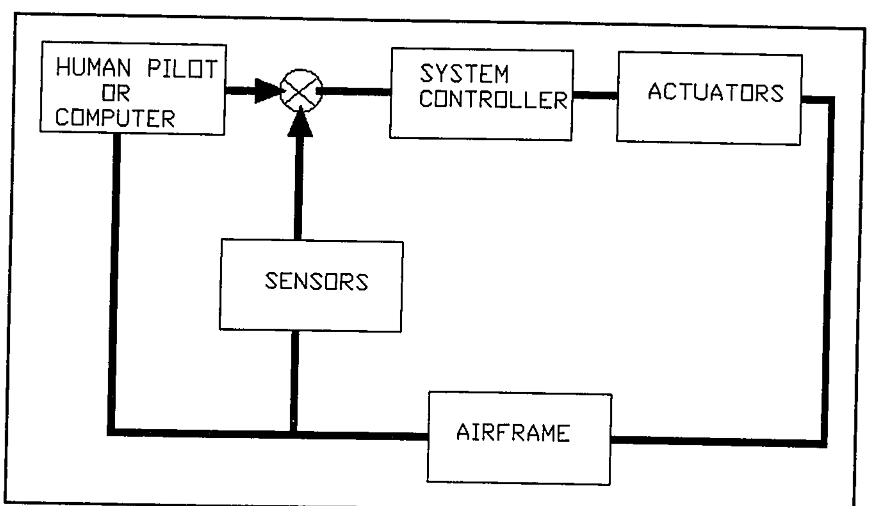


Fig 2.10 AFCS General Structure

The basic components of an AFCS are the following:

(1) The Airframe

This is the actual body of the aircraft. In our case this will be the QF-106 drone aircraft.

(2) The Human Pilot

The pilot is the one that normally closes the control loop. He provides the inputs to the on-board flight control system based on the observations obtained from the sensor readings and from his own sensors (brain, eyes, and ears). For drone control, the pilot can be replaced by a (human) drone controller which can control the drone by remote control. He uses downlink telemetry sensor data and navigation data (displayed on the controller's console) for drone control. The drone controller can also be replaced by a digital ground computer that generates the inputs to the on-board system controller based on pre-programmed responses to some of the sensors readings.

(3) The Sensors

Their function is to "sense" airframe output quantities such as velocity, attitude angles, and body rates. Among the sensors we can include the gyroscopes, accelerometers, airspeed detectors, altitude sensors and air data computers.

(4) The System Controller

This is the nerve system of the AFCS. Its functions are: to accept signals from the sensors and from the pilot (or from the drone controller), to affect signal phase lead or lag as required for desired system response (control law), and to amplify the signals to a power level sufficient for operation of the servo actuators. Subsection 2.2.2.2 briefly describes the present system controller used by the QF-106 drone aircraft. This is the system controller that was used as a BENCHMARK to test the proposed adaptive controller for the QF-106 drone aircraft.

(5) Actuators

These are the electrical-mechanical devices used to move the aircraft control surfaces.

2.3 QF-106 Simulation Model

The objective of this section is to provide an overview of the 6-DOF simulation program developed for the design and evaluation of an adaptive system controller for the QF-106 drone aircraft. The detailed software description of the simulation program is presented in Appendix A. The computer listing of the simulation program is included in appendix B.

The original QF-106 vehicle model was developed in 1978 by the company Industrieanlagen-Betriebsgesellshaft [12]. The IBM corporation implemented this simulation program into the Drone Formation Control System (DFCS) in 1987 [13]. The simulator is presently in operation at White Sands Missile Range, New Mexico. The vehicle simulation used in this thesis is only a subset of the complete program. The QF-106 simulation program was substantially modified to reduce its size and optimize its operation. It should be noted that the original program resides on an IBM 4381 computer. The simulation program used in this thesis resides on a 386-Personal Computer.

Figure 2.11 shows an overview block diagram of the QF-106 6-DOF simulation. The dotted block indicates that the aircraft sensors were not simulated. The frequency bandwidth of these sensors is in the order of 150 radians/second. Their exclusion did not affect the accuracy of the simulation.

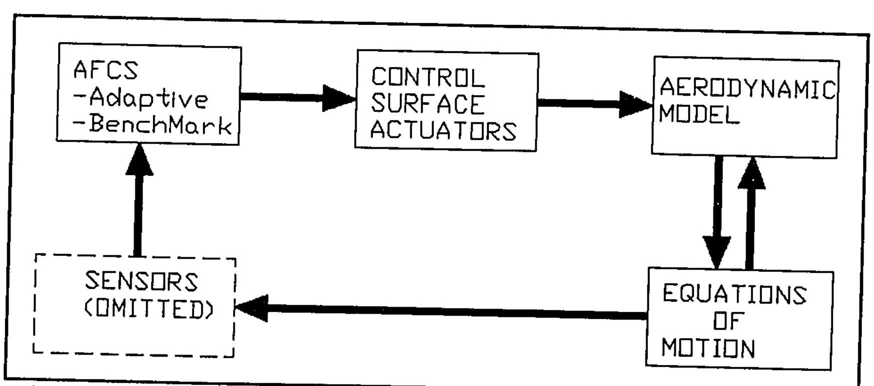


Figure 2.11 QF-106 simulation overview

The simulation includes an aerodynamic model, airframe 6-

DOF equations of motion, an AFCS and control surface actuator models. Two different digital autopilots are simulated: one is the self tuning controller, the other one is a classical feedback controller which is used as a benchmark to evaluate the performance of the adaptive system controller.

The conventional system controller designed for the QF106 drone aircraft was originally designed by the Honeywell corporation [10] in 1988 as part of the effort to drone the F-106 aircraft. The system controller was designed to respond to inputs generated by DFCS. That is, it was design so DFCS could replace the human pilot. The following describes some of the control laws implemented. It should be noted that the actual QF-106 system controller is very complex and only the control laws that were actually simulated and used for system performance evaluation are listed below.

(1) Pitch Attitude Hold System (PAH).

Its function is to provide both, an autopilot attitude hold capability and to respond to attitude changes due to outer loop commands (altitude hold loop) or to pitch steering commands from DFCS. Figure 2.12 depicts the control law block diagram of the PAH system. Pitch attitude damping is provided by pitch attitude rate and normal acceleration feedbacks. Three gain schedulers, dependent on airspeed, mach and altitude are included in the forward path.

(2) Altitude Hold on Pitch System (AH)

Its function is to provide both an autopilot altitude hold capability and to respond to altitude trim commands from DFCS. As depicted in Figure 2.13, this control law includes feedback terms proportional to altitude error, altitude rate and the integral of altitude error. The inner control loop is the PAH system.

(3) Speed Hold on Throttle (SHOT)

The purpose of this control system is to maintain airspeed using throttle control. Normally this control mode is enabled in up-and-away flight when altitude hold via pitch is being maintained. Figure 2.14 shows the SHOT loop coupled with the pitch attitude control system. Pitch feedback is used to enhance the SHOT system response. More throttle is needed when the aircraft is pitching up than when it is pitching down.

(4). Speed Hold on Elevator (SHE)

The objective of this control loop is to control the speed of the aircraft using pitch attitude control. The aircraft will pitch up to reduce its airspeed and will pitch down to increase it. Figure 2.15 illustrates the control law block diagram of the SHE loop. The inner loop of this system is the PAH system.

(5). Roll Attitude Hold System (RAH)

This control system was designed to respond to proportional roll attitude commands. The roll control loop consists basically of a roll attitude feedback path summed with roll rate feedback. Fixed gain values are used in both attitude and rate feedback paths. The roll axis gain scheduler, located in the main forward path is a function of airspeed and it is depicted on the block diagram of the RAH system, in Figure 2.16. Yaw rate damping is provided to stabilize the control loop. An integrator is included in the forward path to null any roll trim biases.

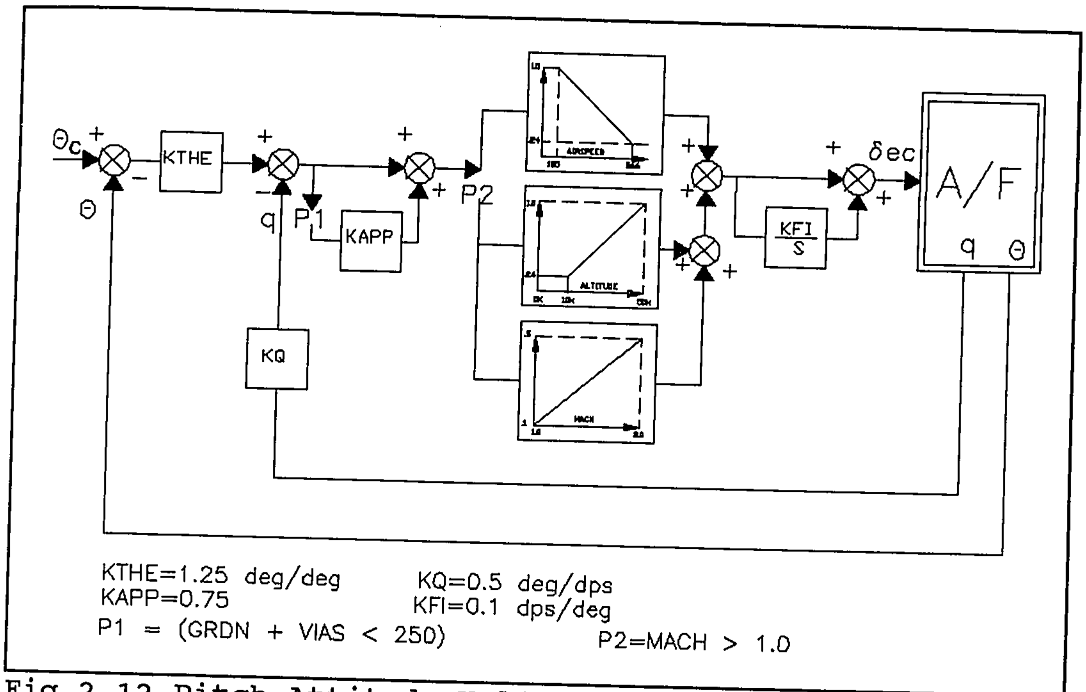


Fig 2.12 Pitch Attitude Hold system block diagram

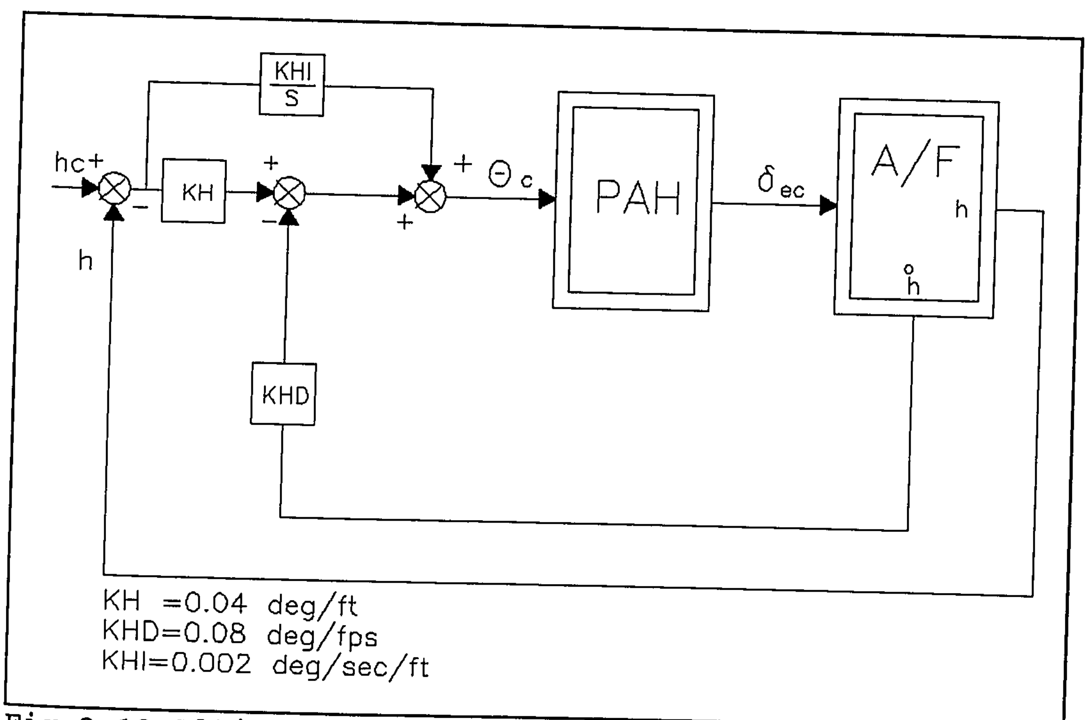


Fig 2.13 Altitude Hold system block diagram

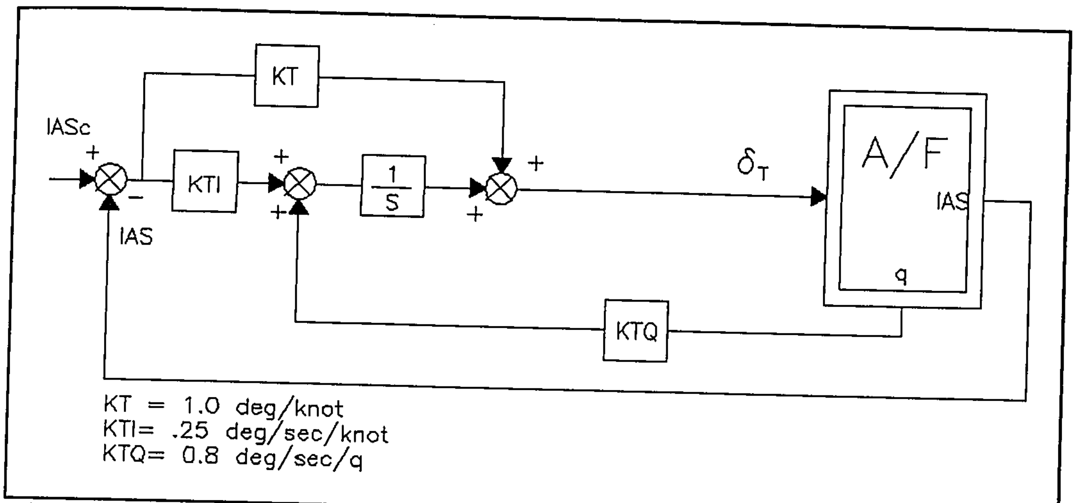


Fig 2.14 Speed Hold on Throttle system block diagram

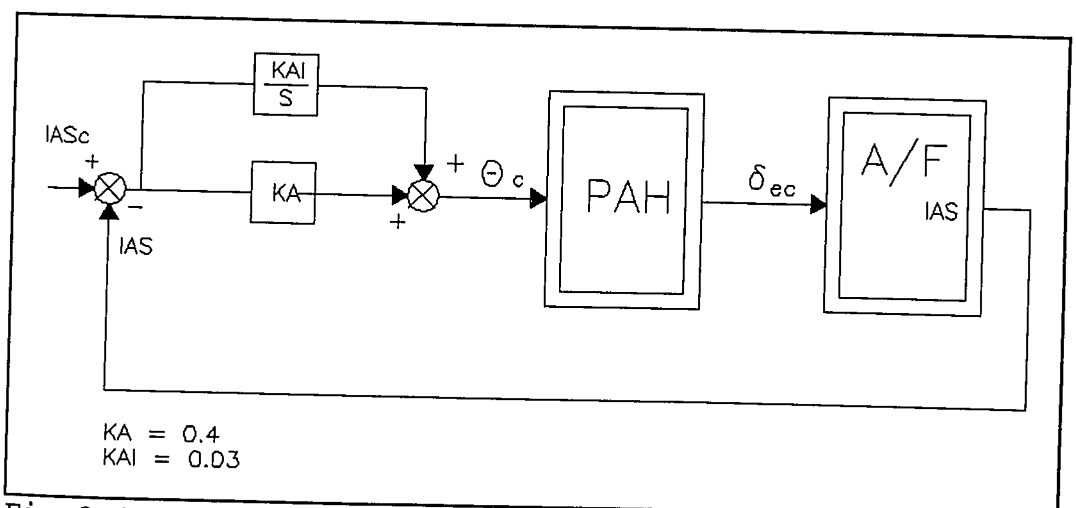


Fig 2.15 Speed Hold on Elevator system block diagram

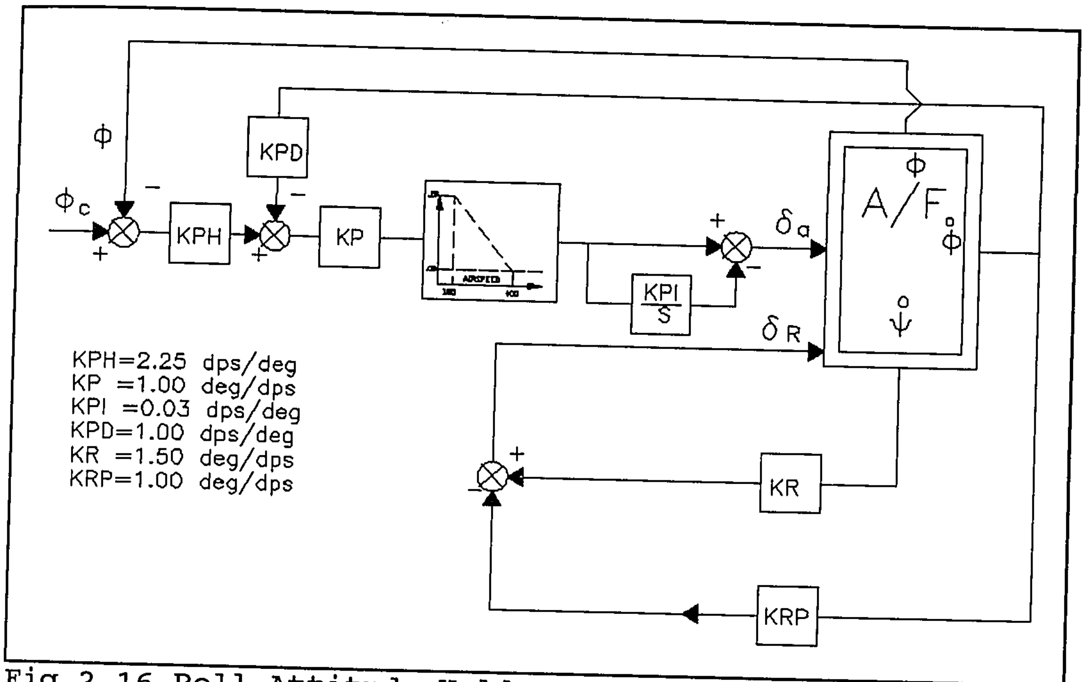


Fig 2.16 Roll Attitude Hold system block diagram

CHAPTER 3

ADAPTIVE FLIGHT CONTROL SYSTEM DESIGN

This chapter describes the design of an adaptive flight control system for the longitudinal axis of the QF-106 drone aircraft. A total of four different adaptive control laws were designed. These control laws are listed below and their functions are described in Subsection 2.3 of this thesis.

(1)	Adaptive Pitch Attitude Hold	(APAH)
(2)	Adaptive Speed Hold on Throttle	(ASHOT)
(3)	Adaptive Altitude Hold on Pitch	(AAH)
(4)	Adaptive Speed Hold on Elevator	(ASHE)

It should be noted that the AAH and ASHE control laws are adaptive only in the sense that they use the APAH system for altitude and speed control.

The above adaptive control laws were used in combination with a conventional Roll Attitude Hold (RAH) control law for lateral control. Together, they provide the necessary control capabilities for straight and level flight, moderate climbs and dives and low bank angle turns. By low bank angle turns we allude to roll angles of less than 75 degrees. This limitation of the above control laws is the result of a simplification made in the system identification process of the longitudinal dynamics of the aircraft. It was assumed that the vertical (pitch) and lateral (roll) dynamics of the aircraft are decoupled when in reality they are not. At high bank angles, the cross coupling effects of the roll and pitch control axes are considerable and need to be considered for control. Therefore, a more sophisticated system identification process and control laws would be required for high G maneuvers.

3.1 Control Problem

As stated before the objective of the proposed adaptive flight control system is to control the longitudinal axis of the QF-106 drone aircraft. This includes the control of the pitch angle and the control of the velocity of the aircraft.

As mentioned in Subsection 2.2.1.2 the dynamics of the longitudinal axis can be represented by a system of four linear difference equations. The state space representation of these equations is as follows:

$$\mathbf{X'} = \mathbf{AX} + \mathbf{BU} \tag{3.1}$$

where the vector

$\mathbf{X} = [\ u,\ \alpha,\ q,\ \theta\ ]^T$

includes the following variables:

u = forward velocity (true airspeed)

α = angle of attack

q = pitch rate

θ = pitch angle.

The vector **U** consists of two variables

δe = elevator deflection

δt = throttle position.

The elevator, δe, is used to control the pitch attitude of the aircraft and the throttle, δt, is commonly used to control its speed during level flight. The control surfaces and the throttle are moved by actuators (servos). Because of the fact that these actuators normally operate at very high frequencies (20 radians/sec) the dynamics of the actuators was not taken into consideration in the design of the adaptive controller. However, the dynamics of the actuators was included in the simulation program that was used to evaluate the performance of the adaptive controller.

The matrices A and B of the state Equation (3.1) are defined as follows:

$$A = \begin{vmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ 0 & 0 & 1 & 0 \end{vmatrix} \qquad B = \begin{vmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \\ b_{31} & b_{32} \\ 0 & 0 \end{vmatrix} \qquad (3.2)$$

where the coefficients a_{ij} and b_{ij} are the stability and control derivatives that represent the dynamics of the longitudinal axis of the aircraft. It should be noted that these derivatives vary with time depending on the altitude and speed (Mach) of the aircraft.

The following subsections describe the design of the control laws listed at the beginning of the section.

3.1.1 Adaptive Pitch Attitude Hold System

First, we consider a simplified mathematical model for the pitch axis of the aircraft. The simplified model can be obtained from Equation (3.1) by assuming that the forward speed variations of the aircraft are independent of the pitch attitude angle and that the pitch attitude angle can be obtained by simply integrating pitch attitude rate q. The validity of this model has been proven on numerous occasions by several authors [9]. This model is usually known as the short period approximation for the pitch dynamics of the aircraft.

$$\begin{vmatrix} q' \\ \alpha' \end{vmatrix} = \begin{vmatrix} a_{22} & a_{23} \\ a_{32} & a_{33} \end{vmatrix} \begin{vmatrix} q \\ \alpha \end{vmatrix} + \begin{vmatrix} b_{21} \\ b_{31} \end{vmatrix} \delta e. \tag{3.3}$$

By reason of reliability and costs, the use of an angle of attack sensor should be avoided in stability systems. A normal accelerometer can be used instead of the angle of attack sensor. It is well known that normal acceleration measured at an appropriate location of the aircraft is a very good alternative for the angle of attack signal. Note that the relation between the normal acceleration and the angle of attack is approximately given by

$$a_z = Z_\alpha \cdot \alpha \tag{3.5}$$

where the normal force due to the elevator deflection, δe, is neglected. The coefficient Z_α is defined as the dimensional variation of the Z-force with angle of attack.

For practical realizations we are interested in a discrete time version of the control system. Equation (3.6) is a discrete time version of Equation (3.1) where we include the preceding relationship between angle of attack and normal acceleration given in Equation (3.5).

$$\begin{vmatrix} q(k+1) \\ a_z(k+1) \end{vmatrix} = \begin{vmatrix} f_{11} & f_{12} \\ f_{21} & f_{22} \end{vmatrix} \begin{vmatrix} q(k) \\ a_z(k) \end{vmatrix} + \begin{vmatrix} h_1 \\ h_2 \end{vmatrix} \delta e(k) . \tag{3.6}$$

The design strategy consists of first developing an adaptive pitch rate control system and then using this system to control the pitch attitude of the aircraft. Equation (3.7) shows the control law used for pitch rate control.

$$\delta e(k) = - K_{qi}\,\theta(k) + K_q\,q(k) + K_{az}\,a_z. \tag{3.7}$$

Equation (3.8) is obtained from Equations (3.6) and (3.7).

$$\begin{aligned} q(k+1) = {} & f_{11}\,q(k) + f_{12}\,a_z(k) + h_1\,K_{az}\,a_z(k) \\ & - h_1\,K_\theta\,\theta(k) + h_1\,K_q\,q(k). \end{aligned} \tag{3.8}$$

Assuming that, q and a_z, are de-coupled we obtain Equations (3.9) and (3.10).

$$q(k+1) = (f_{11} + h_1\,K_q)\,q(k) - h_1\,K_\theta\,\theta(k), \tag{3.9}$$

$$(f_{12} + h_1\,K_{az})\,a_z(k) = 0. \tag{3.10}$$

The control gain K_{az} is calculated from Equation (3.10)

$$K_{az} = - \frac{f_{12}}{h_1} \tag{3.11}$$

Using the Euler approximation and the sampling period T we express:

$$q'(k) = \frac{q(k+1) - q(k)}{T}, \qquad (3.12)$$

where q'(k) = is the time derivative of q.

Using Equation (3.12), the equivalent expression of Equation (3.9) in the time domain is

$$q'(t)T + q(t)[1 - f_{11} - h_1K_q] - h_1\theta(t)K_\theta = 0$$
$$\text{where} \quad \theta(t) = \int(q(t)dt. \qquad (3.13)$$

The characteristic equation of Equation (3.9) is obtained by taking the Laplace Transform of Equation (3.13)

$$s^2\, T\, q(s) + s\, q(s)\ (1 - f_{11} - h_1\, K_q) - h_1\, q(s)\ K_{qi} = 0. \qquad (3.14)$$

Equation (3.14) represents a second order system of the form

$$s^2 + 2\,\zeta\, Wn\, s + Wn^2 = 0 \qquad (3.15)$$

where

ζ = damping factor

Wn = natural frequency.

The damping factor and natural frequency selected were

$\zeta = 0.707$

Wn = 5 rad/sec for subsonic speeds

= 7 rad/sec for supersonic speeds

and by comparing Equation (3.14) with the desired characteristic Equation (3.15) the control parameters can be calculated as a function of ζ and Wn.

$$K_{qi} = -\frac{T\,Wn^2}{h_1}, \qquad K_q = \frac{1 - f_{11} - 2\,\zeta\,Wn\,T}{h_1}. \tag{3.16}$$

Therefore, to control pitch rate, q, the identification process must calculate the parameters

$$f_{11}, \quad f_{12}, \quad h_1$$

of the equation

$$q(k+1) = f_{11}\,q(k) + f_{12}\,a_z(k) + h_1\,\delta e(k). \tag{3.17}$$

By examining Equation (3.17) we can observe that this equation assumes that the pitch acceleration of the aircraft is zero for zero elevator angle, δe, and zero angle of attack (remember that a_z is proportional to angle of attack, Equation 3.5). This is obviously not true since all the airframes are built to have an inherent pitch moment, M_0, that varies with the aerodynamic pressure. This pitch moment can be theoretically calculated by the equation

$$M_0 = CM_0 \, Q_B \, S_w \, c_B \tag{3.18}$$

where

CM_0 = pitch moment coefficient for $\alpha = \delta e = 0$

Q_B = aerodynamic pressure

S_w = wing surface area

c_B = mean aerodynamic chord.

To take into account this moment, the bias term, b_1, was added to Equation (3.17).

$$q(k+1) = f_{11}\, q(k) + f_{12}\, a_z(k) + h_1\, \delta e(k) + b1. \tag{3.19}$$

Finally, pitch attitude control is obtained using the control law, Equation (3.20)

$$q_c(k) = K_\theta\, (\theta_c - \theta) \tag{3.20}$$

where

q_c = desired pitch rate

K_Θ = constant pitch gain

Θ_c = desired pitch attitude

Θ = pitch attitude.

Figure 3.1 shows the block diagram of the APAH system for the QF-106 drone aircraft.

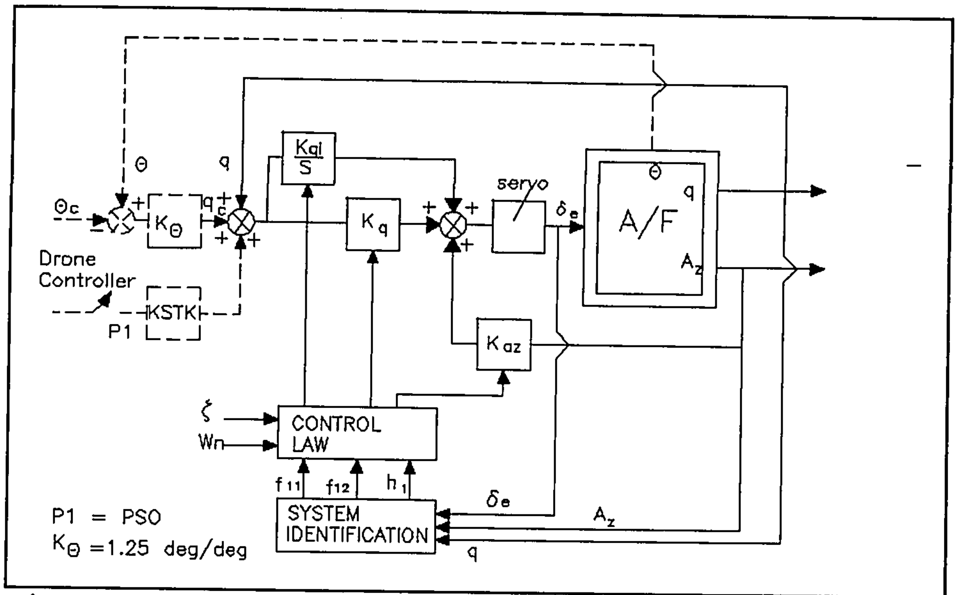


Fig 3.1 APAH system block diagram

In this figure, the box labeled A/F represents the QF-106 drone airframe. This figure also shows that the drone controller has the capability to control the pitch attitude of the aircraft. When the drone controller moves the stick (PSO-Pitch stick out of detente) the pitch attitude feedback is deactivated, ($\Theta_c = \Theta$), and the control signal is a pitch rate command directly into the pitch rate loop. As soon as the drone controller releases the stick, the pitch attitude loop is reactivated and the APAH system will maintain the pitch reference established by the controller.

3.1.2 Adaptive Speed Hold on Throttle system

The most direct way to control the velocity of an aircraft is by controlling the throttle position. The design of this control system assumes that the airspeed of the aircraft, v, can be measured directly.

The discrete time version of the ASHOT system was derived from Equation (3.1). The acceleration of the aircraft is proportional to the throttle position.

$$v(k+1) = v(k) + b_1 \delta t \tag{3.21}$$

where

v = airspeed

δt = throttle position

b_1 = unknown parameter.

The PI control law for the ASHOT system is

$$\delta t(k) = -K_v v(k) - k_p p(k), \quad \text{where} \quad p(t) = \int v(t)\,dt. \tag{3.22}$$

Equation (3.23) is obtained using Equations (3.21) and (3.22).

$$v(k+1) = v(k) - b_1 K_v v(k) - b_1 K_p p(k). \tag{3.23}$$

Using the Euler approximation, Equation (3.12), the equivalent expression of Equation (3.23) in the continuous time domain is the following:

$$v'(t)T + v(t)\ b_1K_v + b_1K_p \int v(t)dt = 0. \qquad (3.24)$$

The characteristic equation of the system is obtained by calculating the Laplace Transform of Equation (3.24)

$$s^2\, T\, v(s) \;+\; s\, v(s)\, b_1\, K_v \;+\; b_1\, K_p\, v(s) \;=\; 0. \qquad (3.25)$$

The desired characteristic equation is

$$s^2 \;+\; 2\,\zeta\, Wn\, s \;+\; Wn^2 \;=\; 0 \qquad (3.26)$$

where

ζ = damping factor

Wn = natural frequency.

The damping factor and natural frequency selected were

ζ = .707

Wn = .35 rad/sec.

By comparing Equations (3.25) and (3.26) the control parameters can be calculated as a function of ζ and Wn.

$$K_p = \frac{T\, Wn^2}{b_1}, \qquad K_v = \frac{2\,\zeta\, Wn\, T}{b_1} \qquad (3.27)$$

Figure 3.2 depicts the control law block diagram for the ASHOT system.

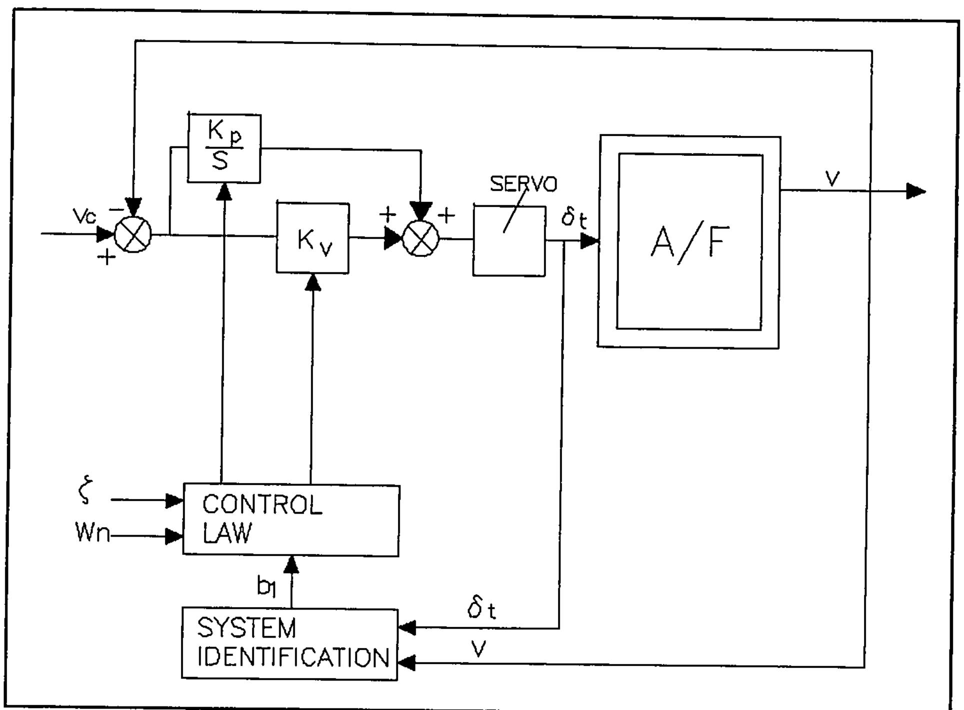


Fig 3.2 ASHOT system block diagram

Not shown in Figure 3.2 are the limits of the throttle command, δt, and of the system identification parameter b_1. The upper and lower throttle command limits are 0 degrees and 45 degrees, respectively. System identification is frozen when the throttle requested is outside these limits. The lower limit of the system identification parameter b_1 was set to 0.015. This was done to make sure that the ASHOT gains K_p and K_v do not become too large or negative (see Equation

3.27). Also, not illustrated in Figure 3.2, is the fact that the control integral is frozen when the throttle command is outside the limits.

3.1.3 Adaptive Altitude Hold on Pitch System

The function of the AAH system is to provide both an altitude hold capability and to respond to altitude trim commands. This control system is adaptive in the sense that it uses the APAH system described in the preceding section to control the altitude of the drone aircraft.

The AAH system derives a pitch attitude command using aircraft altitude, altitude rate and the integral of altitude error. Altitude, h, can be provided by a barometric and/or radar altimeter. Altitude rate, h', is computed by taking the derivative of altitude.

$$\theta_c = K_h \cdot (h_c - h) - K_{hd} \cdot h' + K_{hi} \int (h_c - h)\, dt. \qquad (3.28)$$

Figure 3.3 shows the control law block diagram of the AAH system.

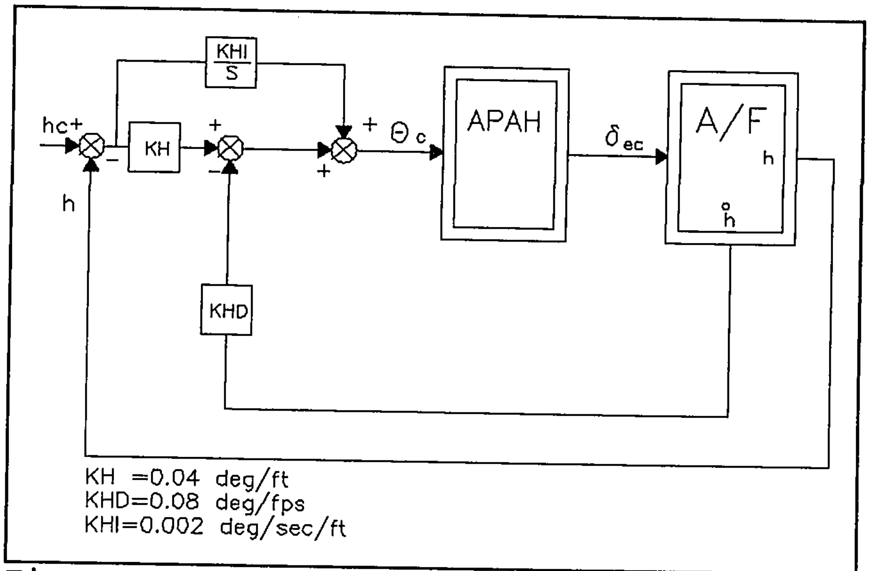


Fig 3.3 AAH system block diagram

4.2.3 Adaptive Speed Hold on Elevator

The objective of this control system is to control the speed of the aircraft using pitch attitude. This system is adaptive in the sense that it uses the adaptive pitch attitude hold system, described in Subsection 3.1.1, to control the speed of the aircraft. This control system will command the aircraft to pitch down if the desired airspeed reference is greater than the actual airspeed of the aircraft. On the other hand, the control system will command the aircraft to pitch up if the airspeed reference is less than the actual speed of the aircraft. This control mode is normally used for climbs and dives.

The control law of the ASHE loop is

$$\theta_c = K_a \cdot (v_c - v) + K_{ai} \int (v_c - v)\,dt. \tag{3.29}$$

Figure 3.4 shows the system block diagram of the ASHE loop.

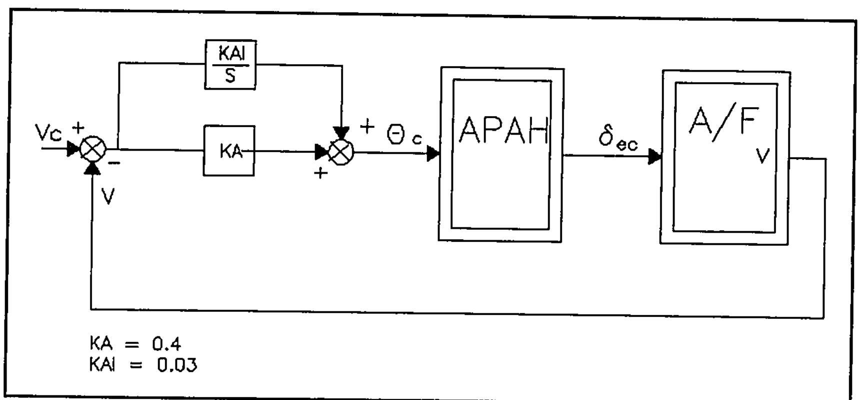


Fig 3.4 ASHE system block diagram

3.2 System Identification Algorithm

The recursive weighted least-squares estimation algorithm was selected for the identification process of the aircraft parameters mentioned in the preceding subsections. This algorithm delivers sufficiently fast and accurate on-line parameter estimates. A complete derivation of this algorithm was presented in Chapter 2.

The following equations summarize this algorithm:

$$P_r = \frac{1}{\lambda}\left[P_{r-1} - P_{r-1}\phi(r)\left[\phi^T(r)P_{r-1}\phi(r) + \lambda\right]^{-1}\phi^T(r)P_{r-1}\right] \qquad (3.30)$$

$$\theta_r = \theta_{r-1} + P_r\phi(r)\left[y(r) - \phi^T(r)\theta_{r-1}\right] \qquad (3.31)$$

where

P_r is the covariance matrix

$\Phi(r)$ is the input vector

Θ_r is the unknown parameter vector

$y(r)$ is the output measurement

λ is the forgetting factor ($\lambda = 0.98$).

Exponential weighting of past data was used in the estimation algorithm to ensure tracking of time-variable parameters. The optimum forgetting factor, λ, was selected through extensive simulations of the adaptive control system.

Persistent excitation was required to ensure a good system identification of the time varying parameters. This was obtained by adding white noise to the controller inputs.

CHAPTER 4

SIMULATION RESULTS

This chapter summarizes the simulation results of the adaptive controller described in Chapter 3. The adaptive controller was tested under different flight conditions and the simulation results were compared against those obtained with a conventional flight control system.

4.1 Summary of Simulation Results

The adaptive control system worked satisfactorily for all of the simulated flight conditions. The simulation tests included low and high altitude tests at subsonic and supersonic speeds. From the simulations tests, the following observations can be made:

(1) Due to the persistent excitation required for system identification, the adaptive controller tends to be noisier that the conventional controller. This could be annoying for the safety pilot, especially at supersonic speeds, during flight testing.

(2) A tighter control was obtained with the adaptive Speed Hold on Throttle control system. This was expected since the conventional SHOT system is a fixed gain control system that

does not take into account that the engine performance varies with the altitude of the drone. On the other hand, the adaptive SHOT system varies the control gains of the SHOT loop according to the drone acceleration to a throttle increase.

(3) Simulation tests showed that the performance of the adaptive PAH system was acceptable for drone control. The test results also showed that a higher natural frequency (Wn) was required at higher airspeeds. For the simulation tests, a natural frequency of 5 rad/sec was assigned for subsonic speeds. A natural frequency of 7 rad/sec was selected for supersonic speeds. It is well known that the ideal frequency requirements for the pitch axis is given by the load factor, n/α, which is defined as the steady state normal acceleration change per unit of angle of attack. It is also known that this factor varies with the aerodynamic pressure which in turn changes with the airspeed of the aircraft (see Equation 2.35).

(4) The APAH system was used without any problem for altitude (AAH) and speed control (ASHE). The simulation tests showed that the performance of these two control systems is equivalent, if not better, than the performance obtained with the conventional control laws.

4.2 Simulation Tests

Table 4.1 summarizes some of the test cases used to evaluate the performance of the adaptive controller. The table shows for each test case, the initial altitude and speed of the aircraft, its roll attitude, the configuration of the adaptive autopilot and a brief description of the simulation test. Not shown in this table is the fact that the Roll Attitude Hold mode was active during all of the test cases to provide automatic lateral control to the aircraft.

Test No.	H (ft)	Mach	Roll (deg)	Adaptive AFCS Configuration				Objective
				APAH	ASHOT	AAH	ASHE	
1	10000	0.8	0	ON	ON	OFF	OFF	APAH response
2	10000	0.8	0	ON	ON	ON	OFF	ASHOT response
3	20000	0.9	60	ON	ON	ON	OFF	Test APAH,AAH,ASHOT
4	20000	0.9	-60	ON	ON	ON	OFF	Same as 3 with winds
5	5000	0.6	0	ON	ON	ON	OFF	APAH adaptability
6	5000	0.8	0	ON	OFF	OFF	ON	ASHE response
7	25000	1.1	0	ON	ON	ON	OFF	Supersonic AAH test

Table 4.1 Simulation Test Cases

(1) Test No. 1

The purpose of this test was to evaluate the overall performance of the adaptive pitch attitude hold (APAH) system. For this simulation test, the aircraft was initialized at an altitude of 10,000 feet MSL. Its speed was initialized at Mach 0.8. Figure 4.1 is a time plot of the desired pitch attitude (dotted line), and of the pitch attitude angle (solid line).

Also shown, is the elevator deflection (DELTAE). The initial pitch attitude of the aircraft was 2.5 degrees. A five degrees pitch change was commanded and the pitch attitude of the aircraft increased to 7.5 degrees. A few seconds later, the pitch reference was commanded to follow a sawtooth wave pattern. Figure 4.2 depicts the system parameters f_{11}, f_{12}, h_1 and the bias term b_1. These parameters were identified by the recursive least squares system identification algorithm. From Figure 4.2 it can be observed that the steady state values of the parameters f_{11}, f_{12}, and h_1 are 0.6, -1.2, and -1.9, respectively. Figure 4.3 compares the actual pitch attitude rate of the aircraft (q) with the pitch rate reconstructed (qhat) using the system parameters f_{11}, f_{12}, h_1, b_1 and the Equation (3.17) which is repeated below for clarity.

$$\hat{q}(k+1) = f_{11}\, \hat{q}(k) + f_{12}\, a_z(k) + h_1\, \delta e(k) + b_1. \qquad (4.1)$$

As described in Subsection 2.2.1, this is a commonly applied procedure to test the validity of a theoretical model. The system is simulated with the actual input (in this case elevator deflection, δe) and the simulated output is compared to the measured output.

The same test was repeated but this time using a conventional PAH control system. Figure 4.4 illustrates time plots of the pitch attitude and elevator deflection angles. By simply comparing Figures 4.1 and 4.4 it can be observed that the performance of the APAH system was comparable to the performance of the conventional controller. It can also be observed that the APAH system was noisier than the conventional controller. This result was expected since persistent excitation of the APAH system is required to ensure a good system identification.

(2) Test No. 2

The purpose of this test was to assess the overall performance of the adaptive speed hold on throttle (ASHOT) system. As in test case 1, the aircraft altitude was 10,000 feet and its speed was Mach 0.8. As depicted in Figure 4.5, the aircraft was traveling at an airspeed (V) of 442 knots when the airspeed reference (dotted line) was increased to 452 knots. The ASHOT system responded immediately by commanding a momentary throttle increase (DELTAT) from 15 to 30 degrees as portrayed in Figure 4.6. The drone speed overshot the reference by about 3 knots. A few seconds later, the airspeed came down to 452 knots and started following the airspeed command very closely. Figure 4.7 shows the ASHOT system identification parameter b_1. Figure 4.8 compares the actual airspeed (V) of the aircraft with the airspeed (Vhat) estimated using the parameter b_1 and the throttle position, δt.

$$\hat{V}(k+1) = \hat{V}(k) + b_1\, \delta t(k). \tag{4.2}$$

To evaluate the above simulation results it was required to re-run test case No. 2 using, this time, a conventional SHOT control law. Figures 4.9 and 4.10 show the performance of the conventional controller. Notice that the system response of the conventional SHOT controller is slower and sloppier than the system response of the adaptive controller. It should be noted also that the time response of the adaptive controller can be changed easily by simply modifying the desired natural frequency, Wn, of the SHOT control law (see Equation 3.24).

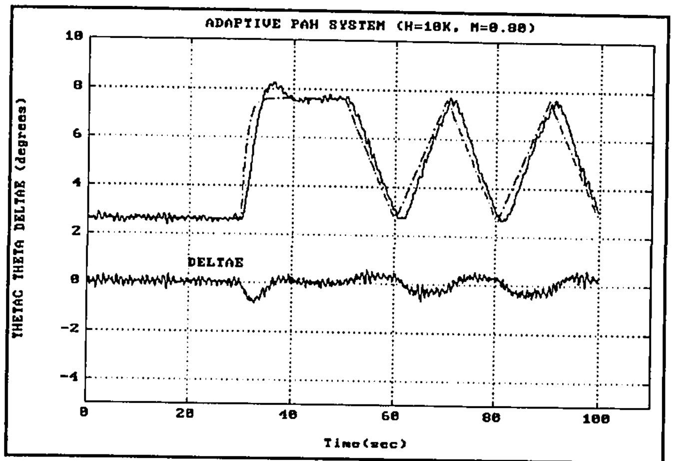


Fig 4.1 APAH-pitch/elevator angles

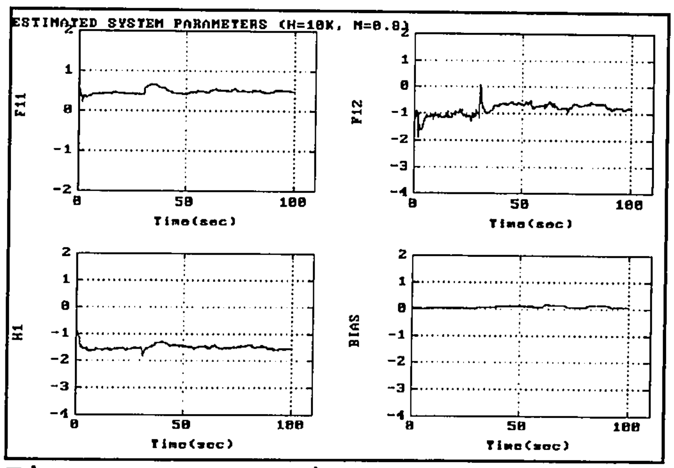


Fig 4.2 APAH-estimated parameters

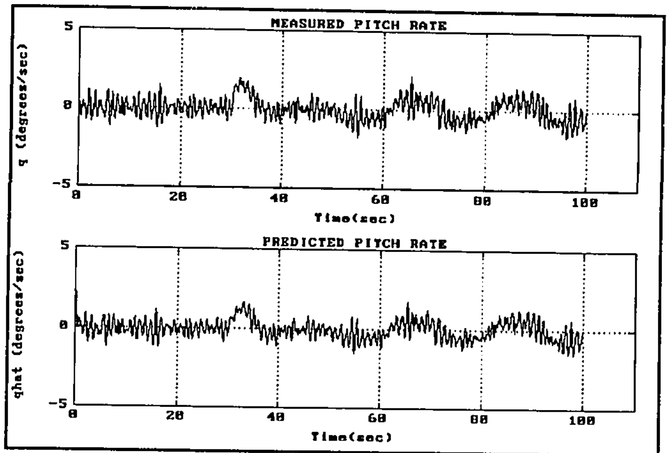


Fig 4.3 Measure/predicted pitch rate

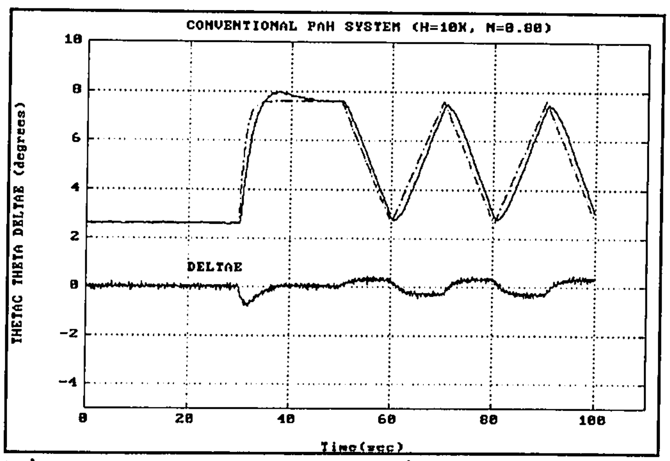


Fig 4.4 Conv. PAH - pitch/elevator

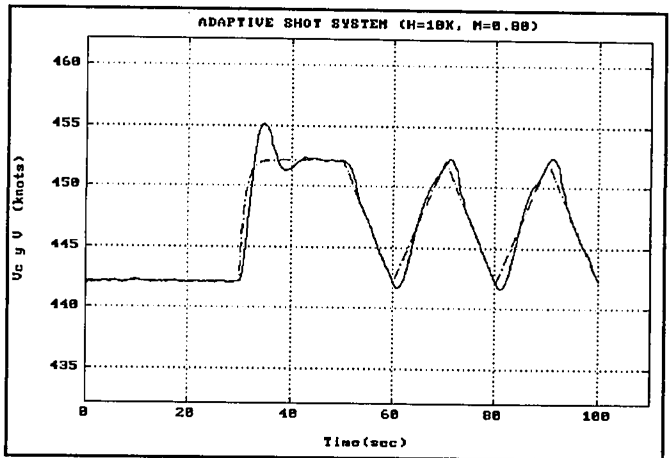
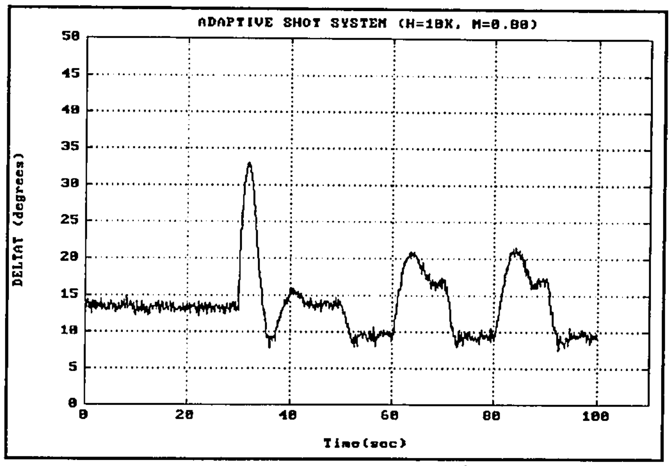


Fig 4.5 ASHOT - airspeed



(4.6) ASHOT-throttle position

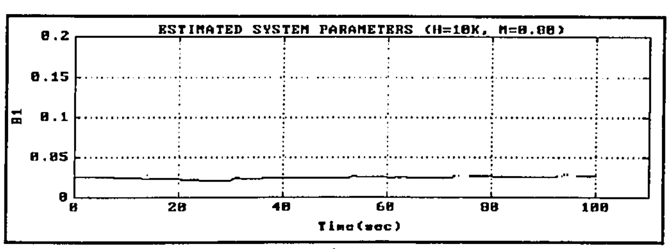


Fig 4.7 ASHOT-estimated parameter

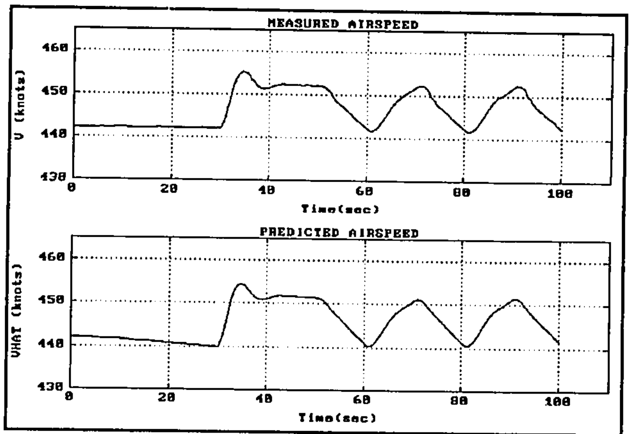


Fig 4.8 Measured/predicted speed

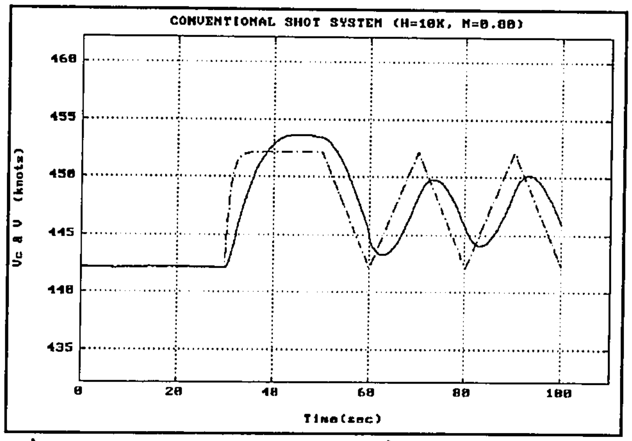


Fig 4.9 Conv. SHOT-airspeed

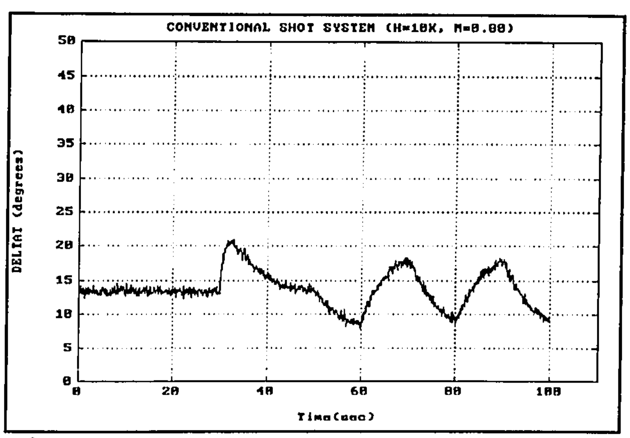


Fig 4.10 Conv. SHOT-throttle

(3) Test No. 3

The objective of this test was to study the overall performance of the AAH and ASHOT systems during turns. The drone was initialized at an altitude of 20,000 feet and at a speed of Mach 0.92. Thirty seconds later, a right turn was commanded. As depicted in Figure 4.11, the roll attitude (PHI) of the aircraft changed from 0 to 60 degrees. Figure 4.12 shows that the drone descended momentarily 50 feet due to the reduction of the aerodynamic lift force during the turn. The pitch attitude of the drone aircraft is plotted in Figure 4.13. The APAH system parameters are shown in Figure 4.14. As expected, during the turn, the drag force increased forcing the ASHOT system to command a throttle increase (see Figures 4.15 and 4.16), to maintain the desired airspeed.

(4) Test No. 4

The purpose of this test was to evaluate the overall performance of the AAH and ASHOT systems under gusty flight conditions. The drone was initialized at 20,000 feet MSL, at Mach 0.92. Thirty seconds into the simulation, a 60 degree left turn was commanded as portrayed in Figure 4.17. Figure 4.18 shows the inertial axis wind velocity components Vwx, Vwy and Vwz. In the simulator, the wind velocity components are subtracted from the inertial velocity of the aircraft to calculate the total aerodynamic velocity of the aircraft (see

Appendix A, Equation A.6). A sinusoidal head wind of ± 50 feet per second was simulated. Also simulated was a vertical sinusoidal wind of ± 5 feet per second. Figures 4.19 and 4.20 show the altitude and pitch transients caused by the left turn and the gusty winds. Figure 4.21 depicts the APAH system parameters identified during this simulation test. Figure 4.22 shows the airspeed variations of the aircraft. The ASHOT system attempted to maintain, despite of the winds, an airspeed reference of 432 knots by controlling the throttle as portrayed in Figure 4.23.

For comparison, the preceding simulation test was rerun using a conventional controller. The performance of the conventional controller is illustrated in Figures 4.24 through 4.27. Notice that the performance of the adaptive AH system was as good as the performance of the conventional AH controller. Notice also that the performance of the adaptive SHOT system was superior to the performance obtained with the conventional controller.

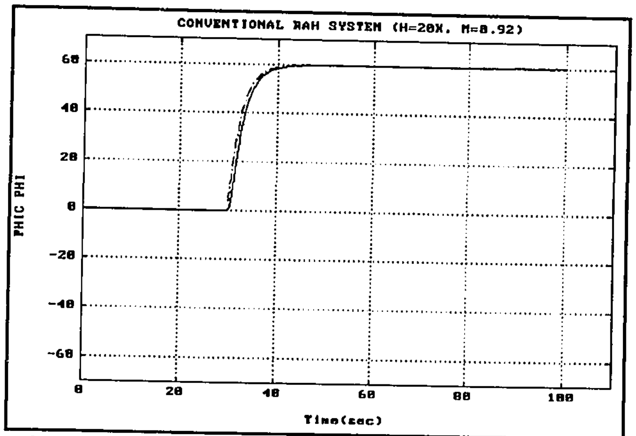


Fig 4.11 Conv. RAH - Roll attitude

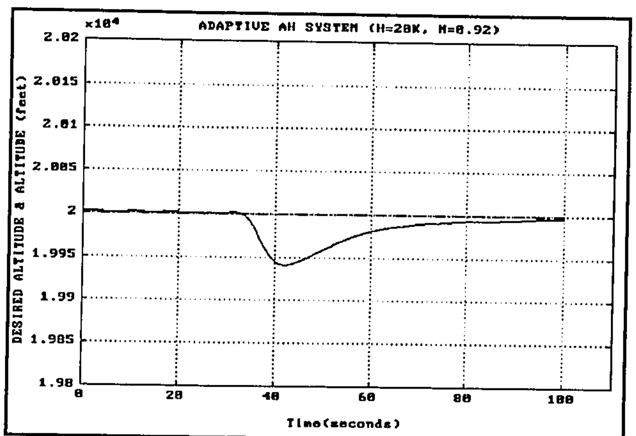


Fig 4.12 AAH - altitude

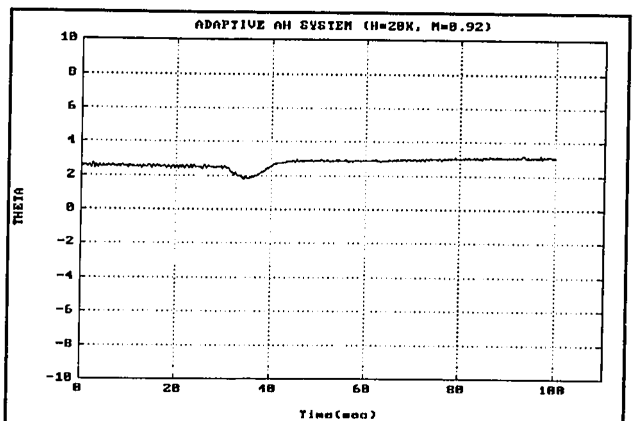


Fig 4.13 AAH - pitch attitude

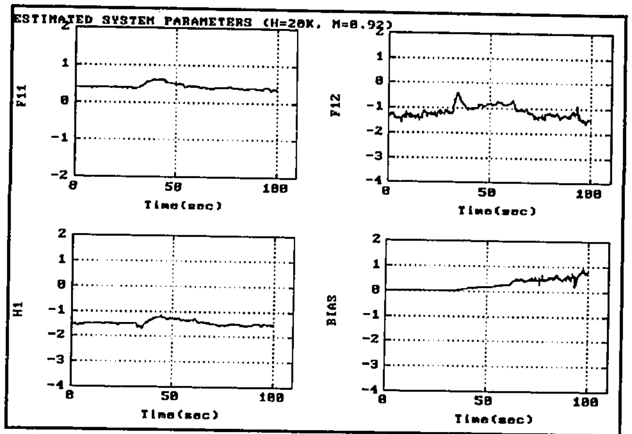


Fig 4.14 AAH-estimated parameters

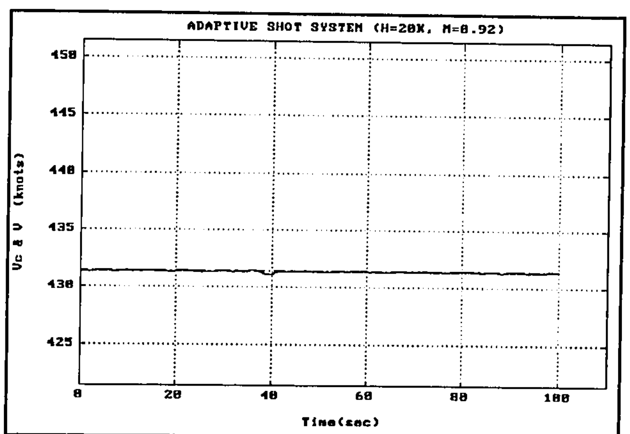


Fig 4.15 ASHOT - airspeed

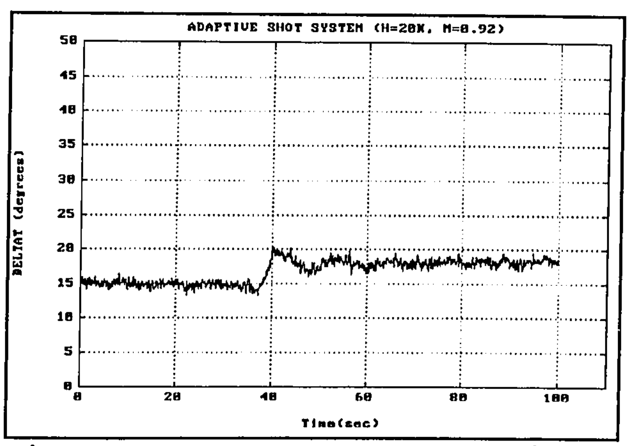


Fig 4.16 ASHOT - throttle position

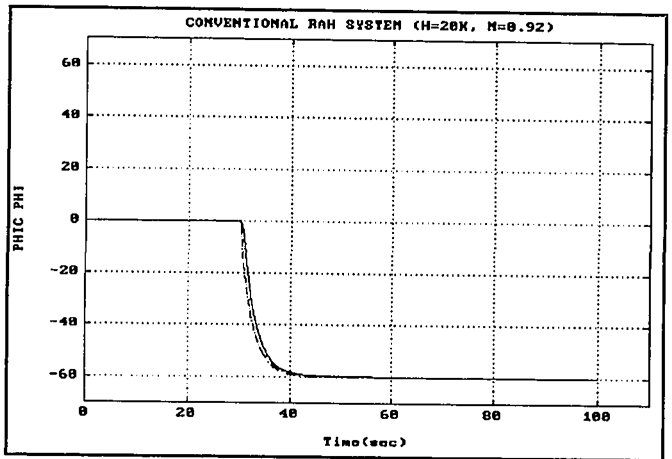


Fig 4.17 Conv. RAH - Roll attitude

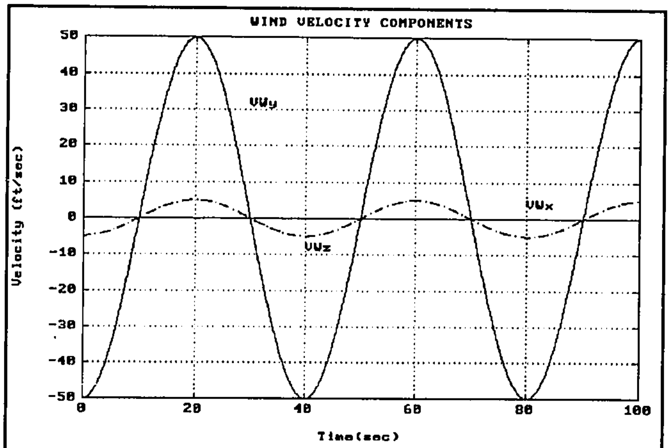


Fig 4.18 Wind velocity components

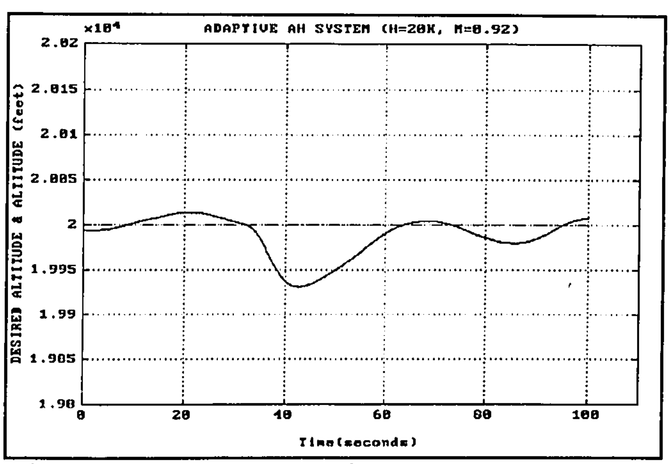


Fig 4.19 AAH - altitude

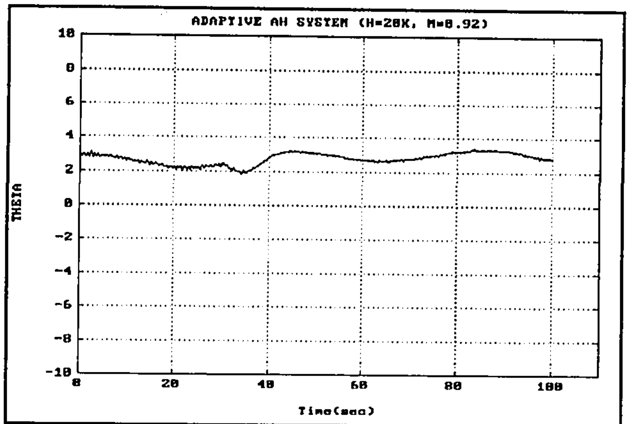


Fig 4.20 AAH - pitch attitude

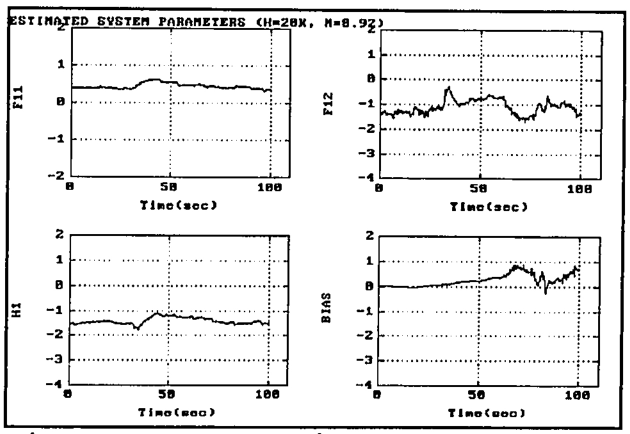


Fig 4.21 APAH-estimated parameters

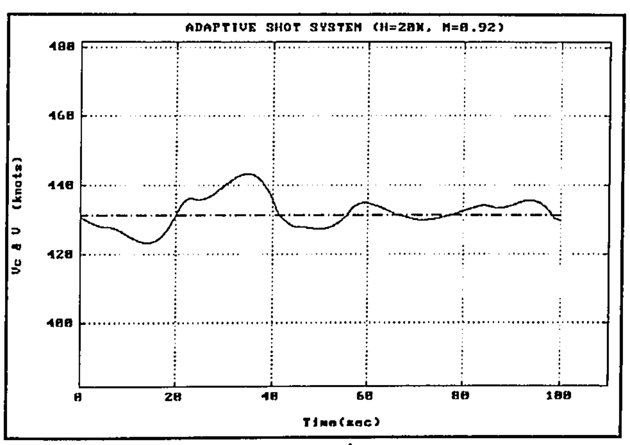


Fig 4.22 ASHOT - airspeed

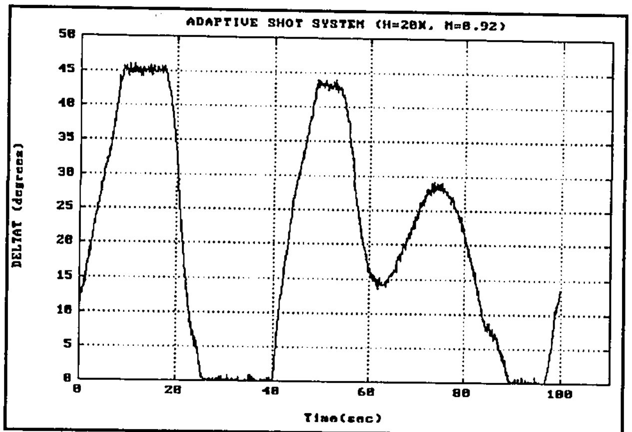


Fig 4.23 ASHOT - throttle position

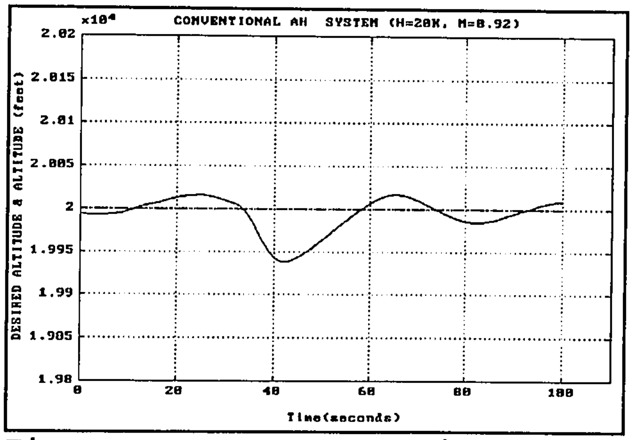


Fig 4.24 Conv. AH - altitude

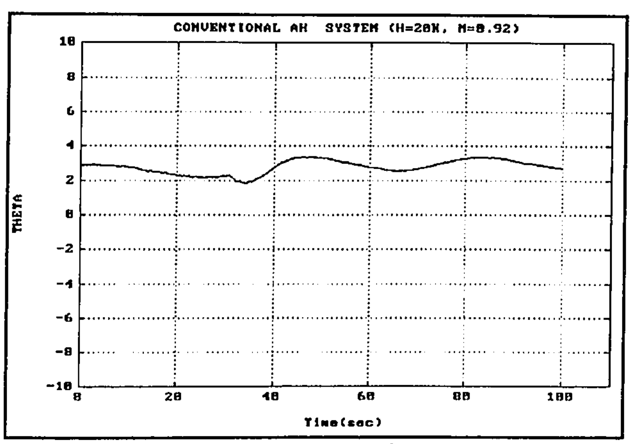


Fig 4.25 Conv. AH - pitch attitude

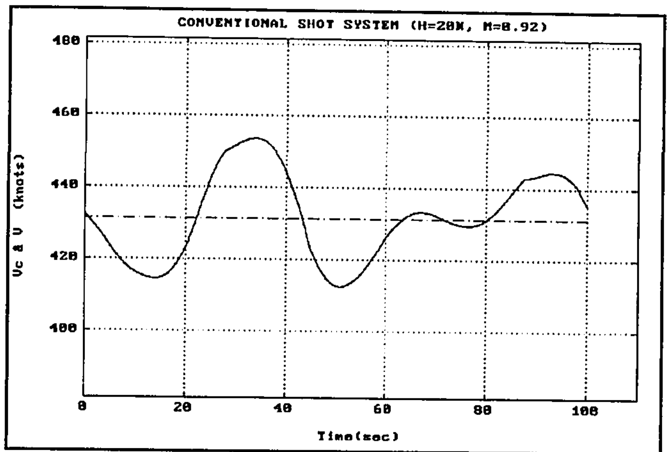


Fig 4.26 Conv. SHOT - airspeed

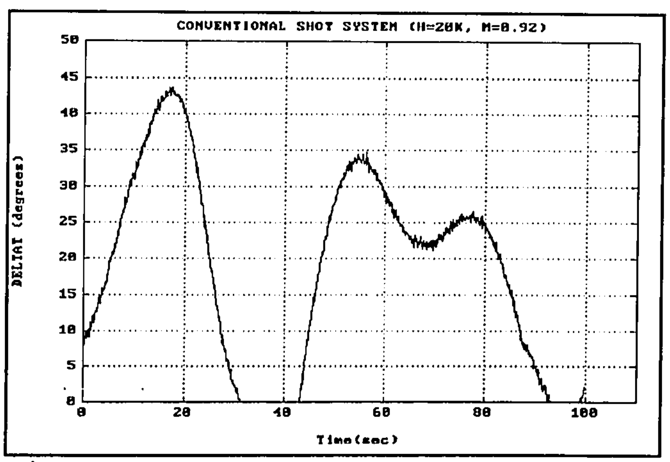


Fig 4.27 Conv. SHOT - throttle

(5) Test No. 5

The objective of this simulation test was to check the adaptability of the APAH system to airspeed changes. As mentioned in Chapter 2, Subsection 2.2.1.1, the dynamics of the longitudinal axis of the aircraft varies with the velocity of the aircraft relative to the air-mass or more precisely with the aerodynamic pressure of the aircraft. For this simulation test the speed of the aircraft was increased from mach .68 to mach .83 as illustrated in Figure 4.29. The magnitude of the APAH control gains K_q (KQ), K_{qi} (KTH) and K_{az} (KNZ) decreased as the airspeed of the aircraft increased as depicted in Figure 4.28. It should be noted that this is consistent with the theory and with the airspeed schedule of the conventional PAH system shown in Chapter 2, Figure 2.12. In the conventional PAH controller the total gain of the PAH system is reduced as the airspeed of the aircraft increases. The pitch moment is directly proportional to the aerodynamic pressure of the aircraft as shown in Equation (A.15) which in turn is proportional to the true airspeed of the aircraft (Equation 2.35). Therefore, as the airspeed of the aircraft increases, the pitch moment increases and a lower control loop gain is required to maintain the stability of the system. Figure 4.30 shows that the altitude of the aircraft increased a few feet momentarily as the speed of the aircraft increased. This is consistent with the theory also. As the dynamic pressure increased, the lift force acting on the aircraft increased (see Equation A.11). As portrayed in Figure 4.31, the APAH system commanded a lower pitch attitude angle to try to compensate for this increase in lift. Finally, Figure 4.32 shows the APAH system identification parameters computed during this simulation test.

(6) Test No. 6

The purpose of this simulation test was to evaluate the performance of the adaptive speed hold on elevator (ASHE) system. As mentioned before, the function of this system is to control the airspeed of the aircraft using pitch attitude control. This control system is normally used for climbs and dives. This system is called adaptive because it uses the APAH system described in Subsection 3.1.1. For this simulation test, the drone aircraft was initialized at an altitude of 5,000 feet MSL at Mach 0.8. The airspeed reference of the drone was decreased from 503 to 483 knots as shown in Figure 4.33 (dotted line). The ASHE system immediately responded by commanding an increase in the pitch attitude of the drone (Figure 4.34). Notice in Figure 4.33 (solid line), that as the pitch angle of the aircraft increased, the drone airspeed decreased and started tracking the airspeed reference. The pitch attitude increase caused the drone to climb from 5,000 feet to almost 7,000 feet MSL as illustrated in Figure 4.35.

Figure 4.36 shows the APAH system identification parameters. Finally, for comparison, Figures 4.37 and 4.38 show the performance of a conventional SHE system for the same test case.

(7) Test No. 7

The purpose of this test was to assess the effectiveness of the AAH system at supersonic speeds. The drone was initialized at 25,000 feet MSL. The drone speed was Mach 1.1. The drone was in altitude hold mode. The altitude reference of the autopilot was changed by 250 feet. Immediately, the aircraft pitched up and started climbing as depicted in Figure 4.39. In this figure the altitude command is portrayed by the dotted line. The real altitude of the drone is indicated by the solid line. The pitch angle increased momentarily from 2 to 4 degrees as shown in Figure 4.40. The RLS algorithm estimated the correct system parameters as plotted in Figure 4.41. Figure 4.42 depicts time plots of the Mach and airspeed of the drone aircraft. Finally, Figures 4.43 and 4.44 show the performance of the gain-scheduling controller for the same test case.

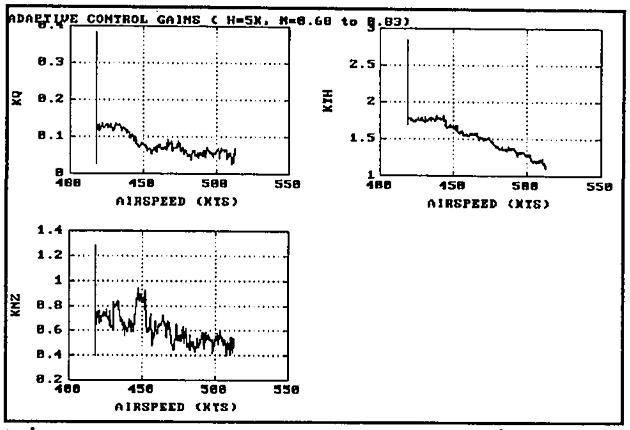


Fig 4.28 APAH - control gains

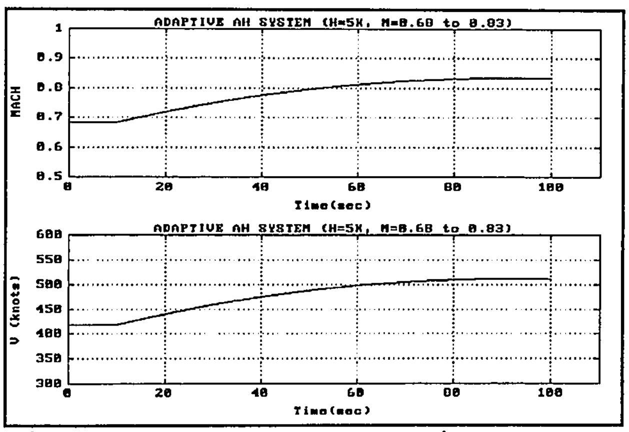


Fig 4.29 AAH - Mach & airspeed

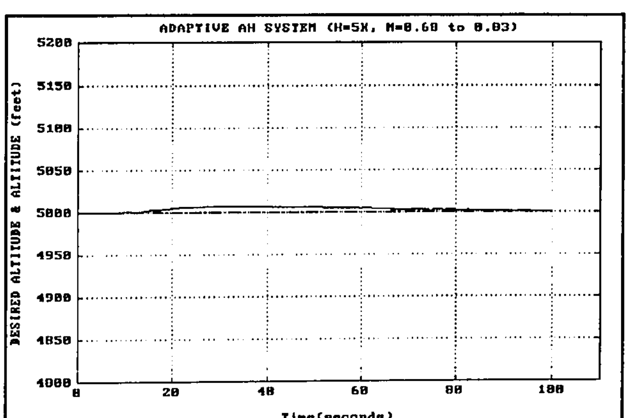


Fig 4.30 AAH - altitude

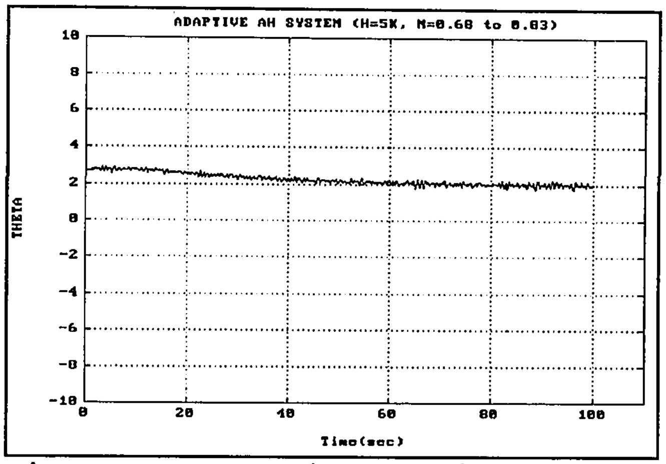


Fig 4.31 AAH - pitch attitude

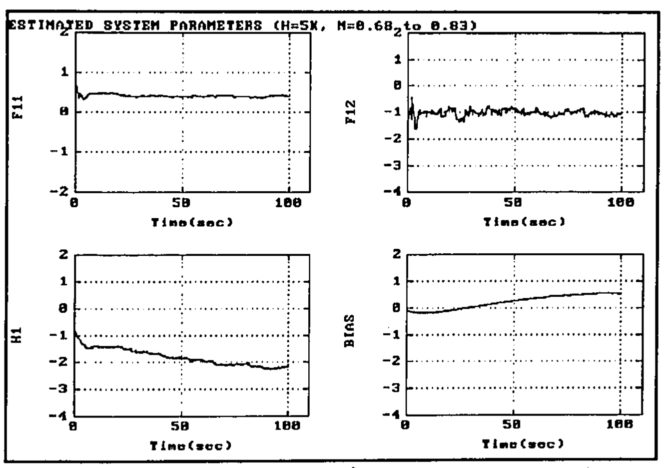


Fig 4.32 APAH - estimated parameters

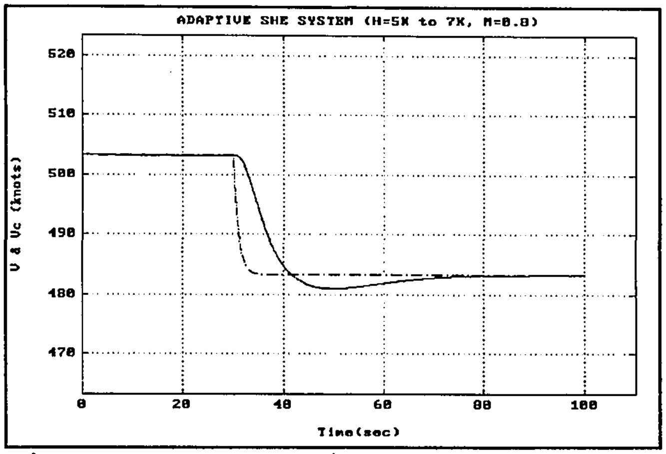


Fig 4.33 ASHE - Airspeed change

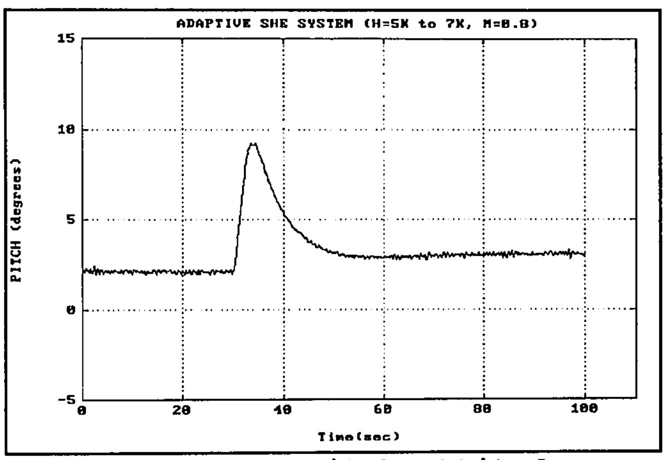


Fig 4.34 ASHE - pitch attitude

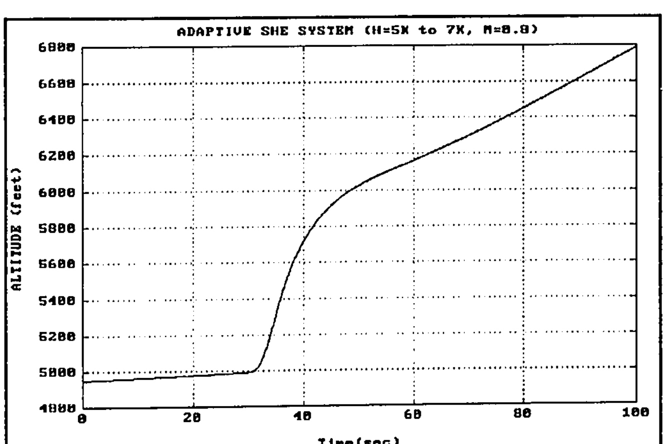


Fig 4.35 ASHE - altitude

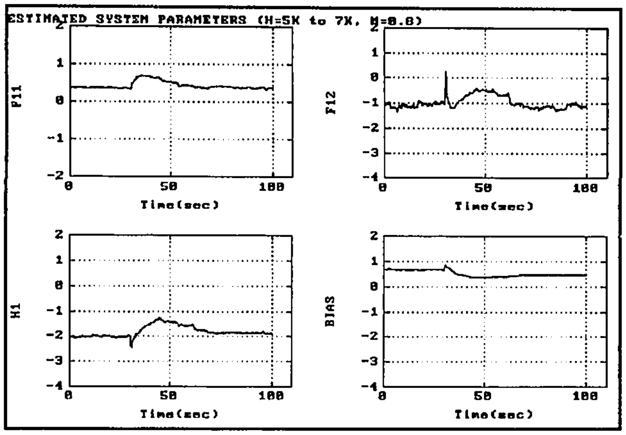


Fig 4.36 APAH-estimated parameters

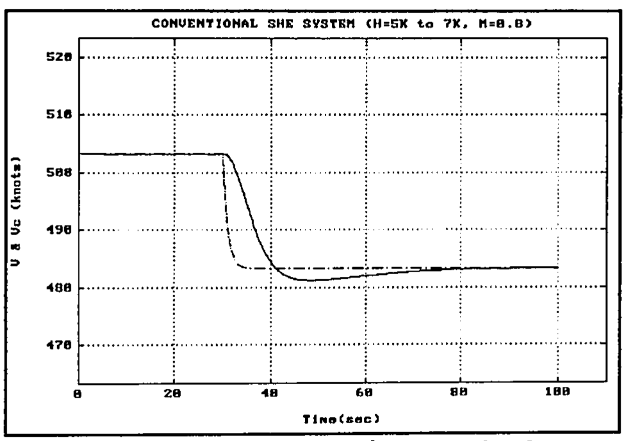


Fig 4.37 Conv. SHE-airspeed change

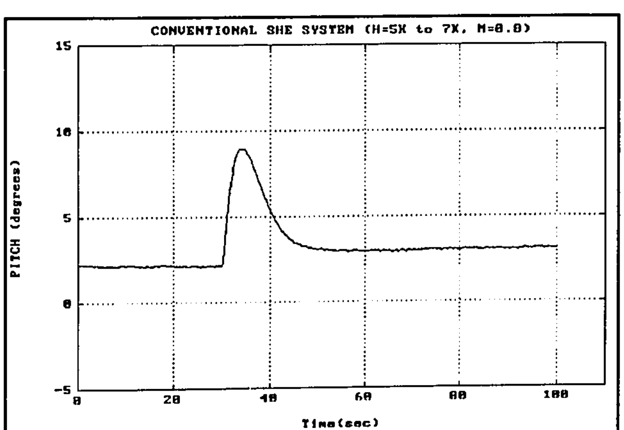


Fig 4.38 Conv. SHE - pitch attitude

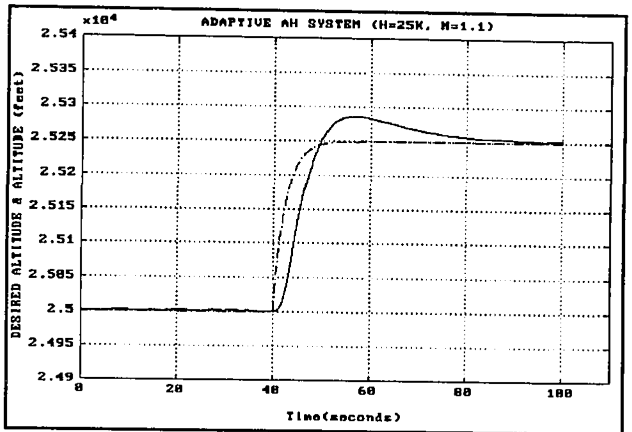


Fig 4.39 AAH-supersonic-altitude

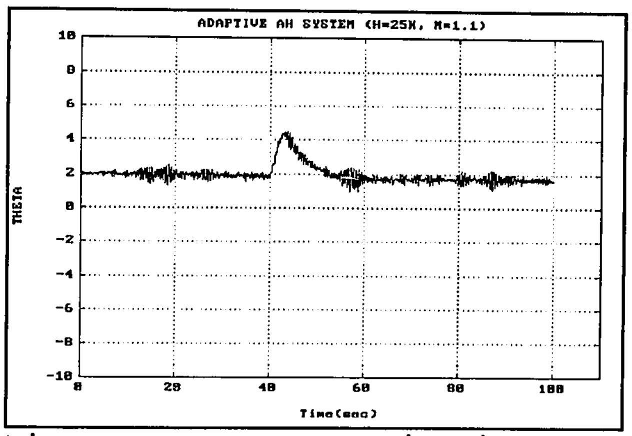


Fig 4.40 AAH-supersonic-pitch

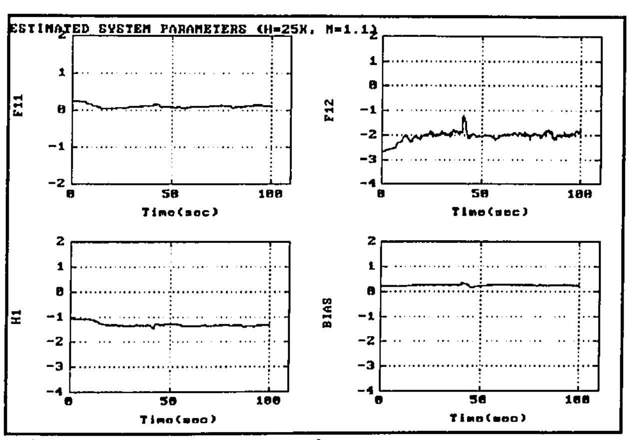


Fig 4.41 APAH estimated parameters

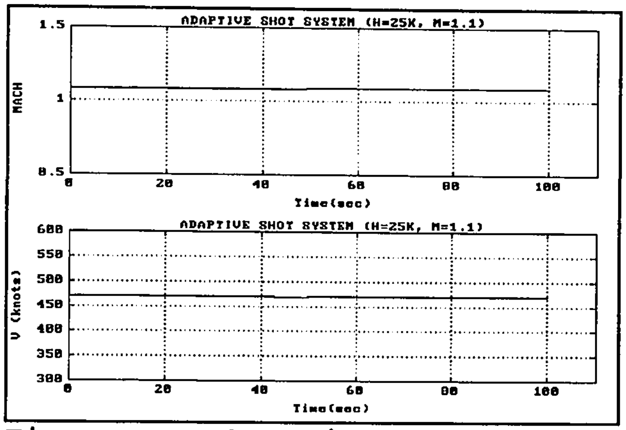


Fig 4.42 Mach & airspeed

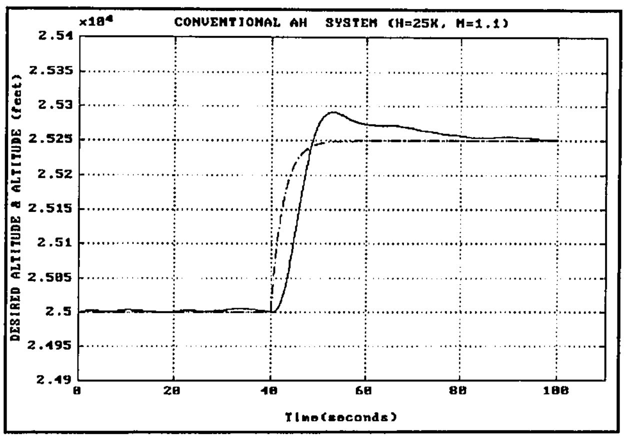


Fig 4.43 Conv. AH supersonic

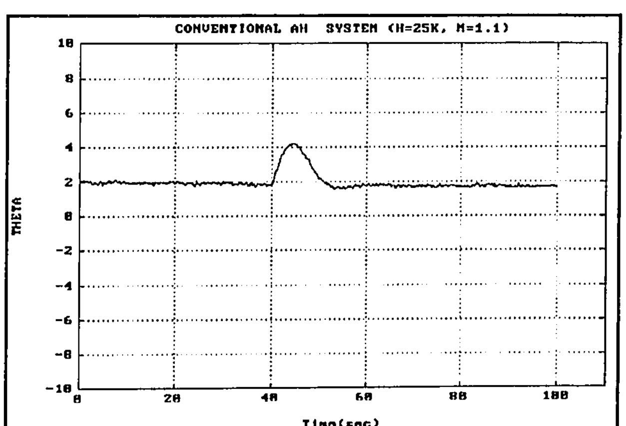


Fig 4.44 Conv. AH supersonic-pitch

CHAPTER 5

CONCLUSIONS AND RECOMMENDATIONS

5.1 Conclusions

The proposed adaptive system controller for the longitudinal axis of the QF-106 drone aircraft performed satisfactorily during the simulation tests. It is an efficient and very easy to implement controller that challenges the performance of the current gain-scheduling flight control systems. Based on theoretical and simulation studies the following conclusions are drawn:

(1) The second order mathematical model (short period approximation) used to represent the dynamics of the pitch axis is sufficiently accurate for adaptive pitch control.

(2) The proposed adaptive pole placement control technique provided good control for all of the simulated flight conditions including under gusty wind conditions.

(3) The adaptive flight controller is noisier than the conventional flight control system.

(4) The adaptive PAH system can be used without any problems for altitude and speed control.

5.2 Recommendations

Based on the theory and observations made during the investigation, the following recommendations are proposed for further study:

(1) During the simulation tests, the natural frequency, Wn, of the APAH system was provided by the investigator and maintained constant throughout the simulation test run. In the real world there is a need for automatically selecting the correct natural frequency for a given flight condition. Since the ideal frequency requirements for the pitch axis is given by the load factor, n/α, further study is needed to determine if this factor can be approximated using aerodynamic pressure and if this approximation can be used for pitch control.

(2) The proposed adaptive flight control system needs to be evaluated with the real QF-106 drone aircraft.

(3) Theoretically, the proposed adaptive control technique can be used for lateral control. Further study is required to determine if the known approximations for the lateral control axis such as the rolling and dutch roll mathematical models can be used for lateral adaptive control.

REFERENCES

[1] P.C. Gregory. "Proceedings of the Self Adaptive Flight Control Systems Symposium". WADC Technical Report, Wright Air Development Center, WPAFB, Ohio, 1959.

[2] K.J. Astrom and T. Hagglund. Automatic Tuning of PID Controllers. Instrument Society of America, 1988.

[3] V. V. Chalam. Adaptive Control Systems. New York: Marcel Dekker Inc., 1987.

[4] G.C. Goodwin and K. S. Sin. Adaptive Filtering Prediction and Control. Englewood Cliffs, NJ: Prentice-Hall, 1984

[5] D.C Nemir. Analysis and Self Tuning Control of a Rotating Compliant Link, PHD Thesis, Purdue University, December 1986.

[6] H.P. Wittaker and J. Yamron. "Design of Model Reference Adaptive Control Systems for Aircraft", Report No. R-164, Instrumentation Lab., MIT, 1958.

[7] K.J. Astrom and B. Wittenmark. "Self Tuning Regulators", Automatica, vol 9, pages 185-199, 1973.

[8] L. Ljung. System Identification, Englewood Cliffs, NJ: Prentice-Hall,1987.

[9] J. Roskam. Airplane Flight Dynamics and Automatic Flight Controls Parts I & 2, Kansas: Roskam Aviation and Engineering Corporation, 1982.

[10] B Friedlan. Control System Design, McGraw-Hill, 1986.

[11] "QF-106 Primary and Backup AFCS System Software Diagrams", Honeywell Corporation, 1989.

[12] "Simulation Model F-106A (Simulation Mode, Mass Model and Engine Model)", Report No. B-WT1238101, Industviasnlogen-Betriebsgesellshaft, 1978

[13] "DFCS/QF-106 Simulator", Report No. 87-F24-006, IBM Corporation, 1987.

[14] "Atmospheric Structure, Part 3", White Sands Missile Range, 1969.

[15] "MATLAB Program Users Guide", MATHWORKS Inc, 1990.

APPENDIX A

QF-106 SIMULATION PROGRAM DOCUMENTATION

A.1 Scope and Background

Appendix A describes the QF-106 aircraft batch simulation program. This simulation program was used to evaluate the performance of the proposed adaptive flight control system for the QF-106 drone aircraft.

The simulation program provides aircraft rotational and translational motion with six degrees of freedom; an atmospheric representation; mathematical models of aerodynamic, gravity, and engine forces and moments acting on the vehicle; and finally, engine dynamics. It also simulates two different flight control systems: an adaptive controller and a conventional gain-scheduling controller.

As part of this thesis, the QF106 simulator was substantially modified so it could be executed (in batch mode) on a 386 personal computer. A lot of changes were also made to speed up the execution of the program and to reduce its size. Some of the changes made are listed below:

(1) Deleted the simulation of the ground steering dynamics including the simulation of the nosewheel servo and landing gear.

(2) Deleted the simulation of the rate gyros.

(3) Simplified considerably the engine model.

(4) Re-designed the data commons.

(5) Simplified the atmospheric model.

Substantial modifications were also incorporated into the simulator to include an adaptive flight control system for the QF106 drone aircraft. Furthermore, the simulation program was redesigned to provide data to an off-line analysis program called PC-MATLAB.

A.2 Software Description

A.2.1 SIM06 Executive Program

Figure A.2.1 shows the subroutine hierarchy of the executive program SIM06. Figure A.2.2 is a simplified functional flow of the executive program SIM06. The intent of this simplified flow chart is to illustrate the general structure of the program. It is not intended to show its actual code. The FORTRAN listing of the SIM06 program is included in Appendix B.

In the first step of this program, a group of simulator variables, such as, aircraft position, sampling time and initial heading are initialized. After that, the program, reads from the user, the control parameters for the particular flight condition that will be simulated (i.e. true airspeed, aircraft altitude, and fuel weight). Next, the program reads parameters, provided by the user, to control the simulation time, to select autopilot configuration (adaptive or conventional control system) and to select the time responses that will be tested. Next, the simulation initialization program INIT06 is called. This program iteratively calls the vehicle subroutines ATMS06, ENGIN6, SERV06 and MOTN06 to establish a trim condition where all the inertial and angular accelerations are nulled. Once the trim condition has been established, SIM06 implements another iterative loop to continue the simulation. The simulation loop includes:

(1) The generation of the step and ramp commands used to test the adaptive controller.

(2) The simulation of the adaptive and conventional flight control systems (see Figure A.2.3)

(3) The execution of the vehicle model subroutines ATMS06, ENGIN6, SERV06 and MOTN06

(4) The execution of the system identification algorithms, SYSIDP and SYSIDT, for the pitch and throttle axes, respectively.

(5) The print out of important data and the generation of ASCII files for the analysis program PC-MATLAB.

Figure A.2.1 QF-106 Simulator Program Hierarchy

```
SIM06 - Executive Program
CALLS:
* INIT06-Initializes the QF-106 Simulator
   CALLS: ATMS06, ENGIN6, SERV06, and MOTN06
* ATMS06 - Generates atmospheric data; airspeed, mach#,...
*      CALLS:
*      INERT6 - Calculates moments of-inertia and CG location
*           CALLS:
*           INTPAR - Interpolation utility
* ENGIN6 - Engine Model
* SERV06 - Simulates pitch and  roll elevon and rudder
*           servo responses.
* MOTN06 - Simulates  the QF106 aircraft dynamics.
*       CALLS:
*       WINDS - Winds Simulation
*       COEF06 - Calculates aerodynamic coefficients
*          CALLS:
*          COEF03 -Table lookup for Mach and altitude
                 dependent coefficients
*          COEF08 -Table lookup for Drag Coefficient
* WNOISE - White noise simulation
*       CALLS:
*          RANDOM - Random number generator
* SYSIDP   - System identification algorithm for the pitch
             axis
* SYSIDT   - System identification algorithm for the throttle
             axis
*       CALLS:
*          MULTM - Matrix multiplication algorithm
```

```
         |-------------------|
         |  SIM06            |
|--------|-------------------|---------------------------------------------------------|
|        Initialize Simulation Variables                                              |
|        PSI = 0.0 { Initial Heading (North) }       ENU(2) = 0.0 {Y-drone initial position}
|        ENU(1)=0.0 {X-Aircraft initial  position}   ENU(3) = 10000.  {Z-drone init. position}
|-------------------------------------------------------------------------------------|
| Input Initial Flight  VR    { Total velocity in fps }        WFUEL { Fuel Weight }
| Conditions:           HA    { Aircraft altitude in feet }    ABURNR  {After Burner ON/OFF}
|-------------------------------------------------------------------------------------|
|   Input Simulation Analysis Control Parameters       PMLAB1,PMLAB2,PMLAB3 {Print Control Flags}
|   TSTIME   {Total Simulation Time}                   TPRNT {Time to Print data }
|   TSTEP, TRAMP {Times for step or ramp }             AHM,SHOTM,SHEM   {Autopilot Modes selected}
|   PSTEP, PRAMP {Pitch step and Ramp command}         SDEVP,SDEVS,SDEVR {White Noise St. dev.}
|   RSTEP, RRAMP {Roll step and Ramp command           LAMDS,LAMDP,LAMDR {SI forgetting factors}
|   HSTEP, HRAMP {Altitude Step and Ramp command}      WNS,WNP,WNR {STR natural frequencies}
|   VSTEP, VRAMP {Velocity Step and Ramp Command}      DMPS,DMPP,DMPR  {STR damping factors}
|   TSELF    {Time to turn on adaptive controller      WTYPE,WVEL,WDIR,WGUST,WCYCLE,WTIME
|   APAH, ASHOT {Adaptive PAH and SHOT mode flags}
|=====================================================================================|
|   Subroutine to initialize simulation, establish trim conditions and calculate small perturbation|
|        CALL INIT06
|=====================================================================================|
|   Initialize variables and save trim biases
|   TIME = 0         { Current simulation time }   PPTRIM = DELTAE     {Elevator bias}  HREF = H
|   THET = THETA     { Pitch trim bias }           PRTRIM = DELTAA     {Aileron bias}   VIASR = VIAS
|   PHIT = PHI6D     { Roll trim bias }            THBIAS = DELTAT     {Throttle bias}
|-------|-----------------------------------------------------------------------------|
  >>>>> | \                     Check for time to step input command                 /
   /\   | T \                                    TIME > TSTEP and TIME < TRAMP?      / F
   /\   |--------------------------------------------------------|--------------------|
   /\   |   Command Exponential Step responses                   |  Hold trim meanwhile
   /\   |   Pitch Step          Roll Step                        |
   /\   |   Velocity Step       Altitude Step                    |
   /\   |-----------------------------------------------------------------------------|
   /\   | \                          Check for time to ramp command                   /
   /\   | T \                                       TIME > TRAMP?                     / F
   /\   |--------------------------------------------------------|--------------------|
   /\   |   Pitch Ramp          Roll Ramp                        |  Hold trim mean while
   /\   |   Velocity Ramp       Altitude Ramp                    |
   /\   |--------------------------------------------------------|--------------------|
   /\   |   Simulate White Noise for the PAH, RAH, and SHOT control systems
   /\   |   CALL WNOISE ( mean, sdev, noise )
   /\   |-----------------------------------------------------------------------------|
   /\   | AFCS SIMULATION {ADAPTIVE & CONVENTIONAL AUTOMATIC FLIGHT CONTROL SYSTEM}
   /\   |=============================================================================|
   /\   |   Call Vehicle Model  CALL ATMS06   {Atmospheric Data; Mach, Airspeed,...  }
   /\   |   Subroutines         CALL ENGIN6   {Engine Dynamics; thrust }
   /\   |                       CALL SERV06   {Servos Dynamics; surface deflection }
   /\   |                       CALL MOTN06   {Aircraft motion dynamics }
   /\   |=============================================================================|
   /\   |   System Identification Algorithms   CALL SYSIDP {RLS algorithm for Pitch Axis }
   /\   |                                      CALL SYSIDT {RLS algorithm for Throttle axis}
   /\   |-----------------------------------------------------------------------------|
   /\   |                PRINT DATA for plotting and analysis
   /\   |-----------------------------------------------------------------------------|
   /\   |                Increment Time                TIME = TIME + DT25
   /\   |-----------------------------------------------------------------------------|
   /\   | \                         Time less than Simulation time duration ?         /
   /\   | T \                                TIME < TSTIME                           / F
   /\   |--------------------------|--------------------------------------------------|
   /\   |   Branch to start        |
  <<<<< |   of the loop            |
        |--------------------------|=================|--------------------------------|
```

Fig A.2.2 SIM06 Batch Simulation Executive Program-simplified flow diagram

```
          |=================|
          ||   AFCS LOGIC  ||
|---------|=================|--------------------------------------------------------|
| \                                  Is SHOT mode  on ?                           /  |
| F \                                  SHOTM .EQ. 1                             / T  |
|----|-------------------------------------------------------------------------------|
|    | \  Is Time to turn on the adaptive controller and adaptive SHOT control is required?  / |
|    | T \                            TIME > TSELF .and. ASHOT .EQ. 1          / F  |
|    |--------------------------------------------|-----------------------------------|
|    |  ADAPTIVE SHOT CONTROLLER                  |  CONVENTIONAL SHOT                |
|    |                                            |  CONTROLLER                       |
|----|--------------------------------------------|-----------------------------------|
| \                                  Is SHE Mode ON ?                             /  |
| F \                                  SHEM .EQ. 1                              / T  |
|-----------------------------------------------------------|-----------------------|
| \                Is AH Mode ON ?                        / |SHE CONTROL  LAW      |
| F \               AHM .EQ. 1                          / T |{ Control Law for     |
|----|------------------------------------------------------|adaptive and          |
|    |          AH CONTROL LAW                              |Conventional          |
|    |  {Control Law for Adaptive and Conventional Controllers}|Control systems }    |
|----|------------------------------------------------------|-----------------------|
| \         Is time to turn on the adaptive controller and PAH adaptive control is required? |
| T \                              TIME > TSELF .and. APAH .EQ. 1                   |
|--------------------------------------------|--------------------------------------|
|  ADAPTIVE PAH CONTROL LAW                  |  CONVENTIONAL PAH                    |
|                                            |  CONTROL LAW                         |
|--------------------------------------------|--------------------------------------|
|          CONVENTIONAL LATERAL MOTION CONTROL -- ROLL AND RUDDER CONTROL            |
|                                                                                   |
|=====================|-------------------------------------------------------------|
||  END OF           ||
||  LOGIC            ||
|=====================|
```

Fig A.2.3 AFCS logic - simplified flow diagram

A.2.2 INIT06 Subroutine

The subroutine INIT06 provides executive logic to control the initialization of the batch simulation program for the QF-106 drone aircraft. INIT06 is called by the simulation executive subroutine SIM06. Figure A.2.4 is a simplified block diagram showing the flow of the subroutine INIT06.

It should be noted that the variable names used herein are intended to depict functional definitions and may be different from those used in the actual code of INIT06 and other simulation subroutines.

First, this program calls the subroutine INERT6 to compute the moments of inertia and the center of gravity of the aircraft for the flight condition specified by the user. Next, INIT06 converts the desired altitude into the equivalent vertical position coordinate Z, and resolves the desired speed through the heading angle to obtain east and north components of velocity. Radar altitude and some altitude reference points are also computed.

The remainder of the program is a loop which iterates to establish airframe trim conditions. The simulation subroutines which represent the basic vehicle are called in their normal sequence to calculate accelerations corresponding to the current attitude and control surface settings. The position and velocity terms are then reset to the initial values and new control surface commands are developed as inputs for the next pass. The commands are incremented in a direction which reduces the acceleration so that after a number of passes the loop will establish a state of equilibrium where the acceleration and the body rates are nulled.

```
         |=================|
         |    INIT06       |
|--------|=================|------------------------------------------------------------|
|  KTRIM1 = 0.08        {Normal acceleration feedback gain}                              |
|  KTRIM2 = -2.25       {Roll feedback gain}          KTRAT1 = 28.6  { pitch rate feedback gain}
|  KTRIM3 = -1.5        {Gain for throttle loop }     KTRAT2 = -57.2 { Roll rate feedback gain }
|-----------------------------------------------------------------------------------------|
|  Initialize Constants     RE = 20847225.4 {Radius of the earth }                        |
|                           MAXN = 500 {Max.number of iterations allowed to reach trim conditions}
|-----------------------------------------------------------------------------------------|
|  Calculate Moments of Inertia and Center of Gravity:   CALL INERT6                      |
|-----------------------------------------------------------------------------------------|
|  Set simulation geometry using X, Y, Altitude, Speed, and Heading                       |
|       SINPSI = SIN(PSI*D2R)     {SINE of aircraft heading}                               |
|       COSPSI = COS(PSI*D2R)     {Cosine of aircraft heading}                             |
|       ENUD(1) = ENU(1)          { X - position  }                                        |
|       ENUD(2) = ENU(2)          { Y - position  }                                        |
|       ENUD(3) = DSQRT( (HA+RE)**2 - ENUD(1)**2 - ENUD(2)**2 ) - RE                       |
|  Initialize local (inertial) velocities and accelerations and define altitude parameters
|       VL(1) = VR*SINPSI         {X-local axis velocity component}                        |
|       VL(2) = VR*COSPSI         {y-local axis velocity component}                        |
|       VL(3) = -(ENUD(1)*VL(1) + ENUD(2)*VL(2) ) / (RE + ENUD(3))                         |
|       AL(1) = 0    AL(2) = 0    AL(3) = 0                                               |
|       HGRND = 3950 + .002*ENU(1)     {Simulate terrain altitude in ft}                   |
|       HBARO = HA                     {Simulate baro metric altimeter}                    |
|       HRADAR = HBARO - HGRND         {Simulate radar altimeter}                          |
|-----------------------------------------------------------------------------------------|
|  Save velocity and position     XDS = VL(1)        XS= ENU(1)                           |
|                                 YDS = VL(2)        YS= ENU(2)                           |
|                                 ZDS = VL(3)        ZS= ENU(3)                           |
|----|------------------------------------------------------------------------------------|
>>>>>|   DO LOOP for NLOOP = 1 to MAXN:   Iterates to reach trim condition                 |
/\   |------------------------------------------------------------------------------------|
/\   |   Pitch control law   PPTRIM = PPTRIM + AL(3) * KTRIM1 * 2.75 * DT25               |
/\   |                       RHSCH = 0.24 + .00199 * (566 - VIAS)    {Airspeed gain schedule }
/\   |                       RKSCH = AMIN1(1.0, AMAX1(RKSCH, 0.24))                       |
/\   |                       DELTEC = (PPTRIM + AL(3) * KTRIM1 + KTRAT1 * PQR(2) ) * RKSCH |
/\   |------------------------------------------------------------------------------------|
/\   |   Roll control law    RHSCH = 0.05 + .00093 * (400 - VIAS)                         |
/\   |                       RKSCH = AMIN1(0.25, AMAX1(RKSCH, 0.05)) {Airspeed gain schedule }
/\   |                       RPROP = (PHI6D * KTRIM2 + KTRAT2 * PQR(1) ) * RKSCH          |
/\   |                       PRTRIM = PRTRIM + RPROP * 0.03 * DT25                        |
/\   |                       DELTAC = PRTRIM + RPROP                                      |
/\   |------------------------------------------------------------------------------------|
/\   | Throttle Control Law  DELTAT = DELTAT + KTRIM3 * (AL(2)*COSPSI + AL(1)*SINPSI)     |
/\   |====================================================================================|
/\   | VEHICLE MODEL: CALL ATMS06  {Simulate Atmospheric Data; MACH, IAS,...}             |
/\   |                CALL ENGIN6  {Simulate engine dynamics; thrust}                     |
/\   |                       THRUST = TDCMDL       {Override thrust}                      |
/\   |                       DELTAR = 0            {Force rudder to zero}                 |
/\   |                       YAWCD1 = 0            {Series rudder }                       |
/\   |                       YAWCD2 = 0            {Parallel rudder}                      |
/\   |                CALL SERV06   {Calculate surface deflections}                       |
/\   |                CALL MOTN06   {Simulate Aircraft Motion Dynamics}                   |
/\   |====================================================================================|
/\   |  Restore initial velocity   VL(1) = XDS       ENUD(1)= XS                          |
/\   |  and position               VL(2) = YDS       ENUD(2)= YS                          |
/\   |                             VL(3) = ZDS       ENUD(3)= ZS                          |
/\   |------------------------------------------------------------------------------------|
/\   | \  Continue DO LOOP ?        / |                                                   |
/\   | T \          NLOOP < MAXN   / F |                                                  |
/\   |-------------------------|--------|                                                  |
<<<<<|   Branch to loop start|                                                            |
|-----------------------------------------------------------------------------------------|
```

Figure A.2.4 INIT06 Subroutine- simpflified flow diagram

A.2.3 ATMS06 Subroutine

The subroutine ATMS06 is part of the QF-106 simulation, and provides a mathematical model of atmospheric properties and calculates miscellaneous air-data quantities needed in the simulation of the QF-106 drone aircraft. ATMS06 is called by the executive program SIM06.

Figure A.2.5 is a simplified block diagram showing the flow of the subroutine ATMS06.

The atmospheric model simulated is a simplification of the atmospheric model used at White Sands Missile Range [14]. It is simplified to the extend that it is limited to 60000 feet of altitude and does not compute pressure, temperature nor gravity.

The atmospheric density in slugs per cubic foot, ρ, and the speed of sound, C_{ss}, in feet per second, are computed using interpolative table look up methods. The data is presented such that for every 5000 feet altitude increment there is a corresponding value for speed of sound and air density.

The atmospheric model also calculates aerodynamic quantities such as Mach number, dynamic pressure and indicated airspeed. Mach number is the ratio of true airspeed (total velocity relative to air mass), Vr, and speed of sound C_{ss}.

$$\text{Mach} = \frac{V_r}{C_{ss}} \tag{A.1}$$

As illustrated in Equation (A.2) dynamic pressure is calculated from true airspeed and air density.

$$Q_B = \rho \frac{V_r^2}{2} \tag{A.2}$$

Indicated airspeed, V_{ias}, is a function of the compressibility factor Qc, Equation (A.3) and is calculated as shown in Equation (A.4). The airspeed error due to compressibility, VE, is determined from an airspeed table as function of Qc.

$$Qc = Q\ (1.0 + .269\ \text{Mach}^2) \tag{A.3}$$

$$V_{ias} = 25.68969\ (Qc)^2 + VE \tag{A.4}$$

```
|=================|
|    ATMS06       |
|=================|
|----------------------------------------------------------------------|
|  Find Index IALT and fraction DXH for altitude interpolation         |
|       HTAB(IALT)  =<  ALT  <  HTAB(IALT +1)                            |
|       DXH = ( ALT - HTAB(IALT) )/( HTAB(IALT +1) - HTAB(IALT) )        |
|  Interpolate airdensity and speed of sound                           |
|       RHO = AIRDEN(IALT)*(1.-DXH) + DXH*AIRDEN(IALT+1)                 |
|       SS = SSOUND(IALT)*(1.-DXH) + DXH*SSOUND(IALT+1)                  |
|----------------------------------------------------------------------|
|  Calculate Mach, dynamic pressure, impact pressure                   |
|       MACH = VR/SS                                                    |
|       QBAR = .5*RHO*VR*VR                                             |
|       QC = QBAR*(1. + .269*MACH*MACH)                                 |
|----------------------------------------------------------------------|
|  Find Index IQC and fraction DQC for interpolation                   |
|       QCTAB(IQC)  =<  QC  <  QCTAB(IQC + 1)                           |
|       DQC = ( QC - QCTAB(IQC) )/( QCTAB(IQC + 1) - QCTAB(IQC) )       |
|  Interpolate correction factor for indicated airspeed                |
|       VE = TVIAS(IQC)*(1.-DQC) + TVIAS(IQC + 1)*DQC                   |
|  Calculate Indicated Airspeed -- in knots                            |
|       VIAS = FPS2KT*( 25.6869 + SQRT(QC) + VE )    {FPS2KT=0.5924}    |
|----------------------------------------------------------------------|
|=================|
|    RETURN       |
|=================|
```

Figure A.2.5 ATMS06 Subroutine - simplified flow diagram

A.2.4 INERT6 Subroutine

The subroutine INERT6 provides the capability to calculate moments-of-inertia and center-of-gravity (CG) terms for the QF-106 aircraft. INERT6 is called by the subroutine INIT06 which is called by SIM06. The inertia and CG data are used in the subroutine MOTN06 in the simulation representation of the rotational and translational dynamics of the QF-106 drone aircraft.

The subroutine INERT6 includes a subroutine call to INTPAR as part of a table lookup and interpolation algorithm. Details of the INTPAR subroutine are given in Subsection A.2.10

Figure A.2.6 is a simplified block diagram showing the flow of the subroutine INERT6. The first section calculates the total aircraft weight and mass. The weight is used as the independent variable in a table lookup and interpolation algorithm which calculates a set of moments-of-inertia and center-of-gravity terms for subsonic and supersonic speeds.

```
|=================|
|     INERT6      |
|=================|
|------------------------------------------------------------------------------------------|
|   Calculate total weight and mass                          {WEMPTY = 25693 lbs}          |
|                                                                                          |
|        WEIGHT = WEMPTY + WFUEL                                                           |
|        MASS = WEIGHT / GRAVITY                                                           |
|==========================================================================================|
|   Subroutine to find Index IW and fraction DXW for weight interpolation                  |
|   CALL:INTPAR( WEIGHT , WTAB , 12 , IW , DXW )                                           |
|                  Returns IW and DXW such that:                                           |
|                  WTAB(IW)  =<  WEIGHT  <  WTAB(IW +1)                                    |
|                  DXW = (WEIGHT - WTAB(IW) )/(WTAB(IW +1) - WTAB(IW) )                    |
|------------------------------------------------------------------------------------------|
| \                               Is Aircraft Supersonic ?                              /  |
| T \                                     MACH > 1.0                                  / F  |
|--------------------------------------------------------|---------------------------------|
|                                                        | Interpolate  supersonic         |
|  Interpolate first subsonic moment-of-inertia for weight| moment of inertia for weight   |
|  IXXSUB = TXXSUB( IW )*(1 - DXW) + TXXSUB( IW+1 )*DXW   | IXXSUP=TXXSUP(IW)*(1-DXW)       |
|  Interpolate other subsonic MOIs and CGs                |   + TXXSUP(IW+1)*DXW            |
|  IYYSUB = TYYSUB( IW )*(1 - DXW) + TYYSUB( IW+1 )*DXW   | IYYSUP=TYYSUP(IW)*(1-DXW)       |
|  IZZSUB = TZZSUB.........                               |   + TYYSUP(IW+1)*DXW            |
|  IXZSUB = TXZSUB.........                               | IZZSUP = TZZSUP.........        |
|  XCGSUB = TXCGSB.........                               | IXZSUP = TXZSUP.........        |
|  ZCGSUB = TZCGSB.........                               | XCGSUP = TXCGSP.........        |
|                                                        | ZCGSUP = TZCGSP.........        |
|--------------------------------------------------------|---------------------------------|
|=================|
|     RETURN      |
|=================|
```

Fig A.2.6 INERT6 Subroutine - simplified flow diagram

A.2.5 ENGIN6 Subroutine

This subroutine provides the capability to calculate the total thrust generated by the QF-106 aircraft engine. ENGIN6 is called by the executive routine SIM06.

Figure A.2.7 is a simplified flow diagram of the subroutine ENGIN6.

The subroutine ENGIN6 includes a two dimensional table lookup algorithm of thrust which is a function of mach and altitude. Repeated calls are made to INTPAR with different size Mach lists, including different ranges of Mach, to provide the correct interpolation for different engine thrust tables.

There are two sets of data for steady state thrust versus throttle position. One set is used to compute the thrust when the afterburner is off, the other one is used to calculate the thrust when the afterburner is on. The throttle position is expressed as percent of maximum, rather than in degrees, and thrust is normalized relative to mil-power thrust. The desired thrust, referred to here as thrust command TFCMD, is interpolated as a function of mach and altitude, using endpoint and slope terms established by the table lookup algorithm in ENGIN6.

```
|==================|
|    ENGIN6        |
|==================|
|==========================================================================================|
||  Table lookup of Thrust and Fuel Flow parameters -- functions of Mach and altitude     ||
||                            Thrust Parameters Calculated: TFIDLE,TFMIL,TFNORM,ABMAX      ||
|==========================================================================================|
|   Calculate thrust force for minimum throttle setting in afterburner range               |
|        ABMIN = 0.75*TFMIL + 0.25*ABMAX                                                   |
|------------------------------------------------------------------------------------------|
| \  Throttle at Idle?                                                                   / |
| T \          DELTAT < THRMIN    {THRMIN=8 deg}                                       / F |
|----------------------------|-------------------------------------------------------------|
|   Minimum thrust command   | \  Throttle below AB range?                             /   |
|   TFCMD = 0.9*TFIDLE       | T \            DELTAT < 0.8*THRMAX     {THRMAX=45 deg}  / F |
|   ABTIME = 0.              |----|--------------------------------------------------------|
|                            |    | \  Afterburner command OFF                         /    |
|                            |    | T \          ABURNR = 0 or DELTAT < 25            / F  |
|                            |----|-------------------------------|------------------------|
|                            |   Normal range thrust command     |  AB range thrust command |
|                            |   TFA = TFIDLE                    |  ABTIME = 5.            |
|                            |   TFB = DELTAT                    |  TFA = ABMIN            |
|                            |   TFC = TFMIL - TFIDL             |  TFB = ABMAX - ABMIN    |
|                            |   TFD = 0.8*THRMAX                |  TFC = DELTAT -0.8*THRMAX|
|                            |   TFCLIM = 1.25*TFMIL - 0.25*TFIDLE| TFD = 0.2*THRMAX       |
|                            |                                   |  TFCLIM = ABMAX         |
|                            |-----------------------------------|-------------------------|
|                            |   TFCMD = TFA + TFB*TFC/TFD                                 |
|                            |   Limit; Max TFCMD = TFCLIM                                 |
|----------------------------|-------------------------------------------------------------|
|==================|
|    RETURN       ||
|==================|
```

Fig A.2.7 ENGIN6 Subroutine - simplified flow diagram

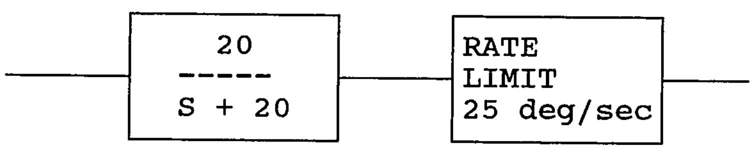
A.2.6 SERV06 Subroutine

The subroutine SERV06 is part of the QF-106 simulation, and provides the capability to simulate the elevon and rudder servos and to calculate the control surface deflections of the QF-106 drone aircraft. SERV06 is called by the vehicle executive program, SIM06.

Figure A.2.8 is a simplified flow diagram of the subroutine SERV06.

(1) Elevon Servo Algorithm

The simulation model for the Hydraulic Elevon Package (HEP-valve) elevon servos is a simple lag with a bandwidth of 20 rad/sec and a rate limit of 25 deg/sec.

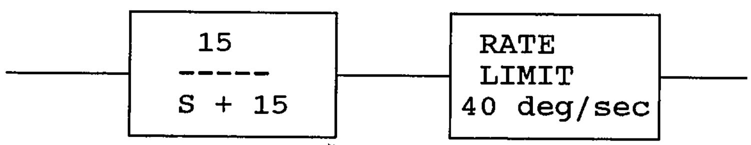


Elevon command limits are applied first to the elevator and to aileron command quantities established by the pitch and roll control loops, respectively. Then, the total right and left elevon commands are formed, limited individually, and integrated with appropriate rated limits to yield individual elevon deflections. These are recombined into equivalent elevator and aileron deflections.

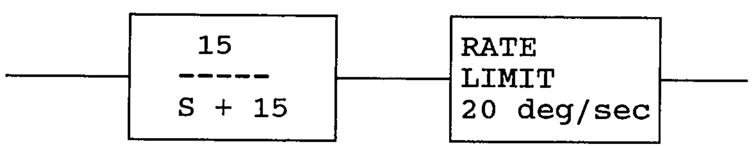
(2)Rudder servo algorithm

The design includes two servos; a "series rudder" servo with 6 degrees of authority, and a "parallel rudder" servo with +&- 20 degrees of authority. Each servo is modeled as a rate-limited simple lag.

Series Rudder Servo



Parallel Rudder Servo



```
|=================|
|    SERV06       |
|=================|
|-------------------------------------------------------------------------------------|
|   Servo parameters                                                                   |
|  HRPBW = 15  {Hydraulic Rudder bandwidth}     ERLIM = 24 {elevon rate limits}        |
|  EMRBW = 15  {Electr Mech Rudder bandwidth}   EBW   = 20  {Elevon bandwidth}         |
|  RRLIMS =40 {Series rudder rate limit}        RRLIMP =20 {parallel rudder rate limit}|
|-------------------------------------------------------------------------------------|
|  Limit surface deflection commands                                                   |
|       Limit elevator: DELTEC between -25 and +8 deg     {Elevator is positive up}     |
|       Limit aileron:  DELTAC between -7 and +7 deg      {Aileron is positive with right up} |
|-------------------------------------------------------------------------------------|
|  Individual Elevon commands                         {Elevons are positive up}        |
|       ERHTC = -DELTEC - DELTAC                               ELFTC =  - DELTEC + DELTAC |
|       Limit : Elevon commands ERHTC & ELFTC between -15 and +32 deg                   |
|-------------------------------------------------------------------------------------|
|  Individual Elevon servos                                                            |
|       EERR = EBW*(ERHTC - ERIGHT)                    EERR = EBW*(ELFTC - ELEFT)        |
|       Limit rate: EERR between +&- ERLIM             Limit rate: EERR between +&- ERLIM |
|       ERIGHT = ERIGHT + DT25*EERR                    ELEFT = ELEFT + DT25*EERR         |
|  Reform equivalent elevator and aileron deflections                                  |
|       DELTAE = (-0.5)*( ERIGHT + ELEFT )                                              |
|       DELTAA = 0.5*( ERIGHT - ELEFT )                                                 |
|-------------------------------------------------------------------------------------|
|   Series Rudder servo and rate limit                                                 |
|       EERR = HRPBW*( YAWCD2 - DELTRS )                                                |
|       Limit rate: EERR between +&- RRLIMS                                             |
|       DELTRS = DELTRS + DT25*EERR                                                     |
|   Parallel Rudder servo and rate limit                                               |
|       EERR = EMRBW*( YAWCD1 - DELTRP )                                                |
|       Limit rate: EERR between +&- RRLIMP                                             |
|       DELTRP = DELTRP + DT25*EERR                                                     |
|                                                                                      |
|  Limit total rudder deflection                                                       |
|       DELTAR = DELTRS + DELTRP                                                        |
|       Limit total rudder: -24 < DELTAR < +24 deg                                      |
|-------------------------------------------------------------------------------------|
|=================|
|    RETURN       |
|=================|
```

Fig A.2.8 SERV06 Subroutine - simplified flow diagram

A.2.7 MOTN06 Subroutine

The subroutine MOTN06 provides a simulation representation of the rotational and translational dynamics of the QF-106 aircraft. This includes subroutine calls for table lookup of aerodynamic coefficients and physical characteristics, calculation of total forces and moments, calculation of angular and translational accelerations, and integration of these accelerations to update the body rates, attitudes, position and velocity of the simulated aircraft. The equations of motion are integrated at 40 times per second. The simulation development include tradeoffs of processing efficiency allowing some aero coefficients to be looked up at a lower rate provided there is no degradation in the integrity of the simulation of aircraft dynamic maneuvers.

Figure A.2.9 is a simplified flow diagram showing the top level program flow for the MOTN06 subroutine. The following paragraphs review this flow in detail.

(1) Earth Fixed (Inertial) to Body Transformation Matrix

$$T_{IB} = \begin{vmatrix} T_{11} & T_{12} & T_{13} \\ T_{21} & T_{22} & T_{23} \\ T_{31} & T_{32} & T_{33} \end{vmatrix} \qquad (A.5)$$

where

$$T_{11} = \sin(\Psi)\cos(\Theta)$$

$$T_{12} = \cos(\Psi)\cos(\Theta)$$

$$T_{13} = \sin(\Theta)$$

$$T_{21} = \sin(\Theta)\sin(\Phi)\sin(\Psi) + \cos(\Theta)\cos(\Psi)$$

$$T_{22} = \sin(\Theta)\sin(\Phi)\cos(\Psi) - \cos(\Phi)\sin(\Psi)$$

$$T_{23} = -\cos(\Theta)\sin(\Phi)$$

$$T_{31} = \sin(\Theta)\cos(\Phi)\sin(\Psi) - \sin(\Phi)\cos(\Psi)$$

$$T_{32} = \sin(\Theta)\cos(\Phi)\cos(\Psi) + \sin(\Phi)\sin(\Psi)$$

$$T_{33} = -\cos(\Theta)\cos(\Phi).$$

The angles Θ, Ψ, Φ describe the orientation of the body axis relative to a translated Earth fixed coordinate system X_L, Y_L, Z_L. These angles are frequently referred to as the Euler angles. Θ is referred to as the pitch angle, Ψ is called the heading angle and Φ is named the roll (or bank) angle

(2) Total Aerodynamic Velocity and Angle of Attack

Total aero velocity (inertial)= velocity (inertial)- Wind velocity

$$X'_i = X'_i - Wx$$

$$Y'_i = Y'_i - Wy \qquad \text{(A.6)}$$

$$Y'_i = Y'_i - Wz$$

```
      |=================|     Subroutine called
      |    MOTN06       |     by simulation
      |=================|     executive SIM06
              \|/
|---------------------------------------------------------------------|
|   COMPUTE:  LOCAL TO BODY TRANSFORMATION MATRIX                      |
|---------------------------------------------------------------------|
              \|/
|---------------------------------------------------------------------|
|   COMPUTE: TOTAL AERODYNAMIC VELOCITY IN INERTIAL FRAME              |
|            TRANSFORM TO BODY FRAME                                   |
|---------------------------------------------------------------------|
              \|/
      |---------------------------|
      |   COMPUTE:                |
      |   ANGLE OF ATTACK         |
      |   AOA RATE                |
      |   SIDESLIP ANGLE          |
      |---------------------------|
              \|/
|=====================================================|
|   SUBROUTINE TO COMPUTE TOTAL AERO COEFFICIENTS      |
|            CALL:COEF06                               |
|=====================================================|
              \|/
|-----------------------------------------------|
|   COMPUTE:  TOTAL FORCES IN STABILITY AXES     |
|             TOTAL MOMENTS IN BODY AXES         |
|-----------------------------------------------|
              \|/
|-----------------------------------------------|
|   TRANSFORM:                                   |
|   FORCES FROM STABILITY TO BODY AXES           |
|-----------------------------------------------|
              \|/
|-----------------------------------------|
|   COMPUTE:                               |
|   ROTATIONAL DYNAMICS EQUATIONS OF MOTION|
|-----------------------------------------|
              \|/
|-----------------------------------------|
|   COMPUTE AND INTEGRATE                  |
|   EULER ANGLE RATES AND ANGLES           |
|-----------------------------------------|
              \|/
|-----------------------------------------------|
|   TRANSFORM:                                   |
|   FORCES FROM BODY TO EARTH FIXED (INERTIAL) AXES |
|-----------------------------------------------|
              \|/
|-----------------------------------------------------|
|   COMPUTE AND INTEGRATE:                             |
|   TRANSLATIONAL ACCELERATIONS, VELOCITIES, POSITION  |
|-----------------------------------------------------|
              \|/
|-----------------------------------------------------------------|
|   COMPUTE :                                                      |
|   ALTITUDE, ALTITUDE RATE, NORMAL AND LONG ACCELERATION OF CG    |
|-----------------------------------------------------------------|
              \|/
      |=================|
      |    RETURN       |
      |=================|
```

Fig A.2.9 MOTN06 Subroutine - simplified flow diagram

Total aero velocity in body coordinates

$$\begin{vmatrix} U \\ V \\ W \end{vmatrix} = T_{IB} \begin{vmatrix} X_i' \\ Y_i' \\ Z_i' \end{vmatrix} \tag{A.7}$$

Angle of attack and angle of attack rate

$$\alpha = \tan^{-1}\left|\frac{W}{U}\right|$$

$$\alpha' = \frac{\alpha(k) - \alpha(k-1)}{T} \tag{A.8}$$

Total velocity relative to air-mass

$$Vr = \sqrt{(U)^2 + (V)^2 + (W)^2} \tag{A.9}$$

Sideslip angle

$$\beta = \sin^{-1}\left|\frac{V}{V_r}\right| \tag{A.10}$$

(3) Total Aerodynamic Coefficients

The subroutine COEF06 is called to calculate the total aerodynamic coefficients. See Subsection A.2.8 for details of COEF06. This subroutine, in turn, calls the COEF3 and COEF8 subroutines to lookup and extrapolate different coefficients. COEF06 returns the following quantities:

CL -- Total lift force coefficient
CD -- Total drag force coefficient
CY -- Total side force coefficient

CM -- Total pitching moment coefficient
CN -- Total yawing moment coefficient
CLL -- Total rolling moment coefficient

(4) Total Forces and Moments in the stability axis

$$L_f = CL\ Q_B\ S_w \quad ---- \quad \text{Lift Force} \tag{A.11}$$

$$D_f = CD\ Q_B\ S_w \quad ---- \quad \text{Drag Force} \tag{A.12}$$

$$S_f = CY\ Q_B\ S_w \quad ---- \quad \text{Side Force} \tag{A.13}$$

$L_s = CLL\ Q_B\ S_w\ B_B$ ---- Rolling Moment (A.14)

$M_s = CM\ Q_B\ S_w\ C_B$ ---- Pitching Moment (A.15)

$N_s = CN\ Q_B\ S_w\ B_B$ ---- Yawing Moment (A.16)

where B_B is the wing span

C_B is the mean aerodynamic chord

S_w is the wing surface area

(5) Transform Total Forces and Moments to the Body Axes

$$F_x = L_f\ \sin(\alpha) - D_f \cos(\alpha) + \text{thrust} \cos(2.06°) \quad (A.17)$$

$$F_y = S_f \quad (A.18)$$

$$F_z = -L_f \cos(\alpha) - D_f \sin(\alpha) - \text{thrust} \quad (A.19)$$

$$L = -L_s\cos(\alpha) - N_s \sin(\alpha) \quad (A.20)$$

$$M = -M_s + 0.125\ \text{thrust} \quad (A.21)$$

$$N = L_s \sin(\alpha) + N_s \cos(\alpha) \quad (A.22)$$

(6) Rotational Dynamics Equations -- Angular Accelerations

Angular Accelerations, Euler angular rates and angles

$$P' = \frac{(I_{zz}\,AA + I_{xz}\,CC)}{I_{xx}\,I_{zz} - I_{xz}^2} \quad \text{---- body roll acceleration} \qquad (A.23)$$

$$Q' = \frac{BB}{I_{yy}} \quad \text{---- body pitch acceleration} \qquad (A.24)$$

$$R' = \frac{(I_{xx}\,CC + I_{xz}\,AA)}{I_{xx}\,I_{zz} - I_{xz}^2} \quad \text{---- body yaw acceleration} \qquad (A.25)$$

where

AA = L - Q * R (Izz - Ixx) + P * Q * Ixz

BB = M - P * R (Ixx - Izz) + (R^2 + P^2) * Ixz

CC = M - P * Q (Iyy - Ixx) + Q * R * Ixz

DD = Ixx * Izz - Ixz * Ixz

where

P = Body roll rate

Q = Body pitch rate

R = Body yaw rate

Ixx = Moment of inertia about x-axis

Iyy = Moment of inertia about y-axis

Izz = Moment of inertia about z-axis

Ixz = Product of inertia about y-axis.

The inertia terms Ixx, Iyy, Izz and Ixz are provided by the subroutine INERT6. This subroutine looks up values tabulated as a function of weight, for subsonic and supersonic flight conditions.

(7) Compute and integrate Euler angle rates and angles

Integrating angular accelerations to get body rates

$$P = \int P' dt$$
$$Q = \int Q' dt \qquad (A.26)$$
$$R = \int R' dt$$

Compute Euler angle rates

$$\Phi' = P + [Q \cdot \sin(\Phi) + R \cdot \cos(\Phi)] \tan(\Theta)$$

$$\Theta' = Q \cdot \cos(\Phi) - R \cdot \sin(\Phi) \qquad (A.27)$$

$$\Psi' = Q \cdot \sin(\Phi) + R \cdot \cos(\Phi) \cdot \sec(\Theta)$$

Obtain the Euler angles; pitch, roll, and heading, by integrating the Euler angular rates.

$$\Phi = \int \Phi' dt$$
$$\theta = \int \Theta' dt \qquad (A.28)$$
$$\Psi = \int \Psi' dt$$

(8) Translational Accelerations, Velocities, and Position

The first step is to transform the body forces to the local, inertial, coordinate system using the inverse (or transpose) of the transformation matrix T_{IB}, Equation (A.5).

$$\begin{vmatrix} F_{xi} \\ F_{yi} \\ F_{zi} \end{vmatrix} = (T_{IB})^{-1} \begin{vmatrix} F_x \\ F_y \\ F_z \end{vmatrix} \qquad \text{(A.29)}$$

Using Equation (A.29), the mass of the aircraft, m, and the earth gravity constant, g, we compute the translational accelerations

$$\begin{aligned} X_i'' &= \frac{F_{Xi}}{m} \\ Y_i'' &= \frac{F_{Yi}}{m} \\ Z_i'' &= \frac{F_{Zi}}{m} - g \end{aligned} \qquad \text{(A.30)}$$

Integrating the above accelerations, Equations (A.30), we obtain the inertial velocities

$$\begin{aligned} X_i' &= \int X_i'' \, dt \\ Y_i' &= \int Y_i'' \, dt \\ Z_i' &= \int Z_i'' \, dt \end{aligned} \qquad \text{(A.32)}$$

Integrating the velocities, Equations (A.31), we obtain the aircraft position in the inertial axis.

$$X_i = \int X_i' \, dt$$
$$Y_i = \int Y_i' \, dt \qquad (A.32)$$
$$Z_i = \int Z_i' \, dt$$

(9) Calculate different simulator parameters such as aircraft altitude, h, altitude rate, h', and normal acceleration, a_z, in g's

$$h = \sqrt{\left[X_i^2 + Y_i^2 + (Z_i + RE)^2 \right]} - RE \qquad (A.33)$$

$$h' = \frac{X_i \cdot X_i' + Y_i \cdot Y_i' + (Z_i + RE) \cdot Z_i'}{h + RE} \qquad (A.34)$$

where RE is the radius of the Earth

$$a_z = \frac{-X_i \cdot T_{31} - Y'' \cdot T_{32} - (Z'' + g) \cdot T_{33}}{g} \qquad (A.35)$$

where g = gravity = 32.2 ft/sec^2

A.2.8 COEF06 Subroutine

The subroutine COEF06 provides the capability to calculate aerodynamic coefficients and other physical properties of the QF-106 drone aircraft which are needed in the simulation representation of the rotational and translational dynamics of the QF-106 drone target. This includes subroutine calls for table lookup of individual aerodynamic coefficients and physical characteristics, and then, calculation of the following total forces and moment coefficients:

CL -- Total lift force coefficient

CD -- Total drag force coefficient

CY -- Total side force coefficient

CM -- Total pitching moment coefficient

CLN -- Total yawing moment coefficient

CLL -- Total rolling moment coefficient

The coefficients are looked up at the rate of 8 times per second. However, the calculation of the total force and moment coefficients is at the same rate that the equations of motion are integrated, 40 times per second to include the effects of rapid changes in surface deflections and body rates. One exception is the drag coefficient, which is looked up every cycle ,since it is a function of CL and the elevator, δe, which may change rapidly.

Figure A.2.10 is a simplified flow diagram showing the top level program flow for the COEF06 subroutine. The following paragraphs review this flow in detail. The discussion presents each of the equations for the six total force and moment coefficients which are the outputs of the subroutine.

(1) Lift force coefficient

$$CL = CL_0 + CL_\alpha \cdot \alpha + CL_{\delta a} \cdot \delta e \qquad (A.36)$$

CL_0 - Lift Coefficient for zero angle of attack

CL_α - Airplane lift curve slope

$CL_{\delta e}$- Variation of lift coefficient with elevator angle.

α - Angle of Attack

δe - Elevator position

(2) Drag force coefficient

$$CD = CD6 \qquad (A.37)$$

CD6 - Airplane drag coefficient without increments for tanks, etc

(3) Side force coefficient

$$CY = CY_\beta \cdot \beta + CY_{\delta a} \cdot \delta a + CY_{\delta r} \cdot \delta r + B_B \cdot (CY_p \cdot P + CY_r\ R)/(2 \cdot Vr) \qquad (A.38)$$

CY_{β} -Variation of side force coefficient with side-slip angle.

$CY_{\delta a}$ -Variation of side force coefficient with aileron angle.

$CY_{\delta r}$ -Variation of side force coefficient with rudder angle.

CY_p -Variation of side force coeff. with roll rate.

CY_r -Variation of side force coeff. with yaw rate.

Vr -True airspeed

β -Side slip angle

δa -Aileron position

δr -Rudder position

P -Roll rate

R -Yaw rate

B_B -Wing span

(4) Rolling moment coefficient

$$CLL=CLL_{\beta}\ \beta + CLL_{\delta a}\ \delta a + CLL_{\delta r}\cdot \delta r + B_B\cdot(CLL_p\cdot P + CLL_r\cdot R)/(2\cdot Vr) \quad (A.39)$$

CLL_{β} -Variation of rolling moment coeff. with sideslip angle.

$C_{\delta a}$-Variation of rolling moment coeff. with aileron angle (i.e. lateral control power).

$C_{\delta r}$-Variation of rolling moment coeff. with rudder angle.

CL_r-Variation of rolling moment coeff. with yaw rate.

CL_p-Variation of rolling moment coeff. with roll rate.

(5) Pitching moment coefficient

$$CM = CM_0 + CM_{\delta e} \cdot \delta e + [\ CL \cdot \cos(\alpha) + CD \cdot \sin(\alpha)\ ] \cdot (xnp - cg) + CM_{qd} \cdot [\ d\alpha/dt\ ] \cdot c_B / (2 \cdot Vr) \qquad (A.40)$$

CM_0 - Pitching moment coefficient for zero angle of attack and zero elevator angle.

$CM_{\delta e}$- Variation of pitch moment coeff. with elevator angle.

CM_{qd}- Variation of pitch moment coeff. with dq/dt.

xnp - Longitudinal center of pressure

cg - center of gravity

c_B - Mean aerodynamic chord

(6) Yawing moment coefficient

$$CN = CN_{\beta} \cdot \beta + CN_{\delta a} \cdot \delta a + CN_{\delta r} \cdot \delta r + B_B \cdot (CN_p \cdot P + CN_r \cdot R) \qquad (A.41)$$

CN_{β} - Variation of yawing moment coeff. with sideslip angle.

$CN_{\delta a}$- Variation of yawing moment coeff. with aileron angle.

$CN_{\delta r}$- Variation of yawing moment coeff. with rudder angle.

CN_p - Variation of yawing moment coeff. with roll rate.

CN_r - Variation of yawing moment coeff. with yaw rate.

```
|=================|
|    COEF06       |
|=================|
|---------------------------------------------------------------------------------------|
|    Calculate dimensional/velocity terms:                                              |
|         CBAR2V =0.5*CBAR/VR ;                BBAR2V = 0.5*BBAR/VR                      |
|---------------------------------------------------------------------------------------|
| \  Test fast loop counter -- time to update coeff?                                  / |
| T \                                          ICNT = 5 ?                            / F |
|===========================================|-------------------------------------------|
||   Moments of Inertia Subroutine          ||                                          |
||        CALL:INERT6                       ||                                          |
|===========================================|                                           |
||   Mach and altitude Aero Table lookup Subr ||                                        |
||        CALL:COEF3                        ||                                          |
|===========================================|-------------------------------------------|
|    Convert Deflections and Angles to Radians                                          |
|         ALPHAR = D2R * ALPHA                                                          |
|         DELTER = D2R * DELTAE                                                         |
|         DELAAR = D2R * DELTAA                                                         |
|         DELTRR = D2R * DELTAR                                                         |
|         BETAR  = D2R * BETA                                                           |
|---------------------------------------------------------------------------------------|
|    Calculate total lift force coefficient                                             |
|         CL = ...see text....                                                          |
|=======================================================================================|
||   Three-dimensional Drag Coeff Table lookup Subr                                    ||
||        CALL:COEF8                                                                   ||
|=======================================================================================|
|    Calculate total drag force coefficient                                             |
|         CD = ...see text....                                                          |
|---------------------------------------------------------------------------------------|
|    Calculate total side force coefficient                                             |
|         CY = ...see text....                                                          |
|---------------------------------------------------------------------------------------|
|    Calculate total pitching moment coefficient                                        |
|         CM = ...see text....                                                          |
|---------------------------------------------------------------------------------------|
|    Calculate total yawing moment coefficient                                          |
|         CLN = ...see text....                                                         |
|---------------------------------------------------------------------------------------|
|    Calculate total rolling moment coefficient                                         |
|         CLL = ...see text....                                                         |
|---------------------------------------------------------------------------------------|
|=================|
|    RETURN       |
|=================|
```

Fig A.2.10 COEF06 Subroutine - simplified flow diagram

A.2.9 COEF3 and COEF8 Subroutines

Subroutines COEF3 and COEF8 provide the capability for table lookup and interpolation of the aerodynamic coefficients used by COEF06 to compute the total aerodynamic coefficients listed in the preceding subsection. The tabulated aero data is from the AERCOM data common. All functions performed by the subroutines are basically the same. The primary difference between these two subroutines is the size of the tables processed, i.e., the number of independent variables and the number of breakpoints for each variable.

Figures A.2.11 is a simplified block diagrams of COEF3 subroutine which illustrates the table lookup and interpolation performed by this subroutine.

The INTPAR subroutine first uses the input value of Mach to find the interpolation lower index point and the fraction value from the table of Mach break points. Then, similar values are found for the altitude input value. INTPAR includes tests for Mach and altitude values outside the table limits.

For each of the different coefficients processed, two coefficient values are interpolated as a function of Mach for the two altitudes bracketing the input. Then an interpolation is made between these values as a function of altitude to get the output value for the particular coefficient.

Figure A.2.12 is a simplified block diagram for COEF8 showing the three-dimensional table lookup and interpolation procedure needed to determine the drag coefficient, CD6, as a function of Mach number, lift coefficient, and elevator surface deflection. This algorithm is a direct extension of the two-dimensional case to add a third interpolation.

```
|=============|
|   COEF3     |
|=============|
|==========================================================================================
|   Subroutine to find Index IM and fraction DXM for Mach interpolation                  |
|   CALL:INTPAR( MACH, TM43 , 43 , IM , DXM )                                            |
|                  Returns IM and DXM such that:                                         |
|                  TM43(IM)  =<  MACH  <  TM43(IM +1)                                    |
|                  DXM = (MACH - TM43(IM) )/(TM43(IM +1) - TM43(IM) )                    |
|==========================================================================================
|   Subroutine to find Index IH and fraction DXH for altitude interpolation              |
|   CALL:INTPAR( ALT , TALT06 , 6 , IH , DXH )                                           |
|                  Returns IH and DXH such that:                                         |
|                  TALT06(IH)  =<  ALT  <  TALT06(IH +1)                                 |
|                  DXH = (ALT - TALT06(IH) )/(TALT06(IH +1) - TALT06(IH) )               |
|==========================================================================================
|   Interpolate coefficient for Mach at lower altitude                                   |
|        VAL1 = TCLO(IM,IH)*(1 - DXM) + TCLO(IM+1,IH)*DXM                                |
|   Interpolate coefficient for Mach at higher altitude                                  |
|        VAL2 = TCLO(IM,IH+1)*(1 - DXM) + TCLO(IM+1,IH+1)*DXM                            |
|   Interpolate for altitude to get coefficient                                          |
|        CLO = VAL1*(1 - DXH) + VAL2*DXH                                                 |
|-----------------------------------------------------------------------------------------|
|   Interpolate TCLA ; Calculate CLA           Interpolate TCLLPA2 ; Calculate CLLPA2    |
|   Interpolate TCLDE : Calculate CLDE         Interpolate TCNP : Calculate CNP          |
|   Interpolate TXNP : Calculate XNP           Interpolate TCYR : Calculate CYR          |
|   Interpolate TCYBA2 ; Calculate CYBA2       Interpolate TCLLRO ; Calculate CLLRO      |
|   Interpolate TCLLBO : Calculate CLLBO       Interpolate TCLLRA : Calculate CLLRA      |
|   Interpolate TCLBB2 : Calculate CLLBA2      Interpolate TCNR : Calculate CNR          |
|   Interpolate TCYP ; Calculate CYP           Interpolate TCLLDA ; Calculate CLLDA      |
|   Interpolate TCLLPO : Calculate CLLPO       Interpolate TCLNDAO : Calculate CNDAO     |
|   Interpolate TCLLPA : Calculate CLLPA       Interpolate TCYDR : Calculate CYDR        |
|   Interpolate TCLLDR ; Calculate CLLDR                                                 |
|   Interpolate TCYBB : Calculate CYBB                                                   |
|   Interpolate TCLLBB : Calculate CLLBB                                                 |
|   Interpolate TCNBB ; Calculate CNBB                                                   |
|   Interpolate TCYBO : Calculate CYBO                                                   |
|   Interpolate TCNBA2 : Calculate CNBA2                                                 |
|   Interpolate TCNDAA  :Calculate CNDAA                                                 |
|   Interpolate TVNDAD : Calculate CNDADE                                                |
|==========================================================================================
|   Subroutine to find Index IM and fraction DXM for Mach interpolation                  |
|   CALL:INTPAR( MACH, TM66, 66 , IM , DXM )                                             |
|   Interpolate TCMO ; Calculate CMO                                                     |
|-----------------------------------------------------------------------------------------|
|   Subroutine to find Index IM and fraction DXM for Mach interpolation                  |
|   CALL:INTPAR( MACH, TM57, 57 , IM , DXM )                                             |
|-----------------------------------------------------------------------------------------|
|   Interpolate TCMQD ; Calculate CMQD      Interpolate TCNBA : Calculate CNBA           |
|   Interpolate TCMDE : Calculate CMDE                                                   |
|   Interpolate TCNBO : Calculate CNBO                                                   |
|-----------------------------------------------------------------------------------------|
|   Subroutine to find Index IM and fraction DXM for Mach interpolation                  |
|   CALL:INTPAR( MACH, T2M41, 41 , IM , DXM )                                            |
|-----------------------------------------------------------------------------------------|
|   Interpolate TCLLBA ; Calculate CLLBA                                                 |
|   Interpolate TCNDR : Calculate CNDR                                                   |
|---|================|--------------------------------------------------------------------|
    ||  RETURN       ||
    ||               ||
    |================|
```

Fig A.2.11 COEF3 Subroputine- simplified flow diagram

```
|=============|
|    COEF8    |
|=============|
|==========================================================================================
|    Subroutine to find Index IM and fraction DXM for Mach interpolation                  |
|    CALL:INTPAR( MACH, TM19 , 19 , IM , DXM )                                            |
|                   Returns IM and DXM such that:                                         |
|                   TM19(IM)  =<  MACH  <  TM19(IM +1)                                    |
|                   DXM = (MACH - TM19(IM) )/(TM19(IM +1) - TM19(IM) )                    |
|==========================================================================================
|    Subroutine to find Index IL and fraction DXL for lift coefficient interpolation      |
|    CALL:INTPAR( CL , TCL27 , 27 , IL , DXL )                                            |
|                   Returns IL and DXL such that:                                         |
|                   TCL27(IL)  =<  CL  <  TCL27(IL +1)                                    |
|                   DXL = ( CL - TCL27(IL) )/(TCL27(IL +1) - TCL27(IL) )                  |
|==========================================================================================
|    Subroutine to find Index IE and fraction DXE for elevator interpolation              |
|    CALL:INTPAR( DELTAE , TDE07 , 7 , IE , DXE )                                         |
|                   Returns IE and DXE such that:                                         |
|                   TDE07(IE)  =<  DELTAE  <  TDE07(IE +1)                                |
|                   DXEL = ( DELTAE - TDE07(IL) )/(TDE07(IL +1) - TDE07(IL) )             |
|==========================================================================================
|    Interpolate coefficient for Mach at lower lift and lower elevator                    |
|        VAL1 = CDTAB(LM,IL,IE)*(1-DXM) + CDTAB(IM+1,IL,IE)                               |
|    Interpolate coefficient for Mach at higher lift but lower elevator                   |
|        VAL2 = CDTAB(LM,IL+1,IE)*(1-DXM) + CDTAB(IM+1,IL+1,IE)                           |
|    Interpolate for mach and lift at lower elevator                                      |
|        VAL12 = VAL1*(1 - DXL) + VAL2*DXL                                                |
|    Interpolate coefficient for Mach at lower lift and higher elevator                   |
|        VAL3 = CDTAB(LM,IL,IE+1)*(1-DXM) + CDTAB(IM+1,IL,IE+1)                           |
|    Interpolate coefficient for Mach at higher lift and higher elevator                  |
|        VAL4 = CDTAB(LM,IL+1,IE+1)*(1-DXM) + CDTAB(IM+1,IL+1,IE+1)                       |
|    Interpolate for mach and lift at higher elevator                                     |
|        VAL34 = VAL3*(1 - DXL) + VAL4*DXL                                                |
|    Interpolate drag coefficient for elevator                                            |
|        CD = VAL12(1 - DXE) + VAL34*DXE                                                  |
|-----------------------------------------------------------------------------------------|
|=================|
|   RETURN        |
|=================|
```

Fig A.2.12 COEF8 Subroutine - simplified flow diagram

A.2.10 INTPAR SUBROUTINE

The subroutine INTPAR provides information used in table lookup and interpolation of aerodynamic coefficients and other tabulated physical characteristics of the QF-106 drone aircraft. INTPAR is called by other subroutines such as ENGIN6, COEF8, COEF3 which interpolate tabulated data.

Figure A.2.13 is a functional flow diagram for INTPAR. The subroutine has three input calling arguments. VALUE is the input quantity (e.g., Mach or altitude) being used as the independent variable in a table lookup and interpolation. TABLE represents a list of breakpoints for that variable and NMAX defines the size of the table in terms of the number of breakpoints.

INTPAR has two outputs. INDEX is an integer defining the breakpoint in TABLE which is equal to or just below the quantity VALUE. FRACT is the fraction defining the incremental position of VALUE between TABLE(INDEX) and TABLE(INDEX + 1).

The subroutine first checks to see whether the input VALUE is outside the range of TABLE, in which case it sets the outputs for the appropriate end point. If VALUE is within the limits, then the algorithm loops to find the correct INDEX and computes the corresponding FRACT. It should be noted that this search logic assumes that the breakpoint values are in increasing order, i.e., a higher value of index corresponds to a higher value of the independent variable.

```
|===========================================================|
||   INTPAR( VALUE , TABLE , INDEX , FRACT , NMAX)         ||
||        VALUE--Input value of Mach or Alt                ||
||        TABLE--Input list of breakpoints                 ||
||        NMAX --Input number of breakpoints               ||
||        INDEX--Output index for breakpoint<value         ||
||        FRACT--Output fraction > breakpoint              ||
|===========================================================|
|------------------------------------------------------------------------------------------|
| \  Input exceed maximum table value?                                                   / |
| T \          VALUE > TABLE( NMAX)                                                    / F |
|------------------|-----------------------------------------------------------------------|
|    Output        | \  Input below first table value?                                   / |
|    Maximum       | T \          VALUE < TABLE( 1)                                    / F |
|    Condition     |--------------|---------------------------------------------------|----|
|    INDEX = NMAX-1|   Output     |   DO LOOP                                         |<<<<<
|    FRACT = 1.0   |   Minimum    |   NLOOP = 2 to NMAX                               |   /\
|                  |   Condition  |---------------------------------------------------|   /\
|                  |   INDEX = 1  | \                                               / |   /\
|                  |   FRACT = 0. | T \          VALUE < TABLE( NLOOP )           / F |   /\
|                  |              |-------------------------------------|-------------|   /\
|                  |              |                                     | \     done?/ |   /\
|                  |              |                                     | T \     / F  |>>>>>
|                  |              |-------------------------------------|-------------|
|                  |              |   INDEX = NLOOP - 1                            |
|                  |              |FRACT=(VALUE-TABLE(INDEX))/(TABLE(NLOOP)-TABLE(INDX))|
|                  |              |                                                |
|------------------|--------------|------------------------------------------------|
|=================|
||   RETURN      ||
|=================|
```

Fig A.2.13 INTPAR Subroutine - simplified flow diagram

A.2.11 SYSIDP/SYSIDT Subroutines

These two programs contain the Recursive Least Squares system identification algorithms for the pitch and throttle axes of the adaptive flight control system.

The input and output arguments of the SYSIDP subroutine are the following:

SYSIDP(U, X, Y, N, λ, Θ_i)

where

U - System Input

X - System Variable

Y - System Output

N - System Order

λ - Forgetting factor

Θ_i- Array of unknown parameters.

SYSIDP Identification Model

$$y(k) = \Theta_1\, y(k-1) + \Theta_2\, x(k-1) + \Theta_3\, u(k-1) + \text{bias term}$$

The input and output arguments of the SYSIDT subroutine are the following:

SYSIDT(U, Y, N, λ, Θ_i)

where

U - System Input

Y - System Output

N - System Order

λ - Forgetting factor

θ_i - Array of unknown parameters.

SYSIDT Identification Model

$$y(k) = y(k-1) + \theta_i \, u(k-1)$$

A.2.12 MULTM Subroutine

The function of this subroutine is to multiply two matrices. This subroutine is used by the system identification programs SYSIDP and SYSIDT. The input and output arguments are the following:

MULTM(A, B, C, L, M, N)

where

A, B -- Matrices to be multiplied

C -- Matrix product

L, M -- Number of rows and columns of matrix A

N -- Number of columns of matrix B.

A.2.13 WNOISE Subroutine

The function of this program is to generate random numbers of a normal distribution. This program calls the subroutine RANDOM to generate uniform random numbers. This subroutine uses the central limit theorem and 12 uniform random numbers to generate a normal random number. This program is called by the executive program SIM06 to simulate white noise.

A.2.14 WINDS Subroutine

The function of this subroutine is to simulate "sinusoidal" winds. To simulate "sinusoidal" winds the user defines the maximum wind velocity, wind direction and the time period of the sinusoidal wind. Based on the wind direction, and velocity, the program computes the corresponding wind components in the inertial axis. As shown in Subsection, A.2.7, the wind velocity is subtracted from the aerodynamic velocity in the local inertial axis.

APPENDIX B

QF-106 SIMULATION PROGRAM LISTING

```
      PROGRAM SIM06
C*********************************************************************
C                                                                    *
C  SUBROUTINE NAME: SIM06                                            *
C                                                                    *
C  FUNCTION: SIM06  IS THE EXECUTIVE   PROGRAM USED TO TEST THE      *
C       QF106 ADAPTIVE CONTROLLER.THIS PROGRAM IS USED TO GENERATE   *
C       STEP RESPONSES FOR THE PITCH, ROLL AND THROTTLE CONTROL      *
C       LOOPS. APPC  CALLS INIT06 TO ESTABLISH THE DESIRED TRIM      *
C       CONDITIONS. THE USER CAN SPECIFY THROUGH NAMELIST INPUTS THE *
C       TRIM CONDITIONS AS WELL AS THE STEP RESPONSES DESIRED.       *
C                                                                    *
C      7/30/90   NEW PITCH CONTROLLER                                *
C      8/01/90   NEW SHOT CONTROLLER                                 *
C*********************************************************************
C
$FLOATCALLS
$INCLUDE: 'DRONE.COM'
$INCLUDE: 'AERO2.COM'
$INCLUDE: 'AERO3.COM'
$INCLUDE: 'AERO8.COM'
$INCLUDE: 'GENRAL.COM'
C
C*************************************************
      REAL    *4 KTRIM1,KTRIM2,KTRIM3,KTRAT1,KTRAT2
      REAL    *4 KR,KCFP,TAO,TM
      REAL    *4 TSTEP,TSTIME,PSTEP,RSTEP,PITREF,ALTINT,DTINT
      REAL    *4 AHM, SHOTM, PMLAB,SHEM
      REAL    *8 TH(5),THS(5),THR(5),LAMDS,LAMDP,LAMDR
      REAL    *4 KNZ,KTHT,KQ,KASI,KAS
      REAL    *4 GALT(7)
      INTEGER*4 JCNT
      LOGICAL   STOPSI
      CHARACTER*60 LABEL
C*************************************************
C
C FEEDBACK PITCH GAIN
      DATA     KTRIM1/ 1.25/
C FEEDBACK ROLL GAIN
      DATA     KTRIM2/ 2.25/
C THROTTLE LOOP GAIN
      DATA     KTRIM3/-1.5/
C PITCH RATE GAIN
      DATA     KTRAT1/ 28.6/
C ROLL RATE GAIN
      DATA     KTRAT2/ 57.2/
C TIME BEFORE A STEP RESPONSE IS COMMANDED
      DATA     TSTEP      /0.0/
C TOTAL SIMULATION TIME
      DATA     TSTIME     /0.0/
C PITCH STEP IN DEGREES
      DATA     PSTEP      /0.0/
C ROLL STEP IN DEGREES
      DATA     RSTEP      /0.0/
C THROTTLE STEP IN DEGREES
      DATA     THSTEP     /0.0/
      DATA     JCNT       /3/
```

```
        DATA      JJ          /0/
        DATA      LL          /0/
C ALTITUDE HOLD MODE
        DATA      AHM         /0.0/
C SPEED HOLD ON THROTTLE MODE
        DATA      SHOTM       /0.0/
        DATA      PITREF      /0.0/
        DATA      ALTINT      /0.0/
        DATA      DTINT       /0.0/
C YAW RATE DAMPING GAINS
        DATA      KR          /1.5/
        DATA      KCFP        /0.2/
C TIME CONSTANT IN SEC FOR EXPONENTIAL STEP INPUTS
        DATA      TAOR        /2.5/
        DATA      TAOP        /1.0/
        DATA      TAOV        /1.0/
        DATA      TAOH        /2.5/
        DATA      TM          /0.0/
C SAMPLING FOR SYSTEM IDENTIFICATION (1/10 SEC)
        DATA      SAMP        /4.0/
        DATA      DT          /0.1/
C PITCH GAIN SCHEDULE
        DATA      GALT        /.24,.24,.42,.59,.74,.87,.87/
C
        OPEN( 6,FILE='d:SIM.INI',STATUS='NEW')
        OPEN( 9,FILE='d:SIM1.PLT',STATUS='NEW')
        OPEN(10,FILE='d:SIM2.PLT',STATUS='NEW')
        OPEN(11,FILE='d:SIM3.PLT',STATUS='NEW')
        OPEN(12,FILE='d:SIM4.PLT',STATUS='NEW')
        OPEN(5,FILE='C:USER.DAT',STATUS='OLD')
C
C READ USER INPUT VARIABLES
C
        READ(5,13) LABEL
        READ(5,12) VR, HA, WFUEL
        READ(5,13) LABEL
        READ(5,12) TSTIME, TSELF, TSTEP, TRAMP, TPRNT
        READ(5,13) LABEL
        READ(5,12) PSTEP,  RSTEP, VSTEP, HSTEP
        READ(5,13) LABEL
        READ(5,12) PRAMP,  RRAMP, VRAMP, HRAMP
        READ(5,13) LABEL
        READ(5,12) AHM,SHOTM,SHEM
        READ(5,13) LABEL
        READ(5,12) APAH,ASHOT,ARAH
        READ(5,13) LABEL
        READ(5,12) SDEVP,SDEVS,SDEVR
        READ(5,13) LABEL
        READ(5,12) LAMDS, WNS, DMPS
        READ(5,13) LABEL
        READ(5,12) LAMDP, WNP, DMPP
        READ(5,13) LABEL
        READ(5,12) LAMDR, WNR, DMPR
        READ(5,13) LABEL
        READ(5,12) WTYPE, WVEL, WDIR, WCYCLE, WGUST, WTIME
        READ(5,13) LABEL
        READ(5,12) PMLAB1, PMLAB2, PMLAB3
C
```

```
12        FORMAT(6F10.0)
13        FORMAT(A70)
          NSAMP=(50-10)/(SAMP*DT25) + 1
          WRITE(6,16) TSTIME, TSELF, TSTEP, TRAMP, TPRNT
16        FORMAT(' TSTIME,TSELF,TSTEP,TRAMP,TPRNT : ',5F10.0)
          WRITE(6,17) PSTEP, RSTEP, VSTEP, HSTEP
17        FORMAT(' PSTEP,RSTEP,VSTEP,HSTEP: ',4F10.0)
          WRITE(6,18) AHM, SHOTM, SHEM
18        FORMAT(' AHM,SHOTM,SHEM,:',3F8.1)
          WRITE(6,215) APAH,ASHOT,ARAH
215       FORMAT(' APAH,ASHOT,ARAH:',3F8.1)
          WRITE(6,220) SDEVP,SDEVS,SDEVR
220       FORMAT(' SDEVP,SDEVS,SDEVR :',3F8.2)
          WRITE(6,19) LAMDS,WNS,DMPS
19        FORMAT(' LAMDS,WNS,DMPS:',3F10.3)
          WRITE(6,20) LAMDP,WNP,DMPP
20        FORMAT(' LAMDP,WNP,DMPP:',3F10.3)
          WRITE(6,21) LAMDR,WNR,DMPR
21        FORMAT(' LAMDR,WNR,DMPR:',3F10.3)
          WRITE(6,61) WTYPE,WVEL,WDIR,WCYCLE,WGUST,WTIME
61        FORMAT(' WTYPE,WVEL,WDIR,WCYCLE,WGUST,WTIME',6F10.3)
C
          write(*,22)
22        format(' ** SIMULATION INITIALIZATION **')
C
C
        HREF=HA
        PSI = 0.0
        ENU(1) = 0.0
        ENU(2) = 0.0
        ENU(3) = 10000.0
        ICNT = 5
        COSTHE = 1.0
C
C CALL INIT06 TO INITIALIZE SIMULATION AND ESTABLISH
C TRIM FLIGHT CONDITIONS
C
        CALL INIT06
C
         write(*,29)
29       format(' ** SIMULATION IN PROGRESS **')
C
C CALL PRINT ROUTINE FOR DEBUGGING
C
        CALL DBGPRT
C
C INITIALIZE VARIABLES AND SAVE TRIM BIASES
C
        HREF=HA
        VIASR = VIAS
        TIME = 0
        INDX = 0
        THET = THETA
        ALTINT=THET
        PHIT = PHI6D
        ICNT = 5
        PWO    = 0
        PWN    = 0
```

```
         SWN    = 0
         PPTRIM = DELTAE
         PRTRIM = DELTAA
         DTINT  = DELTAT
C
100      CONTINUE
C
         AZDD = NZCG - 1.0
C
C  CHECK FOR TIME TO STEP INPUT COMMAND
C
        IF( (TIME .GE. TSTEP) .AND. (TIME .LT. TRAMP) ) THEN
          TM = TM + DT25
          PDELT = PSTEP * (1.0 - EXP(-TM/TAOP))
          VDELT = VSTEP * (1.0 - EXP(-TM/TAOV))
          RDELT = RSTEP * (1.0 - EXP(-TM/TAOR))
          HDELT = HSTEP * (1.0 - EXP(-TM/TAOH))
        ENDIF
C
C CHECK FOR TIME TO RAMP COMMAND
C
        IF( TIME .GE. TRAMP ) THEN
         IF( PRAMP .NE. 0 )THEN
          IF((PDELT .LE. 0).OR.(PDELT .GT. ABS(PSTEP)))PSTEP=-PSTEP
          PDELT = PDELT - PSTEP*DT25/PRAMP
         ELSEIF( VRAMP .NE. 0 ) THEN
          IF((VDELT .LE. 0).OR.(VDELT .GT. ABS(VSTEP)))VSTEP=-VSTEP
          VDELT = VDELT - VSTEP*DT25/VRAMP
         ELSEIF( HRAMP .NE. 0 ) THEN
          IF((HDELT .LE. 0).OR.(HDELT .GT. ABS(HSTEP)))HSTEP=-HSTEP
          HDELT = HDELT - HSTEP*DT25/HRAMP
         ELSEIF( RRAMP .NE. 0 ) THEN
          IF((RDELT .LE. 0).OR.(RDELT .GT. ABS(RSTEP)))RSTEP=-RSTEP
          RDELT = RDELT - RSTEP*DT25/RRAMP
         ENDIF
        ENDIF
C
C  WHITE NOISE FOR SYSTEM IDENTIFICATION
C
            LL=LL+1
            IF( LL .GE. SAMP) THEN
             CALL WNOISE(0.0, SDEVP, PWN)
             CALL WNOISE(0.0, SDEVS, SWN)
             CALL WNOISE(0.0, SDEVR, RWN)
             LL = 0
            ENDIF
C
         AZDD =  NZCG - 1.0
C
C ALTITUDE HOLD/SHOT AUTOPILOT LOOPS
C
C DRONE IN SHOT MODE ?
C
         IF( SHOTM .EQ. 1) THEN
         VRF   = VIASR + VDELT
         VREF = VRF
          IF( TIME .GT. TSELF .AND. ASHOT .EQ. 1  ) THEN
C
```

```
C ADAPTIVE SHOT CONTROL LAW
C
            A1 = THS(1)
            B1 = THS(2)
C LIMIT B1 TO .01
            IF( B1 .LE. .015) B1 = .015
C
            KASI  =  (DT * WNS * WNS)/B1
            KAS   = (A1 + 2 * WNS * DMPS * DT - 1 )/B1
C
            VERR1= AMIN1( 20.0, AMAX1( -20., VREF-VIAS) )
            THCMD = KAS * VERR1
            CMDTHR= THCMD + THINT
C STOP SYSTEM IDENTIFICATION IF THROTTLE CLOSE TO NON-LINEAR REGION
C AND DO NOT UPDATE INTEGRAL
            IF ( (CMDTHR .LE. 0) .OR. (CMDTHR .GE. 43) ) THEN
              STOPSI = .TRUE.
            ELSE
              STOPSI = .FALSE.
              THINT = THINT + KASI * VERR1 * DT25
            ENDIF
            DELTAT= AMAX1( 0., AMIN1(45.0, CMDTHR ) )+SWN
            PRTDAT = DELTAT
C
          ELSE
C
C BENCHMARK SHOT CONTROL LAW
C
            VERR1= AMIN1( 20.0, AMAX1( -20., VREF-VIAS) )
            DTPROP= VERR1 * .10
            DTINT = DTINT + DTPROP * DT25
            THCMD = DTINT + VERR1
            PRDTAT=THCMD + SWN*0.5
            DELTAT=THCMD + SWN*0.5
          ENDIF
C PAST VALUES OF AIRPEED REFERENCE AND AIRSPEED
          VIAS1=VIAS
          VREF1=VREF
         ENDIF
C
C DRONE IN SPEED HOLD ON ELEVATOR MODE
C
         IF( SHEM .EQ. 1 ) THEN
          VRF   = VIASR + VDELT
          VREF = VRF
          VERR1= AMIN1( 10.0, AMAX1( -10., VREF-VIAS) )
          VERR2= AMIN1( 20.0, AMAX1( -20., VREF-VIAS) )
          ASPINT = ASPINT - 0.03 * VERR1 * DT25
          PITREF = ASPINT - 0.40 * VERR2
C
C DRONE IN ALTITUDE HOLD MODE ?
C
         ELSEIF( AHM .EQ. 1) THEN
          HCMD=HREF + HDELT
          ALTINT = ALTINT + .002 * (HCMD-HA) * DT25
          PITREF= (HCMD-HA)*0.03 - HDOT6D*.06 + ALTINT
C
C DRONE IN DIRECT PITCH ATTITUDE CONTROL
```

```
C
         ELSE
          PITRF  = THET + PDELT
          PITREF = PITRF
         ENDIF
C
C  PITCH, ROLL AND THROTTLE AUTOPILOT INNER LOOPS
C
           IF( TIME .GT. TSELF .AND. APAH .EQ. 1) THEN
C
C ADAPTIVE PITCH CONTROL LAW
C
            A11 = TH(1)
            A12 = TH(2)
            H1  = TH(3)
C
            KNZ =   A12/H1
            KTHT= -(DT * WNP * WNP)/H1
            KQ  = (1 - A11 - 2 * WNP * DMPP * DT )/H1
C
            PPROP = (THETA - PITREF) * KTRIM1 + RAWPR
            PITINT = PITINT + KTHT * PPROP * DT25
            DELTEC = PITINT + KQ * PPROP + KNZ * AZDD + PWN
            PRELEV = DELTEC
C
           ELSE
C
C BENCHMARK PITCH CONTROL LAW
C
C GAIN SCHEDULE
C
            KNZ = 3.6
            IF(MACH .GT. 1) THEN
             NA = 1 + INT(HA/10000)
             FR = (HA/10000) - (NA -1)
             IF( NA .GT. 6 ) THEN
               RKSCH = 1.0 + .5 * (MACH - 1)
             ELSE
               RKSCH = .5*(MACH -1)+(1-FR)*GALT(NA)+FR*GALT(NA+1)
             ENDIF
            ELSE
               RKSCH = 1 - .76*(VIAS - 185)/381
            ENDIF
C
            PRATE = PQR(2) * KTRAT1
            PPROP = (THETA - PITREF ) * KTRIM1 + PRATE + KNZ*AZDD
            PPROP = PPROP * RKSCH
            DELTEC = PTRIMI + PPROP + PWN*.5
            PRELEV = DELTEC
            PTRIMI = PTRIMI + .25 * PPROP * DT25
           ENDIF
C
          ROLRF =PHIT + RDELT
          ROLREF=ROLRF
          IF( TIME .GT. TSELF .AND. ARAH .EQ. 1) THEN
C
C ADAPTIVE RAH CONTROL LAW
C
```

```
              BH1 = THR(3)
              AH1 = THR(1)
              AH2 = THR(2)
C
              ROLCMD = (-AR1+AH1)*(ROLREF-PHI6D)+(-AR2+AH2)*(RREF1-PHI6D1)
              ROLCMD = ROLCMD/BH1
              DELTAC = ROLCMD + RWN
             ELSE
C
C BENCHMARK RAH CONTROL LAW
C
              RKSCH = .05 + .00093 * (400 - VIAS)
              RKSCH = AMIN1(.25,AMAX1(RKSCH, 0.05))
              RPROP=((ROLREF-PHI6D)*KTRIM2-KTRAT2*PQR(1))*RKSCH
              DELTAC = RTRIMI + RPROP
              RTRIMI = RTRIMI +  0.03 * RPROP * DT25
             ENDIF
             PHI6D1 = PHI6D
             RREF1  = ROLREF
C
C YAW RATE DAMPING
C
             YAWCD2 = KR*RAWYR - KCFP*RAWRR
C
C  CALL AIRFRAME MODEL SUBROUTINES
C
            CALL ATMS06
            CALL ENGIN6
            THRUST=TFCMD
            CALL SERV06
            DELTAE=DELTEC
            CALL MOTN06
C
            KCNT = KCNT + 1
            IF( KCNT .EQ. 200) THEN
             KCNT = 0
             ITIME=TIME+1
             WRITE(*,34) ITIME
34           FORMAT(' SIMULATION TIME (SEC) =',I10)
            ENDIF
C
C***********************************************
C ONLINE SYSTEM IDENTIFICATION
C***********************************************
C
            jj = jj + 1
            IF( jj .GE. SAMP) THEN
            jj = 0
C
C CALL SYSTEM IDENTIFICATION PROGRAMS
C
            IF(APAH .EQ.1) CALL SYSIDP(DELTEC,AZDD,RAWPR,4,LAMDP, TH )
            IF(ASHOT.EQ.1 .AND. .NOT. STOPSI ) THEN
             CALL SYSIDT ( DELTAT, VIAS,  2, LAMDS, THS)
            ENDIF
            IF(ARAH .EQ.1) CALL SYSID2 ( DELTAC, PHI6D, 3, LAMDS, THR)
C
C PRINT DATA FOR PLOTTING
```

```
C
          IF( TIME .GT. TPRNT ) THEN
            IF(AHM .EQ. 1) THEN
             WRITE(9,40)HCMD,HA,THETA,(TH(L),L=1,4)
40           FORMAT(4F9.2,4F11.6)
            ELSEIF (SHEM .EQ. 1)THEN
             WRITE(9,41)VRF,VIAS,THETA,(TH(L),L=1,4)
            ELSE
             WRITE(9,41)PITRF,THETA,PRELEV,(TH(L),L=1,4)
41           FORMAT(3F9.2,4F11.6)
            ENDIF
C
            IF( PMLAB1 .EQ. 1 ) THEN
             WRITE(10,42) VRF,VIAS,PRDTAT,THS(2),HA
42           FORMAT(3F9.2,F11.5,F9.1)
            ENDIF
C
            IF( PMLAB2 .EQ. 1 ) THEN
C******      WRITE(11,43)ROLRF,PHI6D,DELTAA,DELTAR,(THR(L),L=1,3)
             WRITE(11,43)ROLRF,PHI6D,DELTAA,DELTAR
43           FORMAT(4F9.2,3F11.6)
            ENDIF
C
            IF(PMLAB3 .EQ. 1) THEN
            WRITE(12,47) NZCG,RAWPR,MACH,VWA(1),VWA(2),VWA(3)
47           FORMAT(3F11.6,3F11.2)
            ENDIF
          ENDIF
C
          ENDIF
C
C        INCREMENT TIME
C
         TIME = TIME + DT25
C  TIME LESS THAN TEST DURATION
       IF(TIME .LE. TSTIME) GO TO 100
       WRITE(6,200)
200      FORMAT(5X,' *** END OF SIMULATION ')
C
         STOP
         END
```

```
      SUBROUTINE INIT06
C*********************************************************************
C                                                                    *
C  SUBROUTINE NAME: INIT06                                           *
C                                                                    *
C  FUNCTION:  PROVIDES EXECUTIVE LOGIC TO CONTROL INITIALIZATION     *
C OF THE SIMULATION OF THE AIRCRAFT.  INIT06 IS CALLED               *
C BY THE SIMULATION EXECUTIVE SUBROUTINE SIM06
C                                                                    *
C  FIRST THE ALGORITHM CONVERTS THE DESIRED ALTITUDE INTO THE        *
C EQUIVALENT VERTICAL POSITION  COORDINATE (Z), AND RESOLVES THE     *
C DESIRED SPEED THROUGH THE HEADING ANGLE TO OBTAIN EAST AND NORTH   *
C COMPONENTS OF VELOCITY.   RADAR ALTITUDE AND SOME ALTITUDE         *
C REFERENCE POINTS ARE ALSO SET.                                     *
C                                                                    *
C  THE REST OF THE PROGRAM IS A LOOP WHICH ITERATES TO ESTABLISH     *
C AIRFRAME TRIM CONDITIONS.  THE SIMUALTION ROUTINES ARE CALLED IN   *
C THE NORMAL ORDER TO CALCULATE ACCELERATIONS CORRESPONDING TO       *
C THE CURRENT ATTITUDE AND CONTROL SURFACE SETTINGS. THE POSITION    *
C AND VELOCITY ARE RESET AND SIMPLIFIED CONTROL LAWS ARE USE TO      *
C COMPUTE NEW SURFACE CONTROL COMMANDS FOR INPUT INTO THE NEXT PASS. *
C                                                                    *
C                                                                    *
C  SUBROUTINES CALLED: ATMS06, ENGIN6, SERV06, MOTN06                *
C                                                                    *
C*********************************************************************
C
$FLOATCALLS
$INCLUDE: 'DRONE.COM'
$INCLUDE: 'AERO2.COM'
C
      REAL*4     KTRIM1,KTRIM2,KTRIM3,KTRAT1,KTRAT2,RKSCH,RPROP,
     1           XS,YS,ZS,HS,XDS,YDS,ZDS,RE
      INTEGER*4  MAXN
C
C NORMAL ACCELERATION FEEDBACK GAIN
      DATA     KTRIM1/ .08/
C ROLL FEEDBACK GAIN
      DATA     KTRIM2/-2.25/
C THROTTLE LOOP GAIN
      DATA     KTRIM3/-1.5/
C PITCH RATE FEEDBACK GAIN
      DATA     KTRAT1/28.6/
C ROLL RATE FEEDBACK GAIN
      DATA     KTRAT2/-57.2/
C GAIN SCHEDULE
      DATA     RKSCH /0.0/
C PROPORTIONAL ROLL COMMAND
      DATA     RPROP /0.0/
C RADIUS OF THE EARTH
      DATA     RE / 20847225.4 /
C SAVE X,Y,Z POSITION AND VELOCITIES AND ALTITUDE
      DATA     XS   /0.0/
      DATA     YS   /0.0/
      DATA     ZS   /0.0/
      DATA     HS   /0.0/
      DATA     XDS  /0.0/
      DATA     YDS  /0.0/
```

```
      DATA     ZDS  /0.0/
C MAXIMUM NUMBER OF ITERATIONS ALLOWED FOR CONVERGENCE
C     DATA     MAXN/500/
      DATA     MAXN/500/
C
C SET SIMULATION GEOMETRY USING X,Y, ALTITUDE, SPEED, AND HEADING
C
      SINPSI   = SIN(D2R*PSI)
      COSPSI   = COS(D2R*PSI)
      ENUD(1) = ENU(1)
      ENUD(2) = ENU(2)
      ENUD(3)=  SQRT( (HA+RE)**2 - ENUD(1)**2 - ENUD(2)**2 ) -RE
      ENU(3) = ENUD(3)
      VL(1) = VR * SINPSI
      VL(2) = VR * COSPSI
      VL(3)=-(ENUD(1)*VL(1) + ENUD(2)*VL(2))/(RE+ENUD(3))
      HGRND = 3950.0 + 0.002 * ENU(1)
      HRADAR = HA - HGRND
C
C SAVE POSITION AND VELOCITY
C
      XS   =  ENU(1)
      YS   =  ENU(2)
      ZS   =  ENU(3)
      XDS  =  VL(1)
      YDS  =  VL(2)
      ZDS  =  VL(3)
      HS   =  HA
C
C ITERATE TO REACH TRIM CONDITION
C
      DO 100   NLOOP = 1,MAXN
C
       PTRIMI = PTRIMI + ALX(3) * KTRIM1 * 2.75 * DT25
       RKSCH = 0.24 + .00199 * (566 - VIAS)
       RKSCH = AMIN1(1.0,AMAX1(RKSCH, 0.24))
       DELTEC=(PTRIMI+ALX(3)* KTRIM1 + KTRAT1 * PQR(2)) * RKSCH
C
       RKSCH = .05 + .00093 * (400 - VIAS)
       RKSCH = AMIN1(.25,AMAX1(RKSCH, 0.05))
       RPROP = ( PHI6D * KTRIM2 + KTRAT2 * PQR(1) ) * RKSCH
       RTRIMI = RTRIMI + RPROP * 0.03 * DT25
       DELTAC = RTRIMI + RPROP
C
       DELTAT=DELTAT + KTRIM3*(ALX(2)*COSPSI + ALX(1)*SINPSI)
C
C CALCULATE AIRDATA
C
       CALL ATMS06
C
C CALCULATE ENGINE THRUST
C
       CALL ENGIN6
C
C OVERRIDE THRUST AND FUEL FLOW COMPUTED BY ENGIN6
       THRUST = TFCMD
C
C FORCE RUDDER AND NOSE WHEEL TO ZERO
```

```
C
         DELTAR = 0.0
         YAWCD1 = 0.0
         YAWCD2 = 0.0
C
C CALCULATE SURFACE DEFLECTIONS
C
         CALL SERV06
C
C CALCULATE AIRCRAFT MOTION OF DYNAMICS
C
         CALL MOTN06
C
C CALL PRINT ROUTINE
C
C        IF(NLOOP .LE. 10 ) CALL DBGPRT
C
C RESTORE INITIAL POSITION AND VELOCITY
C
          ENUD(1)    = XS
          ENUD(2)    = YS
          ENUD(3)    = ZS
          VL(1)      = XDS
          VL(2)      = YDS
          VL(3)      = ZDS
C
C EXIT AFTER FIRST PASS IF ON GROUND
C
         IF (VR.LT.250) THEN
C
C  RESET THROTTLE ON GROUND
C
          DELTAT = 0.0
C
              GOTO 200
         ENDIF
 100     CONTINUE
 200     CONTINUE
         RETURN
         END
```

```
      SUBROUTINE ATMS06
C*********************************************************************
C                                                                    *
C  SUBROUTINE NAME: ATMS06                                           *
C                                                                    *
C  FUNCTION:  THIS SUBROUTINE IS PART OF THE QF-106 SIMULATION, AND  *
C             PROVIDES A MATHEMATICAL MODEL OF ATMOSPHERIC PROPERTIES*
C             AND CALCULATES MISCELLANEOUS AIR-DATA QUANTITIES       *
C             NEEDED, MACH NUMBER, DYNAMIC PRESSURE, IMPACT PRESSURE *
C             AND INDICATED AIRSPEED.                                *
C                                                                    *
C*********************************************************************
C
$FLOATCALLS
$INCLUDE: 'DRONE.COM'
C
C LOCAL VARIABLE DECLARATION/DEFINITION
C
         REAL*4 DXH3,DXXH3,DXVE,DXXVE,FPS2KT,QC,VE,AIRDEN(12),
     1          SSOUND(12),TVIAS(11),H3IDX(12),VEIDX(11)
C
         INTEGER*4  I,J,IALT
C
C LOCAL VARAIBLE FOR SPEED OF SOUND
         DATA   DXH3     /0/
         DATA   DXXH3    /0/
         DATA   DXVE     /0/
         DATA   DXXVE    /0/
C CONVERSION FACTOR--FPS TO KNOTS
         DATA   FPS2KT /0.5924850/
         DATA   I        /0/
         DATA   J        /0/
         DATA   IALT     /0/
C IMPACT PRESSURE
         DATA   QC       /0/
C AIRSPEED ERROR DUE TO COMPRESSIBILITY
         DATA   VE       /0/
C
         DATA AIRDEN      / 0.237692E-2, 0.175556E-2, 0.149616E-2,
     X                       0.126726E-2, 0.106626E-2, 0.890683E-3,
     X                       0.704505E-3, 0.587281E-3, 0.462266E-3,
     X                       0.363905E-3, 0.286505E-3, 0.225595E-3/
C
         DATA SSOUND     / 1116.440, 1077.400, 1057.350, 1036.930,
     X                      1016.100,  994.848,  968.075,  968.075,
     X                       968.075,  968.075,  968.075,  968.075/
C
         DATA TVIAS       / -50.670,   11.800,   25.300,   35.500,
     X                        47.300,    50.67,   43.100,   12.300,
     X                       -27.000,  -327.00,  -627.00/
C
C INTERPOLATION STEPS WITHIN AERO TABLES
C
         DATA H3IDX       /     0.0,  10000.0,  15000.0,  20000.0,
     X                       25000.0,  30000.0,  36200.0,  40000.0,
     X                       45000.0,  50000.0,  55000.0,  60000.0/
C
         DATA VEIDX       /   0.0,    20.0,    60.0,   150.0,   320.0,
```

```
     X                            560.0,  960.0, 1660.0, 2280.0, 7280.0,
     X                           12280.0/
C
C
C GENERATE LINEAR INTERPOLATION INDICES
C
         I = IALT - 1
         IF (I .LE. 0) I = 1
C CHECK HEIGHT
         IF ( HA .GE. 60000. ) GO TO 115
         DO 100 J = I , 12
            IF ( HA .GE. H3IDX( J ) ) GO TO 100
            IALT = J
            GO TO 110
100      CONTINUE
110      CONTINUE
         IALT   = MAX0(1,IALT - 1)
         DXH3    = (HA-H3IDX(IALT))/(H3IDX(IALT+1) - H3IDX(IALT))
         DXXH3  = 1.0 - DXH3
         GO TO 120
C
C ABOVE 60000 FT.   TOO HIGH FOR INTERPOLATION
C
115      CONTINUE
         IALT   = 12
         DXH3    = 0.0
         DXXH3  = 1.0
120      CONTINUE
C
C AIR DENSITY
C
         RHO = AIRDEN(IALT)*DXXH3 + AIRDEN(IALT+1)*DXH3
C
C SPEED OF SOUND
C
         SS   = SSOUND(IALT)*DXXH3 + SSOUND(IALT+1)*DXH3
C
C MACH NUMBER
C
         MACH = VR / SS
C
C DYNAMIC PRESSURE
C
       QBAR = 0.5 * RHO * VR * VR
C
C TABLE LOOK*UP FOR VE = F(QC)
C
         QC    = QBAR*(1.0 + 0.269*MACH*MACH)
C
C INDEX FOR VE = F(QC)
C
         I = IVEQC - 1
         IF ( I .LT. 1 )  I = 1
         IF ( QC .LT. 7280.0) GOTO 126
         IVEQC = 11
         GO TO 140
126      CONTINUE
         DO 130 J = I , 11
```

```
            IVEQC = J
            IF ( QC .LT. VEIDX( J ) ) GOTO 140
130      CONTINUE
140      CONTINUE
         IVEQC = IVEQC - 1
         DXVE   = (QC - VEIDX(IVEQC))/(VEIDX(IVEQC+1) - VEIDX(IVEQC))
         DXXVE  = 1.0 - DXVE
         VE    = TVIAS(IVEQC)*DXXVE + TVIAS(IVEQC+1)*DXVE
C
C INDICATED AIRSPEED CALCULATION
C
         QC = ABS ( QC )
         VIAS = FPS2KT*(25.6869*SQRT(QC) + VE)
C
C LIMIT AIRSPEED TO POSITIVE VALUES
C
         IF( VIAS .LT. 0.0 ) VIAS = 0.0
C
         RETURN
         END
```

```
      SUBROUTINE INERT6
C*********************************************************************
C                                                                    *
C  SUBROUTINE NAME: INERT6                                           *
C                                                                    *
C  FUNCTION:  THIS ROUTINE IS PART OF THE QF-106 SIMULATION AND      *
C             WILL PROVIDE THE CAPABILITY TO CALCULATE MOMENT-       *
C             OF-INERTIA AND CENTER-OF-GRAVITY TERMS FOR THE QF-106  *
C             AIRCRAFT. TABLE LOOKUP AND INTERPOLATION IS THEN       *
C             PERFORMED TO CALCULATE MOMENTS OF INERTIA AND CENTER   *
C             OF GRAVITY TERMS, FOR SUBSONIC AND SUPERSONIC FLIGHT   *
C             CONDITIONS                                             *
C                                                                    *
C        SUBROUTINE: INTPAR--> INTERPOLATION UTILITY                 *
C                                                                    *
C *****************************************************************  *
C
$FLOATCALLS
$INCLUDE: 'DRONE.COM'
C
      REAL *4 G,IXGEAR,IXZGR,IYGEAR,IZGEAR,WEMPTY,DXW,
     1        WTAB(12),TXXSUB(12),TYYSUB(12),TZZSUB(12),
     2        TXZSUB(12),TXCGSB(12),TZCGSB(12),TXXSUP(12),
     3        TYYSUP(12),TZZSUP(12),TXZSUP(12),TXCGSP(12),
     4        TZCGSP(12)
C
      INTEGER*4 IW,IDX1
C *****************************************************************
C CONSTANT OF GRAVITY
      DATA  G        /32.2/
C INCREMENTAL EFFECTS OF GEAR --MOMENTS OF INERTIA
      DATA  IXGEAR   /1740.0/
      DATA  IXZGR    /0.0/
      DATA  IYGEAR   /70.0/
      DATA  IZGEAR   /560.0/
C TOTAL VEHICLE WEIGHT WITHOUT FUEL
      DATA  WEMPTY   /25693.0/
      DATA  IW       /0/
      DATA  IDX1     /2/
      DATA  DXW      /0.0/
C
C  TABLES FOR MOMENTS OF INERTIA AND CENTERS OF GRAVITY
C
      DATA  WTAB         /25693.0, 26693.0, 27193.0, 28193.0,
     1                    29629.0, 30613.0, 31597.0, 33073.0,
     2                    35042.0, 35534.0, 37861.0, 40188.0/
C
      DATA  TXXSUB       /14933.0, 15800.0, 16500.0, 17200.0,
     1                    17500.0, 20250.0, 23000.0, 25800.0,
     2                    28000.0, 28500.0, 35300.0, 42000.0/
C
      DATA  TYYSUB       /173688.0, 176000.0, 177000.0, 179000.0,
     1                    188800.0, 190600.0, 192500.0, 194000.0,
     2                    201500.0, 202500.0, 205300.0, 208000.0/
C
      DATA  TZZSUB       /183002.0, 186500.0, 188000.0, 190300.0,
     1                    200500.0, 205300.0, 210200.0, 213800.0,
     2                    223500.0, 225500.0, 234000.0, 242500.0/
```

```
C
        DATA TXZSUB          /6773.0, 6540.0, 6450.0, 6350.0,
     1                        5850.0, 5700.0, 5550.0, 5550.0,
     2                        5100.0, 5020.0, 4690.0, 4350.0/
C
        DATA  TXCGSB         /419.0, 423.0, 424.5, 427.5, 421.5, 423.7,
     1                        426.0, 427.0, 428.5, 429.0, 430.0, 431.5/
C
        DATA  TZCGSB         /92.2, 91.7, 91.5, 91.5, 91.7, 91.6,
     1                        91.5, 91.0, 91.2, 91.2, 89.2, 87.2/
C
        DATA  TXXSUP         /14933.0, 15800.0, 16500.0, 17200.0,
     1                         19700.0, 22600.0, 25500.0, 27500.0,
     2                         28000.0, 28500.0, 35300.0, 42000.0/
C
        DATA  TYYSUP         /173688.0, 176000.0, 177000.0, 179000.0,
     1                         185800.0, 187100.0, 188500.0, 189300.0,
     2                         201500.0, 202500.0, 205300.0, 208000.0/
C
        DATA  TZZSUP         /183002.0, 186500.0, 188000.0, 190300.0,
     1                         198800.0, 198000.0, 197200.0, 211000.0,
     2                         223500.0, 225500.0, 234000.0, 242500.0/
C
        DATA TXZSUP          /6773.0, 6540.0, 6450.0, 6350.0,
     1                        6050.0, 5950.0, 5850.0, 5900.0,
     2                        5100.0, 5020.0, 4690.0, 4350.0/
C
        DATA  TXCGSP         /419.0, 423.0, 424.5, 427.5, 434.0, 436.5,
     1                        439.0, 438.5, 430.0, 429.0, 430.0, 431.5/
C
        DATA  TZCGSP        /92.2, 91.7, 91.5, 91.5, 91.2, 91.0,
     1                        90.8, 90.5, 91.2, 91.2, 89.2, 87.2/
C
C  CALCULATE TOTAL WEIGHT AND MASS
C
        WEIGHT = WEMPTY + WFUEL
        MASS = WEIGHT / G
C
C SUBROUTINE TO FIND INDEX IW AND FRACTION DXW FOR WEIGHT INTERPOLATION
C
        IW=IDX1
        CALL INTRP(WEIGHT, WTAB, 12, IW, DXW)
        IDX1=IW
C
C FUEL TRANSFER AT HIGH MACH SPEEDS OR AT HIGH ALTITUDE IF
C THERE IS ENOUGH FUEL
C
       IF( (MACH .GT. 1.14).AND.(HA .GT.13000).AND.(WFUEL.GT.2500))THEN
C
C  INTERPOLATE SUPERSONIC MOIS AND CGS FOR WEIGHT
C
        IXX = TXXSUP(IW)*(1-DXW) + TXXSUP(IW+1) * (DXW) + GEAR*IXGEAR
        IYY = TYYSUP(IW)*(1-DXW) + TYYSUP(IW+1) * (DXW) + GEAR*IYGEAR
        IZZ = TZZSUP(IW)*(1-DXW) + TZZSUP(IW+1) * (DXW) + GEAR*IZGEAR
        IXZ = TXZSUP(IW)*(1-DXW) + TXZSUP(IW+1) * (DXW) + GEAR*IXZGR
        XCG = TXCGSP(IW)*(1-DXW) + TXCGSP(IW+1) * (DXW)
        ZCG = TZCGSP(IW)*(1-DXW) + TZCGSP(IW+1) * (DXW)
        YCG = 0
```

```
C
      ELSE
C
C  INTERPOLATE SUBSONIC MOMENTS AND CGS FOR WEIGHT
C
        IXX = TXXSUB(IW)*(1-DXW) + TXXSUB(IW+1) * (DXW) + GEAR*IXGEAR
        IYY = TYYSUB(IW)*(1-DXW) + TYYSUB(IW+1) * (DXW) + GEAR*IYGEAR
        IZZ = TZZSUB(IW)*(1-DXW) + TZZSUB(IW+1) * (DXW) + GEAR*IZGEAR
        IXZ = TXZSUB(IW)*(1-DXW) + TXZSUB(IW+1) * (DXW) + GEAR*IXZGR
        XCG = TXCGSB(IW)*(1-DXW) + TXCGSB(IW+1) * (DXW)
        ZCG = TZCGSB(IW)*(1-DXW) + TZCGSB(IW+1) * (DXW)
        YCG = 0
C
      ENDIF
c
        RETURN
        END
```

```
      SUBROUTINE ENGIN6
C*********************************************************************
C                                                                    *
C  SUBROUTINE NAME: ENGIN6                                           *
C                                                                    *
C  FUNCTION:  THE F106 ENGIN6 SUBROUTINE PROVIDES THE                *
C             CAPABILITY TO SIMULATE THE THROTTLE SERVOS AND         *
C             CALCULATE THRUST, AND RPM                              *
C             FOR THE QF-106 AIRCRAFT. ENGIN6 IS CALLED BY THE       *
C             VEHICLE SUBEXECUTIVE SUBROUTINE SIM06                  *
C                                                                    *
C                                                                    *
C*********************************************************************
C
$FLOATCALLS
$INCLUDE: 'AERO2.COM'
$INCLUDE: 'DRONE.COM'
C
C LOCAL VARIABLE  DECLARATION/DEFINITION
C
        REAL*4 RHZRO,FT,TFA,TFB,TFC,TFD,TFCLIM,
     1         TAUENG,RPMIDL,IDLREF,MILREF,VAL1,VAL2,DXM,
     2         DXH
        INTEGER*4 IM,IH
        LOGICAL*2 SWT5,SWT6
C
C AIR DENSITY AT SEA LEVEL
        DATA  RHOZRO /0.237692E-2/
C NORMALIZED THRUST
        DATA  FT     /0/
        DATA  TFA    /0/
        DATA  TFB    /0/
        DATA  TFC    /0/
        DATA  TFD    /0/
C THRUST LIMIT
        DATA  TFCLIM      /0/
C TIME CONSTANT FOR THROTTLE RESPONSE
        DATA  TAUENG      /0/
C IDLE RPM
        DATA  RPMIDL      /0/
C IDLE AND MILITARY THROTTLE REFERENCES
        DATA  IDLREF      /8.0/
        DATA  MILREF      /43.0/
C VARIABLES USED IN TABLE LOOKUP
        DATA  VAL1        /0/
        DATA  VAL2        /0/
        DATA  DXH         /0/
        DATA  DXM         /0/
        DATA  IH          /0/
        DATA  IM          /0/
C
C**********************************************************************
C AUTOPILOT THROTTLE SERVO ALGORITHM
C OUTPUT: DELTAT (THROTTLE POSITION)
C CHECK FOR THROTTLE MIL STOP CONDITON  (1 DEGEREE HYSTERESIS)
C
        IF (DELTAT .GT. MILREF) THEN
C
```

```
C SET THROTTLE MIL LIMIT DISCRETE
C
            SWT5 = .TRUE.
         ELSE
            IF (DELTAT .LT. (MILREF - 1.)) THEN
               SWT5 = .FALSE.
            ENDIF
C
C IS THROTTLE AT IDLE STOP?
C
           IF (DELTAT .LT. IDLREF) THEN
              SWT6 = .TRUE.
           ELSE
              IF (DELTAT .GT. (IDLREF + 1.)) THEN
                 SWT6 = .FALSE.
              ENDIF
           ENDIF
         ENDIF
C
           IF (DELTAT .LE. 0.0) DELTAT = 0.0
           IF (DELTAT .GT. 45.0) DELTAT = 45.0
C
C TABLE LOOKUP OF THRUST--FUNCTION OF MACH AND ALTITUDE
C OUTPUTS: TFIDLE, TFMIL, TFNORM, ABMAX
C
         IF( NLOOP .EQ. 1 ) THEN
C FIND INDEX IH AND FRACTION DXH FOR ALTITUDE INTERPOLATION
C
         IH=IDX1
         CALL INTRP (HA,TALT07,7,IH,DXH)
         IDX1=IH

C
C FIND INDEX IM AND FRACTION DXM
C
         IM=IDX2
         CALL INTRP (MACH,T2M41,41,IM,DXM)
         IDX2=IM
C
C INTERPOLATE TFNMAB-- THRUST FOR MAX THROTTLE AND AFTERBURNER
C
         VAL1=TFNMAB(IM,IH)  *(1-DXM) + TFNMAB(IM+1,IH)  *DXM
         VAL2=TFNMAB(IM,IH+1)*(1-DXM) + TFNMAB(IM+1,IH+1)*DXM
         ABMAX= VAL1*(1-DXH) + VAL2*DXH
C
C INTERPOLATE TFNMIL-- THRUST FOR MAX THROTTLE AND NO AFTERBURNER
C
         VAL1=TFNMIL(IM,IH)  *(1-DXM) + TFNMIL(IM+1,IH)  *DXM
         VAL2=TFNMIL(IM,IH+1)*(1-DXM) + TFNMIL(IM+1,IH+1)*DXM
         TFMIL= VAL1*(1-DXH) + VAL2*DXH
C
C INTERPOLATE TFNIDL-- THRUST FOR THROTTLE AT IDLE
C
         VAL1=TFNIDL(IM,IH)  *(1-DXM) + TFNIDL(IM+1,IH)  *DXM
         VAL2=TFNIDL(IM,IH+1)*(1-DXM) + TFNIDL(IM+1,IH+1)*DXM
         TFIDLE= VAL1*(1-DXH) + VAL2*DXH
C
C INTERPOLATE TFNNRM-- THRUST FOR NORMAL THROTTLE (80% OF MIL)
```

```
C
        VAL1=TFNNRM(IM,IH)  *(1-DXM) + TFNNRM(IM+1,IH)   *DXM
        VAL2=TFNNRM(IM,IH+1)*(1-DXM) + TFNNRM(IM+1,IH+1)*DXM
        TFNORM= VAL1*(1-DXH) + VAL2*DXH
C
        ENDIF
C
C IS THROTTLE AT IDLE?
C
        ABMIN = TFMIL + 0.25 * (ABMAX - TFMIL)
        IF (SWT6) THEN
          TFCMD = 0.9 * TFIDLE
        ELSE
C
C IS THROTTLE IN AB RANGE?
C IS AFTERBURNER COMMANDED?
C
            IF ( (ABURNR.EQ.1) .AND. (DELTAT .GE.25) ) THEN
C
C AB RANGE THRUST COMMAND
C
                TFA = ABMIN
                TFB = ABMAX - ABMIN
                TFC = DELTAT - 25.0
                TFD = .2 * THRMAX
                TFCLIM = ABMAX
            ELSE
                TFA = TFIDLE
                TFB = DELTAT
                TFC = TFMIL - TFIDLE
                TFD = 25.0
                TFCLIM = TFMIL*1.25 + 0.25*TFIDLE
           ENDIF
C
            TFCMD = TFA + TFB * TFC/TFD
            IF (TFCMD .GT. TFCLIM) TFCMD = TFCLIM
        ENDIF
C
        RETURN
        END
```

```
          SUBROUTINE SERV06
C*******************************************************************
C                                                                  *
C  SUBROUTINE NAME: SERV06                                          *
C                                                                  *
C  FUNCTION:  THIS ROUTINE IS PART OF THE QF-106 SIMULATION AND     *
C             WILL PROVIDE THE CAPABILITY TO SIMULATE THE ELEVON,   *
C             RUDDER, SPEEDBRAKE AND LANDING GEAR SERVOS AND        *
C             CALCULATE CONTROL SURFACE DEFLECTIONS.                *
C                                                                  *
C ***************************************************************  *
$FLOATCALLS
$INCLUDE: 'DRONE.COM'
C
C LOCAL VARIABLE DECLARATION/DEFINITION
C
         REAL*4 HRPBW,EMRBW,RRLIM(2),EBW,ERLIM,EERR,ELFTC,ERHTC
C HYDRAULIC RUDDER PACKAGE BANDWIDTH
         DATA       HRPBW  /15./
C EM RUDDER ACTUATOR BANDWIDTH
         DATA       EMRBW  /15./
C RUDDER RATE LIMITS
         DATA       RRLIM  /20.,40./
C ELEVON BAND WIDTH -- INTEGRATOR GAIN
         DATA       EBW     /20./
C ELEVON RATE LIMIT
         DATA       ERLIM   /25./
C SERVO ERROR
         DATA       EERR    /0/
C LEFT ELEVON COMMAND
         DATA       ELFTC   /0/
C RIGHT ELEVON COMMAND
         DATA       ERHTC   /0/
C
C LIMIT SURFACE DEFLECTION COMMANDS
C LIMIT ELEVATOR
C
        IF (DELTEC .LT. -25.0)  DELTEC = -25.0
        IF (DELTEC .GT.   8.0)  DELTEC =   8.0
C
C LIMIT AILERON
C
        IF (DELTAC .LT.  -7.0)  DELTAC = -7.0
        IF (DELTAC .GT.   7.0)  DELTAC =  7.0
C
C INDIVIDUAL ELEVON COMMANDS -- POSITIVE ELEVON IS TRAILING EDGE UP
C
        ERHTC = -DELTEC - DELTAC
        ELFTC = -DELTEC + DELTAC
C
C LIMIT ELEVON COMMANDS ERHTC & ELFTC
C
        IF (ERHTC .LT.  -15.0)  ERHTC =  -15.0
        IF (ERHTC .GT.   32.0)  ERHTC =  32.0
        IF (ELFTC .LT.  -15.0)  ELFTC = -15.0
        IF (ELFTC .GT.   32.0)  ELFTC =  32.0
C
C INDIVIDUAL ELEVON SERVOS
```

```
C
         EERR =EBW*(ERHTC - ERIGHT)
         IF (EERR .LT. -ERLIM)  EERR = -ERLIM
         IF (EERR .GT.  ERLIM)  EERR =  ERLIM
         ERIGHT = ERIGHT + EERR*DT25
         EERR =EBW*(ELFTC - ELEFT)
         IF (EERR .LT. -ERLIM)  EERR = -ERLIM
         IF (EERR .GT.  ERLIM)  EERR =  ERLIM
         ELEFT = ELEFT + EERR*DT25
C
C REFORM EQUIVALENT ELEVATOR AND AILERON DEFLECTIONS
C CONVENTIONS-ELEVONS ARE POSITIVE TRAILING EDGE UP
C CONVENTION- ELEVATOR IS POSITIVE DOWN
C CONVENTION- AILERON IS POSITIVE WITH RIGHT UP; LEFT DOWN
C
         DELTAE = -0.5*(ERIGHT + ELEFT)
         DELTAA = -0.5*(ELEFT  - ERIGHT)
C
C  USE HRP AND EMR ACTUATORS
C  PARALLEL RUDDER- EMR SERVO AND RATE LIMIT
C
        EERR = EMRBW * (YAWCD1 - DELTRP)
        TEMP = RRLIM(1)
        EERR = AMAX1( -TEMP, AMIN1( TEMP, EERR) )
        DELTRP = DELTRP + EERR * DT25
        IF( ABS(DELTRP) .LT. 1.0 E-9 ) DELTRP = 0
C
C LIMIT PARALLEL RUDDER TO +/- 24 DEG
C
        DELTRP = AMAX1(-24., AMIN1(24., DELTRP) )
C
C SERIES RUDDER - HRP SERVO AND RATE LIMIT
C
        EERR = HRPBW * (YAWCD2 - DELTRS)
        TEMP = RRLIM(2)
        EERR = AMAX1( -TEMP, AMIN1( TEMP, EERR) )
        DELTRS = DELTRS + EERR * DT25
        IF( ABS(DELTRS) .LT. 1.0 E-9 ) DELTRS = 0
C
C LIMIT SERIES  RUDDER TO +/- 6 DEG
C
        DELTRS = AMAX1(-6., AMIN1(6., DELTRS) )
C
C COMPUTE TOTAL RUDDER DEFLECTION
C
        DELTAR = DELTRP + DELTRS
C LIMIT TOTAL RUDDER DEFLECTION
        DELTAR = AMAX1( -24.0, AMIN1( 24.0, DELTAR) )
         RETURN
         END
```

```
      SUBROUTINE MOTN06
C*********************************************************************
C                                                                    *
C  SUBROUTINE NAME: MOTN06                                           *
C                                                                    *
C  FUNCTION: THIS ROUTINE PROVIDES A SIMULATION REPRESENTATION OF    *
C       THE ROTATIONAL AND TRANSLATIONAL DYNAMICS OF THE QF106       *
C       AIRCRAFT. INCLUDES SUBROUTINE CALLS  FOR TABLE LOOKUP OF     *
C       AERODYNAMIC COEFFICIENTS AND PHYSICAL CHARACTERISTICS, CAL-  *
C       CULATION OF TOTAL FORCES AND MOMENTS, CALCULATION OF ANGULAR *
C       AND TRANSLATIONAL ACCELERATIONS, AND INTEGRATION OF THESE    *
C       ACCELERATION TO UPDATE THE BODY RATES, ATTITUDES, POSITION   *
C       AND VELOCITY OF THE SIMULATED AIRCRAFT.                      *
C                                                                    *
C*********************************************************************
C
$FLOATCALLS
$INCLUDE: 'DRONE.COM'
$INCLUDE: 'GENRAL.COM'
C
C LOCAL VARIABLE DEFINITION
C
        REAL*4  AA,ALPHAP,BB,BF,CC,COS206,DD,DELZCG,IEWE,QS,QSB,QSC,
     1          SIN206,SSB,THETAD,L2B(3,3)
C
        REAL*8  DH,DHDOT,RE
C
        INTEGER*4  I
C
        DATA     AA          /0/
        DATA     ALPHAP      /0/
        DATA     BB          /0/
        DATA     CC          /0/
C COS(2.06 DEG)
        DATA     COS206      /0.999353/
        DATA     DD          /0/
C Z-CENTER OF GRAVITY
        DATA     DELZCG      /0/
C DOUBLE PRECISION ALTITUDE, AND ALTITUDE RATE
        DATA     DH          /0/
        DATA     DHDOT       /0/
C
        DATA     I           /0/
C ENGINE ROTATIONAL MOMENTUM
        DATA     IEWE        /0/
C PRODUCT OF WING AREA * QBAR
        DATA     QS          /0/
C PRODUCT OF QS * WING SPAN
        DATA     QSB         /0/
C PRODUCT OF QS * CHORD
        DATA     QSC         /0/
C RADIUS OF THE EARTH
        DATA     RE          /20847225.4/
        DATA     SIN206      /0.035946/
C SPEED BRAKE EFFECTIVE AREA (sq ft)
        DATA     SSB         /12.6/
C THETA RATE IN RAD/SEC
        DATA     THETAD      /0/
```

```
      DATA     L2B       /9*0.0/
C
C LOCAL (INERTIAL) TO BODY TRANSFORMATION MATRIX
C
      L2B(1,1) =  SINPSI*COSTHE
      L2B(1,2) =  COSPSI*COSTHE
      L2B(1,3) =  SINTHE
      L2B(2,1) =  SINTHE*SINPHI*SINPSI + COSPHI*COSPSI
      L2B(2,2) =  SINTHE*SINPHI*COSPSI - COSPHI*SINPSI
      L2B(2,3) = -COSTHE*SINPHI
      L2B(3,1) =  SINTHE*COSPHI*SINPSI - SINPHI*COSPSI
      L2B(3,2) =  SINTHE*COSPHI*COSPSI + SINPHI*SINPSI
      L2B(3,3) = -COSTHE*COSPHI
C
C SIMULATE WIND GUSTS (VWA)
C
      IF( (WTYPE .NE. 0) .AND. (TIME .GE. WTIME) ) THEN
          CALL WINDS
      ELSE
          VWA(1)=0
          VWA(2)=0
          VWA(3)=0
      ENDIF
C
C TOTAL AERODYNAMIC VELOCITY IN LOCAL FRAME
C
      VWL(1) = VL(1) - VWA(1)
      VWL(2) = VL(2) - VWA(2)
      VWL(3) = VL(3) - VWA(3)
C
C TOTAL VELOCITY IN BODY FRAME = LOCAL TO BODY TRANSFORMATION
C                                * TOTAL V IN LOCAL FRAME
C
      VWB(1) = L2B(1,1)*VWL(1) + L2B(1,2)*VWL(2) + L2B(1,3)*VWL(3)
      VWB(2) = L2B(2,1)*VWL(1) + L2B(2,2)*VWL(2) + L2B(2,3)*VWL(3)
      VWB(3) = L2B(3,1)*VWL(1) + L2B(3,2)*VWL(2) + L2B(3,3)*VWL(3)
C
C ANGLE OF ATTACK & RATE
C
      ALPHAP = ALPHA
      VWB(1) = AMAX1(.001,VWB(1))
      ALPHA  = ATAN(VWB(3)/VWB(1))
      SINALF = SIN(ALPHA)
      COSALF = COS(ALPHA)
      ALPHA  = ALPHA*R2D
      ABSALF = ABS(ALPHA)
      ALPHAD = D2R*(ALPHA - ALPHAP)/DT25
C
C SIDE SLIP ANGLE
C
      VR     = SQRT(VWL(1)**2 + VWL(2)**2 + VWL(3)**2)
      VR     = AMAX1(.001,VR)
      BETA   = ASIN ( VWB(2) / VR )
      SINBET = SIN(BETA)
      COSBET = COS(BETA)
      BETA   = BETA*R2D
      ABSBET = ABS(BETA)
C***************************************************
```

```
C CALCULATE TOATAL FORCE AND MOMENT COEFFICIENTS
C
        CALL COEF06
C
C**************************************************
        QS     =  S*QBAR
        QSB    =  BBAR*QS
        QSC    =  CBAR*QS
C
C TOTAL FORCES IN STABILITY (LOCAL WIND) AXIS
C
        FS(1)  = CLX*QS
        FS(2)  = CD*QS
        FS(3)  = CY*QS
C
C ADD SPEED BRAKES DRAG
C
        FS(2) = FS(2) + SBRAKE * QBAR * SSB

C CONVERT Z CENTER-OF-GRAVITY TO THRUST MOM ARM IN FT
C MOMENT DECREASES AS CG MOVES DOWN
C
        DELZCG = .125 + ( (ZCG - 92.2) / 12.0)
C
C  TOTAL MOMENTS IN BODY AXES
C
        MB(1)  = CLL*QSB
        MB(2)  = CM *QSC  + DELZCG * THRUST
        MB(3)  = CLN *QSB
C
C BODY AXIS FORCES AND MOMENTS &
C TRANSFORM FORCES FROM STABILITY AXES TO BODY AXES
C AND ADD THRUST
C
        FB(1)  = FS(1)*SINALF - FS(2)*COSALF + THRUST*COS206
        FB(2)  = FS(3)
        FB(3)  = -(FS(1)*COSALF + FS(2)*SINALF)-THRUST*SIN206
C
        HGRND = 3950.0 + 0.0017 * ENU(1)
C
C GROUND SPEED
C
        VG = SQRT( VL(1)**2 + VL(2)**2)  + .1
C
        IF ( (VG .LE. 5.0) .OR. (HA .LT. HGRND) ) THEN
C
C STOP THE DRONE BY ZEROING ALL DYNAMICS
C
           DO 202 I = 1, 3
              VL(I) =0.
              ALX(I) =0.
 202       CONTINUE
           PQR(1)   = 0.0
           PQR(2)   = 0.0
           PQR(3)   = 0.0
           PQRD(1)  = 0.0
           PQRD(2)  = 0.0
           PQRD(3)  = 0.0
```

```
            PSI       = 227.5
            MB(1) = 0.
            MB(2) = 0.0
            MB(3)=0.0
         ENDIF
C
C EQUATIONS OF MOTION
C
C ROTATIONAL DYNAMICS
C
         IEWE = 200.0*RPM
         AA = MB(1) - PQR(2)*PQR(3)*(IZZ-IYY) +  PQR(1)*PQR(2)*IXZ
         BB = MB(2) - PQR(1)*PQR(3)*(IXX-IZZ)+(PQR(3)**2-PQR(1)**2)*IXZ
     X        - IEWE*PQR(3)
         CC = MB(3) - PQR(1)*PQR(2)*(IYY-IXX) - PQR(2)*PQR(3)*IXZ
         DD = IXX*IZZ - IXZ**2
C
C NOTE THE FOLLOWING CONVENTION **
C  ANGULAR ACCELERATIONS = RADIANS/SEC**2
C  ANGULAR VELOCITIES    = RADIANS/SEC
C  ANGLES                = DEGREES (NOT RADIANS)
C
C BODY ANGULAR ACCELERATIONS
C
         PQRD(1) = (IZZ*AA + IXZ*CC + IEWE*IXZ*PQR(2))/DD
         PQRD(2) = BB/IYY
         PQRD(3) = (IXX*CC + IXZ*AA + IEWE*IXX*PQR(2))/DD
C
C BODY ANGULAR RATES
C
         PQR(1)  = PQR(1) + PQRD(1)*DT25
         PQR(2)  = PQR(2) + PQRD(2)*DT25
         PQR(3)  = PQR(3) + PQRD(3)*DT25
C
C BODY EULER ANGLES & RATES
C
         COSTHE  = AMAX1(1.E-10,COSTHE)
         PSID    = (PQR(2)*SINPHI + PQR(3)*COSPHI)/COSTHE
         THETAD  = (PQR(2)*COSPHI - PQR(3)*SINPHI)
         PHID    =  PQR(1) + PSID*SINTHE
         PSI     = (D2R*PSI) + PSID*DT25
         SINPSI  = SIN(PSI)
         COSPSI  = COS(PSI)
         PSI     = PSI*R2D
         PSI     = ANGL( PSI)
         ABSPSI  = ABS(PSI)
         THETA   = (D2R*THETA) + THETAD*DT25
         SINTHE  = SIN(THETA)
         COSTHE  = COS(THETA)
         THETA   = THETA*R2D
         THETA   = ANGL(THETA)
         ABSTHE  = ABS(THETA)
         PHI6D   = (D2R*PHI6D) + PHID*DT25
         SINPHI  = SIN(PHI6D)
         COSPHI  = COS(PHI6D)
         PHI6D   = PHI6D*R2D
         PHI6D   = ANGL( PHI6D)
         ABSPHI  = ABS(PHI6D)
```

```
C
C TRANSFORM BODY FORCES TO LOCAL (INERTIAL) FRAME
C
        FL(1)   = L2B(1,1)*FB(1) + L2B(2,1)*FB(2) + L2B(3,1)*FB(3)
        FL(2)   = L2B(1,2)*FB(1) + L2B(2,2)*FB(2) + L2B(3,2)*FB(3)
        FL(3)   = L2B(1,3)*FB(1) + L2B(2,3)*FB(2) + L2B(3,3)*FB(3)
C
C INERTIAL ACCELERATIONS
C
        ALX(1) =  FL(1) / MASS
        ALX(2) =  FL(2) / MASS
        ALX(3) =  FL(3) / MASS - 32.2
C
C INERTIAL VELOCITIES
C
        VL(1) = VL(1) + ALX(1) * DT25
        VL(2) = VL(2) + ALX(2) * DT25
        VL(3) = VL(3) + ALX(3) * DT25
        VFW = VL(1)*SINPSI + VL(2)*COSPSI
C
C INERTIAL POSITIONS
C
        ENUD(1) = ENUD(1) + VL(1) * DT25
        ENUD(2) = ENUD(2) + VL(2) * DT25
        ENUD(3) = ENUD(3) + VL(3) * DT25
        ENU(1)  =  ENUD(1)
        ENU(2)  =  ENUD(2)
        ENU(3)  =  ENUD(3)
C
C ALTITUDE: DOUBLE PRECISION
C
        DH = DSQRT(ENUD(1)**2 + ENUD(2)**2 + (ENUD(3) + RE)**2) -RE
C
C FORCE ALTITUDE BELOW GROUND LEVEL
C
        IF(  DH .LT. HGRND) THEN
         DH = HGRND
         ENUD(3)=DSQRT((DH + RE)**2 - ENUD(1)**2 - ENUD(2)**2)-RE
         ENU(3) = ENUD(3)
        ENDIF
C
        HA= DH
C
C ALTITUDE RATE: DOUBLE PRECISION
C
         DHDOT  =(ENUD(1) * VL(1) + ENUD(2)* VL(2) +
     &          (ENUD(3) + RE) * VL(3)) / ( DH + RE)
         HDOT6D = DHDOT
C
C C.G. NORMAL ACCELERATION  AND LONGITUDINAL DECELERATION
C
        NZCG   = (-ALX(1)*L2B(3,1) - ALX(2) * L2B(3,2) -
     1          (ALX(3) + 32.2 ) * L2B(3,3)) / 32.2
C
C LONGITUDINAL DECELERATION IN G'S
C
        LACC = -(ALX(1) * L2B(1,1) + ALX(2) * L2B(1,2) +
     1           (ALX(3) + 32.2) * L2B(1,3)) / 32.2
```

```
C
C COMPUTE BODY RATES IN DEG/SEC/SEC
C
        RAWPR = PQR(2) * R2D
        RAWRR = PQR(1) * R2D
        RAWYR = PQR(3) * R2D
C
        RETURN
        END
```

```
      SUBROUTINE WINDS
C*******************************************************************
C                                                                  *
C                                                                  *
C  FUNCTION:  SIMULATE WIND GUSTS                                  *
C                                                                  *
C                                                                  *
C  SUBROUTINES CALLED: WNOISE                                      *
C                                                                  *
C*******************************************************************
C
$FLOATCALLS
$INCLUDE: 'DRONE.COM'
$INCLUDE: 'GENRAL.COM'
C
C
        DATA TWOPI   /6.28/
        DATA TSEC    /0.0/
        DATA TAOW    /1.0/
C
        IF(WTYPE .EQ. 1 ) THEN
C
C SIMULATE SINUSOIDAL WINDS
C
          TSEC = TSEC + DT25
          ANG  =(TWOPI/WCYCLE)*TSEC
          IF( ANG .GE. TWOPI ) THEN
           ANG = 0
           TSEC = 0
          ENDIF
          WV = WVEL * SIN(ANG)
          VWY = WV * COS(WDIR * D2R)
          VWX = WV * SIN(WDIR * D2R)
          VWZ = WV * WGUST
         ELSE
          VWY = 0
          VWX = 0
          VWZ = 0
         ENDIF
C
        VWA(1) = VWX
        VWA(2) = VWY
        VWA(3) = VWZ
C
        RETURN
        END
```

```
      SUBROUTINE COEF06
C*********************************************************************
C                                                                    *
C  SUBROUTINE NAME: COEF06                                           *
C                                                                    *
C  FUNCTION: THIS SUBROUTINE CALCULATES TOTAL FORCE AND MOMENTS      *
C         COEFFICIENTS; CL, CY, CD, CLL, CM AND CLN FOR THE QF106    *
C         SIMULATION VEHICLE MODEL                                   *
C         THE SUBROUTINES CALLED ARE:  INERT6, AERO3 AND AERO8       *
C                                                                    *
C                                                                    *
C*********************************************************************
C
$FLOATCALLS
$INCLUDE: 'DRONE.COM'
$INCLUDE: 'AERO3.COM'
$INCLUDE: 'AERO8.COM'
C
C TOTAL AERODYNAMIC COEFFICIENTS
C
C          CLX = LIFT FORCE COEFFICIENT
C          CD  = DRAG FORCE COEFFICIENT
C          CM  = PITCH MOMENT COEFFICIENT
C          CY  = SIDE FORCE COEFFICIENT
C          CLN = YAW MOMENT COEFFICIENT  (C-LITTLE-N)
C          CLL = ROLL MOMENT COEFFICIENT (C-LITTLE-L)
C
        REAL*4 ALPHAR,CBAR2V,CNB,CG,FCMQ,CLBB,CLLP,CLLR,CYBETA,
     1        DCLSB,DCLGR,DCDSB,DCDGR,DCMSB,DCMGR
C
        DATA ALPHAR     /0/
C CHORD/VELOCITY RATIO
        DATA CBAR2V     /0/
C YAWING MOMENT COEFFIECIENT FOR BETA
        DATA CNB        /0/
C CENTER OF GRAVITY/CHORD RATIO
        DATA CG         /0/
        DATA FCMQ       /0/
C ROLLING MOMENT COEFFICIENT FOR BETA
        DATA CLLB       /0/
C ROLLING MOMENT COEFFICIENT FOR ROLL
        DATA CLLP       /0/
C ROLLING MOMENT COEFFICIENT FOR YAW
        DATA CLLR       /0/
C SIDE FORCE COEFFICIENT FOR BETA
        DATA CYBETA     /0/
C
C THESE NEED TO BE INITIALIZED TO THEIR PROPER VALUE
C
C COEFF-LIFT-INCR DUE TO SPEED BRAKES
        DATA DCLSB     /0/
C COEFF-LIFT-INCR DUE TO LANDING GEAR
        DATA DCLGR     /0/
C COEFF-DRAG-INCR DUE TO SPEED BRAKES
        DATA DCDSB     /0/
C COEFF-DRAG-INCR DUE TO LANDING GEAR
        DATA DCDGR     /0/
C COEFF-MOMENT-INCR DUE TO SPEED BRAKES
```

```
        DATA DCMSB      /0/
C COEFF-MOMENT-INCR DUE TO LANDING GEAR
        DATA DCMGR      /0/
C
C CALCULATE DIMENSIONAL/VELOCITY TERMS
C INCREMENT COUNTER
C
        BBAR2V = 0.5*BBAR/VR
        CBAR2V = 0.5*CBAR/VR
C
        IF (NLOOP .EQ. 1) THEN
C
C CALL MOMENTS OF INERTIA SUBROUTINE
C
          CALL INERT6
C
C CALL MACH, ALTITUDE AEROTABLE LOOKUP SUBROUTINE
C
          CALL COEF3
C
        ENDIF
C
C       CONVERT DEFLECTIONS AND ANGLES TO RADIANS
C
        ALPHAR = D2R * ALPHA
        DELTER = D2R * DELTAE
        DELAAR = D2R * DELTAA
        DELTRR = D2R * DELTAR
        BETAR  = D2R * BETA
C
C CALCULATE TOTAL LIFT FORCE COEFFICIENT
C
        CLX =CL0 + CLA*ALPHAR + CLDE*DELTER + DCLSB*SBRAKE
     1       + DCLGR*GEAR
C
C THREE DIMENTIONAL DRAG COEFFICIENT TABLE LOOK UP SUBROUTINE
C
        CALL COEF8
C
C CALCULATE TOTAL LIFT DRAG FORCE COEFFICIENT
C
        CD = CD6 + DCDSB*SBRAKE + DCDGR*GEAR
C
C CALCULATE TOTAL SIDE FORCE COEFFICIENT
C
        CYBETA = CYB0 + (CYBA + CYBA2*ALPHA)*ALPHA + CYBB*ABSBET
C
        CY = CYBETA*BETAR + CYDA*DELAAR + CYDR*DELTRR
     1                  +BBAR2V*( CYP*PQR(1) + CYR*PQR(3) )
C
C CALCULATE TOTAL PITCHING MOMENT COEFFICIENT
C
        CG = (345.9 - XCG) / (12*CBAR)
        IF (ALPHA .LE. 11) THEN
           DELALF = ALPHA - 4.64
        ELSE
           DELALF = 8.56 - 0.2*ALPHA
        ENDIF
```

```
         FCMQ = 1. + DELALF*( .07874 + .024134*DELALF )
         CM =CMO + CMDE*DELTER+ DCMSB*SBRAKE + DCMGR*GEAR
     1        + ( CLX*COSALF + CD*SINALF )*( XNP - CG)
     2        + CMQD*( FCMQ*PQR(2) + ALPHAD )*CBAR2V
C
C CALCULATE TOTAL YAWING MOMENT COEFFICIENT
C
         CNB = CNBO + ( CNBA + CNBA2*ALPHA )*ALPHA +CNBB*ABSBET
         CNDA = CNDAO + CNDAA*ALPHA + CNDADE*DELTAE
         CLN = CNB*BETAR + CNDA*DELAAR + CNDR*DELTRR
     1                 +BBAR2V*( CNP*PQR(1) + CNR*PQR(3) )
C
C CALCULATE TOTAL ROLLING MOMENT COEFFICIENT
C
         CLLB = CLLBO + ( CLLBA + CLLBA2*ALPHA ) * ALPHA
     1                 + CLLBB*ABSBET
         CLLR = CLLRO + CLLRA*ALPHA
         CLLP = CLLPO + ( CLLPA + CLLPA2*ALPHA ) * ALPHA
         CLL = CLLB*BETAR + CLLDA*DELAAR + CLLDR*DELTRR
     1                 +   BBAR2V* (CLLR*PQR(3) + CLLP*PQR(1))
C
      RETURN
      END
```

```
      SUBROUTINE COEF3
C*********************************************************************
C                                                                    *
C  SUBROUTINE NAME: COEF3                                            *
C                                                                    *
C  FUNCTION:   THIS ROUTINE IS PART OF THE QF-106 SIMULATION AND     *
C              WILL PROVIDE THE CAPABILITY FOR TABLE LOOKUP AND      *
C              INTERPOLATION OF AERODYNAMIC COEFFICIENTS AND OTHER   *
C              PHYSICAL CHARACTERISTICS OF THE QF-106 AIRCRAFT.      *
C                                                                    *
C           SUBROUTINE: INTPAR--> INTERPOLATION UTILITY              *
C                                                                    *
C*********************************************************************
C
$FLOATCALLS
$INCLUDE: 'DRONE.COM'
$INCLUDE: 'AERO2.COM'
$INCLUDE: 'AERO3.COM'
C
C  LOCAL VARIABLES
C
      REAL*4 VAL1,VAL2,DXH,DXM
      INTEGER*4 IH,IM,IDX1,IDX2,IDX3,IDX4,IDX5,IDX6
C
      DATA VAL1     /0/
      DATA VAL2     /0/
      DATA DXH      /0/
      DATA DXM      /0/
      DATA IH       /0/
      DATA IM       /0/
      DATA IDX1     /2/
      DATA IDX2     /2/
      DATA IDX3     /2/
      DATA IDX4     /2/
      DATA IDX5     /2/
      DATA IDX6     /2/
      DATA IDX7     /2/
C
C START OF PROCESSING
C
C FIND INDEX IM AND FRACTION DXM FOR MACH INTERPOLATION
C
      IM=IDX1
      CALL INTRP(MACH,TM43,43,IM,DXM)
      IDX1=IM
C
C FIND INDEX IH AND FRACTION DXH FOR MACH INTERPOLATION
C
      IH=IDX2
      CALL INTRP(HA,TALT06,6,IH,DXH)
      IDX2=IH
C
C INTERPOLATE COEFFICIENT FOR MACH AT LOWER ALTITUDE
C
      VAL1 = TCLO(IM,IH)*(1-DXM) + TCLO(IM+1,IH)*DXM
C
C INTERPOLATE COEFFICIENT FOR MACH AT HIGHER ALTITUDE
C
```

```
      VAL2 = TCLO(IM,IH+1)*(1-DXM) + TCLO(IM+1,IH+1)*DXM
C
C INTERPOLATE COEFFICIENT - CLO
C
      CLO = VAL1*(1-DXH) + VAL2*DXH
C
C INTERPOLATE COEFFICIENT FOR MACH AT LOWER ALTITUDE
C
      VAL1 = TCLA(IM,IH)*(1-DXM) + TCLA(IM+1,IH)*DXM
C
C INTERPOLATE COEFFICIENT FOR MACH AT HIGHER ALTITUDE
C
      VAL2 = TCLA(IM,IH+1)*(1-DXM) + TCLA(IM+1,IH+1)*DXM
C
C INTERPOLATE COEFFICIENT - CLA
C
      CLA = VAL1*(1-DXH) + VAL2*DXH
C
C INTERPOLATE COEFFICIENT - CLDE
C
      VAL1 = TCLDE(IM,IH)*(1-DXM) + TCLDE(IM+1,IH)*DXM
      VAL2 = TCLDE(IM,IH+1)*(1-DXM) + TCLDE(IM+1,IH+1)*DXM
      CLDE = VAL1*(1-DXH) + VAL2*DXH
C
C INTERPOLATE COEFFICIENT - XNP
C
      VAL1 = TXNP(IM,IH)*(1-DXM) + TXNP(IM+1,IH)*DXM
      VAL2 = TXNP(IM,IH+1)*(1-DXM) + TXNP(IM+1,IH+1)*DXM
      XNP  = VAL1*(1-DXH) + VAL2*DXH
C
C INTERPOLATE COEFFICIENT - CYBA2
C
      VAL1 = TCYBA2(IM,IH)*(1-DXM) + TCYBA2(IM+1,IH)*DXM
      VAL2 = TCYBA2(IM,IH+1)*(1-DXM) + TCYBA2(IM+1,IH+1)*DXM
      CYBA2 = VAL1*(1-DXH) + VAL2*DXH
C
C INTERPOLATE COEFFICIENT - CLLBO
C
      VAL1 = TCLLBO(IM,IH)*(1-DXM) + TCLLBO(IM+1,IH)*DXM
      VAL2 = TCLLBO(IM,IH+1)*(1-DXM) + TCLLBO(IM+1,IH+1)*DXM
      CLLBO= VAL1*(1-DXH) + VAL2*DXH
C
C INTERPOLATE COEFFICIENT - CLLBA2
C
      VAL1 = TCLLB2(IM,IH)*(1-DXM) + TCLLB2(IM+1,IH)*DXM
      VAL2 = TCLLB2(IM,IH+1)*(1-DXM) + TCLLB2(IM+1,IH+1)*DXM
      CLLBA2 = VAL1*(1-DXH) + VAL2*DXH
C
C INTERPOLATE COEFFICIENT - CYP
C
      VAL1 = TCYP(IM,IH)*(1-DXM) + TCYP(IM+1,IH)*DXM
      VAL2 = TCYP(IM,IH+1)*(1-DXM) + TCYP(IM+1,IH+1)*DXM
      CYP= VAL1*(1-DXH) + VAL2*DXH
C
C INTERPOLATE COEFFICIENT - CLLPO
C
      VAL1 = TCLLPO(IM,IH)*(1-DXM) + TCLLPO(IM+1,IH)*DXM
      VAL2 = TCLLPO(IM,IH+1)*(1-DXM) + TCLLPO(IM+1,IH+1)*DXM
```

```
      CLLPO= VAL1*(1-DXH) + VAL2*DXH
C
C INTERPOLATE COEFFICIENT - CLLPA
C
      VAL1 = TCLLPA(IM,IH)*(1-DXM) + TCLLPA(IM+1,IH)*DXM
      VAL2 = TCLLPA(IM,IH+1)*(1-DXM) + TCLLPA(IM+1,IH+1)*DXM
      CLLPA= VAL1*(1-DXH) + VAL2*DXH
C
C INTERPOLATE COEFFICIENT - CLLPA2
C
      VAL1 = TCLLP2(IM,IH)*(1-DXM) + TCLLP2(IM+1,IH)*DXM
      VAL2 = TCLLP2(IM,IH+1)*(1-DXM) + TCLLP2(IM+1,IH+1)*DXM
      CLLPA2= VAL1*(1-DXH) + VAL2*DXH
C
C INTERPOLATE COEFFICIENT - CNP
C
      VAL1 = TCNP(IM,IH)*(1-DXM) + TCNP(IM+1,IH)*DXM
      VAL2 = TCNP(IM,IH+1)*(1-DXM) + TCNP(IM+1,IH+1)*DXM
      CNP= VAL1*(1-DXH) + VAL2*DXH
C
C INTERPOLATE COEFFICIENT - CYR
C
      VAL1 = TCYR(IM,IH)*(1-DXM) + TCYR(IM+1,IH)*DXM
      VAL2 = TCYR(IM,IH+1)*(1-DXM) + TCYR(IM+1,IH+1)*DXM
      CYR= VAL1*(1-DXH) + VAL2*DXH
C
C INTERPOLATE COEFFICIENT - CLLRO
C
      VAL1 = TCLLRO(IM,IH)*(1-DXM) + TCLLRO(IM+1,IH)*DXM
      VAL2 = TCLLRO(IM,IH+1)*(1-DXM) + TCLLRO(IM+1,IH+1)*DXM
      CLLRO= VAL1*(1-DXH) + VAL2*DXH
C
C INTERPOLATE COEFFICIENT - CLLRA
C
      VAL1 = TCLLRA(IM,IH)*(1-DXM) + TCLLRA(IM+1,IH)*DXM
      VAL2 = TCLLRA(IM,IH+1)*(1-DXM) + TCLLRA(IM+1,IH+1)*DXM
      CLLRA= VAL1*(1-DXH) + VAL2*DXH
C
C INTERPOLATE COEFFICIENT - CNR
C
      VAL1 = TCNR(IM,IH)*(1-DXM) + TCNR(IM+1,IH)*DXM
      VAL2 = TCNR(IM,IH+1)*(1-DXM) + TCNR(IM+1,IH+1)*DXM
      CNR= VAL1*(1-DXH) + VAL2*DXH
C
C INTERPOLATE COEFFICIENT - CYDA
C
      VAL1 = TCYDA(IM,IH)*(1-DXM) + TCYDA(IM+1,IH)*DXM
      VAL2 = TCYDA(IM,IH+1)*(1-DXM) + TCYDA(IM+1,IH+1)*DXM
      CYDA= VAL1*(1-DXH) + VAL2*DXH
C
C INTERPOLATE COEFFICIENT - CLLDA
C
      VAL1 = TCLLDA(IM,IH)*(1-DXM) + TCLLDA(IM+1,IH)*DXM
      VAL2 = TCLLDA(IM,IH+1)*(1-DXM) + TCLLDA(IM+1,IH+1)*DXM
      CLLDA= VAL1*(1-DXH) + VAL2*DXH
C
C INTERPOLATE COEFFICIENT - CNDAO
C
```

```
      VAL1 = TCNDA0(IM,IH)*(1-DXM) + TCNDA0(IM+1,IH)*DXM
      VAL2 = TCNDA0(IM,IH+1)*(1-DXM) + TCNDA0(IM+1,IH+1)*DXM
      CNDA0= VAL1*(1-DXH) + VAL2*DXH
C
C INTERPOLATE COEFFICIENT - CYDR
C
      VAL1 = TCYDR(IM,IH)*(1-DXM) + TCYDR(IM+1,IH)*DXM
      VAL2 = TCYDR(IM,IH+1)*(1-DXM) + TCYDR(IM+1,IH+1)*DXM
      CYDR= VAL1*(1-DXH) + VAL2*DXH
C
C INTERPOLATE COEFFICIENT - CLLDR
C
      VAL1 = TCLLDR(IM,IH)*(1-DXM) + TCLLDR(IM+1,IH)*DXM
      VAL2 = TCLLDR(IM,IH+1)*(1-DXM) + TCLLDR(IM+1,IH+1)*DXM
      CLLDR= VAL1*(1-DXH) + VAL2*DXH
C
C INTERPOLATE COEFFICIENT - CYBB
C
      VAL1 = TCYBB(IM,IH)*(1-DXM) + TCYBB(IM+1,IH)*DXM
      VAL2 = TCYBB(IM,IH+1)*(1-DXM) + TCYBB(IM+1,IH+1)*DXM
      CYBB= VAL1*(1-DXH) + VAL2*DXH
C
C INTERPOLATE COEFFICIENT - CLLBB
C
      VAL1 = TCLLBB(IM,IH)*(1-DXM) + TCLLBB(IM+1,IH)*DXM
      VAL2 = TCLLBB(IM,IH+1)*(1-DXM) + TCLLBB(IM+1,IH+1)*DXM
      CLLBB= VAL1*(1-DXH) + VAL2*DXH
C
C INTERPOLATE COEFFICIENT - CNBB
C
      VAL1 = TCNBB(IM,IH)*(1-DXM) + TCNBB(IM+1,IH)*DXM
      VAL2 = TCNBB(IM,IH+1)*(1-DXM) + TCNBB(IM+1,IH+1)*DXM
      CNBB= VAL1*(1-DXH) + VAL2*DXH
C
C INTERPOLATE COEFFICIENT - CYB0
C
      VAL1 = TCYB0(IM,IH)*(1-DXM) + TCYB0(IM+1,IH)*DXM
      VAL2 = TCYB0(IM,IH+1)*(1-DXM) + TCYB0(IM+1,IH+1)*DXM
      CYB0= VAL1*(1-DXH) + VAL2*DXH
C
C INTERPOLATE COEFFICIENT - CNBA2
C
      VAL1 = TCNBA2(IM,IH)*(1-DXM) + TCNBA2(IM+1,IH)*DXM
      VAL2 = TCNBA2(IM,IH+1)*(1-DXM) + TCNBA2(IM+1,IH+1)*DXM
      CNBA2= VAL1*(1-DXH) + VAL2*DXH
C
C FIND INDEX IM AND FRACTION DXM FOR MACH INTERPOLATION
C
      IM=IDX3
      CALL INTRP(MACH,TM66,66,IM,DXM)
      IDX3=IM
C
C INTERPOLATE COEFFICIENT FOR MACH AT LOWER ALTITUDE
C
      VAL1 = TCM0(IM,IH)*(1-DXM) + TCM0(IM+1,IH)*DXM
C
C INTERPOLATE COEFFICIENT FOR MACH AT HIGHER ALTITUDE
C
```

```
      VAL2 = TCMO(IM,IH+1)*(1-DXM) + TCMO(IM+1,IH+1)*DXM
C
C INTERPOLATE COEFFICIENT - CMO
C
      CMO = VAL1*(1-DXH) + VAL2*DXH
C
C FIND INDEX IM AND FRACTION DXM FOR MACH INTERPOLATION
C
      IM=IDX4
      CALL INTRP(MACH,TM57,57,IM,DXM)
      IDX4=IM
C
C INTERPOLATE COEFFICIENT FOR MACH AT LOWER ALTITUDE
C
      VAL1 = TCMQD(IM,IH)*(1-DXM) + TCMQD(IM+1,IH)*DXM
C
C INTERPOLATE COEFFICIENT FOR MACH AT HIGHER ALTITUDE
C
      VAL2 = TCMQD(IM,IH+1)*(1-DXM) + TCMQD(IM+1,IH+1)*DXM
C
C INTERPOLATE COEFFICIENT - CMQD
C
      CMQD = VAL1*(1-DXH) + VAL2*DXH
C
C INTERPOLATE COEFFICIENT FOR MACH AT LOWER ALTITUDE
C
      VAL1 = TCMDE(IM,IH)*(1-DXM) + TCMDE(IM+1,IH)*DXM
C
C INTERPOLATE COEFFICIENT FOR MACH AT HIGHER ALTITUDE
C
      VAL2 = TCMDE(IM,IH+1)*(1-DXM) + TCMDE(IM+1,IH+1)*DXM
C
C INTERPOLATE COEFFICIENT - CMDE
C
      CMDE = VAL1*(1-DXH) + VAL2*DXH
C
C INTERPOLATE COEFFICIENT - CNBO
C
      VAL1 = TCNBO(IM,IH)*(1-DXM) + TCNBO(IM+1,IH)*DXM
      VAL2 = TCNBO(IM,IH+1)*(1-DXM) + TCNBO(IM+1,IH+1)*DXM
      CNBO = VAL1*(1-DXH) + VAL2*DXH
C
C FIND INDEX IM AND FRACTION DXM FOR MACH INTERPOLATION
C
      IM=IDX5
      CALL INTRP(MACH,TM43,43,IM,DXM)
      IDX5=IM
C
C INTERPOLATE COEFFICIENT FOR MACH - CYBA
C
      CYBA = TCYBA(IM)*(1-DXM) + TCYBA(IM+1)*DXM
C
C INTERPOLATE COEFFICIENT FOR MACH -CNDAA
C
      CNDAA = TCNDAA(IM)*(1-DXM) + TCNDAA(IM+1)*DXM
C
C INTERPOLATE COEFFICIENT FOR MACH - CNDADE
C
```

```
      CNDADE = TCNDAD(IM)*(1-DXM) + TCNDAD(IM+1)*DXM
C
C FIND INDEX IM AND FRACTION DXM FOR NEXT MACH INTERPOLATION
C
      IM=IDX6
      CALL INTRP(MACH,TM57,57,IM,DXM)
      IDX6=IM
C
C INTERPOLATE COEFFICIENT FOR MACH - CNBA
C
      CNBA = TCNBA(IM)*(1-DXM) + TCNBA(IM+1)*DXM
C
C FIND INDEX IM AND FRACTION DXM FOR MACH INTERPOLATION
C
      IM=IDX7
      CALL INTRP(MACH,T2M41,41,IM,DXM)
      IDX7=IM
C
C INTERPOLATE COEFFICIENT FOR MACH AT LOWER ALTITUDE
C
      VAL1 = TCLLBA(IM,IH)*(1-DXM) + TCLLBA(IM+1,IH)*DXM
C
C INTERPOLATE COEFFICIENT FOR MACH AT HIGHER ALTITUDE
C
      VAL2 = TCLLBA(IM,IH+1)*(1-DXM) + TCLLBA(IM+1,IH+1)*DXM
C
C INTERPOLATE COEFFICIENT - CLLBA
C
      CLLBA = VAL1*(1-DXH) + VAL2*DXH
C
C INTERPOLATE NEXT COEFFICIENT - CNDR
C
      VAL1 = TCNDR(IM,IH)*(1-DXM) + TCNDR(IM+1,IH)*DXM
      VAL2 = TCNDR(IM,IH+1)*(1-DXM) + TCNDR(IM+1,IH+1)*DXM
      CNDR = VAL1*(1-DXH) + VAL2*DXH
C
      RETURN
      END
```

```
      SUBROUTINE COEF8
C*******************************************************************
C                                                                  *
C  SUBROUTINE NAME: COEF8                                          *
C                                                                  *
C  FUNCTION:  THIS ROUTINE IS PART OF THE QF-106 SIMULATION AND    *
C             WILL PROVIDE THE CAPABILITY FOR TABLE LOOKUP AND     *
C             INTERPOLATION OF AERODYNAMIC COEFFICIENT AND OTHER   *
C             PHYSICAL CHARACTERISTICS OF THE QF-106 AIRCRAFT.     *
C
C        SUBROUTINE: INTPAR--> INTERPOLATION UTILITY               *
C                                                                  *
C ***************************************************************** *
$FLOATCALLS
$INCLUDE: 'DRONE.COM'
$INCLUDE: 'AERO8.COM'
C
C  LOCAL VARIABLES
C
        REAL*4 VAL1,VAL2,VAL3,VAL4,VAL12,VAL34,DXE,DXL,DXM
        INTEGER*4 IE,IL,IM,IDX1,IDX2,IDX3
C
         DATA VAL1    /0/
         DATA VAL2    /0/
         DATA VAL3    /0/
         DATA VAL4    /0/
         DATA VAL12   /0/
         DATA VAL34   /0/
         DATA DXE     /0/
         DATA DXL     /0/
         DATA DXM     /0/
         DATA IE      /0/
         DATA IL      /0/
         DATA IM      /0/
         DATA IDX1    /2/
         DATA IDX2    /2/
         DATA IDX3    /2/
C
C START OF PROCESSING
C
C FIND INDEX IM AND FRACTION DXM FOR MACH INTERPOLATION
C
         IM=IDX1
         CALL INTRP(MACH,TM19,19,IM,DXM)
         IDX1=IM
C
C FIND INDEX CL AND FRACTION DXH FOR LIFT COEFFICIENT INTERPOLATION
C
         IL=IDX2
         CALL INTRP(CLX,TCL27,27,IL,DXL)
         IDX2=IL
C
C FIND INDEX IE AND FRACTION DXE FOR ELEVATOR INTERPOLATION
C
         IE=IDX3
         CALL INTRP(DELTAE,TDE07,7,IE,DXE)
         IDX3=IE
C
```

```
CINTERPOLATE COEFFICIENT FOR MACH AT LOWER LIFT AND LOWER ELEVATOR
C
         VAL1 = TCD6(IM,IL,IE)*(1-DXM) + TCD6(IM+1,IL,IE)*DXM
C
C INTERPOLATE COEFFICIENT FOR MACH AT HIGHER LIFT BUT LOWER ELEVATOR
C
         VAL2 = TCD6(IM,IL+1,IE)*(1-DXM) + TCD6(IM+1,IL+1,IE)*DXM
C
C INTERPOLATE COEFFICIENT FOR MACH ANFD LIFT AT LOWER ELEVATOR
C
         VAL12 = VAL1*(1-DXL) + VAL2*DXL
C
C INTERPOLATE COEFFICIENT FOR MACH AT LOWER LIFT BUT HIGHER ELEVATOR
C
         VAL3 = TCD6(IM,IL,IE+1)*(1-DXM) + TCD6(IM+1,IL,IE+1)*DXM
C
C INTERPOLATE COEFFICIENT FOR MACH AT HIGHER LIFT BUT HIGHER ELEVATOR
C
         VAL4 = TCD6(IM,IL+1,IE+1)*(1-DXM) + TCD6(IM+1,IL+1,IE+1)*DXM
C
C INTERPOLATE COEFFICIENT FOR MACH ANFD LIFT AT HIGHER ELEVATOR
C
         VAL34 = VAL3*(1-DXL) + VAL4*DXL
C
C INTERPOLATE DRAG COEFFICIENT FOR ELEVATOR
C
         CD6 = VAL12*(1-DXE) + VAL34*DXE
         RETURN
         END
```

```
      SUBROUTINE WNOISE(MEAN,STD,WN)
C*********************************************************************
C                                                                    *
C  SUBROUTINE NAME: WNOISE                                           *
C                                                                    *
C  FUNCTION: GENERATES RANDOM NUMBERS WITH NORMAL DISTRIBUTION      *
C       THIS SUBROUTINE USES 12 UNIFORM RANDOM NUMBERS TO COMPUTE    *
C       NORMAL RANDOM NUMBERS USING THE CENTRAL LIMIT THEOREM        *
C*********************************************************************
C
$FLOATCALLS
      REAL*4 MEAN,WN
      REAL*4 STD,Z
C
      SUM=0.0
      DO 10 K=1,12
      CALL RANDOM(Z)
      SUM = SUM + Z
10    CONTINUE
C
      WN = (SUM - 6.0)*STD + MEAN
C
      RETURN
      END
```

```
      SUBROUTINE RANDOM(Z)
C*********************************************************************
C                                                                    *
C  SUBROUTINE NAME: RANDOM                                           *
C                                                                    *
C  FUNCTION: GENERATES RANDOM NUMBERS HAVING A UNIFORM DISTRIBUTION  *
C       BY THE MIX MULTIPLICATIVE CONGRUENTIAL METHOD                *
C*********************************************************************
C
$FLOATCALLS
      INTEGER*4 A,X
      DATA  I    /1/
C
      IF( I .NE. 0) THEN
       I=0
       M=2**20
       FM=M
       X=566387
       A=2**10 + 3
      ENDIF
C
      X = MOD(A*X, M)
      FX=X
      Z=FX/M
C
      RETURN
      END
```

```
      SUBROUTINE SYSIDP( U, Q, Y, N, LAMDA, TH )
C***********************************************************************
C                                                                      *
C                                                                      *
C  FUNCTION:  SYSTEM IDENTIFICATION PROGRAM                            *
C                                                                      *
C   Y(t) = A11*Y(t-1) + A12*Q(t-1) + H1*U(t-1) + K                     *
C                                                                      *
C  SUBROUTINES CALLED: MULTM - MATRIX MULTIPLICATION                   *
C                                                                      *
C***********************************************************************
C
$FLOATCALLS
C
        REAL* 8 PR(5,5),  PHIR(5,5), PHIRT(5,5), NUM(5,5)
        REAL* 8 TM1(5,5),  TM2(5,5),  THR(5,5), TH(5)
        REAL* 8 LAMDA, LAMDAI, DEN, CONST
        INTEGER IDLY
C
        DATA IDLY       /2/
        DATA PR         /25*0.0/
        DATA PHIR       /25*0.0/
        DATA PHIRT      /25*0.0/
C
C FIRST PASS INITIALIZATION LOGIC
C
      IF( IDLY .EQ. 2 ) THEN
          DO 10 I=1,N
          PR(I,I) = 1000
10        CONTINUE
          DO 13 I=1,N
          DO 13 J=1,N
          PHIR(I,J)=0
          PHIRT(I,J)=0
13        CONTINUE
          Y2 = Y
          LAMDAI = 1./LAMDA
          IDLY = IDLY - 1
      ELSEIF (IDLY .EQ. 1 ) THEN
          Y1 = Y
          Q1 = Q
          U1 = U
          IDLY = IDLY - 1
      ELSE
C
C  INITIALIZE MATRICES PHIR AND PHIR TRANSPOSE
C
        PHIRT(1,1)=Y1
        PHIRT(1,2)=Q1
        PHIRT(1,3)=U1
        PHIRT(1,4)=1
        PHIR (1,1)=Y1
        PHIR (2,1)=Q1
        PHIR (3,1)=U1
        PHIR (4,1)=1
C
        Y1 = Y
        Q1 = Q
```

```
        U1 = U
C
C
C COMPUTE MATRIX PRODUCTS   TM1 = PHIRT * PR
C                           TM2 = PHIR * (PHIRT * PR)
C                           NUM = PR * ( PHIR * PHIRT * PR )
C                           TM2 = (PHIRT * PR) * PHIR
C
        CALL MULTM( PHIRT,   PR,  TM1, 1, N, N)
        CALL MULTM(  PHIR,  TM1,  TM2, N, 1, N)
        CALL MULTM(    PR,  TM2,  NUM, N, N, N)
        CALL MULTM(   TM1, PHIR,  TM2, 1, N, 1)
C
        DEN = TM2(1,1) + LAMDA
C
C COMPUTE SCALAR PRODUCTS  ( PR - (NUM/DEN) )* LAMDAI
C
        DO 20 I=1, N
        DO 20 J=1, N
        PR(I,J) = (PR(I,J) - NUM(I,J)/DEN) * LAMDAI
20      CONTINUE
C
C COMPUTE MATRIX PRODUCTS  TM1 = PR * PHIR
C                          TM2 = PHIRT * THR
C
        CALL MULTM(    PR,  PHIR, TM1, N, N, 1 )
        CALL MULTM( PHIRT,   THR, TM2, 1, N, 1 )
C
C COMPUTE   THR = THR + PR * PHIR * (  Y - (PHIRT * THR) )
C
        CONST = Y - TM2(1,1)
        DO 30 I=1, N
        THR(I,1) = THR(I,1) + TM1(I,1) * CONST
        TH (I)   = THR(I,1)
30      CONTINUE
C
      ENDIF
C
        RETURN
        END
```

```
      SUBROUTINE SYSIDT( U, Y, N, LAMDA, TH )
C*********************************************************************
C                                                                    *
C                                                                    *
C  FUNCTION:  SYSTEM IDENTIFICATION PROGRAM                          *
C                                                                    *
C   Y(t) = A1*Y(t-1) + B1*U(t-1)                                     *
C    (Note for throttle axis A1 is always = 1.0)                     *
C  SUBROUTINES CALLED: MULTM - MATRIX MULTIPLICATION                 *
C                                                                    *
C*********************************************************************
C
$FLOATCALLS
C
        REAL* 8 PR(5,5),  PHIR(5,5), PHIRT(5,5), NUM(5,5)
        REAL* 8 TM1(5,5),  TM2(5,5),  THR(5,5), TH(5)
        REAL* 8 LAMDA, LAMDAI, DEN, CONST
        INTEGER IDLY
C
        DATA IDLY       /2/
        DATA PR         /25*0.0/
        DATA PHIR       /25*0.0/
        DATA PHIRT      /25*0.0/
C
C FIRST PASS INITIALIZATION LOGIC
C
      IF( IDLY .EQ. 2 ) THEN
          DO 10 I=1,N
          PR(I,I) = 1000
10        CONTINUE
          DO 13 I=1,N
          DO 13 J=1,N
          PHIR(I,J)=0
          PHIRT(I,J)=0
13        CONTINUE
          Y2 = Y
          LAMDAI = 1./LAMDA
          IDLY = IDLY - 1
      ELSEIF (IDLY .EQ. 1 ) THEN
          Y1 = Y
          U1 = U
          IDLY = IDLY - 1
      ELSE
C
C  INITIALIZE MATRICES PHIR AND PHIR TRANSPOSE
C
        PHIRT(1,1)=Y1
        PHIRT(1,2)=U1
        PHIR (1,1)=Y1
        PHIR (2,1)=U1
C
        U1 = U
        Y1 = Y
C

C
C COMPUTE MATRIX PRODUCTS    TM1 = PHIRT * PR
C                            TM2 = PHIR * (PHIRT * PR)
```

```
C                           NUM = PR * ( PHIR * PHIRT * PR )
C                           TM2 = (PHIRT * PR) * PHIR
C
         CALL MULTM( PHIRT,   PR,  TM1, 1, N, N)
         CALL MULTM(  PHIR,  TM1,  TM2, N, 1, N)
         CALL MULTM(    PR,  TM2,  NUM, N, N, N)
         CALL MULTM(   TM1, PHIR,  TM2, 1, N, 1)
C
         DEN = TM2(1,1) + LAMDA
C
C COMPUTE SCALAR PRODUCTS  ( PR - (NUM/DEN) )* LAMDAI
C
         DO 20 I=1, N
         DO 20 J=1, N
         PR(I,J) = (PR(I,J) - NUM(I,J)/DEN) * LAMDAI
20       CONTINUE
C
C COMPUTE MATRIX PRODUCTS  TM1 = PR * PHIR
C                          TM2 = PHIRT * THR
C
         CALL MULTM(    PR,  PHIR, TM1, N, N, 1 )
         CALL MULTM( PHIRT,   THR, TM2, 1, N, 1 )
C
C COMPUTE   THR = THR + PR * PHIR * (  Y - (PHIRT * THR) )
C
         CONST = Y - TM2(1,1)
         DO 30 I=1, N
         THR(I,1) = THR(I,1) + TM1(I,1) * CONST
         TH (I)   = THR(I,1)
30       CONTINUE
C
       ENDIF
C
         RETURN
         END
```

```
      SUBROUTINE MULTM( A,B,C,L,M,N)
C
      REAL*8 A(5,5),B(5,5),C(5,5),AD(5,5),BD(5,5),CD(5,5)
C
      DO 108 J=1,M
      DO 104 I=1,L
104   AD(I,J)=A(I,J)
      DO 108 K=1,N
108   BD(J,K)=B(J,K)
      DO 112 I=1,L
      DO 112 J=1,N
      CD(I,J)=0.0
      DO 112 K=1,M
112   CD(I,J)=CD(I,J) + AD(I,K)*BD(K,J)
      DO 116 I=1,L
      DO 116 J=1,N
116   C(I,J)=CD(I,J)
      RETURN
      END
```

```
      SUBROUTINE INTPAR(VALUE,TABLE,NMAX,INDX,FRACT)
C**********************************************************************
C                                                                     *
C SUBROUTINE NAME: INTPAR                                             *
C                                                                     *
C FUNCTION: INTPAR IS PART OF THE QF-106 SIMULATION AND               *
C           PROVIDES INFORMATION USED IN THE TABLE LOOK-UP AND        *
C           INTERPOLATION OF AERODYNAMIC COEFFICIENTS AND OTHER       *
C           TABULATED PHYSICAL CHARACTERISTICS OF QF-106 AIRCRAFT.    *
C           INTPAR IS CALLED BY THE AERO SUBROUTINES.                 *
C                                                                     *
C                      VALUE -- INPUT VALUE OF MACH OR ALT            *
C                      TABLE -- INPUT LIST OF BREAKPOINTS             *
C                      NMAX  -- INPUT NUMBER OF BREAKPOINTS           *
C                      INDX  -- OUTPUT INDEX FOR BREAKPOINT < VALUE   *
C                      FRACT -- OUTPUT FRACTION > BREAKPOINT          *
C                                                                     *
C**********************************************************************
$FLOATCALLS
         REAL*4 TABLE(1),VALUE,FRACT
         INTEGER*4 NMAX,INDX,NLOOP
C LOOP COUNTER FOR NEXT HIGHER VALUE IN TABLE
         DATA   NLOOP       /0/
C
C INPUT EXCEEDS MAXIMUM TABLE VALUE?
C
         IF (VALUE .GE. TABLE(NMAX)) THEN
           INDX = NMAX - 1
           FRACT = 1.0
C
C INPUT BELOW FIRST TABLE VALUE
C
         ELSE
            IF (VALUE .LT. TABLE(1)) THEN
              INDX = 1
              FRACT = 0.0
            ELSE
              IF( VALUE .GT. TABLE(INDX) ) THEN
                DO 50 NLOOP = INDX,NMAX
                  IF (VALUE .LT. TABLE(NLOOP)) THEN
                  GO TO 100
                  ENDIF
50              CONTINUE
              ELSE
                DO 60 J=1,NMAX
                   INDX = INDX - 1
                   IF( VALUE .GT. TABLE(INDX) ) THEN
                     NLOOP = INDX + 1
                     GO TO 100
                   ENDIF
60              CONTINUE
              ENDIF
C
C IF YOU GET HERE, INDEX IS POINTING TO LOWER TABLE VALUE AND THE FRAC
C TO RETURN IS FROM "TABLE(INDX)" TO "VAL" WITH VAL>TABLE
C
 100          INDX = NLOOP - 1
              FRACT = (VALUE - TABLE(INDX))
```

```
     &                     /(TABLE(NLOOP) - TABLE(INDX))
          ENDIF
        ENDIF
C
        RETURN
        END
```

```
          SUBROUTINE DBGPRT
C*********************************************************************
C                                                                    *
C  SUBROUTINE NAME: DBGPRT                                           *
C  FUNCTION:                                                         *
C  THIS PROGRAM PRINTS DATA FOR DEBUGGING PURPOSES ONLY              *
C                                                                    *
C*********************************************************************
C
$FLOATCALLS
$INCLUDE: 'DRONE.COM'
$INCLUDE: 'AERO3.COM'
C
        WRITE(6,10)
10      FORMAT(' *********** INIT06 DATA ***************')
        WRITE(6,15) HA, VR, PSI, WFUEL
15      FORMAT(' HA, VR, PSI, WFUEL : ',4F12.2)
        WRITE(6,16) DELTEC, DELTAC, DELTAT
16      FORMAT(' DELTEC, DELTAC, DELTAT : ',3F12.5)
        WRITE(6,20)
                              10       FORMAT(5X,'*********** INIT06 DATA
***************')
20      FORMAT(' *********** ATMS06 DATA ***************')
        WRITE(6,25) RHO, MACH, QBAR, VIAS
25      FORMAT(' RHO, MACH, QBAR, VIAS : ',4F11.4)
        WRITE(6,30)
30      FORMAT(' *********** ENGIN6 DATA ***************')
        WRITE(6,35) DELTAT,THRUST,RPM
35      FORMAT(' DELTAT,THRUST,RPM : ',3F12.2)
        WRITE(6,45)
45      FORMAT(' *********** SERV06 DATA ***************')
        WRITE(6,50) DELTAE, DELTAA, DELTAR
50      FORMAT(' DELTAE,DELTAA,DELTAR : ',3F12.5)
        WRITE(6,55)
55      FORMAT(' *********** COEF06 DATA **************')
        WRITE(6,57) CLX,CD,CM,CLN,CLL
57      FORMAT(' CL,CD,CM,CLN,CLL:',5F9.6)
        WRITE(6,59)CM0, CMQD, CMDE, XNP
59      FORMAT(' CM0,CMQD,CMDE,XNP :',4F10.6)
        WRITE(6,70)
70      FORMAT(' *********** INERT6 DATA ***************')
        WRITE(6,75) IXX,IYY,IZZ,IXZ
75      FORMAT(' IXX,IYY,IZZ,IXZ: ',4F10.0)
        WRITE(6,77) XCG,ZCG,WEIGHT
77      FORMAT(' XCG,ZCG,WEIGHT :',3F12.0)
        WRITE(6,80)
80      FORMAT(' *********** MOTN06 DATA *************')
        WRITE(6,85) ALPHA,THETA,BETA,PSI,PHI6D
85      FORMAT(' ALPHA THETA,BETA,PSI,PHI,PHI:',5F8.4)
        WRITE(6,95) ALX(1),ALX(2),ALX(3)
95      FORMAT(' AL(1) AL(2) AL(3) ',3F12.5)
        WRITE(6,100) FL(1),FL(2),FL(3)
100     FORMAT(' FL(1) FL(2) FL(3) ',3F12.5)
        WRITE(6,103) FS(1),FS(2),FS(3)
103     FORMAT(' FS(1) FS(2) FS(3) ',3F12.5)
        WRITE(6,105) VL(1),VL(2),VL(3)
105     FORMAT(' VL(1) VL(2) VL(3) ',3F12.5)
        WRITE(6,110) PQRD(1),PQRD(2),PQRD(3)
```

```
110     FORMAT(' PQRD(1) PQRD(2) PQRD(3) ',3F12.5)
        WRITE(6,115) PQR(1),PQR(2),PQR(3)
115     FORMAT(' PQR(1) PQR(2) PQR(3) ',3F12.5)
c
         RETURN
         END
```

LIST OF ABBREVIATIONS

AFCS	Automatic Flight Control System
AH	Altitude Hold
AAH	Adaptive Altitude Hold
APAH	Adaptive Pitch Attitude Hold
ASHE	Adaptive Speed Hold on Elevator
ASHOT	Adaptive Speed Hold on Throttle
DFCS	Drone Formation Control System
DME	Distance Measurement Equipment
DOF	Degrees of Freedom
DOS	Disk Operating System
IBM	International Business Machines
LS	Least Squares
MAL	Maximum-likelihood estimation method
MATLAB	Matrix Laboratory
MSL	Mean Sea Level
MRAC	Model Reference Adaptive Controller
PAH	Pitch Attitude Hold
PID	Proportional, Integral and Derivative
RAH	Roll Attitude Hold
RLS	Recursive Least Squares Algorithm
SHE	Speed Hold on Elevator
SHOT	Speed Hold on Throttle

STR	Self Tuning Regulator
WSMR	White Sands Missile Range

LIST OF SYMBOLS

SYMBOL	DEFINITION	DIMENSION
a_z	Normal acceleration	ft/sec^2
CD	Drag coefficient	-
CL	Lift coefficient	-
CLL	Rolling moment coefficient	-
CM	Pitching moment coefficient	-
CN	Yawing moment coefficient	-
CY	Side force coefficient	-
C_{ss}	Speed of sound	ft/sec
D_f	Drag force	lbs
g	Earth gravity constant	ft/sec^2
Ixx	Moment of Inertia about the X axis	slug ft^2
Iyy	Moment of Inertia about the Y axis	slug ft^2
Izz	Moment of Inertia about the Z axis	slug ft^2
L_f	Lift force	lbs
m	Mass of the aircraft	slugs
P	Body roll rate	deg/sec
Q	Body pitch rate	deg/sec
Q_B	Dynamic pressure	lb/ft^2
R	Body yaw rate	deg/sec
S_f	Side force	lbs
T	Sampling Time	seconds

SYMBOL	DEFINITION	DIMENSION
U	Forward velocity (Body axis)	ft/sec
u	Perturbed forward velocity	ft/sec
V	Lateral velocity (Body axis)	ft/sec
v	Perturbed lateral velocity	ft/sec
V_r	True airspeed	ft/sec
W	Downward velocity (Body axis)	ft/sec
w	Perturbed downward velocity	ft/sec
Wn	Natural Frequency	rad/sec
X,Y,Z	Body axis coordinates	ft
X_i, Y_i, Z_i	Inertial axis coordinates	ft
α	Angle of attack	degrees
β	Side-slip angle	degrees
δa	Aileron deflection	degrees
δe	Elevator deflection	degrees
δr	Rudder deflection	degrees
δt	Throttle deflection	degrees
ζ	Damping factor	-
Θ	Pitch attitude	degrees
λ	SI forgetting factor	-
ρ	Air density	$slugs/ft^3$
Φ	Roll angle	degrees
ψ	Yaw angle	degrees

CURRICULUM VITAE

Luis Eduardo Alvarado was born in Brownsville, Texas, on November 2, 1957, the son of Zoila C. de Alvarado and Jesus A. Rodarte. He received his Bachelor of Science Degree (High Honors) at the University of Texas at El Paso in May 1980. From 1980 to 1983 he worked as an assistant engineer for the IBM Corporation in Westlake California. In the fall of 1983 he entered the Graduate School at the University of Texas at El Paso. Presently he works for IBM as an advisory engineer for the Drone Formation Control System at White Sands Missile Range, New Mexico. His area of expertise is automatic control systems, navigation and guidance for ground and aerial drone targets.

Permanent address: 4724 R.T Cassidy
El Paso, Texas 79924

This thesis was typed by Luis E. Alvarado